Linux and Network SysAdmin Companion Reference

Quick Refreshers, Primers, and Overviews

Tristan Mendoza 2024

Over the years I have made my own documentation, dissatisfied with existing material for not being thorough enough or otherwise. For networking An absurd amount of time is spent showing IP addressing and subnetting to the point of confusing the student, and who the hell needs 4 books filled with white space and giant font text to explain CCNP topics?

Much of these are made for 8.5x11 so you can print out ones you think you'll use most often and have them handy in your cubicle or whatever (I prefer printing two pages on one side of a page) I didn't originally make these for the public but found like myself, it helps to have something around for a topic you may have either not seen, haven't worked on in months or years, and want to get up to speed quick, so I tried to make them more digestable for others to use.

Parts too nonverbose? Then go online for longer overviews or tutorials with examples. I don't like wasting page space.

Have fun and I hope it helps! I salute all my brothers and sisters out there struggling with the insane IT job market. There have been times of my life when I have said "I didn't leave the tech job market- it left ME- I never left tech". I am leaving the option to get this for free, but if you can afford it, I definitely need the money.

I intend to add something on cryptography, and there is a lot more to add about BGP-VXLAN-EVPN, ASA firewall, other routing configuration, but that stuff is a bit sloppy and not cleaned up for others to read yet.

-Tristan Mendoza
Austin, Texas, USA
tristanm.tx@gmail.com
http://tristanmtx.github.io/
https://www.linkedin.com/in/tristanmendozatx
Celebrating 30 years in I.T. management, network analysis, and systems administration
Cisco CCNAx2 (Route/ Switch & CyberOps/ InfoSec); AWS Solutions Architect Associate, MEF SDN/NFV Professional
EC-Council Certified Ethical Hacker (CEH), and Forensics Investigator (CHFI);
CompTIA Sec+, Net+, and CASP

Linux Topics (mostly RHEL)

01. Managing the Kernel - Loadable Kernel Modules
02. Changing Parameters of Kernel Modules (Devices)
03. Tuning Resources with sysctl
04. More on /proc and /sys
05. /dev - Device Nodes
06. dmesg
07. Important Log File Locations and Using journalctl
08. Integrating and Configuring rsyslogd and journald
09. Basic Hardware Utilities
10. Process Management
11. The sysstat Suite and sar (vmstat, iostat, etc)
12. Linux Performance Observability Tools as shown by Brendan Gregg
13. SysVinit and systemd Command Equivalents
14. systemd Command Overview
15. SysVinit - Directory Structures and inittab
16. Runlevel Service Management Tools -SysVinit initscript utilities
17. Example systemd Unit Script
18. Contents of the systemd Package
19. ifconfig, netstat, ip - net-tools and iproute2 equivalents
20. NetworkManager Service - Persistent Changes with nmcli
21. Getting Rid of Network Manager
22. systemd-networkd
23. Using iw and wpa_supplicant
24. WiFi Pentesting Tools
25. Netfilter/ iptables
26. Everything about securing systems with SELinux
27. Reviewing RAID Types (Prep to add mdadm, NFS/iSCSI/NAS/SAN, etc stuff)
28. Setting up and securing SMB and NFS
29. Red Hat Identity Management (IdM) - FreeIPA Identity Policy Audit
30. Installing Standalone Kerberos Server (no FreeIPA)
31. Standalone Enterprise-ready LDAP - 389 Directory Server (no FreeIPA)
32. RHEL BIND Enterprise DNS Setup and Overview
33. Mail Server Provisioning on RHEL 8/9 (Enterprise-ready)
34. Git Commands and Quick Overview
35. Docker Commands and Quick Overview (needs more added)
36. Ansible Commands and Quick Overview w/ VMWare/Cloud Modules
37. Overview of GCP/AWS/Azure Offering Names and Equivalents

Network Topics (Net+ CCNA CCNP Etc)

01. OSI and DARPA Network Models/ Network basics I Transport protocols
- Network protocols
- Data-Link protocols
- Ping and Traceroute(and their relatives)
- Round trip ping trip explained PC1-〉router-〉PC2 and back

02. Subnetting and binary overview- Addressing crash course
- IPv4 Fast subnetting technique, VLSM, binary practice
- IPv4 quick reference sheet and worksheet (for dry erase marker)
- IPv6 overview and SLAAC/DHCPv6, Cisco routing in IPv6
- IPv6 subnetting large ISP blocks down /32 to /60 (intro only)

03. Cisco-based switching topics
- Cisco switchports: access and trunks, port security IVR/SVI VLANs, VTP, RoaS
- Spanning Tree, EtherChannel (PAgP and LACP)

04. Cisco Misc. topics
Load Balancing with VRRP, HSRP, GLBP
Cisco-based DHCP, NAT (non-ASA), Access control lists

05. Routing on Cisco Devices
- OSPF and LSAs, Stub areas, NSSA, etc.
- EIGRP, k-values and metric, RD/FD, SR/FS in topology IPv4 Interior routing- static, RIP, OSPF, EIGRP examples IPv6 version (show changing routing to another type)

06. BGP

07. Short subjects and other stuff
- BGP route leaks and BGP hijacking intro (more of a 1 pg "what is...")
- Setting up GRE with IPSEC for a Typical VPN
- DMVPN - Dynamic Multipoint VPN (multipoint GRE, IPSec, NHRP)
- TCP Ports - Interactions and Scanning (nmap concepts, port responses)
- SSH Tunnels; port forwarding, jump hosts, etc (Linux)
- Socat to Netcat Commands Task Quick Reference
- Cables and connectors, line speeds (sortof Network+ junk)

Managing the Kernel - Loadable Kernel Modules

Linux typically modules have the extension .ko ("kernel object") since version 2.6 (previous versions used the .o extension). Other OS's - kernel loadable module (kld) in FreeBSD; kernel extension (kext) in macOS; kernel extension module in AIX; kernel-mode driver in Windows NT. Linux modules are generally in subdirectories of **/lib/modules/** named for kernel version. Numbering format same as Linux versioning- major.minor.patch Odd numbers for minor version are developmental. /lib/modules/*.ko files i.e, bridge.ko for network support

After **/etc/initab** specifies the default runlevel, it kicks off **/etc/rc.sysinit** to load modules
/etc/modprobe.conf -- associates/aliases drivers to devices such as eth0
"alias eth0 natsemi" says use natsemi driver/module for eth0
/etc/modprobe.d/modprobe.conf.dist -- large set of standard autoloaded aliases
/etc/modprobe.d/blacklist and **blacklist-compat** -- aliases that are not loaded

modprobe <modulename> - loads/adds modules AND auto-adds their dependencies
-v verbose; -r remove; -a add; -l list all modules; -t [dir] list modules in directory
insmod - to insert 3rd party drivers/modules; common error: dependency error: "unresolved symbol…." inserts only specified module -not it's dependencies. **modprobe** is preferable
-e make persistent; -f force; -L prevent simultaneous loading of the same module; -o specify optional module name
modinfo <modulename> - param fields have variables and such used in the system calls. -V version, -n name -a author -d description -p parameters
depmod - is run at startup right before **modprobe** to give it dependency info- creates **/lib/modules/modules.dep**
lsmod - lists loaded modules. Indicates size, number of dependencies and what it is used by
modprobe -l - will do sort of the same, with dependencies
cat /proc/modules/ - all modules currently loaded- subdirectories for /pcmcia, /net, /arch, /fs, /drivers
/sys/module also has sub-directories that contains information about each kernel module installed

rmmod remove modules from the kernel but it does not check for dependencies
modprobe -r <modulename> - removes a module from the kernel after checking for dependencies

udev is a device manager that manages the automatic detection and kernel module loading for both coldplug and hotplug devices; in charge of the **/dev** virtual file system to dynamically creates device files as devices are added and removed. When providing new hardware like a USB key, udev wakes up, intitializes the new HW with kernel so kernel can load proper modules automatically.

udevadm monitor
Upon plugging in a usb stick, devices and bus messages (truncated below), see module listed for fat and vfat

```
[root@rhelserver ~]# udevadm monitor
monitor will print the received events for:
UDEV - the event which udev sends out after rule processing
KERNEL - the kernel uevent

KERNEL[69484.521158] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1 (usb)
KERNEL[69484.554385] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/1-1:1.0 (usb)
UDEV  [69484.564011] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1 (usb)
KERNEL[69484.583125] add      /module/usb_storage (module)
UDEV  [69484.608070] add      /module/usb_storage (module)
KERNEL[69484.619712] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/1-1:1.0/host3 (scsi)
UDEV  [69484.619737] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/1-1:1.0 (usb)
KERNEL[69484.619745] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/1-1:1.0/host3/scsi_host/host3 (scsi_host)
KERNEL[69484.619752] add      /bus/usb/drivers/usb-storage (drivers)
UDEV  [69484.621580] add      /bus/usb/drivers/usb-storage (drivers)
UDEV  [69484.622757] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/1-1:1.0/host3 (scsi)
UDEV  [69484.623611] add      /devices/pci0000:00/0000:00:11.0/0000:02:03.0/usb1/1-1/1-1:1.0/host3/scsi_host/host3 (scsi_host)
KERNEL[69486.685029] add      /module/fat (module)
KERNEL[69486.685064] add      /kernel/slab/fat_cache (slab)
KERNEL[69486.685076] add      /kernel/slab/fat_inode_cache (slab)
UDEV  [69486.687754] add      /module/fat (module)
UDEV  [69486.687778] add      /kernel/slab/fat_cache (slab)
UDEV  [69486.687785] add      /kernel/slab/fat_inode_cache (slab)
KERNEL[69486.689217] add      /module/vfat (module)
UDEV  [69486.690176] add      /module/vfat (module)
```

Do it manually: unplug USB, then
lsmod | grep fat - shows vfat module still loaded. Don't rely on **udev** to unload what it activated
modprobe -r vfat --removes modules and dependencies no longer needed
modprobe vfat -- loads the module and dependencies (which udev is expected to accomplish automatically)
---
udevadm controls **systemd-udevd**, requests kernel events, manages event queue, and simple debugging.
udevadm [**info** | **trigger** | **settle** | **monitor**] <options> AND **udevadm control** <command>
The **/etc/udev/rules.d/** - directory allows naming devices when they are connected

Changing Parameters of Kernel Modules (Devices)

modinfo cdrom - reports params. These can be changed, but only by unloading the module with **modprobe -r cdrom**, then **modprobe cdrom lockdoor=0** (for example, to turn off the lockdoor boolean param)

```
[root@rhelserver ~]# lsmod | grep cdrom
cdrom                  42556  1 sr_mod
[root@rhelserver ~]# modinfo cdrom
filename:       /lib/modules/3.10.0-121.el7.x86_64/kernel/drivers/cdrom/cdrom.ko
license:        GPL
srcversion:     B5F2D59440347DFFB175E71
depends:
intree:         Y
vermagic:       3.10.0-121.el7.x86_64 SMP mod_unload modversions
signer:         Red Hat Enterprise Linux kernel signing key
sig_key:        42:49:68:9E:EF:C7:7E:95:88:0B:13:DF:E4:67:EB:1B:7A:91:D1:08
sig_hashalgo:   sha256
parm:           debug:bool
parm:           autoclose:bool
parm:           autoeject:bool
parm:           lockdoor:bool
parm:           check_media_type:bool
parm:           mrw_format_restart:bool
```

It used to be modifying modprobe.conf could make changes but in RHEL7 it changed
These are default settings for kernel modules, from the associated rpm packages:
/lib/modprobe.d/dist-alsa.conf and **/lib/modprobe.d/dist-blacklist.conf**
You don't want to edit them.

Instead, edit files in **/etc/modprobe.d/** By default it is empty- it is the place to put custom conf files.
(see **man 5 modprobe.d** - "options" section*). In this directory create/ **vim cdrom.conf** and add:
options cdrom lockdoor=0

Generally you need to restart to see the effects and truly reload- reloading the module isn't enough.
For some modules you can look in **/sys/module/**, find a directory for the module, and see a file called parameters (or something), but it is up to the programmers to provide this kind of file. For cdrom it isn't. Looking in **dmesg | grep <modulename>** might help find something about when the module was initialized.

```
[root@rhelserver module]# cd cdrom/
[root@rhelserver cdrom]# ls
coresize  holders  initsize  initstate  notes  refcnt  sections  srcversion  taint  uevent
[root@rhelserver cdrom]# dmesg | grep cdrom
[    2.738573] cdrom: Uniform CD-ROM driver Revision: 3.20
[root@rhelserver cdrom]# dmesg | grep -A5 cdrom
[    2.738573] cdrom: Uniform CD-ROM driver Revision: 3.20
[    2.739174] sr 1:0:0:0: Attached scsi CD-ROM sr0
[    2.777003] usb 2-2: New USB device found, idVendor=0e0f, idProduct=0002
[    2.777006] usb 2-2: New USB device strings: Mfr=0, Product=1, SerialNumber=0
[    2.777008] usb 2-2: Product: VMware Virtual USB Hub
[    2.782794] hub 2-2:1.0: USB hub found
```

*man page refers to locations **/etc/modprobe.d/*.conf**, **/lib/modprobe.d/*.conf**, and **/run/modprobe.d/*.conf**

Fields in /etc/modprobe.d/modprobe.conf (or **/etc/modprobe.conf**)

alias {wildcard} {module name}	Specify an alternate name for a module with a long name.
include {file name}	Add configuration files to a module.
options {module name} {option}	Options to be added to each module before insertion into the kernel.
install {module name} {command}	Run the command specified without inserting the module into the kernel.

Don't confuse with shared library files! (aren't kernel modules)

The **/usr/lib** and **/lib** directories are the default system library file locations where the system libraries are kept. Contains routines, which are used by various applications; loaded into memory when executable that links to them is loaded. They are then shared with other applications.
When added, new library file details are passed on to **/etc/ld.so.conf** (default system library info)
Running **ldconfig** updates changes in that file and loads the shared libraries from locations specified by **/etc/ld.so.cache**.

ldconfig -f <config-file>
- -C <cache-file> where library updates will be stored
- -v view details of library file, rebuilds cache
- -p - show shared library cache
- -n /<location> update the library file info in the specified location instead of the default

ldd -v <program-name> - List dynamic dependencies of executable files or shared objects.
LD_LIBRARY_PATH environment variable

Tuning Resources with sysctl

/proc/sys/ - has directories for all interfaces the kernel offers: abi, crypto, debug, dev, fs, kernel, net, sunrpc, vm
/proc/sys/kernel/ for core kernel functionality, **/proc/sys/net/** for networking, **/proc/sys/vm/** for virtual memory
kernel files for different parameters like "osrelease" or "hostname"
cat /proc/sys/net/ipv4 spits out 1. Change this to off with **echo 0 > ip_forward** (not persistent)
vm contains "swappiness" holding a value 0-100 (eagerness of kernel to swapping out unneeded memory)
Default is 30. Greater value (60) might help kernel swap out faster. Change it the same way but it wouldn't stick after reboot (not persistent).

/etc/sysctl.conf contains the default system configurations which hold these sort of values. It should be edited if you want to change these values and have them stick after a reboot.

sysctl -a -- to display **/proc/sys/** values
sysctl -a | grep forward ---to show forwarding/ routing files/values

Just like you see in Java, net.ipv6.conf.lo.mc would refer to the file **/proc/net/ipv6/conf/lo/mc**

```
[root@rhelserver ~]# sysctl -a | grep forward
net.ipv4.conf.all.forwarding = 0
net.ipv4.conf.all.mc_forwarding = 0
net.ipv4.conf.default.forwarding = 0
net.ipv4.conf.default.mc_forwarding = 0
net.ipv4.conf.ens33.forwarding = 0
net.ipv4.conf.ens33.mc_forwarding = 0
net.ipv4.conf.lo.forwarding = 0
net.ipv4.conf.lo.mc_forwarding = 0
net.ipv4.ip_forward = 0
net.ipv6.conf.all.forwarding = 0
net.ipv6.conf.all.mc_forwarding = 0
net.ipv6.conf.default.forwarding = 0
net.ipv6.conf.default.mc_forwarding = 0
net.ipv6.conf.ens33.forwarding = 0
net.ipv6.conf.ens33.mc_forwarding = 0
net.ipv6.conf.lo.forwarding = 0
net.ipv6.conf.lo.mc_forwarding = 0
```

On boot the **sysctl** process is started. In the past, it just checked **/etc/sysctl.conf**, which is now empty (as of RHEL7). The file instructs that "default settings live in **/usr/lib/sysctl.d/00-system.conf** To override those settings enter new settings here, or in an **/etc/sysctl.d/<name>.conf** file"
So you can still put your custom stuff in **/etc/sysctl.conf**, but putting them in an **/etc/sysctl.d/<name>.conf** might be considered more organized if many custom changes are applied
Before you add custom scripts to **/etc/sysctl.d/** it will likely only contain **/etc/sysctl.d/00-sysctl.conf** which it turns out is simply a symlink to **/etc/sysctl.conf**

/usr/lib/sysctl.d/ contains: 00-system.conf 50-default.conf libvirtd.conf
The numbers in the filename represent the read-order

So, in **/etc/sysctl.d/** run **vim 50-ipforward.conf** and put in it **net.ipv4.ip_forward = 1**, save and on reboot you should be able to route IP packets.

The **sysctl** command promises to set the value and write to the **sysctl** files. Sander Vugt (whose examples I just used in this section on /proc and kernel modules) recommends against trusting it, instead suggesting to make sure by going directly to the source and echoing the value in **/proc** as demonstrated.

<u>sysctl command options</u>
sysctl to modify kernel parameters at runtime. The parameters available are those listed under /proc/sys/
-w {variable}={value} Write a parameter value/ change the sysctl setting.
{variable}={value} Set a key parameter value.
-n Disable the printing of the key name while displaying the kernel parameters.
-e Ignore errors about unknown keys.
-a Display all the parameter values that are currently available.
-A Display those in a tabular format.

More on /proc and /sys

cat /proc/cpuinfo - to display cpu info
/proc/net/ has files containing protocol information and settings, such as routing tables (used by netstat, ss)
/proc/meminfo - the used and unused memory and the shared memory and buffers used by the kernel.
/proc/version - produces same mostly as uname
/proc/cmdline - command line boot options passed to the kernel by the boot loader at boot time.
/proc/devices -list of device drivers (hardware) configured into the currently running kernel.
/proc/filesystems - list of filesystems that are configured into the kernel for mounting
/proc/partitions - partition info: the major and minor number of each partition, name, and number of blocks.
/proc/dma direct memory access. DMA gives hardware devices direct access to memory independent of the CPU.
/proc/interrupt lists the interrupt request (IRQ) channels in use.
/proc/iomem - mapping of the memory allocated to each device and the input/output port assignments for memory.
/proc/modules - lists the kernel modules that the computer is currently using.
/proc/bus - contains a file or directory for each USB device attached

Original UNIX systems needed to access structured data so user-space applications could find out about process attributes. Early programs like ps found out about the running processes by directly accessing **/dev/mem** or **/dev/kmem**, and interpreting the raw data). That requires root access, so the applications that use them have to be setuid or SGID. Also it's not good to expose system data directly to user-space. One solution was to make these types of info available through system calls, but creating new system calls over and over to export small process data wasn't so good either. The /proc filesystem was the answer- where interfaces and structures (directories and files) could be kept the same, even as the underlying data structures in the kernel changed.

On kernel startup virtual filesystems are made to reference hardware and resources:
- **udev** creates the virtual filesystem **/dev** to put hardware definitions (like addresses) into files representing them as interfaces so other things can talk to the them
- **sysctl** then helps set up **/proc** and **/sys** virtual filesystem representations of how the kernel modules and drivers to interact with those /dev device files

The **procfs** /proc virtual filesystem was only meant to hold legacy ***process*** information, system attributes from a few main systems. Since it is easily accessible from both kernel and user-space it eventually got the reputation as the convenient place to put other read/write system files to adjust settings, kernel and subsystem operation (cpuinfo, memory statistics, device information). Things then arbitrarily got thrown in created clutter with device data stuck in different spots all over the place.

The **sysfs** /sys virtual filesystem was implemented in kernel 2.5-2.6 (2003 or so) to organize things better, separate device and driver ***system*** information from **/proc** to add structure, provide a uniform way to expose system information and control points (settable system and driver attributes) to user-space from the kernel. For each object found to put in **/dev** on the system, the kernel automatically creates directories in **/sys** when drivers are registered based on the driver type and their values (representing the device hierarchy too)

Many of the legacy system information and control points are still accessible in /proc - entries already added to /proc were allowed to remain for backward compatibility (especially traditional items like CPU and memory). All new device busses and drivers are expected to expose their info and control points via sysfs.
Note: Many traditional UNIX and Unix-like operating systems use /sys as a symlink to the kernel source tree

<u>The /sys Directory Structure</u>
/sys/block - has an entry for each block device (mostly drives use data blocks)
/sys/bus/devices/ - symlinks for each device - point to device's directory under root/
/sys/bus/drivers/ has a directory for each device driver that is loaded
/sys/class/- has files for each class of devices
/sys/devices lists devices discovered, in a directory tree reflecting/representing the device hierarchy
/sys/firmware/
/sys/net/
/sys/fs/ - where each filesystem exporting attributes creates its hierarchy- see ./fuse.txt
/sys/dev/char/ and **/sys/dev/block/** hold symlinks named <major>:<minor> pointing to the appropriate

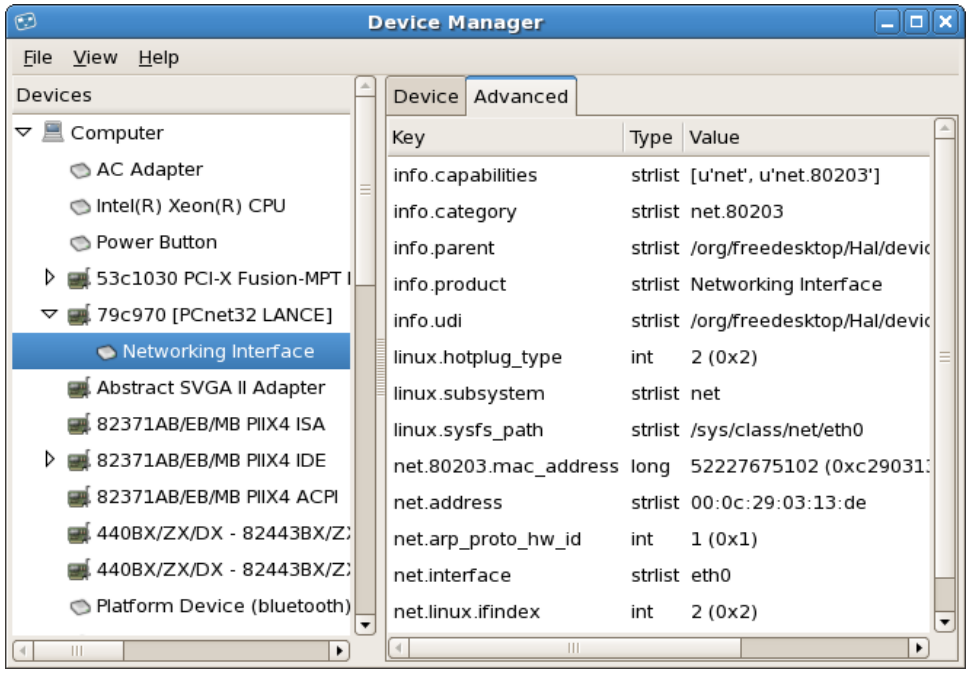
Quick examples:

Setting laptop brightness (not persistent)	**echo N > /sys/class/backlight/acpi_video0/brightness**
Get a network card's MAC address	cat /sys/class/net/enp1s0/address

/dev - Device Nodes

- Device drivers mapping service requests to device access represent HW resources in a device tree
- contains vital information such as the device type, with a minor # to identify device, major # to identify driver
- acts as an interface between the OS and hardware; part of OS or installed on-demand

Device Tree
- "a structure that lists all HW installed on a system and assigns device nodes to them."
- auto generated by the computer's RAM on startup, when a new device is installed, or when a device or system configuration is modified.



Special Devices

/dev/zero Provides unlimited null characters (0 bytes) for writing into any program or file. It is used for generating an empty file of certain size.

/dev/null Does not provide any data to a program or file. It discards all data written to it. It is used as an output file when the output is not required by the user and should be trashed.

/dev/random Functions as a random number generator; gathers random input from device drivers and other sources, saves it as bits in an entropy pool, output as bytes to applications.
When the pool is exhausted, will block the reading application until more random input is collected.

/dev/urandom Like /dev/random, except doesn't block the reading application if the entropy pool is exhausted; also uses pseudorandom input (less secure)

mknod [OPTION] {NAME} {TYPE} [MAJOR MINOR].
Create device files that are not present. Makes use of the major and minor node numbers of a device

dmesg

dmesg shows info about all the hardware controlled by the kernel. CPU, memory, disk drives, network cards, etc. Shows the contents of **/var/log/dmesg**, which holds the kernel ring buffer (ring because it dumps old messages as new ones come in, maintaining the same file size)

It includes actions taken at startup, like configuring hardware devices.
See error messages as they occur with **watch "dmesg | tail -20"**
Setting the level to number 1 (-n 1) only allows panic messages to be seen.

dmesg > kernel_msgs.txt - dump current ring buffer contents into a file, handy for emailing for help troubleshooting
← Pipe output to more, less, head, tail, or grep such as **dmesg | grep -i memory**
← **cat /var/log/dmesg** can generate similar output, but running dmesg provides more control
←
Some options for dmesg:

-C, **--clear**	Clear the ring buffer. you can still view logs stored in '**/var/log/dmesg**' files
-c, **--read-clear**	Clear the ring buffer contents after printing.
-f, **--facility** *list*	Restrict output to defined (comma separated) list of facilities
-k, **--kernel**	Print kernel messages.
-l, **--level** *list*	Restrict output to defined (comma separated) list of levels.
-r, **--raw**	Print the raw message buffer, i.e., don't strip the log level prefixes.
-s, **--buffer-size** *size*	Output **size** to query the buffer; 16392 by default. If you set larger, can view entire buffer.
-u, **--userspace**	Print userspace messages.
-x, **--decode**	Decode facility and level (priority) number to human readable prefixes.
-n, **--console-level** *level*	Set priority level that messages are logged using syslog level number or name (*-n 1* or *-n alert* prevents all messages except emergency) All levels are still written to **/proc/kmsg**, so syslogd can still be used to control exactly where kernel messages appear. When the *-n* option is used, dmesg will not print or clear the kernel ring buffer.

- "dmesg Explained" article from Linux Gazette website explains **dmesg** lines
- *http://www.tldp.org/LDP/LG/issue59/nazario.html*

hald - Hardware Abstraction Layer (HAL) daemon provides all applications with data about current hardware.
dbus - Desktop Bus - Inter-Process Communication (IPC) system
- Allows processes to communicate with each other provides notification system for HAL events
- Can start services for an application's needs- applications register with dbus daemon to participate.
- Can send system wide alerts such as “new hardware detected” and “print queue modified.”

Important Log File Locations and Using journalctl

Commands for viewing, including those with 'relevant command' listed: cat, tail, grep, less, zcat, zgrep, zmore, etc.

Log location	***Relevant command***	***Comments***
/**var/log/messages**		Includes some security related info- where PAM is logged (Ubuntu after Natty instead uses /var/log/syslog)
/var/log/boot.log		Self-explanatory see also boot.log.* for older logs
/var/log/wtmp	**last**	Successful (mnemonic "working") logins, logouts, reboots
(same as above)	**last reboot**	Shows just reboots, same with keyword logins, etc
/var/log/btmp	**lastb**	Shows history of failed login attempts (mnemonic "bad")
/var/log/lastlog	**lastlog**	Shows recent user logins
/var/log/secure		Authpriv messages, more
/var/log/dmesg	**dmesg**	Boot and dbus messages
/root/install.log		Is updated - good to keep copies after fresh installs of things
/var/log/anaconda.log		System installation info on Red Hat systems
/var/log/dpkg.log		dpkg logs for Debian installs/removes (see also /apt/ directory)
/var/log/yum.log		yum command log for Red Hat installs/removes
/var/log/kern.log		Kernel logs
/var/log/daemon.log		Info from various background daemons
/var/log/user.log		Information about all user level logs
/var/log/audit/		Audit daemon (auditd).
/var/log/setroubleshoot/		SELinux's setroubleshootd
/var/log/sssd/		System security services (remote directory access, auth)
/var/log/cron.log		Crond logs
/var/log/sa/		Daily sar files for systat
/var/log/maillog		Mail server logs (sendmail/ dovecot)
/var/log/qmail/		Qmail log directory
/var/log/httpd/		Apache access and error logs directory
/var/log/lighttpd/		Lighttpd access and error logs directory
/var/log/mysqld.log		MySQL database server log file
/var/log/cups		Printer and printing related log messages
/var/log/samba/		Samba info, Windows support

- For authenication, Debian-based use **/var/log/auth.log** while Red Hat-based use **/var/log/secure**
- **/var/log/faillog** - if available can also provide info on failed login attempts
- Similar to **wtmp**, but perhaps not as useful is **/var/log/utmp**

Remember the syslog standard levels/priorities:
0= emerg, 1=alert, 2=crit, 3= err, 4=warning, 5=notice, 6=info, 7=debug
Levels with a higher numerical level give less information (7 least, 0 most)

journald and journalctl options

-f	New log entries as they are added	journalctl -u mysql -f
-k	Kernel messages (example: 5 boots previous	journalctl -k -b -5
-u	Messages for specified systemd service	journalctl -u httpd
-b	Boot msgs; last boot, use -1; two boots ago -2; etc.	(see above -k example)
--list-boots	List system boots	
-r	Show in reverse order; most recent entries first	
-p	Display messages by priority	journalctl -p err
--since, --until	Time range; formats: 09:00; "1 hour ago", 2 days ago	journalctl --since "2017-05-23 23:15:00"
-o	Output options, includes short, verbose export > filename	journalctl -o json-pretty
_PID, _UID	Messages produced by a specific PID, UID, GID	journalctl _UID=100 (remember **id** command)
_COMM, etc.	Name of executable or path, hostname. Similar options	journalctl _HOSTNAME=myhost
	Various attributes supported- see man page for list	_SELINUX_CONTEXT= system_r:policykit_t

The journal is saved in the **/var/log/journal/**

Integrating and Configuring rsyslogd and journald

Rsyslog is still central to logging - journald doesn't have all the mechanisms to do things
journalctl: -b for booting info, --since=yesterday or , -o for verbose, u= service (or PID) for process
journalctl without options just dumps from the binary to screen.
/etc/systemd/journald.conf

Sending journald logs to rsyslog: *In **/etc/syslog.conf** add:*
$modload imuxsock - (input module unix socket)
$OmitLocalLogging off
- and -
*In **/etc/rsyslog.d/listend.conf**, add:* $SystemLogSocketName /run/systemd/journal/syslog

Sending rsyslog to journald *In **/etc/rsyslog.conf** add:*
$modload omjournal *.* :omjournal:
(this tells it, from any facility, and any priority, send to omjournal

Other input modules (Apache into rsyslog example)

```
$ModLoad imfile

$InputFileName /var/log/httpd/error_log
$InputFileTag apache-error:
$InputFileStateFile state-apache-error
$InputRunFileMonitor
```

Exporting to a DB using an output module:

```
$ModLoad ommysql

$ActionOmmysqlServerPort 1234
*.* :ommysql:database-servername,database-name,database-
userid,database-password
```

Enabling remote logging in /etc/rsyslog.conf (these are there for us in the file, just commented out)
Provides UDP syslog reception - classical method - best backward compat but you can lose messages
$ModLoad imudp
$UDPServerRun 514

Provides TCP syslog reception - the better option
$ModLoad imtcp
$TCPServerRun 514

For sending out, look at the forwarding rules and find this:
Replace remote-host with IP addy or servername in hosts files
. @@remote-host:514

Sample conf file lines. Basic syntax is facility.level … target

```
*.info:mail.none:authpriv.none:cron.none    /var/log/messages
        .none is exclusion, * is wildcard.  This line logs everything of level 1 or higher except as noted
 authpriv.*                 /var/log/secure
        This catches all messages from authpriv and puts into /secure
*.emerg                     *
        Sends all emergency messages to all tty's and logs (local- not remote)
uucp.news.crit              /var/log/spooler
        News errors using uucp facility
local7.*                    /var/log/boot.log
        local7 is a boot facility.  See more facilities in the syslog man page
```

Logrotate - /etc/logrotate.conf **-** Specifies to rotate logs, daily, weekly, monthly; how long to keep logs before deleting; a create directive to replace the moved log with a blank empty file to use; dateext directive to use date as a filename extension; compress or not
There is also an include directive pointing to /etc/logrotate.d as a place for specific RPMs to throw logs /etc/logrotate.d/ to hold more granular rule files for syslog, http, yum, up2date, samba, etc., processes).

Basic Hardware Utilities

lspci Display info about all the PCI buses and all the peripheral components connected to a computer.
-k shows the kernel drivers that support the device.**-v** for verbose on most of these ls commands
-t displays a tree diagram that shows connections between all busses, bridges, and devices.
lsusb Display all the USB components connected to a computer. gives more info
-v for verbose, **-s bus_name** to specify a bus
There is also **lshw, lshal**, **lscpu**, **lsblk**, **lspcmcia**
lshw -X - a graphic frontend
lshw -html > hardware.html -- Create a html overview of the hardware

hwinfo --<hw_item> - displays HW info using libhd **- <hw_item>** is one of the following: all, bios, block, bluetooth, braille, bridge, camera, cdrom, chipcard, cpu, disk, dsl, dvb, fingerprint, floppy, framebuffer, gfxcard, hub, ide, isapnp, isdn, joystick, keyboard, memory, modem, monitor, mouse, netcard, network, partition, pci, pcmcia, pcmcia-ctrl, pppoe, printer, scanner, scsi, smp, sound, storage-ctrl, sys, tape, tv, usb, usb-ctrl, vbe, wlan, zip
--short - for brief output
--debug level - set debuglevel
hwinfo --disk show all disks
hwinfo --short --block just an overview
hwinfo --disk --only /dev/sdb - show a particular disk
hwinfo --disk --save-config=all - save disk config state
hwprobe=bios.ddc.ports=4 hwinfo --monitor - try 4 graphics card ports for monitor data (default: 3)

kudzu [options]
- deprecated in RHEL6 - probed and compared with /etc/sysconfig/hwconfig. Configuration options, ran at boot time.

Kernel State Monitoring Utilities
uname OS name, version, license, processor, and hardware details.
uptime Duration the system has been running, number of users currently logged on, and load averages
tload Graphical representation of load average for the past 1 minute, 5 minutes, and 15 minutes.
w is essentially a combination of uptime, who, and ps -a

Memory Monitoring Utilities
free Total memory: free, used, shared, buffered, and cached. -m for megabytes, -k for kb, -s delay
vmstat Virtual memory usage; I/O address info, processor allocation currently running
pmap Display the mapping of processes with memory resources.
iostat Reports on CPU and device utilization. Usage statistics for storage devices and partitions.
memstat -a
systat -m

Disk and Filesystem Utilities
partprobe - tells the kernel to re-read the partition table
-d Cancel any updates; **-s** Display the storage devices and their partitions.
df -h list volumes- memnonic disk free
du -h -d1 file sizes, human readable, depth=1 --max-depth=1; **-c** shows combined total at bottom
cifsiostat - Average reads and writes/sec vs those ops issued per sec, files opened/closed/deleted per sec

Other useful stuff:
id - lists current user's uid, gid, and group memberships
file <filename> Dumps file info including file type
stat <filename.txt> Shows full attributes of file
which <command> -a Shows the pathnames for the command using $path env var
lsof | grep <keyword> List open files, filter by keyword
Remember you can use the watch command (like **watch free**) to have it update on the screen
watch -n 1 -d free Sets a 1 sec interval and -d will highlight values that changed

Make a baseline of system performance on a newly installed Linux system
/proc/meminfo - the used and unused memory and the shared memory and buffers used by the kernel.
/proc/cpuinfo - the CPU information and system architecture dependent items
dmesg displays a snapshot of information about the hardware that is controlled by the kernel, and that output can be redirected to a file for use in system baseline documentation.

Process Management

Moving a running job from foreground to background
- Typing ctrl-z *stops* and places in the background, then type **bg** to *resume* in background
- Type **jobs -l** to see it listed as running (lists all background jobs)
- Executing **fg <job#>** with the number it is listed as will bring it to the foreground
- **dd if=/dev/zero of=/dev/null &** - Copying nothing to nowhere;
- The **&** sends it to the background upon execution without having to stop and resume

[A "shell job" is any command run from the shell. Processes belong to the shell from which they were started]
nohup ./test > test.output - makes process keep running when terminal/tty has been closed
(means to run without attaching process to parent terminal)

Kill Signals
kill -9 <PID> - brute-force terminate process
killall <program-name> - affects all matching processes running

Num.	Name	Default Action	Description
1	SIGHUP	terminate process	Terminal line hangup- restarts the process. After doing so, the process will keep the same PID that it had before. Useful for restarting a service after making changes to a configuration file.
2	SIGINT	terminate process	Interrupt program - sends a ctrl-c key sequence
3	SIGQUIT	create core image	Quit program
6	SIGABRT	create core image	Abort program (formerly SIGIOT)
9	SIGKILL	terminate process	Brute-force kill; process may not clean up - resources allocated may remain allocated until the system is restarted
15	SIGTERM	terminate process	Default- allows process to clean up before exiting

CPU Priority
"Niceness" values for CPU priority - remember that -20 is highest priority and 19 is the lowest. Default is 0
- Use **nice** for *starting* programs: **nice -19 ./test** (or PID) - sets to 19 (is not minus 19, which would be set with **--19**)
- Use **renice** for adjusting currently running processes: **renice <priority> -user** (or PID)
- **renice 17 -p 1234** changes the nice value of the job with process id 1134 to 17; no dash for command option
- Change the nice value of process 1234 to -3 with: **renice -3 -p 1234**
- Modify the priority of all processes that belong to a group or user with the -g and -u (instead of -p for process)
- It is recommended changing nice in steps of -5 at a time
- *Only root can apply negative nice values*
- You can set the default nice value of a particular user or group in the /etc/security/limits.conf
- It uses this syntax: [*username*] [hard | soft] priority [*nice value*]
  user12 hard priority 1

The **-n** modifier produces different behavior in nice and renice
- In nice, using -n option adjusts the value from the default (0) and the "-" will actually work as minus.
- Increase the priority: **nice -n -5 ./test.sh** - Decrease the priority: **nice -n 5 ./test.sh**
- So using the -n makes it less confusing with the whole issue of whether it is a minus or a simple hyphen
- In renice, the -n option specifies the actual number so **renice -n -19 -p 3534** sets the nice value to -19

If you have to mess with nice alot, then it is a sign to increase system resources (RAM, but primarily CPU) to handle the load on the server)

Using PS to Inspect Processes

The ps command 3 syntax styles. BSD style - options are not preceded with a dash (**ps aux**); UNIX/LINUX style - options are preceded by a dash (**ps -ef**) "ps aux" is not the same as "ps -aux". "-u" is used to show process of that user, but plain "u" means show detailed information in the other mode. Usually these can be mixed in Linux: For example "ps ax -f". Finally, some options use the GNU style with double-dashes preceding a word, like --forest.

Most often used is display all processes. The "u" or "-f" options display more detailed info

$ ps ax --or-- **$ ps aux**
$ ps -ef --or-- **$ ps -ef -f** *Using -F can sometimes be more effective than -f*

Other options (there are many):

-u <username>	Display process by user -Multiple usernames can be provided separated by a comma
-f --or-- -F	Show detailed info
-C <processname>	Search by name (must be exact)
-p	Search by PID (separate with comma for multiple e.g., **$ ps -f -p 3150,7298,6544**
--sort=	Prefix with a "-" or "+" to sort descending/ascending (e.g., **$ ps aux --sort=-pcpu,+pmem**)
--forest	Displays ASCII art of process tree. Sort and forest don't work well together
--ppid <PID>	Search by parent PID (show child processes)
-o	Show listed columns sep with commas (e.g., **$ ps -e -o pid,uname,pcpu,pmem**)
-L	Show threads of a process, (e.g., **$ ps -p 3150 -L or ps -C firefox-bin -L -o pcpu,state**)

- Piping to grep instead of using the -C option can be more convenient **$ ps -ef | grep apache**
- Fields for --sort and -o options are available in the manpage section "Standard Format Specifiers" (there are tons)
- Rename the column labels using the format "-o pid,uname=USERNAME,pcpu=CPU_USAGE,pmem,comm"
- When listing usernames, if the length is greater than 8 characters then ps will show the UID instead of username.
- As usual pipe to "less" to ease reading. If sorting remember to consider piping to head 5 or tail -5 as needed, etc.

Common attributes to query/ sort by, etc.
The man page lists many attributes you can interrogate using ps. Here are some of the commonly used ones. pid, comm (just the command name), cmd (the command with all its arguments as a string), uname (username running process), pmem (percent of physical memory being used), pcpu (percent of CPU being used), nice (the niceness value), etime (elapsed time running), state (one letter state code), stat (multiple character state code listing)

```
[root@localhost ~]# ps afx
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss     0:00 init [5]
    2 ?        S<     0:00 [migration/0]
 2438 tty5     Ss+    0:00 /sbin/mingetty tty5
 2439 tty6     Ss+    0:00 /sbin/mingetty tty6
 2440 ?        Ss     0:00 /usr/sbin/gdm-binary -nodaemon
 2506 ?        S      0:00  \_ /usr/sbin/gdm-binary -nodaemon
 2511 tty7     Ss+    0:21      \_ /usr/bin/Xorg :0 -br -audit 0 -auth /var/gdm/:0.Xauth -nolist
 2525 ?        Ss     0:00      \_ /usr/bin/gnome-session
 2561 ?        Ss     0:00          \_ /usr/bin/ssh-agent /bin/sh -c exec -l /bin/bash -c "/usr/
```

The "f" option shows child process as shown above

```
[root@localhost ~]# ps alx
F   UID   PID  PPID PRI  NI    VSZ   RSS WCHAN  STAT TTY        TIME COMMAND
4     0     1     0  15   0   2072   588 -      Ss   ?          0:00 init [5]
1     0     2     1 -100  -      0     0 migrat S<   ?          0:00 [migration/0]
1     0     3     1  34  19      0     0 ksofti SN   ?          0:00 [ksoftirqd/0]
```

The "l" option lists more info

```
[root@localhost ~]# ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   2072   588 ?        Ss   13:40   0:00 init [5]
root         2  0.0  0.0      0     0 ?        S<   13:40   0:00 [migration/0]
root         3  0.0  0.0      0     0 ?        SN   13:40   0:00 [ksoftirqd/0]
```

The "x" option shows info on CPU and RAM usage

PS output columns:
VSZ= virtual memory size, RSS= resident memory size, STAT= state (S=sleep, R=running, etc), F represents number of forked processes, PPID is parent PID. WCHAN is the name of the kernel function the process is sleeping in

States:
I - Idle (sleeping >20 seconds).
R - Running or runnable (on run queue)
S - Sleeping <20 seconds. (waiting for an event to complete)
T - A stopped process (either by a job control signal or because it is being traced)
D - Uninterruptible sleep (usually IO)
U - Process in uninterruptible wait.
Z - Marks a dead ("zombie") process
W - Paging (not valid since the 2.6.xx kernel)
X - Dead (should never be seen)
Additional state codes
+ - Is in the foreground process group of its control terminal.
< - Has raised CPU scheduling priority.
N - Has reduced CPU scheduling priority (see setpriority(2)).
> - Has specified a soft limit on memory requirements and is exceeding that limit; such a process is not swapped.
A - Process has asked for random page replacement
E - Process is trying to exit.
l - Is multi-threaded (using CLONE_THREAD, like NPTL pthreads do)
L - Has pages locked in memory (for example, for raw I/O).
s - Is a session leader.

Some examples

Display top 5 CPU using processes.	**ps aux --sort=-pcpu | head -5**
Display elapsed time of processes	**ps -e -o pid,comm,etime**
View process in realtime	**watch -n 1 'ps -e -o pid,pmem,pcpu --sort=-pmem | head -15'**
Output PID, command, and nice value	**ps -o pid,comm,nice -p 594**
List by % cpu with sed	**ps -e -o pcpu,cpu,nice,cputime,args –sort pcpu | sed '/^ 0.0 /d'**
List by mem (KB) usage	**ps -e -orss=,args= | sort -b -k1,1n | pr -TW$COLUMNS**
Display process hierarchy in a tree style	**ps -f --forest -C apache2**

Display child processes of a parent process
$ ps -o pid,uname,comm -C apache2

PID	USER	COMMAND
2359	root	apache2
4524	www-data	apache2
4525	www-data	apache2

The first process owned by root is the main apache2 process and all other apache2 processes have been forked out of this. We now interrogate that here:
$ ps --ppid 2359

PID	TTY	TIME	CMD
4524	?	00:00:00	apache2
4525	?	00:00:00	apache2

...

pstree - outputs a tree view of processes, which can be preferable over **ps --forest**

```
[root@localhost ~]# pstree
init─┬─acpid
     ├─atd
     ├─auditd─┬─audispd───{audispd}
     │        └─{auditd}
     ├─automount───4*[{automount}]
```

TOP

Combines **ps**, **uptime**, **free** and updates regularly; default sorts task by CPU usage- can kill(k), renice(r)
To hide / show header lines and increase display space, press 'l' for load avg (first line); pressing 't' toggles CPU states (2nd/ 3rd lines); and pressing 'm' toggles memory info (4th/ 5th lines)

The typical header:

```
top - 07:18:59 up 17:38,  2 users,  load average: 0.21, 0.15, 0.10
Tasks: 116 total,   3 running, 113 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.3%us,  0.0%sy,  0.0%ni, 99.7%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1035108k total,  1014336k used,    20772k free,   137732k buffers
Swap:  2097144k total,        0k used,  2097144k free,   646340k cached
```

- First line displays same contents as **uptime** with CPU load average provided in 1, 5, and 15 minute readings
- Numbers are "by average how many runnable processes are in the runqueue waiting to be served by the kernel's scheduler" meaning one-at-a-time per cpu core.
- What is the "optimal setting?" If you have one CPU, the optimal setting would be 1, 16 cpus, 16
- Press "1" to list CPU lines by individual CPUs instead of a total (this is the third line down that says %CPU(#)

The CPU line displays the following:

us - user space
sy - system - kernel space
ni - user nice - time spent on low priority processes
id - idle - time spent idle
wa - I/O wait cpu time - time spent in wait (on disk)
hi - hardware irq - time spent servicing/handling hardware interrupts
si - software irq - time spent servicing/handling software interrupts
st - steal time - involuntary wait by virtual CPU while a hypervisor is servicing another processor (or) %CPU time stolen from a virtual machine

The memory line is identical to running the **free** command
- About 30% memory should always be available for buffer and cache. If you don't have that, or are loading too many programs, the least active programs will release the pages they have allocated to free up memory
- Cache - structured data: filenames, etc - Buffer- unstructured data like parts of files to write to disk
- Swap is allows cache and buffers dedicated to RAM for operations

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
    1 root      15   0  2072  596  512 S  0.0  0.1   0:00.36 init
    2 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.51 ksoftirqd/0
    4 root      RT  -5     0    0    0 S  0.0  0.0   0:00.00 watchdog/0
```

By default, processes are listed by CPU usage, < and > sort by the relatively adjacent column in that direction.
Press 'M' key to sort the process list by memory usage
Press 'P' - to sort the process list by CPU usage.
Press 'N' - to sort the list by process id
Press 'T' - to sort by the running time.
Press 'R' - to sort in reverse order
Type F/f for available fields. Added fields have to be saved to use later by typing 'W' (to write config to **~/.toprc**)
Update the output on-demand by pressing the space bar.
Typing 'x' will boldface the column the processes are sorted by. Use b for a bg color instead
c - will show or hide the command's absolute path, and arguments
n - change the number of processes to display (0 for maximum)
d - change output update frequency, will prompt to enter new time in seconds
O/o - to change the order the columns are displayed (left-to-right)
h or ?- display help for interactive top commands
1 - display all CPUs/ cores separately
A - Split output into 4 individually configurable/sortable views; then use "a" to cycle between different views.
b or z - highlight running process
Type 'k' to kill and send specific signal to process (default is 15 SIGTERM), 'r' to renice; 'S' to just send signal
U/u - Filter processes of a specific user
Typing V gives a 'forest' view similar to the same usage in ps

Typing 'F' brings up this window allowing to choose a column to sort processes by

```
Current Sort Field:  K  for window 3:Mem
Select sort field via field letter, type any other key to return
  a: PID        = Process Id                 u: nFLT       = Page Fault count
  b: PPID       = Parent Process Pid         v: nDRT       = Dirty Pages count
  c: RUSER      = Real user name             w: S          = Process Status
  d: UID        = User Id                    x: COMMAND    = Command name/line
  e: USER       = User Name                  y: WCHAN      = Sleeping in Function
  f: GROUP      = Group Name                 z: Flags      = Task Flags <sched.h>
  g: TTY        = Controlling Tty
  h: PR         = Priority                 Note1:
  i: NI         = Nice value                 If a selected sort field can't be
  j: P          = Last used cpu (SMP)        shown due to screen width or your
* K: %CPU       = CPU usage                  field order, the '<' and '>' keys
  l: TIME       = CPU Time                   will be unavailable until a field
  m: TIME+      = CPU Time, hundredths       within viewable range is chosen.
  n: %MEM       = Memory usage (RES)
  o: VIRT       = Virtual Image (kb)       Note2:
  p: SWAP       = Swapped size (kb)          Field sorting uses internal values,
  q: RES        = Resident size (kb)         not those in column display.  Thus,
  r: CODE       = Code size (kb)             the TTY & WCHAN fields will violate
  s: DATA       = Data+Stack size (kb)       strict ASCII collating sequence.
  t: SHR        = Shared Mem size (kb)       (shame on you if WCHAN is chosen)
```

Different ways of starting up top:
top -u <username> - display process from specific user
top -p 1208, 1425 - to display specific PIDs
top -n 1 -b > top-output.txt --use 'batch' output to a file or pipe (text rather than binary) -n is iterations
top -i --to display only active processes (ignore idle)

Be aware that the top command comes in various variants and each has a slightly different set of options and method of usage. To check your top command version and variant use the -v option
For example, in some versions of top, o/O provides a filter prompt to search processes by various criteria
HTOP offers more ease of use but uses 5x as many system resources

When reading the virt / resident memory columns in top, remember it is showing pages- memory pages by default are about 4k so multiply what you see by 4 to get actual memory size (resident memory is the physical memory)

How Much RAM is Really Free? (disk buffers/cache)

Remember that some RAM is available for applications, and simultaneously used for disk buffer/ cached

$ free -m

	total	used	free	shared	buffers	cached
Mem:	1504	1491	13	0	91	764
-/+ buffers/cache:		635	869			
Swap:	2047	6	2041			

If you don't know how to read the numbers, you'll think the ram is 99% full when it's really just 42% (635MB). Notice also 635+869-13=1491 "used" and 91+764+13=~869 "free"

$ free -m

	total	used	free	shared	buffers	cached
Mem:	1834	1757	76	9	0	1021
-/+ buffers/cache:		735	1098			
Swap:	2047	0	2047			

Here 735MB used.
735+1098-76=1757 called "used" - (used+free in buffers/cache minus "free" in mem = reported used in mem).
Shared is usually shared libs. Buffers is when data needs to be committed to disk
Cache is big because files moved to RAM (especially frequently used files) are kept in RAM as long as possible (optimization). Cache is generally freed up when resources are needed for reallocation.
Also consider 1757 "used"-1021 cached is = 735 really used. (and 735+1098 is 1834 total reported)
Here we really have 1098 available

The -/+ buffers/cache numbers are the real deal.

The sysstat Suite and sar

The sysstat suite includes the programs **iostat** (CPU, I/O stats), **mpstat** (CPU stats), **pidstat** (stats on processes), **nfsiostat** (NFS-related stats), and **cifsiostat** (CIFS stats). The flagship of the suite is the **sar** collection and reporting suite, described below. The other tools have their own expanded descriptions afterward

sadc - System Activity Data Collector. writes binary data to /var/log/sa/ named "sa<dd>" (d is day of month)
sa1- script to run sadc with cron or systemd. Collects and stores binary data in the system activity daily data file
sar - System Activity Reporter produces reports from sadc files. Writes ascii files to /var/log/sa/ named "sar<dd>"
sa2 - script to run sar with cron or systemd. Writes a summarized daily activity report
sadf outputs sar reports in CSV, XML, JSON, etc. so you can output to other programs. Can also SVG graphs.

Installation
- If installing from source: **./configure --enable-install-cron**, then **make**, then **make install**
- The option **--enable-install-cron** makes sure to creates **/etc/rc.d/*** stuff for you
- Installs executables under /usr/local/bin
- (old issue?) Run **echo alias sar='LANG=C sar' >> /etc/bashrc** to ensure sar uses 24 hour clock

Ensuring cron is set up
If your version of sysstat didn't do it, create a file under /etc/cron.d directory that will collect the historical sar data.
vi /etc/cron.d/sysstat
*/10 * * * * root /usr/local/lib/sa/sa1 1 1
53 23 * * * root /usr/local/lib/sa/sa2 -A

- If installed from source, sa1 and sa2 is probably in **/usr/local/lib/sa**
- If installed with a package manager, they may be located in be **/usr/lib/sa/sa1** and **/usr/lib/sa/sa2**
- The above runs sa1 to collect data every 10 minutes, and compile that data about midnight with sa2
- sa1 is executed with options "1 1" - run once with 1 second interval

Log management
Generated log files are only kept for a limited time (usually ~28 days). One option is to have a script back them up before deletion- however, the configuration file /etc/sysconfig/sysstat holds a HISTORY parameter to the number of days you want to keep the log files. The maximum is supposedly 28 days, but if set to more, log files will be stored in a month-by-month directory. Then, log files will be pointing to a symlink such as YYYYMM/saDD

$ sar -u ALL 1 3 - report today's CPU usage (so far) every 1 sec, 3 times; the ALL option gives extended info
$ sar -u -f /var/log/sa/sa10 - reports for a specific day (file); here the 10th day of this month (in the file sa10)
$ sar -P 1 - the -P specifies to report only on CPU core 1 (a quad-core would have 0-3); use **-P ALL** for all cores
$ sar -r -f /var/log/sa/sa10 - most queries use the same syntax as above - this one querying memory stats with **-r**
$ sar -p -d - This query for block devices uses the -p option ("pretty print") so drives listed have familiar names to us.

General options are, **-f** to specify a file (day), **ALL** for extended info or all items (in context), **-p** for human readable, then numbers to specify frequency to run followed by how many times to run before exiting. There is also an option to put in a start time, like **sar -s 09:30:00** for 9:30AM, can be used with **-f** to specify a day. Optional end time with **-e**

The most often used queries are **-u** (CPU), **-r** (memory), **-S** (swap), **-b** (I/O stats), **-d** (block devices), **-n** (network)
Here are some others:
- **-q** - run queue size and load average
- **-w** - processes created per second, and total number of context switches per second
- **-R** - number of memory pages freed, used, and cached per second by the system.
- **-B** - paging statistics. i.e Number of KB paged in (and out) from disk per second.
- -**W** - page swap statistics. i.e Page swap in (and out) per second.

Network activity reporting has a ton of options; **sar -n <keyword>** - where <keyword> can be any of these:

DEV - network devices vital stats for eth0, eth1
EDEV - network device failure stats
NFS - NFS client activities
NFSD - NFS server activities
SOCK - sockets in use for IPv4
IP - IPv4 network traffic
EIP - IPv4 network errors
ICMP - ICMPv4 network traffic
EICMP - ICMPv4 network errors
TCP - TCPv4 network traffic
ETCP - TCPv4 network errors
UDP - UDPv4 network traffic
SOCK6, IP6, EIP6, ICMP6, UDP6 for IPv6
ALL - all options in one output- very long.

vmstat - Memory, Processes, Paging etc

vmstat -a option also displays active and inactive memory. Display units in KB and MB with **--unit k** or **m**
vmstat 2 5 - at every 2 second interval, we want to see 5 polling groups (rows of data). Just like iostat, the first line is going to be bigger because it is generic overview of the last period of total system activity

proc			memory				swap		io		system		cpu			
r	b	w	swpd	free	buff	cache	si	so	bi	bo	in	cs	us	sy	id	wa
2	0	0	120	30180	1736	46696	0	0	424	25	134	77	5	9	86	

Field definitions:
Proc - r: Running waiting for CPU; b: Uninterruptable sleeping processes (waiting for I/O); w: Swapped out processes
Memory - swpd: swap areas (in /proc/swaps); free: Idle memory; buff: buffers; cache: cache used.
Swap - si/so: Memory swapped in and out from disk - use to get more info when top shows huge swap usage
IO - bi/bo: Blocks received from/ to block device (like a hard disk)
System - in: interrupts/ IRQs per second, including the clock; cs: context switches per second. (when the CPU switches tasks among programs)
CPU - us: user time, non-kernel code including nice time; sy: kernel code; system time - network, IO interrupts, etc; id: idle time. Prior to Linux 2.5.41, including IO-wait time.

vmstat -s helps too- lots more info dumped like this:

```
[root@server1 ~]# vmstat -s
      1878184 K total memory
      1804284 K used memory
       329884 K active memory
      1273128 K inactive memory
        73900 K free memory
      1021104 K buffer memory
       151208 K swap cache
      2097148 K total swap
            0 K used swap
      2097148 K free swap
        12824 non-nice user cpu ticks
            3 nice user cpu ticks
         7702 system cpu ticks
       410614 idle cpu ticks
          405 IO-wait cpu ticks
            1 IRQ cpu ticks
         1113 softirq cpu ticks
            0 stolen cpu ticks
     28249840 pages paged in
        49102 pages paged out
            0 pages swapped in
            0 pages swapped out
       552140 interrupts
      1095772 CPU context switches
   1409692776 boot time
         5257 forks
```

Resident memory is the physical memory. swap in/ out- if you see in **top** swap is being used you can check here to see if swap is being used actively. Similarly, bi/bo is blocked in/out so you can tell if blocked processes are spending time reading more or writing. Info is grabbed from **/proc/meminfo**, **/proc/stat**, **/proc/*/stat**

Regular mode is simply "VM mode", but there are others:
Disk Mode [-d, --disk] Use [-D, --disk-sum] for summary statistics
displays these fields for both reads and writes: total (total completed successfully), merged (grouped reads or writes, resulting in one I/O), sectors (sectors read or written successfully), ms (milliseconds spent reading, or writing). Then also displays cur (for I/O currently in progress, and s (seconds spent for I/O)

Disk Partition Mode [-p, --partition device] - Detailed statistics about partition (kernel 2.5.70 or above required). Fields include: reads (reads to this partition), read sectors (read sectors for partition), writes (total number of writes issued), requested writes (number of write requests made)

Slab Mode [-m, --slabs] - Displays the contents of **/proc/slabinfo** Fields include cache (cache name), num (number of currently active objects), total (number of available objects), size (of each object) and pages (the number of pages with at least one active object)

iostat - CPU and I/O Stats for Devices and Partitions

iostat [option] [interval] [count]
Interval specifies time in seconds between each report, and count specifies the number of reports generated before exiting. **iostat 2 5 - indicates** a 2 sec interval, 5 iterations. If you don't specify iterations it will run continuous. When run for the first time, the first report contains information since the system was boot, while each subsequent report covers the time period since the last report was generated, so you may want to ignore the first one.

The first part of the normal output reports the CPU utilization, and should be self-explanatory from other tools like **top** The last two columns show %CPU time idle with an outstanding disk I/O request (lag waiting for disk activity) and without an outstanding disk I/O request. If you only want to get this top part of the report, run **iostat -c**

Linux 2.6.31-17-generic (drt-laptop) 03/24/16 _ii686_ (1 CPU)

avg-cpu:	%user	%nice	%system	%iowait	%idle
	25.99	0.78	7.43	12.77	53.03

Device:	tps	Blk_read/s	Blk_wrtn/s	Blk_read	Blk_wrtn
sda	27.40	797.19	201.27	800902	202208
sr0	0.03	1.24	0.00	1248	0

The device report has transfers per second, number of blocks per second read, number of blocks per second written, total number of blocks read, and total number of blocks written. You can use the **-k** (kilobytes) or **-m** (megabytes) parameter to change the last four columns to be expressed in human-readable terms instead of blocks
For only the device report above, run with **iostat -d**
iostat -x gives stats in an extended report.
Adding a device identifier with **/dev/sdaX** syntax will limit the report to the specified device
Using **-p** in there will show data on partitions, and **-N** gives LVM names and stats.
For NFS info, use the **iostat -n** option. **iostat -z** omits inactive devices.
Like anything, do output redirection into a file, and/or use awk to extract columns you need.
Uses **/proc/stat**, **/proc/uptime**, **/proc/partitions**, **/proc/diskstats**, **/sys**, **/proc/self/mountstats**

mpstat - Multiprocessor Stats
If you don't specify a CPU, it will spit out a global average of all of them. Specify a processor with **-P** and put ALL to get stats on all CPUs.
$ mpstat -P ALL

1:30:26	IST	CPU	%usr	%nice	%sys	%iowait	%irq	%soft	%steal	%guest	%gnice	%idle
1:30:26	IST	all	37.33	0.01	4.57	2.58	0.00	0.07	0.00	0.00	0.00	55.44
1:30:26	IST	0	37.90	0.01	4.96	2.62	0.00	0.03	0.00	0.00	0.00	54.48

$ mpstat -P ALL 2 5 - Just like with iostat - add iteration and interval count The option **-I** shows interrupt stats for each processor, and **-u** says CPU stats in case the other options are used.
*Easiest way: **$ mpstat -A** is the quick and easy equivalent of mpstat **-I ALL -u -P ALL***

pidstat - Stats on Processes

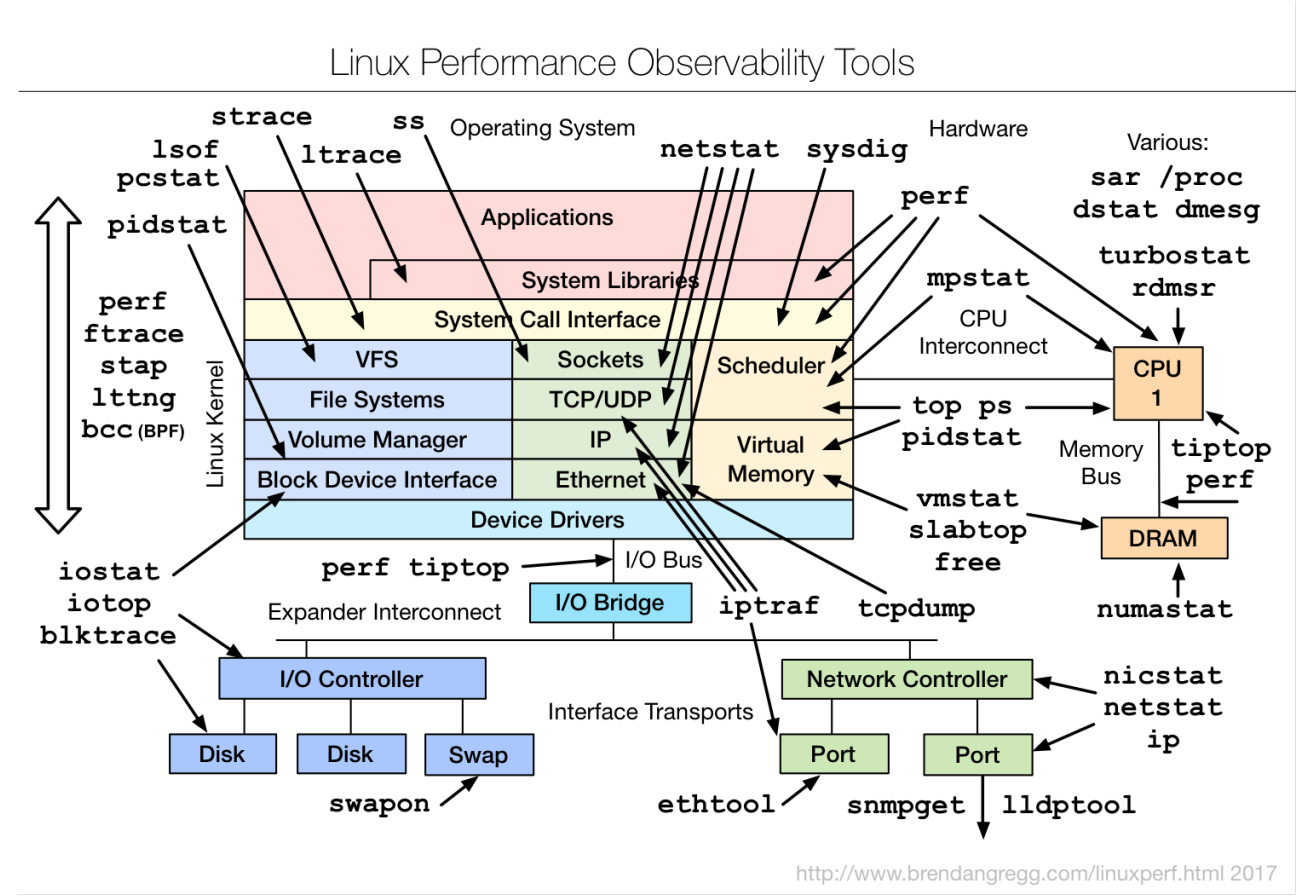
pidstat 2 5 - five reports of CPU statistics for every active task in the system at two second intervals.
pidstat -r - for a focus on page faults in memory statistics:

# Time	UID	PID	minflt/s	majflt/s	VSZ	RSS	%MEM	Command
1409816695	1000	3958	3378.22	0.00	707420	215972	5.32	cinnamon

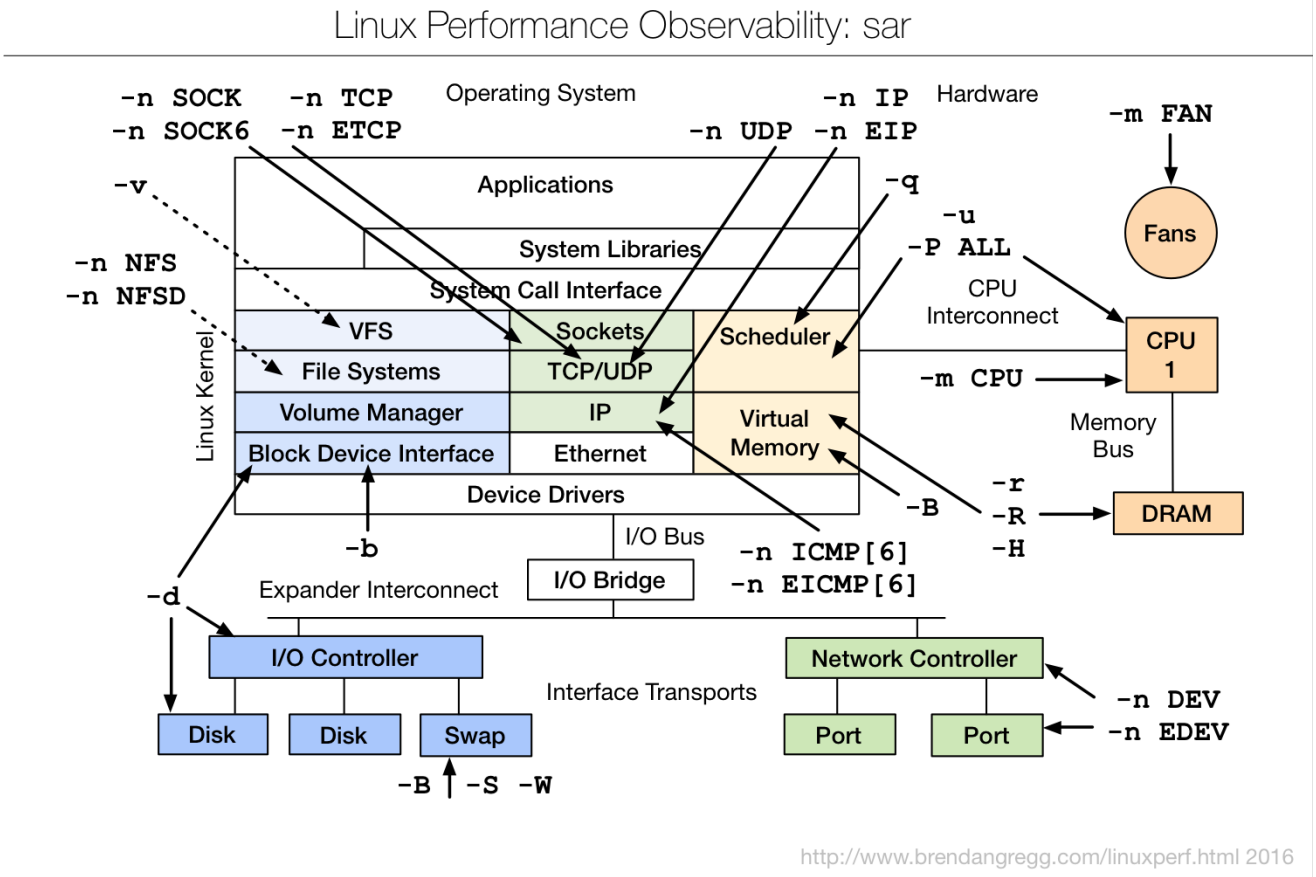
pidstat -d option for disk I/O stats:

	PID	kB_rd/s	kB_wr/s	kB_ccwr/s	Command
03:27:03 EDT	1	0.00	8.00	2.00	init

pidstat -p 1643 for PID 1643. Use **-p ALL** for all processes
Enter a string to match a process name like **-C httpd**; start an executable with **pidstat -e <command>** to monitor it.
pidstat -T CHILD -p 1643 - for the child processes of PID 1643. Only child processes with non-zero statistics values are displayed. **-T ALL** means all tasks and child processes.
pidstat with no options assumes **-p ALL**
For realtime stats use the **-R** option, or **-h** to output one horizontal line of output to easily parse with **awk**, etc.



Strongly recommended: Brendan Gregg's work: http://www.brendangregg.com/linuxperf.html



Command Equivalents - SysVinit and systemd

Action	SysVinit	systemd
Start/stop/restart/reload/status of a service	service ntpd [start | stop | etc...]	systemctl [start | stop etc...] ntpd.service
Restart a service only if already running	service ntpd condrestart	systemctl condrestart httpd.service
Enable or disable service on startup	chkconfig ntpd [off | on]	systemctl [enable | disable] ntpd.service
Is service enabled at startup (this runlevel)?	chkconfig ntpd	systemctl is-enabled ntpd.service
List services that can be started or stopped Used to list all the services and other units	ls /etc/rc.d/init.d/	systemctl OR systemctl list-unit-files --type=service OR ls /lib/systemd/system/*.service AND ls /etc/systemd/system/*.service
Print table of services listing runlevels each is configured on or off	chkconfig --list	systemctl list-unit-files --type=service ls /etc/systemd/system/*.wants/
Print a table of services that will be started when booting into graphical mode	chkconfig --list | grep 5:on	systemctl list-dependencies graphical.target
List what levels this service is config'd on/ off	chkconfig ntpd --list	ls /etc/systemd/system/*.wants/ntpd.service
Create a new service file or modify config	chkconfig ntpd --add	systemctl daemon-reload (this reloads systemd!)
Suspend the system	pm-suspend	systemctl suspend
Hibernate	pm-hibernate	systemctl hibernate
Follow the system log file	tail -f /var/log/messages (or /var/log/syslog)	journalctl -f
System halt	telinit 0, poweroff, halt	systemctl isolate poweroff.target | systemctl poweroff
Change to Single-user mode	telinit 1, s, single	systemctl isolate rescue.target (or runlevel1.target)
Change to Multi-user	telinit 2	systemctl isolate multi-user.target (or runlevel2.target*)
Change to Multi-user with Network	telinit 3	systemctl isolate multi-user.target (or runlevel3.target)
Change to RunLevel 4	telinit 4	systemctl isolate multi-user.target (or runlevel4.target*)
Change to Multi-user, w/ network, x11	telinit 5	systemctl isolate graphical.target (or runlevel5.target)
Reboot	telinit 6, reboot	systemctl isolate reboot.target | systemctl reboot
Emergency Shell	init emergency	emergency.target
Check current runlevel	runlevel	runlevel (deprecated) OR systemctl | **grep (script)**
Change default runlevel	sed s/^id:.*:initdefault:/id:3:initdefault:/	systemctl set-default multi-user.target
Set multi-user target on next boot	sed s/^id:.*:initdefault:/id:3:initdefault:/	ln -sf /lib/systemd/system/multi-user.target /etc/systemd/system/default.target
Execute a systemd cmd on remote host		systemctl dummy.service start -H user@host
Check boot time		systemd-analyze or systemd-analyze time
Kill all processes related to a service		systemctl kill dummy
Get logs for events for today		journalctl --since=today
Hostname and other host information		hostnamectl
Date and time		timedatectl

All recent versions of systemctl assume ".service" if left off. So, 'systemctl start myservicename.service' works like 'systemctl start myservicename'
Default systemd Fedora installs; 0, 1, 3, 5, and 6; have a 1:1 mapping with a specific systemd *target*.
** If you use runlevels 2 or 4 it is suggested that you make a new named systemd *target* as /etc/systemd/system/$YOURTARGET that takes one of the existing runlevels as a base (you can look at /lib/systemd/system/graphical.target as an example), make a directory /etc/systemd/system/$YOURTARGET.wants, and then symlink the additional services that you want to enable into that directory.
Runlevels 2 and 4 are by default just "multi-user" runlevel3 in systemd until defined otherwise

systemd Command Overview

Is systemd is installed on the system? Is it running?	**# systemd-run --version -- # ps -eaf | grep [s]ystemd**
List all the available units . [*.service, *.mount, *.socket, *.device]	**# systemctl list-unit-files**
List all running units. [*.service, *.mount, *.socket, *.device]	**# systemctl list-units**
List all failed units . [*.service, *.mount, *.socket, *.device]	**# systemctl --failed**
Analyze the systemd boot process.	**# systemd-analyze**
Analyze time taken by each process at boot.	**# systemd-analyze blame**
Analyze critical chain at boot (all or a specific service, etc)	**# systemd-analyze critical-chain** (OR **critical-chain httpd.service**)
	"@" = Time after unit is active or started. "+" = Time unit takes to start

Using systemd to Manage Mountpoints, Sockets, Devices Just Like Services

The general systemctl commands that work with services also do the same thing for mountpoints, sockets, and devices (which are seen as service types). Simply specify in the place of "name.service" the proper item, such as tmp.mount, cups.socket, or item.device. The list-unit-files option uses the **--type** directive such as **--type=device** or **--type=socket** accordingly. Starting stopping a mount point simply mounts and unmounts the mountpoint [**systemctl list-unit-files --type=mount** will list all mountpoints, for example]

 # systemctl list-unit-files --type=socket

 # systemctl [start | restart | stop | reload | status | is-active | enable | disable | is-enabled | mask | unmask] tmp.mount

How to enable, disable or check if turned on at boot time (auto start)	**# systemctl [is-active | enable | disable] httpd.service**
How to mask (making it impossible to start) or unmask a service	**# systemctl [mask | unmask] httpd.service**
List all services (including enabled and disabled).	**# systemctl list-unit-files --type=service**
Start, restart, stop, reload and check the status of a service	**# systemctl [start | restart | stop | reload | status] httpd.service**
Check all the configuration details of a service	**# systemctl show httpd**
Get a list of dependencies for a service	**# systemctl list-dependencies httpd.service**
How to a Kill a service using systemctl command.	**# systemctl kill httpd**
Is unit enabled or not right now ("is-active" is for the target's config)	**# systemctl is-enabled crond.service**
Get current CPU Shares of a Service (default CPUShare = 1024)	**# systemctl show -p CPUShares httpd.service**
Increase/decrease CPU share of a process	**# systemctl set-property httpd.service CPUShares=2000**
No unit specified means default.target - Requires=, RequiresOverridable=, Requisite=, RequisiteOverridable=, Wants=, BindsTo= dependencies	
List control groups hierarchically	**# systemd-cgls**
List control groups according to CPU, memory, Input and Output	**# systemd-cgtop**

To enable a service, you must be currently running the target you want the service to start in.
For example, to turn on bluetooth.service in the graphical.target, you have to change to the graphical.target first with **isolate**, then run **enable**.
systemctl isolate graphical.target ; **systemctl enable bluetooth.service**
Makes a symlink **/etc/systemd/system/graphical.target.wants/bluetooth.servic**e pointing to **/usr/lib/systemd/system/bluetooth.service**

How to start system rescue mode	**# systemctl rescue**
How to enter into emergency mode.	**# systemctl emergency**
List current default runlevel in use.	**# systemctl get-default**
Start Runlevel 5 aka graphical mode	**# systemctl isolate runlevel5.target** (OR **graphical.target**)
Set multiuser mode (runlevel 3) as default	**# systemctl set-default runlevel3.target** (OR **multiuser.target**)
*[This set-default line creates a symlink **/etc/systemd/system/default.target** pointing to **/usr/lib/systemd/system/multiuser.target**]*	
Reboot, halt, suspend, hibernate or put system in hybrid-sleep	**# systemctl [reboot | halt | suspend | hibernate | hybrid-sleep]**

Unit and target files in **/usr/lib/systemd/system/** are pointed to by symlinks placed in **/etc/systemd/system/**
Unit files enabled for a specific target will have a symlink in that target's "wants" directory, such as **/etc/systemd/system/multi-user.target.wants**

SysVinit - Directory Structures

Generally, you will find in the **/etc** directory some symlinks to stuff that is actually in the **/etc/rc.d/** directory. This can cause some confusion, since we have some other symlink stuff for backward compatibility for systems that once supported Upstart but no longer do so. This writing ignores all of that and sticks to CentOS 5.5

/etc/init.d is a symlink to the directory **/etc/rc.d/init.d** and the same with **/etc/rc#.d** linking to **/etc/rc.d/rc#.d**, also the same with scripts **rc**, **rc.local** and **rc.sysinit**, who's actual locations is also in the /etc/rc.d/ directory as well Even though you will often see these in **/etc.** Here is where they actually live:

/etc/rc.d/init.d/
/etc/rc.d/rc0.d/
/etc/rc.d/rc1.d/

...and so on....

/etc/rc.d/rc5.d/
/etc/rc.d/rc6.d/
/etc/rc.d/rc
/etc/rc.d/rc.local
/etc/rc.d/rc.sysinit

Service's scripts are in /etc/rc.d/init.d/ (often accessible via the symlink **/etc/init.d**/)

- Each service managed by SystemVinit needs a script in /etc/rc.d/init.d/
- Common elements to the scripts in **/etc/rc.d/init.d/<*servicename*>** are the top several lines, beginning in a prelude declaring the script processor (as in **#!/bin/bash**); followed by a line with name and brief description; another with chkconfig default runlevels the service should be started, and the start and stop priority levels.
- If default is to not be started in any runlevels, a "-" should be used in place of the runlevels list.
- Another entry contains service description (used by ntsysv)
- Finally the general functions container for init.d scripts is defined(usually **/etc/init.d/functions**), followed by lines setting and ENVVARS and functions for the service, (much like in bashrc does for it's purpose).

For example, the beginning of **/etc/rc.d/init.d/kudzu** has these line common to SysVinit scripts:

```
#!/bin/bash
# kudzu          This scripts runs the kudzu hardware probe.
# chkconfig: 345 05 95
# description:   This runs the hardware probe, and optionally configures \
#                changed hardware.
# Source function library.
. /etc/init.d/functions
```

Says that the script should be started in levels 3, 4, and 5, start priority 5, stop priority 95

/etc/rc.d/init.d directory contents:

/etc/rc.d/init.d/acpid
/etc/rc.d/init.d/anacron

...

/etc/rc.d/init.d/ypbind
/etc/rc.d/init.d/yum-updatesd

Files in the**/etc/rc.d/rc#.d** directories are symlinks to the actual scripts for all of SysVinit's managed programs in **/etc/rc.d/init.d** For example, **/etc/rc.d/rc0.d/K99cpuspeed** links to **/etc/rc.d/init.d/cpuspeed**
With those links, the naming convention of K or S means "kill" or "start" and the number (like 99) indicates the numerical order that it is executed in that runlevel's directory, when that runlevel starts. This way, it is directed that things are stopped and started in the proper order.

As an example, here is a sample of some filenames in **/etc/rc.d/rc3.d/**

K88wpa_supplicant	S02lvm2-monitor
K89netplugd	S04readahead_early
K89rdisc	S05kudzu
K91capi	S08ip6tables
K99readahead_later	S08iptables
S00microcode_ctl	S08mcstrans

rc.local is to execute commands during the startup without needing symlinks. "Local system initialization script"
S99local -> softlink for /etc/rc.local in 2,3,4 and 5 runlevels
You can optionally have a similar shutdown items script in **/etc/rc.d/rc.local_shutdown**
rc.sysinit seems to be redhat specific and is executed very early in the process while rc.local is executed later.
rc is typically not used by linux distributions but is used in BSD

The rc stands for "run commands"; runcom (as in .cshrc or /etc/rc) comes from the runcom facility from the MIT CTSS system, ca. 1965. From Kernighan and Ritchie, as told to Vicki Brown: "There was a facility that would execute a bunch of commands stored in a file; it was called runcom for "run commands", and the file began to be called "a runcom". rc in Unix is a fossil from that usage."
The idea of having the command processing shell be an ordinary slave program came from the Multics design, and a predecessor program on CTSS by Louis Pouzin called RUNCOM. The first time I remember the name "shell" for this function was in a Multics design document by Doug Eastwood (of BTL). Commands that return a value into the command line were called "evaluated commands" in the original Multics shell, which used square brackets where Unix uses backticks.

/etc/inittab Main config file for SysVinit

- Specifies runlevels, scripts to run when certain runlevels are selected, and items to respawn (getty).
- Syntax for an entry in inittab: id:runlevels:action:process.
- First is a unique arbitrary identifier, second indicates what runlevels invoke the command, third is how to handle this entry (like execute command once or respawn whenever it exits, fourth is the command and it's arguments
- x:5:respawn:/etc/X11/prefdm -nodaemon Runlevel 5, specify default login screen for X11
- 3:2345:respawn:/sbin/mingetty tty3 Virtual terminal 3, available for runlevels 2 through 5

Sample /etc/initab (truncated):

```
# Set the default runlevel to three - points to line below "l3:3:wait:/etc/rc.d/rc 3"
id:3:initdefault:

# Execute /etc/rc.d/rc.sysinit when the system boots
# starts network, establishes mounted systems, starts SELinux, encryption
si:S:sysinit:/etc/rc.d/rc.sysinit

# Run /etc/rc.d/rc with the runlevel as an argument - e.g., a 5 points it to /etc/rc5.d/
# Runlevels are designated in /etc/rc.d/
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
...
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Executed when we press ctrl-alt-delete
ca::ctrlaltdel:/sbin/shutdown -t3 -rf now

# Start agetty for virtual consoles 1 through 6
c1:12345:respawn:/sbin/agetty 38400 tty1
c2:12345:respawn:/sbin/agetty 38400 tty2
...
c6:45:respawn:/sbin/agetty 38400 tty6
```

Default runlevel is determined, then scripts in appropriate **/etc/rc.d/rcX/** directory are run.
When SysVinit is instructed to change runlevels, it reads initab for what **/etc/rc.d** directory belongs to that runlevel.
The **rc.d** directory contains the daemon scripts which run at boot and when switching runlevels.
Contents in **/etc/rc.d/rcX/** directories are just symlinks to the files in **/etc/rc.d/init.d/** and are named to either start with an S (start) or a K (kill), in order of the number to be processed
For example: take a symlink **S45dhcpd** in in **/etc/rc.d/rc3/** - This means the **/etc/rc.d/init.d/dhcpd** script will be 45th in order to start for that runlevel directory containing this symlink- in this case runlevel 3.

Some types of Linux using SysVinit don't even use this system of symlinks. Slackware uses something similar to BSD where all directives for a runlevel are only put in a runlevel script.

Runlevel Service Management Tools -SysVinit initscript utilities

service <*servicename*> [**start** | **stop** | **restart** | **status** | **list**] Activate, etc., a daemon in current runlevel
OR Go to **/etc/rc.d/init.d/** directory and type ***./<servicename>* start**.
init OR **telinit [0-6]** switches to the specified runlevel
runlevel tells you your runlevel- returns two numbers - 3 5 means that current runlevel is 5 and previous was 3.
chkconfig --list gives you this type of output:

```
[root@localhost rc.d]# chkconfig --list
NetworkManager  0:off   1:off   2:off   3:off   4:off   5:off   6:off
acpid           0:off   1:off   2:on    3:on    4:on    5:on    6:off
anacron         0:off   1:off   2:on    3:on    4:on    5:on    6:off
apmd            0:off   1:off   2:on    3:on    4:on    5:on    6:off
atd             0:off   1:off   2:off   3:on    4:on    5:on    6:off
auditd          0:off   1:off   2:on    3:on    4:on    5:on    6:off
autofs          0:off   1:off   2:off   3:on    4:on    5:on    6:off
avahi-daemon    0:off   1:off   2:off   3:on    4:on    5:on    6:off
```

chkconfig --list <*servicename*> will output the same on one line for that one service
chkconfig --list | grep <*servicename*> might be more helpful to list multiple matches (e.g. "avahi-" services)
chkconfig --level 35 <*servicename*> off | **on** | **reset** | **resetpriorities** - affects service in runlevels 3 and 4
chkconfig --level <*servicename*> on -turns service on in levels 2-5, if 'off' affects 0-6
chkconfig --add OR **--del** adds or removes scripts from **/etc/rc.d/init.d/**
chkconfig --override <*servicename*> - files in **/etc/chkconfig.d/<*servicename*>** can override init service scripts

ntsysv is just a Red Hat TUI interface to turn off and on services in the currently active runlevel. Debian has **rcconf**
ntsysv --runlevel 35 (OR **--level 35**) manages services on levels 3 and 5
redhat-config-services or **system-config-services** - graphical Services Configuration Tool

/sbin/telinit is linked to /sbin/init - takes a one-character argument (0-6 for switching runlevels, s/S/1 all work for single user mode, U/u to resart current runlevel init scrips without checking inittab; Q/q to do so forcing checking inittab. The init binary checks if it is init or telinit by looking at its process id; the real init's process id is always 1.

- ls /etc/init.d/ will also list all of the currently available service files
- Scripts to stop and start processes can be used as an alternative to running **kill**.
- neither ntsysv or chkconfig starts or stops services- only dictates runlevel. The service command does that.
- xinetd services are immediately affected by ntsysv, unlike others

SysVinit Runlevel	***Systemd Target***	***Description***
0	*poweroff.target*	*Halts the system*
1	*rescue.target*	*Single-user mode* (everything mounted, minimal services)
2	*multi-user.target*	*Multiuser mode without networking*
3	*multi-user.target*	*Multiuser mode with networking*
4	*multi-user.target*	*User configurable*
5	*graphical.target*	*Used for the GUI (X11 multiuser mode)*
6	*reboot.target*	*Reboots the system*

systemd's emergency.target
- Is like init=/bin/sh on the kernel command line
- Has no corresponding sysvinit runlevel- would just boot your machine to a shell with really nothing started.
- You get a shell, but almost nothing else (except for systemd in the background)
- No services are started, no mount points mounted, no sockets established.
- Useful for running specific scripts which could then be started independently.
- Allows booting bit-by-bit, starting the various services and other units step-by-step manually.

In SysVinit, services can define arbitrary commands. Examples would be service iptables panic, or service httpd graceful. Native systemd services do not have this ability. Any service that defines an additional command in this way would need to define some other, service-specific, way to accomplish this task when writing a native systemd service definition.Check the package-specific release notes for any services that may have done this.

Example systemd Unit Script Contents

It is *not* advised to edit unit scripts in **/usr/lib/systemd/system/** so they remain default/as-installed by packages. For custom changes, make copy to edit in **/etc/systemd/system** - this directory overrides those defaults for you.

Example Service Unit Script - /usr/lib/systemd/system/httpd.service contents:

```
[Unit]
Description=The Apache HTTP Server
After= network.target remote-fs.target nss-lookup.target

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
ExecStop=/bin/kill -WINCH ${MAINPID}
KillSignal=SIGCONT
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

Example Target Unit Script - /usr/lib/systemd/system/multi-user.target contents:

```
[Unit]
Description= Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
[Install]
Alias=default.target
```

"After" is what is loaded after this finishes activating
"Requires" should be what needs to be loaded before

Example Custom Mount Scripts - /etc/systemd/system/lvdisk.mount and lvdisk.automount

Needs a mount file and automount file with a matching name (mydisk5.automount for mydisk5.mount)
This is the way future versions of RHEL will likely do automount instead of the old /etc/fstab method

```
# vim /etc/systemd/system/lvdisk.mount
[Unit]
Description= Example test mount
[Mount]
what  = /dev/vgdisk/lvdisk
where = /lvdisk
type = xfs
[Install]
WantedBy=multi-user.target
```

```
# vim /etc/systemd/system/lvdisk.automount
[Unit]
Description= Example test automount
[Automount]
where = /lvdisk
[Install]
WantedBy=multi-user.target
```

Test it out - systemctl enable lvdisk.automount; systemctl start lvdisk.automount; mount | grep lvdisk

Contents of the systemd Package

Installed programs: bootctl, busctl, coredumpctl, halt, hostnamectl, init, journalctl, kernel-install, localectl, loginctl, machinectl, networkctl, poweroff, reboot, runlevel, shutdown, systemctl, systemd-analyze, systemd-ask-password, systemd-cat, systemd-cgls, systemd-cgtop, systemd-delta, systemd-detect-virt, systemd-escape, systemd-hwdb, systemd-inhibit, systemd-machine-id-setup, systemd-mount, systemd-notify, systemd-nspawn, systemd-path, systemd-resolve, systemd-run, systemd-socket-activate, systemd-stdio-bridge, systemd-tmpfiles, systemd-tty-ask-password-agent, telinit, timedatectl, and udevadm

Installed libraries: libnss_myhostname.so.2, libnss_mymachines.so.2, libnss_resolve.so.2, libnss_systemd.so.2, libsystemd.so, libsystemd-shared-231.so, and libudev.so

Installed directories: /etc/binfmt.d, /etc/init.d, /etc/kernel, /etc/modules-load.d, /etc/sysctl.d, /etc/systemd, /etc/tmpfiles.d, /etc/udev, /etc/xdg/systemd, /lib/systemd, /lib/udev, /usr/include/systemd, /usr/lib/binfmt.d, /usr/lib/kernel, /usr/lib/modules-load.d, /usr/lib/sysctl.d, /usr/lib/systemd, /usr/lib/tmpfiles.d, /usr/share/doc/systemd-234, /usr/share/factory, /usr/share/systemd, /var/lib/systemd, and /var/log/journal

bootctl	Query the firmware and boot manager settings
busctl	Review logs and monitor the D-Bus bus
coredumpctl	Retrieve coredumps from the systemd Journal
halt	Normally invokes **shutdown** with the *-h* option, except when already in run-level 0, then it tells the kernel to halt the system; it notes in the file /var/log/wtmp that the system is being brought down
hostnamectl	Query and change the system hostname and related settings
init	The first process to be started when the kernel has initialized the hardware which takes over the boot process and starts all the proceses it is instructed to
journalctl	Query the contents of the systemd Journal
kernel-install	Add and remove kernel and initramfs images to and from /boot
localectl	Query and change the system locale and keyboard layout settings
loginctl	Review logs and control the state of the systemd Login Manager
machinectl	Review logs and control the state of the systemd Virtual Machine and Container Registration Manager
networkctl	Review logs and state of the network links as seen by systemd-networkd
poweroff	Tells the kernel to halt the system and switch off the computer (see **halt**)
reboot	Tells the kernel to reboot the system (see **halt**)
runlevel	Reports the previous and the current run-level, as noted in the last run-level record in /var/run/utmp
shutdown	Brings the system down in a secure way, signaling all processes and notifying all logged-in users
systemctl	Review logs and control the state of the systemd system and service manager
systemd-analyze	Determine system boot-up performance of the current boot
systemd-ask-password	Query a system password or passphrase from the user, using a question message specified on the command line
systemd-cat	Connect STDOUT and STDERR of a process with the Journal
systemd-cgls	Recursively shows the contents of the selected Linux control group hierarchy in a tree
systemd-cgtop	Shows the top control groups of the local Linux control group hierarchy, ordered by their CPU, memory and disk I/O load
systemd-delta	Identify and compare configuration files in /etc that override default counterparts in /usr
systemd-detect-virt	Detects execution in a virtualized environment
systemd-escape	Escape strings for inclusion in systemd unit names
systemd-hwdb	Manage hardware database (hwdb)
systemd-inhibit	Execute a program with a shutdown, sleep or idle inhibitor lock taken
systemd-machine-id-setup	Used by system installer tools to initialize the machine ID stored in /etc/machine-id at install time with a randomly generated ID
systemd-mount	A tool to temporarily mount or auto-mount a drive.
systemd-notify	Used by daemon scripts to notify the init system about status changes

systemd-nspawn	Run a command or OS in a light-weight namespace container
systemd-path	Query system and user paths
systemd-resolve	Resolve domain names, IPV4 and IPv6 addresses, DNS resource records, and services
systemd-run	Create and start a transient .service or a .scope unit and run the specified command in it
systemd-socket-activate	A tool to listen on socket devices and launch a process upon connection.
systemd-tmpfiles	Creates, deletes and cleans up volatile and temporary files and directories, based on the configuration file format and location specified in tmpfiles.d directories
systemd-tty-ask-password-agent	Used to list or process pending systemd password requests
telinit	Tells **init** which run-level to change to
timedatectl	Query and change the system clock and its settings
udevadm	Generic Udev administration tool: controls the udevd daemon, provides info from the Udev database, monitors uevents, waits for uevents to finish, tests Udev configuration, and triggers uevents for a given device
libsystemd	systemd utility library
libudev	A library to access Udev device information

-- from http://www.linuxfromscratch.org/lfs/view/systemd/chapter06/systemd.html

You may have noticed that installed items listed contains telinit, and an /etc/init.d directory. According to the Fedora Wiki, the 'service' and 'chkconfig' commands will (surprisingly) mostly continue to work as expected in the systemd world. Presumably, this would be for backward compatibility support for old scripts, etc.

Target unit directories hold symlinks to the real unit files like this:	**/etc/systemd/system/XXXXXX.target.wants/bluetooth.service**
Those symlinks point to the actual service (etc) unit files that reside here:	**/usr/lib/systemd/system/bluetooth.service**
The **default.target** file here is a target/runlevel symlink:	**/etc/systemd/system/default.target**
And the default.target symlink points to the actual target here:	**/usr/lib/systemd/system/XXXXXX.target**

Running **systemctl isolate graphical.target** will not affect the default.target symlink, and merely switches the current runlevel (use **set-default**).

Running **systemctl disable myservice** basically does the same as **rm '/etc/systemd/system/multi-user.target.wants/service.myservice'**
Running **systemctl enable myservice** basically does the same as
ln -s '/usr/lib/systemd/system/myservice.service' '/etc/systemd/system/multi-user.target.wants/service.myservice'

The "**target.wants**" directories in **/usr/lib/systemd/system/** hold symlinks to the corresponding runlevel's unit files just like init's **/etc/rc.d/rc#.d/**
A target is itself a unit file, manages other unit files. Defaults are multi-user.target, graphical.target, rescue.target, emergency.target, poweroff.target, and reboot.target

The ifconfig Command

This command is deprecated in Linux, but it is important to keep sharp with when working with Solaris, AIX, HP-UX, BSD, and SCO UNIX.

- With no arguments will display details all the active interfaces, use **-a** to also list inactive ones
- You can specify the interface to view, like **ifconfig eth0**
- Bring interfaces up or down with **ifconfig eth0 up** (or **down**) or **ifup eth0** - (or **ifdown**)
- (note- routes to interfaces disabled with ifconfig and ifdown are not automatically disabled)

Changes with ifconfig are NOT persistent after reboot
In BSD you may have to edit /etc/rc.conf appropriately
In Linux edit appropriate interface file in /etc/sysconfig/network-scripts/

When you specify an interface, you can add these to change it's properties:
- Set an ip address with the syntax **ifconfig eth0 172.16.25.125**
- Just like setting the address you can add **netmask 255.255.255.224** and/or **broadcast 172.16.25.63**
- Set an mtu with **mtu 1000**
- Change a MAC address with the "hw ether" argument - **ifconfig eth0 hw ether 00:BB:22:DD:44:FF**
- The states broadcast, multicast, and allmulti can all be turned off and on like promisc and -promisc
- When considering packet capture, enter promiscuous mode with **promisc** and disable with **-promisc**
- Similarly you can disable ARP with **-arp** and re-enable it with **arp**
- Adding an alias for an interface is possible, but it's ip address has rules on masks (see ifconfig addendum)
  ifconfig eth0:0 172.16.25.127 - to disable the alias simply bring it down **ifconfig eth0:0 down**
  note: some versions of ifconfig require the use of the keyword **alias**

delete - Removes the specified *network address*. This is used when an alias is incorrectly specified or when it is no longer needed. Incorrectly setting an ns address has the side effect of specifying the host portion of the network address. Removing all ns addresses allows you to re-specify the host portion.

detach - Removes an *interface* from the network interface list. If the last interface is detached, the network interface driver code is unloaded. In order for the interface route of an attached interface to be changed, that interface must be detached and added again with ifconfig.

Other options:

group ID and **-group ID**
Adds/removes a group ID to the group ID list for the interface (used in determining the route to use when forwarding packets that arrived on the interface.

metric <*value*> - Sets the routing metric of the interface to the value specified by the Number variable. The default is 0 (zero). The routing metric is used by the routing protocol (the routed daemon). Higher metrics have the effect of making a route less favorable. Metrics are counted as addition hops to the destination network or host.

monitor and **-monitor**
Enables/disables the underlying adapter to notify the interface layer of link status changes. The adapter must support link status callback notification. If multipath routing is used, alternate routes are selected when a link goes down.

checksum_offload and **-checksum_offload**
Enables/disables the flag to indicate that transmit TCP checksum should be offloaded to the adapter. Also resets the per-interface counter that determines whether TCP should dynamically enable or disable offloading of checksum computation.

The ifconfig command in Linux is part of the net-tools package, which has been deprecated (and not updated since April 2001. Other commands worked with ifconfig, like ifdown, ifup, iwconfig, arp, route, iptunnel, ipmaddr, tunctl, brctl, etc. and even netstat were generally replaced with ip, ss, and more in the iproute2 package, which here has a separate section devoted to it.

netstat

- gets values from /proc/net*
- often generically used as **netstat -netulp** or **-tulpen** (same thing, different mnemonic)
- netstat's successor **ss** (for "socket stats") uses almost identical command options (less to remember)

-a list all ports
-t for tcp
-u for udp
-l listening
-p includes a PID/Program name field
-s statistics for each protocol (separate listings for each protocol udp/tcp/icmp/etc in one listing)
-n numeric only - says don't use hostnames, port names, usernames. **-N** does the opposite (-r in the ss command)
To apply the **-n** option more specific, instead use **--numeric-ports**, **--numeric-hosts**, or **--numeric-users**
-e for extended on certain other subcommands
-c for continuous output, sort of like using watch command
-r displays the routing table
-i displays a list of all network interfaces (kernel interface table)
-ie displays interfaces with extended option, which looks like ifconfig output
--verbose includes info like " netstat: no support for `AF IPX' on this system"
-g will display the multicast group information
-M or **--masquerade** to show NAT info
-x lists legacy UNIX process sockets that are listed as their own protocol
-w to list packets of the type raw

As usual pipe out to other tools. Use grep to search for specific items i.e., **netstat -ap | grep ssh** or use **grep "ESTAB"** - Continuous list of active tcp connections: **watch -d -n0 "netstat -atnp | grep ESTA"**
netstat -l | grep 1000 | wc -l - to see if there is traffic on a certain port

Some options in netstat vary slightly among different versions of UNIX- check your man page

Migrating from the traditional net-tools package to iproute2 package

The net-tools package, which included ifconfig and many other tools, was slated for deprecation over a decade ago, and finally started getting dropped around 2009. This applies to most distributions of Linux- if you are using BSD, Solaris, or another version of UNIX, you will want to stick to traditional net-tools package and ifconfig. On Linux, you can still install it, but there are things it won't work well with modern Linux networking

Summary of changes to be aware of that I'll try to cover here:
- *The net-tools package replacements provided by iproute2*
- *Network Monitor (nmcli) to manage persistent configuration changes*
- *The abandoning of udev naming schemes for devices; a return to hardware-specific names*

Deprecated	*Replacement command(s)*
arp	ip n (ip neighbor)
ifconfig	ip a (ip addr), ip link, ip -s (ip -stats)
iptunnel	ip tunnel
iwconfig	iw
nameif	ip link, ifrename
netstat	ss, ip route (for netstat-r), ip -s link (for netstat -i), ip maddr (for netstat-g)
route	ip r (ip route)

For an exhaustive comparison of these, please see Doug Vitale's blog entry at https://dougvitale.wordpress.com/2011/12/21/deprecated-linux-networking-commands-and-their-replacements/

The ip command

ip address (or addr or a) [add | del | set] dev [interface]
    Address, IPv4 or v6, on the interface
ip link (or l) [set | show] dev [interface]
    Link generally refers to interfaces/ devices
ip route (or r) [add | chg | repl | del | show] cache
    Route mostly replaces route commands
ip neighbor (or n) [add | chg | repl | del | show] dev [interface]
    Neighbor mostly replaces arp commands and also shows IPv6 NDP info

The ip command has a HUGE list of other "objects" like link, route, neighbor, and address that can also be queried or configured. Explained more later, here is a list: addrlabel, rule, ntable, tunnel, tuntap, l2tp,maddr, mroute, mrule, monitor, xfrm, netns, tcp_metrics. The link object also supports a huge list of interface types with their own help pages.

Using the ip command applies settings but will not save the configuration - it's not persistent. In order to make persistent changes, either use the Network Manager package or edit the network scripts directly

Device naming:
Traditionally, we have seen devices use udev naming like eth0 or usb0. udev provides persistent naming for some device types out of the box to make things more human-readable (like hard drives- /dev/sdb1, /dev/hda2). It has been used so long some people don't know or forgot it was an add-on.

BIOS naming based on HW properties (physical naming) has returned, and here is what you will see more of:
- em[1-N] for embedded NICs
- p[slot-number]p[port-number] - p6p1 = pci slot 6 port 1

You might also see logical naming with VLAN and alias naming. You may even see a udev name being used.

If you prefer not to, you can add to your boot options in GRUB:
Add "net iframes=0 biosdevnames=0" to boot options
Then write to grub config on disk after boot : grub2-mkconfig -o /boot/grub2/grub.cfg

Examples of tasks - iproute2 and net-tools package equivalents

Show All Connected Network Interfaces
With net-tools: $ ifconfig -a
With iproute2: $ ip link show
See also: "ip addr" for "ifconfig" and "ip -s link" for "netstat -i"

Show IPv4 Address(es) of a Network Interface
With net-tools: $ ifconfig eth1
With iproute2: $ ip addr show dev eth1

Show IPv6 address(es) of a Network Interface
With net-tools: $ ifconfig eth1
With iproute2: $ ip -6 addr show dev eth1

View the IP Routing Table
With net-tools: $ route -n --or-- $ netstat -rn
With iproute2: $ ip route show

View Socket Statistics
With net-tools: $ netstat --AND-- $ netstat -l
With iproute2: $ ss --AND-- $ ss -l

View the ARP Table
With net-tools: $ arp -an
With iproute2: $ ip neigh

Activate or Deactivate a Network Interface
With net-tools: $ ifconfig eth1 [up | down]
With iproute2: $ ip link set [up | down] eth1

Assign IPv4 address(es) to a Network Interface
With net-tools: $ ifconfig eth1 10.0.0.1/24
With iproute2: $ ip addr add 10.0.0.1/24 dev eth1

Remove an IPv4 address from a Network Interface
In net-tools you end up assigning 0 to the interface. iproute2 can properly remove it.
With net-tools: $ ifconfig eth1 0
With iproute2: $ ip addr del 10.0.0.1/24 dev eth1

Assign or Remove an IPv6 address on a Network Interface
With net-tools: $ ifconfig eth1 inet6 [add | del] 2002:0db5:0:f102::1/64
With iproute2: $ ip -6 addr [add | del] 2002:0db5:0:f102::1/64 dev eth1

Assign Multiple IP Addresses to an Interface
With net-tools (ip subinterface aliases workaround):
$ ifconfig eth0:1 192.168.10.10 netmask 255.255.255.0 up
$ ifconfig eth0:2 192.168.10.15 netmask 255.255.255.0 up
With iproute2:
$ ip addr add 10.0.0.1/24 dev eth1
$ ip addr add 10.0.0.2/24 dev eth1

Change the MAC Address of a Network Interface
Before changing the MAC address, you need to deactivate the interface first.
With net-tools: $ ifconfig eth0 [down | up]; ifconfig eth1 hw ether 08:00:27:75:2a:66
With iproute2: $ ip link set dev eth0 [down | up]
$ ip link set dev eth1 address 08:00:27:75:2a:67

Add or Modify a Default Route
With net-tools: $ route [add | del] default gw 192.168.1.2 eth0
With iproute2: $ ip route [add | replace] default via 192.168.1.2 dev eth0 *(replace is a command)*

Add or Remove a Static Route
With net-tools: $ route add -net 172.16.32.0/24 gw 192.168.1.1 dev eth0
$ route del -net 172.16.32.0/24
With iproute2: $ ip route add 172.16.32.0/24 via 192.168.1.1 dev eth0
$ ip route del 172.16.32.0/24

Add or Remove a Static ARP Entry
With net-tools: $ arp -s 192.168.1.100 00:0c:29:c0:5a:ef
$ arp -d 192.168.1.100
With iproute2: $ ip neigh add 192.168.1.100 --OR-- ip addr 00:0c:29:c0:5a:ef dev eth0
$ ip neigh del 192.168.1.100 dev eth0

Add, Remove or View Multicast Addresses
With net-tools: $ ipmaddr [add | del] 33:44:00:00:00:01 dev eth0
$ ipmaddr show dev eth0 --OR-- $ netstat -g
With iproute2: $ ip maddr [add | del] 33:44:00:00:00:01 dev eth0
$ ip maddr list dev eth0

NetworkManager Service - Persistent Changes with nmcli

In order to make persistent changes, you should either use Network Manager, or manually edit the files it uses When running NM, manually editing those files is not recommended to do unless you have to, such as in a script or something, but saying that isn't a way of babysitting us- Network Manager often clobbers what we put in manually with it's own info, so telling it through it's own mechanisms can avoid that.
[Unsurprisingly, Network Manager disgusts a lot of sysadmins for being so resistant to manual edits]

Network Manager has a GUI, and a text interface quite similar to it. It can actually be effective for doing a good range of tasks, but (as usual) the command line is much more flexible and granular. Usually, we get things done by entering individual commands, but it also has a command prompt of it's own for advanced operations.

nmcli [OPTIONS] OBJECT { COMMAND | help }
When using nmcli, the most important component in the command definition above is "object."
Connections and devices are the most often used object components.
- a device is a network interface
- a connection is a collection of configurations (e.g., home, work configs with different settings for everything)
- so, you can have multiple connections for a device but only one can be active at one time

The general options pertain to output styles, facilitating use by external scripts, etc. These include -t [erse] or -p [retty] for ease of viewing, -m [mode] tabular | multiline, -f [ields] <field1, field2, ...> if you only want to output some columns, etc.

The "device" object is how you refer to specific devices, like your wireless or ethernet interfaces. They are what you are going to add to your various "connection" objects
Common device commands: status, show, connect, set, reapply, disconnect, delete, monitor, wifi, and lldp.
Generally those commands would be followed by the interface name. The set command allows setting autoconnect and/or managed to on or off.
Wifi devices have more specific direcives as illustrated in this excerpt:

```
  wifi [list [ifname <ifname>] [bssid <BSSID>]]
  wifi connect <(B)SSID> [password <password>] [wep-key-type key|phrase] [ifname <ifname>]
                         [bssid <BSSID>] [name <name>] [private yes|no] [hidden yes|no]
  wifi hotspot [ifname <ifname>] [con-name <name>] [ssid <SSID>] [band a|bg] [channel <#>] [password
<password>]
  wifi rescan [ifname <ifname>] [[ssid <SSID to scan>] ...]
```

The nmcli "connection" object has a variety of common commands: show, up, down, add, modify, clone, edit, delete, monitor, reload, load, import and export

The "general" object has 4 commands: status, hostname, permissions, and logging
Status comes up whenever you call nmcli general by itself:

```
[user@localhost ~]$ nmcli g
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

Typing "nmcli general hostname" outputs your hostname, and if you give put one at the end it sets it to that.
Typing "nmcli general logging" outputs or changes the logging level and domains the same way..
Typing "nmcli general permissions" outputs "caller permissions for authenticated operations," as seen below.

```
[user@localhost ~]$ nmcli g permissions
PERMISSION                                        VALUE
org.freedesktop.NetworkManager.enable-disable-network      yes
org.freedesktop.NetworkManager.enable-disable-wifi         yes
org.freedesktop.NetworkManager.enable-disable-wwan         yes
org.freedesktop.NetworkManager.enable-disable-wimax        yes
org.freedesktop.NetworkManager.sleep-wake                  no
org.freedesktop.NetworkManager.network-control             yes
org.freedesktop.NetworkManager.wifi.share.protected        yes
org.freedesktop.NetworkManager.wifi.share.open             yes
org.freedesktop.NetworkManager.settings.modify.system      yes
org.freedesktop.NetworkManager.settings.modify.own         yes
org.freedesktop.NetworkManager.settings.modify.hostname    auth
org.freedesktop.NetworkManager.settings.modify.global-dns  unknown
org.freedesktop.NetworkManager.reload                      unknown
```

The "networking" object is so succinct it is almost disappointing. It merely lets you turn networking on or off with "nmcli networking [on | off]," and lets you query "nmcli net connectivity" and it reports "full" if it's working ok.

The "radio" object also doesn't do a lot. Like the networking object, most of the controls are over in the "device" object. Just by itself, it will give the output below, with WWAN referring to mobile network service and interface. You can turn things off with "nmcli radio [wlan | wan | all] [on | off]"

```
[user@localhost ~]$ nmcli radio
WIFI-HW  WIFI     WWAN-HW  WWAN
enabled  enabled  enabled  disabled
```

The "agent" object allows you to use policy management like polkit to govern permissions on things like turning the network on or off. Mechanisms like polkit are outside of the scope of this document, but that's what the "agents" object enables.

Finally, typing "nmcli monitor" simply turns on (or off) a facility that prints a line to stdout when something in Network Manager changes.

General Use and Examples

So, Network manager and it's NMCLI is when you need persistent configuration solutions, even if you are just testing things out. The ip command is for when you need things done at the moment quick, just don't care if something is going to stick after a reboot or possibly get lost after you log out- or, maybe you just want some information in a different format.

Quick NMCLI examples

As you can see below, most nmcli objects and commands can be truncated down a lot,
nmcli con show - *Find different connections for different devices*
nmcli con show eno1 - *Get details on eno1*
nmcli dev status - *Get status of all devices*
--------We know we have an ethernet device object called "eno1" so let's do stuff with it:
nmcli con add con-name dhcp type ethernet ifname eno1
nmcli con add con-name static ifname eno1 autoconnect no type ethernet ip4 192.168.122.102 gw4 192.168.122.1
nmcli dev status - *Find out which is used*
nmcli con up static; nmcli con up dhcp - *Bring these connections up*
nmcli con show static - *Get this connection's status*
--------So this made two connection objects (static and dhcp) pointing to the device object en01
--------Here, we are going to add more information and a second IP address to the "static" connection object:
nmcli con mod static ipv4.dns 192.168.122.1 --- to specify a dns server
nmcli con mod static +ipv4.dns 8.8.8.8 --- to add a dns server (+ is needed if one has already been defined)
nmcli con mod static +ipv4.addresses "192.168.100.10/24 192.168.100.1" -- modify IP and gateway
nmcli con mod static +ipv4.addresses 10.0.0.10/24 -- add a secondary IP addy
--------All of this writes the settings but doesn't activate them - you have to reload the connection for them to work
nmcli con reload - *Re-reads the config file if you can't take down conn and bring it back up*

So, a few interesting things about this example.
- Note that it uses dotted notation to add the ip4 properties addresses and dns. These are listed in the man page for nm-settings(5). [http://manpages.ubuntu.com/manpages/zesty/man5/nm-settings.5.html]
- If you do not specify a connection name when creating it, one is auto-generated as "type-ifname[-number"

The nmcli Prompt

When you choose the "edit" option on an object, you get the nmcli prompt, and can issue directives that way instead. To turn the connection “net-eth1” to DHCP (auto) instead of static:

```
[user@localhost ~]$  nmcli con edit net-eth1
> print all
> remove ipv4.gateway
> remove ipv4.address
> set ipv4.method auto
> set ipv4.dns 8.8.8.8 8.8.4.4
> verify all
> save persistent
> quit
```

Getting Rid of Network Manager

To disable Network Manager on a systemd system:
    $ sudo systemctl stop NetworkManager.service AND systemctl disable NetworkManager.service
On a systemVinit system:
    $ sudo service NetworkManager stop AND chkconfig NetworkManager off
In Debian 7 or earlier:
    $ sudo /etc/init.d/network-manager stop AND update-rc.d network-manager remove
In Ubuntu or Linux Mint:
    $ sudo stop network-manager AND echo "manual" | sudo tee /etc/init/network-manager.override
Slackware:
    $ /etc/rc.d/rc.networkmanager stop AND chmod a-x /etc/rc.d/rc.networkmanager

In some versions of NM, issuing "stop" may also kill dhcp and wpa_supplicant, so be sure to check.
After disabling Network Manager on Debian or Ubuntu, use /etc/network/interfaces to configure network interfaces.
After disabling Network Manager on Fedora or CentOS, use /etc/sysconfig/network-scripts/ifcfg-ethX files to configure network interfaces.

To disable Network Manager only for eth1 (for example)
Network Manager automatically ignores any interfaces specified in the file /etc/network/interfaces (Debian/Ubuntu), or the proper config file inside the directory /etc/sysconfig/network-scripts/ (RHEL/CentOS/Fedora)
Let's say your interface is eth0.

- For RHEL-compatible, make a file for your interface in /etc/sysconfig/network-scripts/ifcfg-eth0

```
DEVICE="eth0"
NM_CONTROLLED="no"    # this is most important
ONBOOT=yes
HWADDR=A4:BA:DB:37:F1:04
TYPE=Ethernet
BOOTPROTO=static
NAME="System eth0"
UUID=5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03
IPADDR=192.168.1.44
NETMASK=255.255.255.0
# Optionally put these in or add them to this file: # vi /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=centos6
GATEWAY=192.168.1.1
# Same thing with this - or add into in resolv.conf  #vi /etc/resolv.conf
nameserver 8.8.8.8      # Replace with your nameserver ip
nameserver 192.168.1.1 # Replace with your nameserver ip
DNS1=8.8.8.8    # another optional format if it works better for you
DNS2=8.8.4.4
# The reason these are optional is this is how you would specify per-interface file
# different default gateways and DNS if you had too.  Not always guaranteed to work but there it is.
```

- For Debian/Ubuntu - In /etc/network/interfaces, add information about the interface you want to disable NM on

```
$ sudo vi /etc/network/interfaces
# Find/add your eth0 entry to disable Network Manager
allow-hotplug eth0
iface eth0 inet static
address 10.0.0.10
netmask 255.255.255.0
gateway 10.0.0.1
dns-nameservers 8.8.8.8
```

For this to work you need to ensure the network service will bring up eth1 upon boot (since NM isn't doing it)
On systemd systems run: $ sudo systemctl enable network.service
On SysVinit systems run: $ sudo chkconfig network on

Upon rebooting, verify that Network Manager is successfully disabled for eth0 with nmcli command.

systemd-networkd

For some time lacked features offered by NetworkManager (check the version you have). Predictably integrated with the rest of systemd (e.g., resolved for DNS, timesyncd for NTP, udevd for naming), and of course shares the rejection of many sysadmins who despise systemd. The command networkctl to show what networkd sees. It features subcommands list, status, and lldp to display info - query a specific device (such as ens128) or --all

To switch from Network Manager to systemd-networkd run:
$ sudo systemctl disable NetworkManager --AND-- sudo systemctl enable systemd-networkd

You also need to enable systemd-resolved service
$ sudo systemctl enable systemd-resolved --AND-- $ sudo systemctl start systemd-resolved
This daemon will create its own resolv.conf - but many programs still look to /etc/resolv.conf, so it is recommended to create a symlink to /etc/resolv.conf
$ sudo rm /etc/resolv.conf --AND-- $ sudo ln -s /run/systemd/resolve/resolv.conf /etc/resolv.conf

To configure network devices you specify configuration information in text files named *.network - to be stored and loaded from /etc/systemd/network. Use **networkctl list** to see available devices on the system.
$ sudo mkdir /etc/systemd/network

To configure DHCP networking (below "yes" can be "ipv4"):
$ sudo vi /etc/systemd/network/20-dhcp.network
[Match]
Name=enp3*
[Network]
DHCP=yes

[Match] obviously says which network device(s) are configured- this matches any interface whose name *starts with* ens3. For static IP on enp3o2 the network block would contain the following with name= enp3o2. Processed in lexical order - a file named 10-static.network, would take precedence over 20-dhcp.network and retain a static IP.
[Network]
Address=192.168.10.50/24
Gateway=192.168.10.1
DNS=8.8.8.8

Wireless interfaces don't have any special differences in the [match] and [network] fields, but they need configuration from another service (wpa_supplicant). An (example) device named wlp2s0, the corresponding systemd service file to enable would be wpa_supplicant@wlp2s0.service, with the configuration file /etc/wpa_supplicant/wpa_supplicant-wlp2s0.conf. If that file doesn't exist, the service won't start.
When you are done, restart networkd to make the changes take effect - $ sudo systemctl restart systemd-networkd

Virtual Network Devices (bridges, VLANs, tunnel, VXLAN, bonding, etc)
These files have the naming *.netdev (rather than *.network). Here is a bridge (br0) with physical interface (eth1):
Create the bridge file: $ sudo vi /etc/systemd/network/bridge-br0.netdev
[NetDev]
Name=br0
Kind=bridge
Eth1 slave config file named *.network as before $ vi /etc/systemd/network/bridge-br0-slave.network
[Match]
Name=eth1
[Network]
Bridge=br0
The *.netdev file declared a bridge - config with a *.network file $ vi /etc/systemd/network/bridge-br0.network
[Match]
Name=br0
[Network]
Address=192.168.10.100/24
Gateway=192.168.10.1
DNS=8.8.8.8
All done, do restart systemd-networkd: $ systemctl restart systemd-networkd
You can use brctl tool to verify that a bridge br0 has been created.

Using iw and wpa_supplicant

Replacing iwconfig with iw

Action	iwconfig (outdated)	iw replacement
Getting info on wlan0	iwconfig wlan0	iw dev wlan0 link
Connecting	iwconfig wlan0 essid foo	iw wlan0 connect foo
Set channel	iwconfig wlan0 essid foo freq 2432M	iw wlan0 connect foo 2432
WEP	iwconfig wlan0 essid foo key s:abcde	iw wlan0 connect foo keys 0:abcde
Join ad-hoc ibss	iwconfig wlan0 mode ad-hoc iwconfig wlan0 essid foo-adhoc	iw wlan0 set type ibss iw wlan0 ibss join foo-adhoc 2412
Leave ad-hoc ibss	iwconfig wlan0 essid off (sometimes worked)	iw wlan0 ibss leave (always works)

For WPA/WPA2 encryption, you should use wpa_supplicant.

Managing connections with wpa_supplicant / wpa-cli

1. Run **ip a** to get name of the wireless interface. If not showing, the driver might need installing
2. Create a file in /etc/wpa_supplicant named *.conf containing this basic configuration line:
   ctrl_interface=DIR=/run/wpa_supplicant GROUP=wheel update_config=1
   GROUP specifies which groups can manage wpa_supplicant, and leaving blank means only root.
3. Initialize by running **wpa_supplicant -B -i w1linksys7 -c /etc/wpa_supplicant/example.conf**
   -B means run in background, -i specifies interface, and -c points to config file
4. Running **wpa_cli** gives an interactive prompt.

wpa-cli commands

scan - will run a scan

scan_results - will dump the scan results of available networks including ssid, security mode, and bssid/ MAC

add_network - to specify a network - provide ssid listed in scan, the key. Number given at the beginning is arbitrary

```
> add_network
0
> set_network 0 ssid "LOCAL_WIFI"
> set_network 0 psk "passcode"
> enable_network 0     - will attempt to associate with the network just configured
> save_config
```

Running **ip a** should then show new IP info

After running save_config, the following will be appended to your configuration file:

```
network={
        ssid="LOCAL_WIFI"
        psk="passcode"
}
```

Now, just running **wpa_supplicant -B -i w1linksys7 -c /etc/wpa_supplicant/example.conf** will connect.

Notes on using Kismet

Most things are self-explanatory with Kismet so there isn't much to cover. When running, typing "h" gives help screen with most info for current screen. In the network panel, W is WEP (yes, none, other); <no_ssid> means the AP isn't broadcasting it's ssid, T is type: P (probe request- no associated connection); A (access point); H (ad-hoc); T (turbocell aka Karlnet or Lucent); G (group); D (data-only network with no control packets).

Flags field includes F (AP using factory default settings/ not configured); T#, U#, A#, D mean an address range of # octets found via type of traffic, being TCP, UDP, ARP, or DHCP respectively

APs listed are color-coded as follows: yellow: unencrypted network; red: factory default settings in use; green: secure networks (WEP, WPA etc..); and blue means SSID cloaking on / SSID not broadcast

The kismet layout can be modified in **/etc/kismet/kismet_ui.conf**

Helpful key functions: type "c" to see clients on an AP, "i" for detailed info on an AP, "r" can show a stats graph, "a" for general stats on all APs, "w" to show all alerts that have come up in the status window

The program LinSSID is a Linux alternative to inSSIDer -- https://sourceforge.net/projects/linssid/

WiFi Pentesting Tools

The details of these tools can be found online or in man pages,

The Aircrack-ng Package - https://www.aircrack-ng.org

Aircrack-ng is the granddaddy of all wireless CLI suites, and has added a lot since I was first using it in 2005-6

- Monitoring: Packet capture and export of data to text files for further processing by third party tools.
- Attacking: Replay attacks, deauthentication, fake access points and others via packet injection.

airbase-ng - Configure fake access points
aircrack-ng - Wireless password cracker
airdecap-ng - Decrypt WEP/WPA/WPA2 capture files
airdecloak-ng - Removes wep cloaking from a pcap file
airdriver-ng - Provides status information about the wireless drivers on your system
aireplay-ng - Primary function is to generate traffic for the later use in aircrack-ng
airmon-ng and airmon-zc - This script can be used to enable monitor mode on wireless interfaces
airodump-ng - Used for packet capturing of raw 802.11 frames
airodump-ng-oui-update - Downloads and parses IEEE OUI list
airolib-ng - Designed to store and manage essid and password lists
airserv-ng - A wireless card server
airtun-ng - Virtual tunnel interface creator
besside-ng - Automatically crack WEP & WPA network
easside-ng - An auto-magic tool which allows you to communicate via an WEP-encrypted access point
buddy-ng - echoes back decrypted packets to the system running easside-ng in order to access the wireless network without knowing the WEP key
ivstools - This tool handles .ivs files. You can either merge or convert them.
makeivs-ng - Generates initialization vectors
packetforge-ng - Create encrypted packets that can subsequently be used for injection
tkiptun-ng - This tool is able to inject a few frames into a WPA TKIP network with QoS
wesside-ng - Auto-magic tool which incorporates a number of techniques to seamlessly obtain a WEP key

Typical WPA-PSK cracking involves taking down the wireless driver, bringing it back up in monitor mode with airmon-ng, firing up airodump-ng to capture packets, and using aireplay-ng to inject deauthentication packets at a client, so that the four-way handshake can be captured when it attempts to reathenticate with the AP. You then run aircrack-ng to crack the pre-shared key in the pcap against a dictionary file. Chances are slim if it won't match in a dictionary.

The WifiTap Package

- http://sid.rstack.org/static/articles/w/i/f/Wifitap_EN_9613.html and https://github.com/gdssecurity/wifitap/

- - traffic capture and injection over a WiFi network by configuring interface **wj0**

Includes wifiarp, wifidns, wifiping, wifitap

- set an IP address consistent with target network address range and route desired traffic through it
- arbitrary packet injection without specific library.
- bypass inter-client communications prevention systems (e.g. Cisco PSPF), reach SSIDs handled by AP

wifitap - WiFi injection tool through tun/tap device
wifiarp - WiFi injection ARP answering tool based on Wifitap
wifidns - WiFi injection DNS answering tool based on Wifitap
wifiping - WiFi injection based answering tool based on Wifitap

wifite - attack multiple WEP, WPA - made to be automated, crack passwords later, grab as much from APs with strongest signal strength so you can come get the gathered stuff and work with it later

Fern Wifi Cracker - a GUI offering the following which isn't limited to wireless attacks. They advertise:

WEP cracking, WPA/WPA2 Cracking with wordlist or WPS based attacks
Automatic AP attacks possible
Session hijacking (Passive and Ethernet Modes)
Internal MITM engine, bruteforce attacks (HTTP, HTTPS, TELNET, FTP)

cowpatty - This is strictly for WPA-PSK - needs aircrack-ng to grab things- provide with a wordlist and captured hash- it generates hashes from wordlist using SSID as seed. Includes genpmk that can precompute hashes

reaver, bully, pixiewps - Bully is faster, more effective for WPS attacks. Reaver was released as a proof of concept back when WPS attack was discovered. On the other hand pixiewps does an offline attack that is super-fast.

Netfilter/ iptables

service iptables [start | save | stop]
/sbin/iptables-save > filename ---- saves rules to STDOUT by default, so send to file
/sbin/iptables-restore < filename ---- restores rules from STDIN by default, so give it the file
/sbin/iptables ----primary ACL modifier utility
- Packet-processing is done top-down by chain order (so are rules in a chain). Duplicate rules can reside in the same chain.
- Rules that are more likely to be matched should be put in the chain higher up in the list than others (line numbers)
- Modules to proxy or function with other OSI layers: [/usr/lib/iptables/*.so]
- Check if enabled: [grep -i config_netfilter /boot/config*] -- you'll find CONFIG_NETFILTER=y
- /etc/sysconfig/iptables is where rules are stored
- Opening /etc/sysconfig/iptables-config:
    IPTABLES_SAVE_ON_RESTART and ON_STOP set to "yes" to save written and implemented rules.
    IPTABLES_SAVE_COUNTER turn on to keep packet counters going from where they left off after a stop or restart

iptables -t table <chain><action/direction><packet pattern><segment pattern> -j <fate>

Table:	filter (default), NAT (change IP addresses/ports), mangle (alter packets/segments TOS/TTL, etc)
Rule handling actions:	-A (append) -D (delete) -L (list) -F (flush) -I (insert) -R (replace) -N (new) -E (rename)
Chain:	INPUT, OUTPUT, FORWARD; PREROUTING and POSTROUTING
Interface:	-i [eth0 | eth1 | eth+] "+" is always wildcard, use ! for negation. Use -o for output interface
L3 Packet Pattern:	-s ip-addr (source), -d ip-addr (destination), "+" is wildcard
L4 Segment Pattern:	-p [tcp | udp | icmp] ; -dport, -sport ---- port #, or any name in /etc/services
State/ statefullness:	-m for matching; -m state --state NEW(syn), ESTABLISHED(syn-ack), RELATED, INVALID
fate (destination):	DROP, ACCEPT, REJECT/DENY, REDIRECT (NAT prerouting chain- local ports), LOG (syslog)

When applicable, -v or --verbose, --line-numbers use with -L list, enumerate lines (for insert and replace operations)

4 default tables (can't remove) and their chains in processing order:

- Filter (default- inbound and outbound traffic rules)	INPUT, FORWARD, OUTPUT
- NAT (change IP addresses, ports)	PREROUTING, OUTPUT and POSTROUTING
- Mangle (packet alteration)	PREROUTING, INPUT, FORWARD, (OUTPUT) and POSTROUTING
- Raw (special- rarely used)	PREROUTING, OUTPUT

The **NAT** table, consulted when a packet that creates a new connection is seen
- PREROUTING (for altering packets as soon as they come in),
- OUTPUT (for altering locally-generated packets before routing)
- POSTROUTING (for altering packets as they are about to go out)

The **mangle** table, used for specialized packet alteration.
- PREROUTING (for altering incoming packets before routing)
- INPUT (for packets coming into the box itself)
- FORWARD (for altering packets being routed through the box)
- OUTPUT (not recommended to use)
- POSTROUTING (for altering packets as they are about to go out).

The **raw** table, used mainly for configuring exemptions from connection tracking along with the NOTRACK target.
- It registers at the netfilter hooks with higher priority and is thus called before ip_conntrack, or any other IP tables.
- Has built-in chains: PREROUTING (packets arriving, any interface) OUTPUT (packets generated by local processes)

Permit SSH	iptables -A INPUT -p tcp --dport 22 -j ACCEPT
Deny telnet	iptables -A INPUT -p tcp --dport telnet -j DROP
Block all traffic NOT from specified IP	iptables -A INPUT -s ! 192.168.1.72 -j DROP
Drop all inbound traffic	iptables -A INPUT -j DROP

Inserts this as rule #1 in the INPUT chain	iptables -I INPUT 1 -p tcp --dport 22 -j DROP
Delete rule #1 in INPUT chain	iptables -D INPUT 1
Deletes first match of this	iptables -D INPUT -p tcp --dport telnet -j DROP
Replace rule 1 with this rule	iptables -R INPUT 1 -p tcp --dport 22 -j ACCEPT
Z option zeros out byte count for chain	iptables -Z INPUT
Zero byte count in PREROUTING chain's mangle table	iptables -Z PREROUTING -t mangle
Flush all rules from output filter chain	iptables -F OUTPUT
Flush all chains	iptables -F
Lists ICMP types for us to view	iptables -p icmp --help
Deny echo-reply from all hosts	iptables -A INPUT -p icmp --icmp-type echo-reply -j DROP
Don't reply to any hosts	iptables -A OUTPUT -p icmp --icmp-type echo-request -j DROP

Allow ALL Incoming SSH on eth0	iptables -A INPUT -i eth0 -p tcp --dport 22 -m state --state NEW,ESTABLISHED -j ACCEPT iptables -A OUTPUT -o eth0 -p tcp --sport 22 -m state --state ESTABLISHED -j ACCEPT
Restrict syslog access (gateway IP should have access	iptables -A INPUT -p udp --dport 514 -s 192.168.1.1 -j ACCEPT iptables -A INPUT -p udp --dport 514 -s ! 192.168.1.1 -j DROP
Lock down NTP (129.6.15.28 and 29 are NIST)	iptables -A INPUT -p udp --dport 123 --sport 123 -s 129.6.15.2+ -j ACCEPT iptables -A INPUT -p udp --dport 123 --sport 123 -s ! 192.168.1.0/24 -j DROP
MAC address filtering	iptables -A INPUT -p tcp -m mac --mac-source 40:6c:8f:47:84:d0 -dport 8080,23 -j DROP
Set default policy to DENY	iptables -P INPUT DROP
NAT- port redirection	iptables -t nat -A PREROUTING -p tcp --dport 2323 -j REDIRECT --to-ports 23
Drop all webserver connections Multiport matches (15 ports maximum per rule)	iptables -A INPUT -m state --state NEW,ESTABLISHED -p tcp -m multiport --dport 80,443 -j DROP iptables -A OUTPUT -m state --state NEW,ESTABLISHED -p tcp -m multiport --dport 80,443 -j DROP
Make a new chain in mangle table	iptables -N INTRANET -t mangle
Rename a chain	iptables -E INTRANET MY_SUBNET
Replace/modify, all from INPUT from subnet go to MY_SUBNET	iptables -R INPUT 1 -s 192.168.1.0/24 -j MY_SUBNET
If NOT matched, pass back to origin chain, start filtering right after the rule where it left off	iptables -A MY_SUBNET -p tcp --dport 22 -j DROP

Logging goes in /var/log/messages by default. Add exception "kern.none" to line logging anything over "info" level and add line to provision further instructions
/etc/syslog.conf
uncomment #kern.* /dev/console/ and change to /var/log/firewall

iptables -A INPUT -p tcp --dport telnet -j LOG
Put these at the top of the INPUT chain (or even better, a subchain pointed to there) Default level is "warning"
iptables -A INPUT -p tcp -m multiport --dport 8080,23 -j LOG
iptables -A INPUT -p tcp -m multiport --dport ! 8080,23 -j LOG
iptables -A INPUT -p tcp --dport 22 -j LOG --log-prefix "SSH ACCESS ATTEMPT: "
iptables -A INPUT -p tcp --dport 23 -j LOG --log-prefix "UNAUTHORIZED TELNET ACCESS ATTEMPT "
--log-level debug emerg etc
--log-tcp-options
--log-ip-options
--log-tcp-sequence

Security Enhanced Linux (SELinux) - https://selinuxproject.org/

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/using_selinux/index

- Provides an extra layer of security to resources in Linux systems
- Runs as a loadable kernel module, gets a pseudofilesystem mount point like /proc
- Restricts access by subjects (users, processes) to objects (files) by applying and using labels for them.
- Separates users, processes and objects into sandboxes, "confined" domains, and one sandbox for everything else (unconfined_t). i.e., as "targets" httpd and ntpd have their own sandboxes that are isolated from eachother
- If a targeted process tries to access resources outside it's confined domain, access is denied and it's logged.
- Provides mandatory access controls (MACs) to extend the basic Linux discretionary access controls (DACs)
- MAC-based checks happen AFTER the DAC-based checks
- Stores MAC permissions in extended attributes of file system, attaching SELinux "labels"
- An "access vector cache" (AVC) stores decisions made (allow/disallow access) to speed up performance during runtime

Side note: a special pain about SELinux is it doesn't install manpages (!) This will provide *some*: sepolicy manpage -a -p /usr/share/man/man8 after installing the policycoreutils-devel package. This issue has never been resolved for over a decade now.

Basic Concept

Type Enforcement (TE)

This is the foundation of SELinux. Things are given context labels, and rules to say how those things can interact.

- Context, applied to subject types (processes) and object types (files, resources) to define security relations between them. The "objects" can be devices, network interfaces, addresses, ports, sockets, (and many things defined in /proc)

- Rules dictate access control by specifying permissions between subject types (domains) and object types. They determine whether a subject with a specific context is allowed or denied access to objects based on their own contexts.

Security Levels: MCS and MLS (optional security enhancements)

These are optional mostly. If you don't change them they will simply be using default values (so not usually a concern).

- Multi-Category Security (MCS): Utilizes functional or departmental categories for access control within an organization.
- Multi-Level Security (MLS): Utilizes security clearance levels or sensitivity classifications (Bell-La Padula model)

Since these are mostly optional, they will be covered later in this document. These also get inspected after type enforcement.

SELinux Modes

- Multiple modes of SELinux functionality can be applied on a system:
  - permissive - permission is granted, but denials are logged to /var/log/messages (for testing)
  - enforcing - strictly enforces 'targeted' policy rules (default)
  - disabled - only basic DACs are used

Using **getenforce** simply reports current mode: enforcing, permissive, or disabled; **sestatus** gives more details:

```
# sestatus
   SELinux status:         enabled
   SELinuxfs mount:        SELinux
   Current mode:           enforcing     -- current mode of operation
   Mode from config file:  permissive    -- mode set by /etc/sysconfig/SELinux
```

To change temporarily, setenforce 0 (permissive) or setenforce 1 (enforce) -or- "echo 1 > /SELinux/enforce"
To set the persistent mode, edit the file /etc/SELinux/config (symlink to /etc/sysconfig/SELinux)

```
>>> cat /etc/selinux/config
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Generally advice is don't disable SELinux. Some installs may call for it, but permissive mode should usually be enough.
If disabled, unless temporarily or to make changes to non-critical running items, you generally would restart after setting enforcing in /etc/sysconfig/SELinux so it will take effect after checking/relabling the system (same with switching from enforcing to disabled)
Switching between enforcing and permissive does not have that limitation (getenforce/ setenforce)
On startup, you can also switch the settings for SELinux in grub:

```
kernel /vmlinuz-2.6.32-279.el6.x86_64 root=/dev/md3 SELinux=1 enforcing=0
```

SELinux=0 is disabled, 1 is enabled, and with the enforcing setting, permissive is 0 and 1 is enforcing

SELinux mode commands: setenforce, getenforce, sestatus

Viewing a file's SELinux context: Labels
Several commands take the -Z option to display SELinux contexts; output of ls shows us files/directories (objects):
$ ls -Z file1
    -rw-rw-r-- user1 group1 unconfined_u:object_r:user_home_t:s0 file1
ls -Z /var/www/html/file5
    -rw-r--r-- root root unconfined_u:object_r:httpd_sys_content_t:s0 file5
ls -al -Z /var/www/html
    drwxr-xr-x. 2 apache root system_u:object_r:httpd_sys_content_t:s0 4096 Dec 23 20:47 .

These examples, the familiar user, group, DAC permissions and filename are shown, but adding the -Z option, we can see the context labels SELinux provides : a user (unconfined_u), a role (object_r), a type (user_home_t), and a level (s0).
- User labels: Non-privileged user = user_u ; Privileged user = root_u
- Role-based labels: Non-privileged and users = user_r, system_r
- Type/domain labels: 12 default protected daemons: httpd, ntpd, dhcpd, mysqld, named, nscd, portmap, postgres, snmp, squid, winbind, syslogd. All others (unless customized) get unconfined_t domain
- Levels (s0) are part of the MCS/MLS options I am saving till the end to keep this simple.

id -Z	Shows user's security context
ps -Z	Shows context for running processes (subjects and thier sandboxes/ domains/analogous to namespaces).
cp -Z, mv -Z	Maintains/preserves the security context when copying/moving files.
mkdir -Z	Sets the security context for newly created directories.
netstat -Z, ss -Z	Displays SELinux context information for network connections.

Using semanage to configure SELinux
Common options: -a, --add; -d, --delete; -m, --modify; -l, --list, -import and -export <filename> to input or output your configs

File context definitions	Add fcontext for all in /web	semanage fcontext -a -t httpd_sys_content_t "/web(/.*)?"
Network port type defs	Allow Apache to listen on tcp port 81	semanage port -a -t http_port_t -p tcp 81
Network interface type	List all interface definitions	semanage interface -l | grep eth*
Network node type	semanage node -a -t node_t -p ipv4 -M 255.255.255.0 192.168.1.0	
Manage policy modules	Install custom apache module	semanage module -a myapache

login - Manage login mappings; user - Manage confined users (roles and levels); boolean - Manage booleans; dontaudit - Disable/Enable dontaudit rules; ibpkey - infiniband pkey type definitions; ibendport - infiniband end port type definitions

Use-case example: You want your SSH host keys in /data/keys. You create the directory, move all the keys into the new home and change the sshd_config file to match the new mapping. When you attempt to use SSH, it fails.
    semanage fcontext -l | grep sshd   - outputs:
        /etc/ssh/primes regular file system_u:object_r:sshd_key_t:s0
        /etc/ssh/ssh_host.*_key regular file system_u:object_r:sshd_key_t:s0
        /etc/ssh/ssh_host.*_key.pub regular file system_u:object_r:sshd_key_t:s0
    semanage fcontext -a -t sshd_key_t   '/data/keys/*.*'
    restorecon -r /data/keys

Important! semanage only changes the policy: Use restorecon afterward to actually label the filesystem.
The tools chcon/chcat do NOT make persistent changes! These are only useful for temporary changes for testing.

Relabeling files and the filesystem
Restores filesystem to permissions(labels), according to what is specified in /etc/SELinux/targeted/policy/
Create hidden .autorelabel file at root of filesystem- gets init to relabel on startup
    touch / .autorelabel && reboot

If you don't want to reboot, using restorecon or fixfiles command will do the relabeling, - **however** - *existing processes may remain running in incorrect and insecure domains, and it will ask to empty /tmp/ since it can't relabel it, as root, temporary files that applications are relying upon are trashed. Instead work* on specific directories or processes rather than the entire filesystem
    - Use fixfiles to restore contexts of files by the package that installed them with '/sbin/fixfiles -R package_name'
    - Or, use '/sbin/restorecon -Rv /directory/path' -R is recursive, use -n to looks for changed files but won't make changes
If a file is moved and restorecon is run on it, it will be given permissions of it's parent directory
Generally, if a file (object) does not have specific fc/te specified, it inherts that of the enclosing directory.

On a SysVinit machine, /usr/sbin/run_init ensures protected daemon isolation and sets up proper contexts for services during system startup. After making changes to SELinux settings or encountering processes running outside proper contexts, you would kill the parent process and then run /usr/sbin/run_init again for the affected service. (such as '/usr/sbin/run_init /etc/init.d/httpd')
On a systemd machine, after modifying SELinux settings for a process, run load_policy or restorecon, and restart it with a systemctl restart <service_name>.

If you make ANY changes to SELinux that are global (i.e. booleans or installing new SELinux binaries), you will have to restart all processes, so a reboot would be necessary

SELinux Booleans
Booleans conveniently permit runtime adjustments to SELinux without the need to modify or reload the policy, and activate specific functionalities on processes seamlessly.

semanage boolean -l *List of them with descriptions, if on or off, and the default value*

SELinux boolean	State	Default	Description
ftp_home_dir	(off,	off)	Allow ftp to read and write files in the user home directories
xdm_sysadm_login	(off,	off)	Allow xdm logins as sysadm
xen_use_nfs	(off,	off)	Allow xen to manage nfs files
ssh_chroot_rw_homedirs	(off,	off)	Allow ssh with chroot env to read and write files in the user home directories
postgresql_can_rsync	(off,	off)	Allow postgresql to use ssh and rsync for point-in-time recovery
authlogin_shadow	(off,	off)	Allow users login programs to access /etc/shadow.
httpd_can_network_relay	(off,	off)	Allow httpd to act as a relay
openvpn_enable_homedirs	(on,	on)	Allow openvpn to read home directories

getsebool -a | grep httpd *Provides a different view*
httpd_builtin_scripting --> on
httpd_can_network_connect --> off
To change a value with setsebool: setsebool httpd_enable_cgi off ---- Make persistent with -P (on and off can be 0 or 1)
Changing booleans persistently might not be done with semanage boolean, so use setsebool -P instead

Disabling Specific (Targeted) Policies While Running
We need to edit the items in the /booleans directory and toggle the boolean:
echo "1 1" > /SELinux/booleans/http_disable_trans
The file commit_pending_bools is monitored by SELinux to see if it needs to refresh the policies
echo "1" > /SELinux/commit_pending_bools
Restart the affected service: /sbin/service httpd restart
Apache is now running in the unconfined_t domain

MLS/MCS Levels Explained
The order of operations in SELinux is as follows:
1.) DAC (Discretionary Access Control, regular Linux permissions) are considered by the host OS
2.) TE (Type Enforcement, the most basic SELinux operation) is inspected
3.) MLS (Multi-Level Security sensitivity labels determine access control, i.e., secret, top secret)
4.) MCS (check Multi-Category Security for access control based on categories)

~] # semanage login -l

Login Name	SELinux User	MLS/MCS Range
_default__	unconfined_u	s0-s0:c0.c1023
root	unconfined_u	s0-s0:c0.c1023
system_u	system_u	s0-s0:c0.c1023

Login Name column lists Linux users, and the SELinux User column lists which SELinux user the Linux user is mapped to. For processes, the SELinux user limits which roles and levels are accessible. Finally, the ranges of MLS/MCS access are listed

An MLS range is a pair of levels, written as lowlevel-highlevel, or if the levels are identical- for example- s0-s0 is the same as s0. Each level is a sensitivity-category pair, with categories being optional. If there are categories, the level is written as sensitivity:category-set. If there are no categories, it is written as sensitivity.

A contiguous series of categories can be abbreviated, such as c0.c3 means c0,c1,c2,c3.
The /etc/SELinux/targeted/setrans.conf file maps levels (s0:c0) to human-readable form (ie. CompanyConfidential). This file needs to have changes to it made by semanage rather than manually edited.

MLS sensitivity levels range from s0 as the least to s15 as the most sensitive.
Default unconfigured SELinux has s0-s0:c0.c1023, with MLS level s0 authorized for all categories.
MCS has up to 1024 different categories: c0 through to c1023.

MLS is based on the Bell-La Padula MAC model, used in Labeled Security Protection Profile (LSPP) environments. You have to install the package [e.g., dnf install selinux-policy-mls], and configure MLS to be the default SELinux policy. This is still incomplete- you have to configure it specifically for your needs and it won't have what it needs for your programs- upstream SELinux Reference Policy can be built that is more inclusive but MLS isn't something you can just unbox and it's ready to go. A full discussion of it is our of the scope of this writing for that reason.

Star utility for backup (SELinux tar)
Tar does not archive security context labels. Star has it's own package: star-1.5a25-6.i386.rpm
star -xattr -H=exustar -c -f newarchive.star foldername/ ---extended attributes, -c create, -f for normal
star -xattr -x -f newarchive.star --- -x to extract

Logging
System calls are filtered through SELinux policy to see if allowed.
If not allowed, an avc:denied message is generated goes through auditd, which writes event to /var/log/audit/audit.log (config file is /etc/audit/auditd.conf) If SELinux is in enforcing, action stopped, if permissive it is allowed, but logged. For example, when a web browser asks Apache for /foo/index.html, a getattr /foo/index.html syscall is issued. If it has the wrong label, SELinux enforcing stops it there. You can also see the getattr referred to in the AVC alerts in audit.log

For a sample denial in logs, running 'grep AVC /var/log/audit/audit.log' you'll see something like this:
type=AVC msg=audit(1711932009.640:1125): avc: denied { open } for pid=7237 comm="httpd" path="/var/www/html/index.html" dev="dm-1" ino=28668713 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:httpd_sys_content_t:s0 tclass=file

Deciphering the line above:
Type of message (AVC) followed by epoch timestamp, there has been an AVC denial on {open} operation.
Open was denied for PID 7237, where the command (comm) is related to httpd. The denied access occurred at path "/var/www/html/index.html" on device "dm-1", with inode number 28668713.
The source context (scontext) is system_u:system_r:httpd_t:s0, representing the httpd program. The target context (tcontext) is system_u:object_r:httpd_sys_content_t:s0, indicating the SELinux label associated with the target file.
Both contexts follow the SELinux label format (*_u, *_r, *_t for user, role, and type), with the source context representing the program (httpd) and the target context representing the type of item being accessed (httpd_sys_content_t).
In this case, httpd has access to files labeled as system_u:system_r:httpd_t:s0, but the file being accessed has a label of system_u:object_r:httpd_sys_content_t:s0, which doesn't match.
So if you run 'sealert -a /var/log/audit/audit.log', you'll see something like this.

```
SELinux is preventing httpd (httpd_t) from { open } access on the file index.html.
*****  Plugin httpd (72.4 confidence) suggests   ************
If you want to allow httpd to open index.html file
Then you need to change the file context to httpd_sys_content_t.
Do
# semanage fcontext -a -t httpd_sys_content_t '/var/www/html/index.html'
# restorecon -v '/var/www/html/index.html'
*****
Allowing Access:
Do
# semanage fcontext -a -t httpd_sys_content_t '/var/www/html/index.html'
# restorecon -v '/var/www/html/index.html'
Additional Information:
Source Context              system_u:system_r:httpd_t:s0
Target Context              system_u:object_r:httpd_sys_content_t:s0
Target Objects              /var/www/html/index.html [ file ]
```

AuditD Tools

```
# aureport -a
AVC Report
===============================================
# date time comm subj syscall class perm obj result event
===============================================================================
1. 02/16/2020 20:52:51 ? (null) 0 (null) (null) (null) unset 745
2. 02/16/2020 22:35:35 ? (null) 0 (null) (null) (null) unset 1391
3. 02/21/2020 10:29:41 httpd system_u:system_r:httpd_t:s0 49 tcp_socket name_bind system_u:object_r:websm_port_t:s0 denied 1144
4. 02/21/2020 10:29:41 httpd system_u:system_r:httpd_t:s0 49 tcp_socket name_bind system_u:object_r:websm_port_t:s0 denied
```

Logged events in the past 3 days	aureport --start 04/08/2024 00:00:00 --end 04/11/2024 00:00:00
All executable file events	aureport -x
Summarize executable file events	aureport -x --summary
Failed events for all users	aureport -u --failed --summary -i
All failed login attempts per each system user	aureport --login --summary -i
All audit files that are queried and times they included	aureport -t
ausearch userID's file access events, make report	ausearch --start today --loginuid 1000 --raw | aureport -f --summary

aulast	List of last logged-in users with login times	aulast
aulastlog	Last login information of users	aulastlog
ausyscall	Converts between system call #'s and names	ausyscall --name 3 or ausyscall --number open
auvirt	List virtualization-related audit records	auvirt
auditctl	Control the kernel's audit system config	auditctl -a always,exit -F arch=b64 -S unlink
augenrules	Generates rules from text file for the audit framework	auegenrules /etc/audit/audit.rules
aureport	Generates summary reports from audit logs	aureport --summary
ausearch	Searches the audit logs for specific events	ausearch -ua 500
autrace	Traces execution of a program, capturing system calls	autrace /bin/ls

Troubleshooting Using sealert: Be Careful! Sometimes You Get Bad Advice

This example was used in Sander Van Vugt's videos for RHEL 7. One of the better examples of why you need to be careful.

```
[root@localhost]# ssh -p 2022 localhost
ssh: connect to host localhost port 2022: Connection refused
[root@localhost]# ssh -p 443 localhost
ssh: connect to host localhost port 443: Connection refused
[root@localhost]# lsof -l
    COMMAND   PID   USER   FD   TYPE   DEVICE   NODE   NAME
    sshd      3538  root   3u   IPv4   31495    TCP    192.168.4.172:5-192.168.4.1:59438 (ESTABLISHED)
    sshd      3538  root   8u   IPv6   31867    TCP    localhost:x11-ssh-offset (LISTEN)
    sshd      3538  root   9u   IPv4   31868    TCP    localhost:x11-ssh-offset (LISTEN)
[root@localhost]# grep AVC /var/log/audit/audit.log
type=AVC msg=audit(1425663361.745:487): avc: denied {name bind} for pid=4555 com="sshd" src=443 scontext-
system_u:system_r:sshd_t:s0-s0:c.c1023 tcontext-system_u:object_r:http_port_t:s0 tclass=tcp_socket

[root@localhost]# sealert -a /var/log/audit/audit.log
Mar 6 12:48:22 localhost dbus [868]: [system] Successfully activated service 'org.fedoraproject.Setroubleshootd'
Mar 6 12:40:23 localhost setroubleshoot: Plugin Exception restorecon_source
Mar 6 12:40:23 localhost setroubleshoot: SELinux is preventing /usr/sbin/sshd from name_bind access on the tcp_socket. For complete SELinux
messages. run sealert -l 88dc1625-8b9e-4a8f-ad9e-4412068fe9ac
Mar 6 12:48:23 localhost python: SELinux is preventing /usr/sbin/sshd from name_bind access on the tcp_socket.
*****       Plugin catchall (100. confidence) suggests      **********************
If you believe that sshd should be allowed name_bind access on the tcp_socket by default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# grep sshd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -l mypol.pp
Mar 6 12:40:23 localhost setroubleshoot: SELinux is preventing /usr/sbin/sshd from name_bind access on the tcp_socket For complete SELinux
messages. run sealert-1 88dc1d25-8b9e-4a8f-ad9e-4412068fe9ac
```

So there's this socket error. Some options are in "semanage port"
-a is add, -m is modify so for 443 its semanage -m -t sshd_t -p tcp 443
You can also list all port definitions and grep for port (semanage port -l | grep port) and you'll find "Allow sshd to listen on tcp port 8991 #semanage port -a -t ssh_port_t -p tcp 8991"

But think first: what sealert suggested is a blanket allow policy with module to allow all traffic of a particular type if we look at /var/log/messages, we sometimes get more data, but it's telling us to get more info by running sealert -l 188dc1d25-8b9e-4a8f-ad9e-4412068fe9ac, so we'll try that.

```
SELinux is preventing /usr/sbin/sshd from name bind access on the tcp_socket
*****       Plugin catchall (100. confidence) suggests      **********************
If you believe that sshd should be allowed name bind access on the tcp socket by default. Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
# grep sshd /var/log/audit/audit.log | audit2allow -M mypol
# semodule -l mypol.pp
Additional Information:
Source Context                system_u:system_risshd_tise-se:ce.c1023
Target Context                 system u:object r:http_port_t:se
Target Objects                 [  tcp_socket  ]
Source                        sshd
Source Path                   /usr/sbin/sshd
Port                          443
Host                          localhost.localdomain
Source RPM Packages           openssh-server-6.4p1-8.el7.x86_64
Target RPM Packages
Policy RPM                    selinux-policy-3.12.1-153.el7.noarch
                              (trimmed output)
```

What do we get? The same bad advice: to make a policy module change that allows a bunch of junk permissions that shouldn't be there- AND it's 100% confident this is the right answer, after all! This shows the danger this can cause.
Think hard before trusting what the advice sealrt gives. Usually seaudit will give a few options, one *might* be acceptable

SELinux files and locations

- /etc/SELinux/targeted

This directory contains config files specific to the targeted SELinux policy. (policy modules, contexts, and configs)

- /etc/SELinux/targeted/policy contains compiled binaries of policies
- /etc/SELinux/targeted/contexts contains exactly that, i.e. the file default_type cats out system_r:unconfined_t

The /etc/SELinux/targeted/contexts/file_contexts files are what holds the default maps of directories and files to labels.
Syntax for file_contexts content= regexp [-type] (context | <<none>>)
Examples ("--" means a file instead of a directory (-d), and -c for block or character special files) :

```
/home/[^/]+         -d          system_u:object_r:user_home_dir_t
/home/[^/]+/.+                  system_u:object_r:user_home_t
/mnt/[^/]*/.*                   <<none>>
```

The /selinux pseudofilesystem (mounted similarly as /proc) exposes runtime SELinux data, like current security context of processes and files, via virtual files. Is used by the OS and SELinux-aware commands to interact with, obtain status and config info

The /etc/SELinux/targeted/src/ directory is created when you install the selinux-policy-targeted-sources package.

- /etc/SELinux/targeted/src/policy --- source tree --- contains .fc (file context) and .te (type enforcement) files
- /etc/SELinux/targeted/src/policy/file_contexts --- has source info for building the file_contexts file (for files, resources
- /etc/SELinux/targeted/src/policy/file_contexts/program/ contains fc files for specific programs, commands (processes)
- /etc/SELinux/targeted/src/policy/domains --- individual domains/ contexts, the rules for specific programs or services
- /etc/SELinux/targeted/src/policy/modules --- for modules for policy rules for specific functionalities or components

Creating Policies for Unsupported Software/ Items

Custom policies are needed if you have a program not represented by default, or to change the defaults (i.e., httpd).
You need to operate on the source of the targeted policy to make customizations.
Running "rpm -qa | grep SELinux" Brings us SELinux-policy-targeted-x.xx.xx These are just the binaries- we need to run rpm -Uvh SELinux-policy-targeted-source-x.xx.xx.rpm" to get the source files.

The installed /etc/SELinux/targeted/src/ directory is where you can start working.
Files named *.fc contain file context definitions, those named *.te contain SELinux policy, Type Enforcement (TE) rules

Here are the fundamental steps for compiling and installing a custom policy:

1. Edit then compile the .te files into a binary policy module (.mod file) and do error checking

```
checkmodule -M -m -o <module_name>.mod <module_name>.te
```

2. Package the *.mod file into a policy module package (.pp file) The -f option specifies a *.fc file to use

```
semodule_package -o <module_name>.pp -m <module_name>.mod -f <module_name>.fc
```

3. Install the *.pp file into the system's running policy directory's module directory:

```
sudo semodule -i <module_name>.pp
```

Once the .pp file is compiled and installed, the file context info in the *.pp file is accessed to label the filesystem when it is relabeled and that's it.

The *.fc file entries map file paths to SELinux security labels. Here are some examples for httpd:

```
/usr/bin/httpd httpd_exec_t  # assigns httpd_exec_t label to the httpd binary
/var/www/html httpd_sys_content_t  # assigns httpd_sys_content_t to the web root
/var/log/httpd httpd_log_t # assigns httpd_log_t to the log files
```

Other options: read access to configuration files (httpd_config_t), to content files (httpd_content_t); write access to log files (httpd_log_t), network access to specific ports (tcp_port_t)

The *te file entries define how processes with a specific label (e.g., httpd_t) can interact with labeled objects (files, network sockets)
Syntax is allow | neverallow subject object:object_class {permissions}

```
allow httpd_t httpd_sys_content_t { read write append };  # Allow httpd to modify web server content
allow httpd_t tcp_socket connectto port 80, 443;  # Allow httpd to connect to web ports
allow httpd_t var_log_t { write append };  # Allow httpd to write to log files
allow httpd_suexec_t self:capability { setuid setgid };  #Allow to gain elevated privileges for CGI scripts
bool httpd_enable_ftp_server false;  # Boolean for if httpd can run an FTP server
if (httpd_enable_ftp_server) {
 allow httpd_t ftp_port_t:tcp_socket name_bind;   #Allow httpd to bind to the port 21
}
```

/etc/SELinux/: Primary configuration directory.
/etc/sysconfig/SELinux/: A symlink to /etc/SELinux/config which dictates default mode and policy
/etc/sysconfig/SELinux/restorecond.conf: Used for restoring contexts on objects.
/etc/sysconfig/SELinux/semanage.conf: Config file for the semanage utility.
/etc/SELinux/targeted/modules/active/booleans.local: Location for local boolean settings.
/etc/SELinux/targeted/booleans: Directory for SELinux boolean settings.
libsemanage - Library provides an API for the manipulation of SELinux binary policies.

SELINUX COMMANDS AND PACKAGES

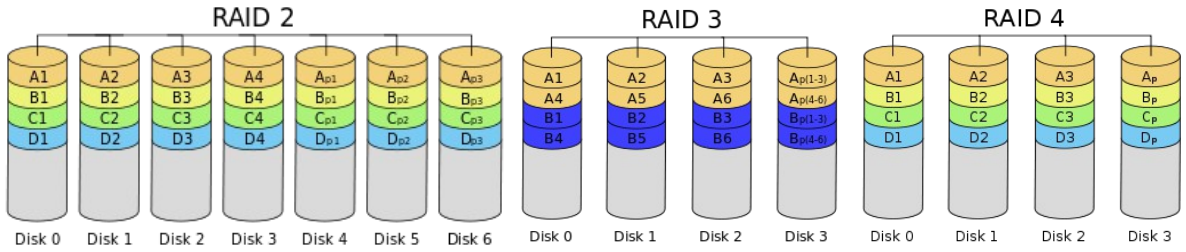
Pkg	Command	Description	Example
☾	system-config-selinux	GUI for configuring policies and settings	system-config-selinux
☆	sesearch	Searches policies for rules matching specified criteria	Find allow rules for Apache- sesearch -A -s httpd_t -p all
◇	sealert	View SELinux-related alerts and recommendations	sealert
○	audit2why	Explain AVC denial messages.	audit2why < AVC_denial_message
△	setfiles	Set default contexts based on file context info stored in the SELinux policy	setfiles -v /path/to/directory
△	restorecon	Relabels files to their default values (or changed by semanage, etc)	restorecon file.txt
△	restorecon_xattr	Restores SELinux extended attributes of files and directories	restorecon_xattr file.txt
□	avcstat	Displays average AVC statistics	avcstat
✶	sedta	Performs domain transition analyses on a policy file	sedta -f policy_file
✶	seinfoflow	Performs detailed information flow analysis	seinfoflow -d /usr/sbin/httpd
POLICY CREATION AND MANAGEMENT			
☾	selinux-polgengui	GUI for generating SELinux policies	(see graphic interface it gives you)
☆	apol	GUI to browse policy (types, classes, roles, users), rules (TE, RBAC, MLS)	(see graphic interface it gives you)
△	semodule	Manage policy modules (install, upgrade, listing, removing)	To install my_module.pp: semodule -i my_module.pp
☆	sechecker	Check SELinux policy for errors and common mistakes	sechecker /path/to/policy
☆	sediff	Compare two policies, reports differences	sediff policy1 policy2
☆	seinfo	Show info about policies, types, and attributes	seinfo /path/to/policy
△	load_policy	Load new SELinux policy into the kernel	load_policy /etc/selinux/targeted/policy/policy
△	sepolicy (semodule is better suited for this)	Manage policies, including loading, querying, and modifying policy rules	sepolicy <subcmd> <policy_rule> (there are query generate compile load and list options)
△	sepolgen	Generate policy interfaces based on input files	sepolgen input_file > output_file
△	sepolgen-ifgen	Generates interfaces in a similar format to sepolgen	sepolgen-ifgen existing_policy.pp > interfaces.cfg
○	audit2allow	Converts SELinux AVC denial messages into policy allow rules	Make rules from AVC denials in denials.log - audit2allow -i denials.log
△	semodule_link	Link a policy module into the current policy	semodule_link -i my_module
△	secon	Convert binary policy files to text	secon -t <policy.bin >policy.txt
△	semodule_expand	Expand modularized policy (pp) into one flat policy file (te)	semodule_expand -o my_policy.te my_module.pp
△	semodule_package/ semodule_unpackage	Create policy module package from current policy source file (or unpackage)	semodule_package -o my_policy.pp / semodule_unpackage my_module.pp
✧	macro-expander	Expands and shows macros used in policy files	macro-expander /path/to/policy
✧	checkmodule	Check module source file for errors, generate binary module	checkmodule -M -m -o /path/to/module.mod /path/to/module.te
✧	checkpolicy	Check policy source file for errors, generate a binary policy file	checkpolicy -M -c /path/to/policy.conf
✧	sedismod	Disassemble a binary policy module	sedismod /path/to/module.mod
✧	sedispol	Disassembles a binary SELinux policy	sedispol /path/to/policy.conf
CONTEXT AND CONTEXT CONFIG TOOLS			
□	selinuxconlist	List SELinux contexts.	selinuxconlist -l
□	selinuxdefcon	Displays the default SELinux context	selinuxdefcon
□	selinuxexeccon	Displays security context of a program (requires full path to program)	selinuxexeccon /bin/netstat
□	getpidprevcon	Get previous security context used by specified process	getpidprevcon 1234
□	matchpathcon	Checks if a file or directory has the correct SELinux context	matchpathcon /path/to/file
□	selabel_lookup	Get security context associated with a specified path	selabel_lookup /path/to/file
□	selabel_lookup_best_match	Find the best-matching security context for a specified path	selabel_lookup_best_match /path/to/file
□	selabel_partial_match	Checks if a path partially matches any SELinux file context	selabel_partial_match /path/to/file
□	selinux_check_access	Checks access against the loaded SELinux policy	selinux_check_access -a target_type -t source_type -p permission
□	validatetrans	Checks if any file with a source type is allowed to transition to the target type	validatetrans -t target_type -s source_type
△	genhomedircon	Generate SELinux file context config for home directories	genhomedircon -r /etc/selinux/targeted/contexts/files/file_contexts
□	sefcontext_compile	Compile file context config files into binary	sefcontext_compile
□	selabel_digest	Compute SHA-256 hash of a specified file's security contexts	selabel_digest /path/to/file
□	selabel_get_digests_all_partial_matches	Get hashes of contexts that partially match a path	selabel_get_digests_all_partial_matches /path/to/partial

△ policycoreutils + -devel, □ libselinux-utils, ○ policycoreutils-python-utils, ◇ setroubleshoot-server, ☾ policycoreutils-gui, ☆ setools-console, ▽ checkpolicy, ✧ selinux-policy-devel, ✶ setools-console-analyses

Striping increases data retrieval *performance* by allowing multiple data readers and writers to work on a single data set at the same time. **Mirroring** provides *redundancy* for recovery. **Parity** *ensures* that complete data can be retrieved from an array even if one or more disks fail

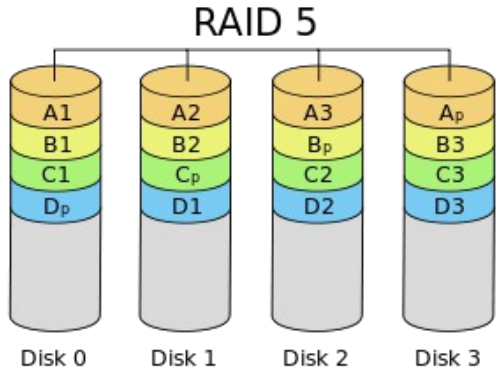
RAID 0: Disk striping
Multiple sources can access bits of data at the same time, performance can be improved.
No redundancy. One disk failure will result in lost data.

RAID 1: Disk mirroring - redundancy - no interruption of data availability
Description: Data is written to two or more disks. No master or primary, the disks are peers.
Performance: Fast read (simultaneous), slower write (writes twice)

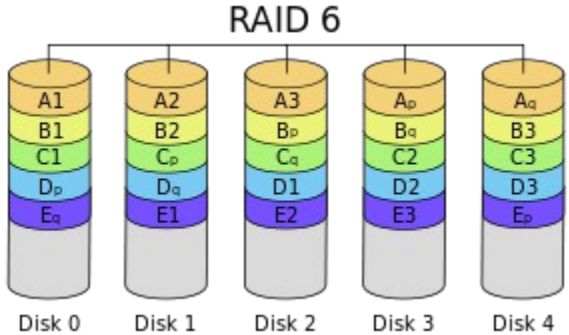


OBSOLETE! RAID2 and 3 - Byte-level striping; RAID4: Block level striping, dedicated parity disk
RAID3 used an additional disk for parity. Since every write touches multiple disks, Obsolete! Slow! All disks spin in synch (lockstep). For highest transfer rates in long sequential reads and writes Raid 4: Block-level striping, added a cache to increases performance over RAID3. **RAID5 killed off all three.**

RAID 5: Striped with distributed parity (3-5 drive minimum)
Description: A second drive failure during drive rebuild is fatal- need a hot spare.
Performance: No single parity disk bottleneck, but rebuilding drives will degrade performance. Balances data availability and read/write performance. During drive rebuilds, write performance suffers if cache isn't used.

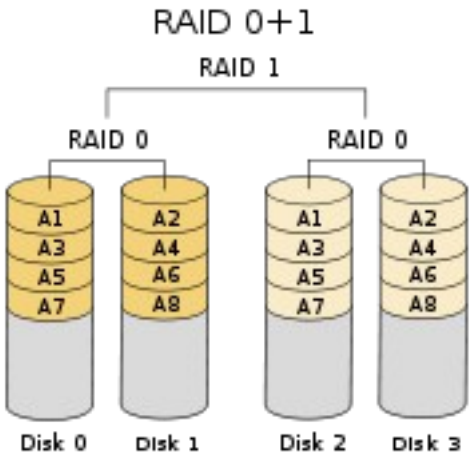


RAID 6: Dual parity
Description: RAID 5 with an additional drive to guard against a second drive failure during a drive rebuild. In the above example, it allows for 2 parity blocks per drive, instead of one.
Performance: Since each parity region is calculated separately, the RAID 5 performance impact is doubled. Some performance loss during multi-drive parity calculations and background drive rebuilds.



Nested RAID Levels

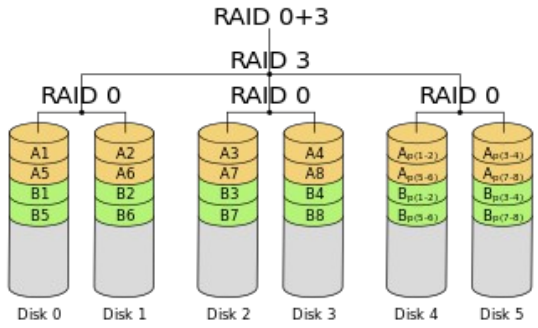
RAID01 (RAID 0+1): Mirror of Stripes - A RAID0 array is Mirrored



RAID01 uses mirror of stripes, achieving both replication and sharing of data between disks. Better performance than RAID1, and better redundancy than RAID0. Data is mirrored and then striped. If you lose a drive in a stripe set, all access to data must be from the other stripe set. Read operations are better because of striping, but write operations mirror the performance degradation of RAID 1.

The usable capacity of a RAID 01 array is the same as in a RAID 1 array made of the same drives, in which one half of the drives is used to mirror the other half. At least four disks are required in a standard RAID 01 configuration, but larger arrays are also used.

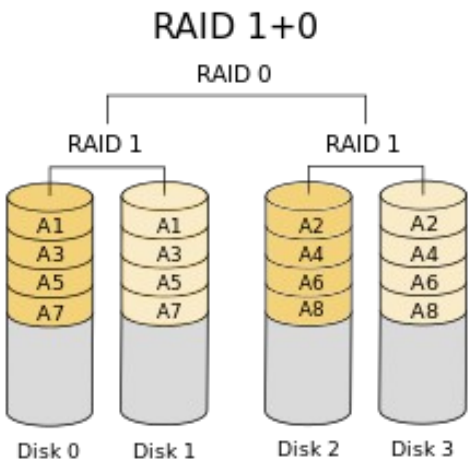
RAID03



RAID 03, also called **RAID 0+3** is byte-level striping with dedicated parity is used. Essentially, a RAID3 array is striped across RAID0 elements

RAID 53 is an accepted term for a series of RAID-5 arrays (striping with distributed parity) striped across a RAID-3 array. For all practical definition it is mostly the same thing, and it's benefits aren't important with other major types.

RAID 10 (RAID 1+0) - Stripe of Mirrors - a RAID0 array is striped across RAID1 elements

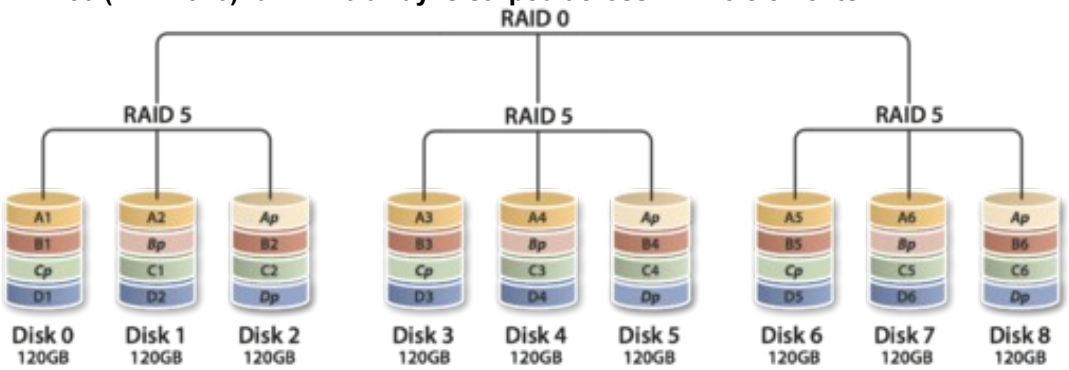


RAID10 is similar to RAID01 with an exception that two used standard RAID levels are layered in the opposite order; thus, RAID 10 is a stripe of mirrors (RAID0 array of RAID1s), which may be two- or three-way mirrors and requires a minimum of four drives.

A nonstandard Linux "RAID10" can be implemented with as few as two disks, and arrays of more than four disks are also possible.

RAID 10 provides better throughput and latency than all other RAID levels except RAID 0 (which wins in throughput). Thus, it is the preferable RAID level for I/O-intensive applications such as database, email, and web servers, as well as for any other use requiring high disk performance.

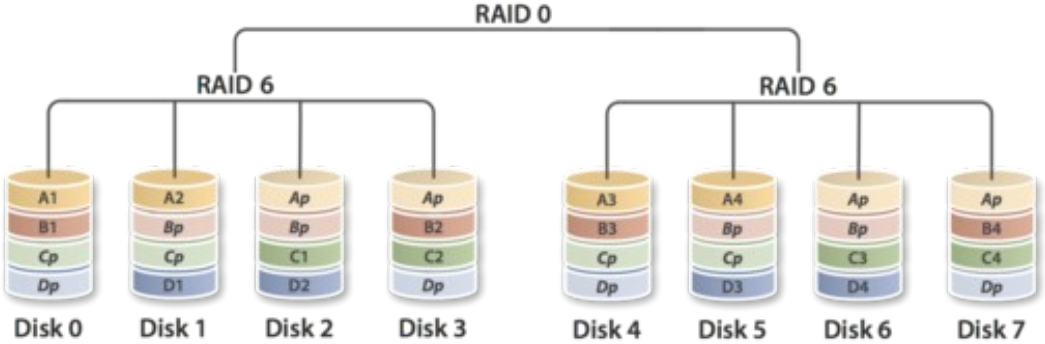
RAID 50 (RAID 5+0): a RAID0 array is striped across RAID5 elements



RAID50 aka **RAID 5+0**, combines the straight block-level striping of RAID0 with the distributed parity of RAID5. As a RAID0 array is striped across RAID5 elements, minimal RAID50 configuration requires six drives. Example shows collections of 120 GB RAID5s striped together to make 720 GB of total storage space.

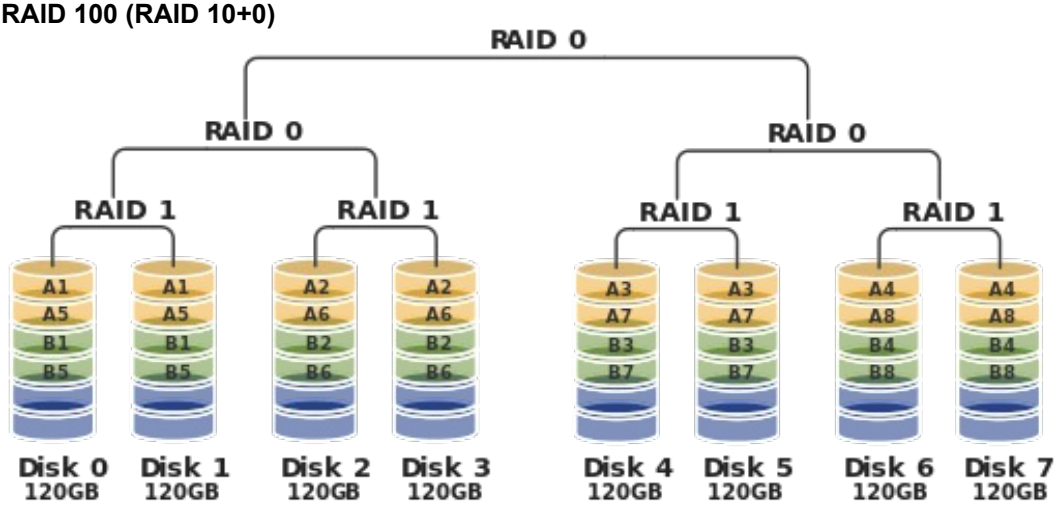
One drive from each of the RAID5 sets could fail without loss of data; for example, a RAID50 configuration including three RAID5 sets can only tolerate three maximum potential drive failures. There is still RAID5's inherent strain to rebuild a drive. As RAID5 was improved by RAID6, so was its nested counterpart, RAID 60

RAID 60 (RAID 6+0): a RAID0 array is striped across RAID6 elements



RAID 60, also called **RAID 6+0**, combines the straight block-level striping of RAID 0 with the distributed double parity of RAID 6, resulting in a RAID 0 array striped across RAID 6 elements. It requires at least eight disks.

RAID 100 (RAID 10+0)



RAID100, sometimes also called **RAID 10+0**, is a stripe of RAID10s. This is logically equivalent to a wider RAID10 array, but is generally implemented using software RAID0 over hardware RAID 10. Being "striped two ways" a RAID100 is described as a "plaid RAID"

NFS- Network File System
- Originally developed by Sun, access to remote file systems with mount points. NFSv4 listens on port 2049

	NFSv3	**NFSv4**
Packages	nfs-utils and nfs-utils-lib	nfs-common, nfs4-acl, rpc-svc-gss, for client. For server, use nfs-kernel-server and nfs-utils
Config files	/etc/sysconfig/nfs, /etc/exports	/etc/exports, /etc/nfs/, /etc/nfs/nfs.conf (server)
Services (SystemD & init)	/etc/init.d/nfs, /etc/init.d/rpcbind service nfs start, service rpcbind start	nfs-server.service and nfs-kernel-server.service, rpc-svc-gss.service
Permissions	/etc/exports file, some options given	ACLs and more options added in /etc/exports
RPC	rpc.lockd, rpc.statd (file locking, status info)	Now automated/ dynamic in NFS4 protocol
Automount	/etc/fstab/	/etc/fstab/ and autofs

NFSv3 Common permission options in the /etc/exports file
Self-explanitory: read-write (rw), read-only (ro), and no access (-)

root_squash	Forces all NFS requests from the root user (UID 0) on the client machine to be mapped to the anonymous user (usually nfsnobody) on the server, to help prevent accidental or malicious modifications by the root user on the client. [also "chown nfsnobody /share" for shared folder]
no_root_squash	Provides access with full permissions (if granted in the ACL). Use this option with caution due to potential security risks.
all_squash	A more aggressive version of root_squash that maps all client UIDs to the anonymous user on the server, regardless of the client user (has significant security implications).
anonuid/anongid	Allow you to specify the UID and GID of the anonymous user on the server used for mapping client users when root_squash or all_squash is enabled.
sync/ async	These have the same functionality as used with the mount command

You can specify access rights for specific hosts or networks. Wildcards can be used to represent groups of hosts or networks in the access control list. you can also use CIDR notation like 192.168.1.0/24

NFSv4-introduced new features and functionalities
Added options for overriding inheritance and setting specific permissions for individual directories in /etc/exports
- Fake root mount: if a server exporting /home and /data, instead of mounting both, just mount /
- Allows for clients to send their UID/GID info for access control decisions
- Access Control Lists (ACLs) on exported directories- the nfs4-acl package provides extensions for setfacl and getfacl to better support NFS, for more granular access permissions for specific users and groups on the server.
- Security options provide more granular authentication and authorization mechanisms like using a Kerberos server. An older option NLM server (Network Lock Manager) is supported but outdated, and NFS delegation tickets (built into NFS) has been proved to have vulnerabilities, thus Kerberos solutions seem to remain as the best security solution. Kerberos options are krb5p, krb5i, and none. Use krb5p for Kerberos security with encryption for data privacy and integrity protection. The krb5pi uses Kerberos but without data encryption. None is obviously not recommended. There was also a systemd sevrvice called nfs-secure-server.service but it integrated into nfs-server.service

Some sample entries in the NFSv4 /etc/exports file
```
/data *(rw,sync,all_squash)
        # Export /data directory with read-write access for all (only all_squash for basic security- this is weak and not advised)
/directory_to_export -sec=krb5p,rw,root_squash,sync  # Consider using ACLs instead of root_squash
        # Export with Kerberos, read-write access, root squashing, and sync
/home/users (rw,sync,nfsv4,user_acl,sec=krb5p)  # Preferred approach
        # Export /home/users with user ACLs, NFSv4, and Kerberos security
```

Commands:
Use exportfs to manage NFS exports. Options include -v to list currently exported directories; -a to export all entries in /etc/exports. To add new exports use 'exportfs -o options /dir client_IP' and to remove exports use 'exportfs -u /directory'
The command 'showmount -e' gives info on NFS shares currently exported; rpcinfo nfs gives info about the NFS server's RPC, and nfsstat gives statistics related to NFS server activity.

Client-side: mount, showmount -e server_IP (show NFS shares on a server), and nfsstat for NFS client statistics

The autofs package:
autofs.service: The systemd service that manages the autofs daemon itself. (start, stop, and restart)
automount: Command for manual interaction with autofs, manually mounting or unmounting specific automount points or debugging automount behavior (viewing logs or checking status of automount points)

/etc/auto.master: main configuration file - defines mount points, links to their map files, global options for autofs
Each line in the file typically follows this format: <mount_point> <map_file> [<options>]
/etc/auto.<identifier>: Map files, referenced by the master map file, contain details on individual automount points.
Each map file defines a specific automount configuration for a particular mount point. The format depends on the chosen map type:
amd.map format: Traditional format that specifies the NFS server location, automount behavior, access control.
auto.master.d format: Newer format that allows for inheritance and modular configuration based on directories.

NFSv3 Mount Options:
rsize/ wsize (read/write size): Set maximum packet size per request the client will try to read or write from the NFS server. Increasing can improve performance for large file reads, but values too large might lead to fragmentation and inefficient network usage.

bg/fg (background/foreground): background allows the system to continue booting while the NFS mount is being established, while foreground forces the mount command to wait until the NFS mount is successfully completed before returning.

async/ sync: The default, asynch allows the client to acknowledge write requests to the server before data is physically written to disk. Can improve performance but might lead to data loss if the client crashes before the write is completed. Using sync ensures that all data written to the NFS share is flushed to the server's disk before the mount command returns. This can improve data integrity but can also impact performance.

noauto: prevents the system from automounting the NFS share during boot

_netdev: Tells system to wait for the network interface to be configured before attempting the NFS mount

tcp/udp: NFSv3 typically defaults to TCP, but UDP can be used in specific scenarios (like low-latency networks) with trade-offs in reliability.

NFSv4 Mount Options:
Building on NFSv3 options, NFSv4 introduces additional options related to security flavors and performance optimizations:

sec: krb5p/ krb5i/ none: Use krb5p for Kerberos security with encryption for data privacy and integrity protection. The krb5pi uses Kerberos but without data encryption. None is obviously not recommended. Kerberos needs to be separately configured on the client and server. rpc_sec is a remnant of NFSv2 and no longer relevant.

nfsvers (NFS version): While the system might negotiate the NFS version with the server, you can explicitly specify nfsvers=4 to force an NFSv4 mount.

minor_version (minor NFS version): This option allows specifying a specific minor version of NFSv4 if your server supports multiple versions.

readdirplus: This option enables the client to request additional information along with directory listings, potentially improving performance for browsing directories on the NFS share.

Many NFSv3 mount options like rsize, wsize, sync/async, noauto, background/foreground and _netdev are still relevant for performance tuning and basic mount behavior in NFSv4.

Kerberos setup

Log in to the kerberos server	ssh <username>@<kerberos_server_ip>
Launch kadmin.local to enter CLI	sudo kadmin.local
Add the service principal	addprinc -randkey nfs/<nfs_server_hostname>
Create a keytab file for NFS Server	ktadd -k /etc/krb5.keytab nfs/<nfs_hostname> (hit return, type quit to exit CLI)
Copy file to NFS Server	scp /etc/krb5.keytab <username>@<nfs_server_ip>:/etc/krb5.keytab (hit return, then exit ssh)
Set write permissions for keytab file	sudo chmod 400 /etc/krb5.keytab; sudo chown root:root /etc/krb5.keytab
Be sure /etc/exports file has shares fixed	sec=krb5p or similar, as demonstrated before
Edit /etc/sysconfig/nfs (enable GSS/Kerb)	sudo vi /etc/sysconfig/nfs Add or uncomment lines RPCGSSDARGS="" and RPCSVCGSSDARGS=""
Edit /etc/krb5.conf to check config	sudo vi /etc/krb5.conf - Ensure [realms], [domain_realm] are configured
Enable, start NFS and kerberos	sudo systemctl enable nfs-server rpcbind && sudo systemctl start nfs-server rpcbind && sudo systemctl restart nfs-server
Run on client to verify it's working	sudo mount -t nfs -o sec=krb5p $server_hostname:/$share_name /mnt/nfs

firewalld configuration: sudo firewall-cmd --permanent --add-service=nfs
iptables configuration: sudo iptables -I INPUT -p tcp --dport 2049,111 -j ACCEPT
List SELinux stuff with semanage boolean -l | grep -i '(nfs_)'; semanage fcontext -l | grep -i '(nfs_)'

Samba - SMB

Client executables (install package: samba-client)

- smbcontrol — Manage Samba shares, view connections to servers), and perform client administrative tasks
- smbclient — Browse, copy, manage files/directories on remote Samba servers.
- smbmount — Mount remote Samba shares as local directories.
- nmblookup — Perform NetBIOS name resolution (find Samba servers on network).
- smbcacls — Remote management tool- view and modify Windows ACLs on files hosted on a Samba server.

SystemD services on the client

- smbd.service — Main SMB/CIFS daemon, handles file/print services.
- nmbd.service — Provides NetBIOS name service.
- cifs.service — Mounts CIFS/SMB shares

Server executables (install package: samba)

- samba-tool — For managing configuration, users, shares, Kerberos settings, and other administrative tasks
- smbstatus — Shows the current status of the Samba server, including active connections and shares.
- testparm — Verifies the syntax of your /etc/samba/smb.conf file before restarting the service.
- smbpasswd — Changes Samba user account passwords.

SystemD services on the server (also includes the client list)

- smbd.service — Main SMB/CIFS daemon, providing access to clients, handles file/print services.
- nmbd.service — Provides NetBIOS name service.
- samba.service — Alias for smbd.service.
- winbind.service — Allows Windows domain authentication.
- samba-ad-dc.service — Samba Active Directory Domain Controller service

Main config file is /etc/samba/smb.conf (resources, user authentication, security, etc.)
Optional configs: /etc/samba/smb-security.conf (security settings) or /etc/samba/smb.secrets (for sensitive info)

/etc/samba/smb.conf - specify shared directories or files; access permissions for users and/or groups; LDAP and encryption options, and security settings

- Shares: Define shared directories or files using the [sharename] section.
- Permissions: read only = yes, writeable = yes, and specific user/group entries.
- LDAP Integration: security = ads option and specifying LDAP server details.
- Security Settings: Enhance security with options like:
  - encrypt passwords = yes - Encrypts user passwords during storage.
  - map to guest = bad user - Disables guest access.
  - browsable = no - Hides the share from browse lists.
  - valid users = @users - Restricts access to the local usernames or users in group specified.

Example smb.conf:

```
[Global]
  workgroup = MYWORKGROUP  # Name of your workgroup for network browsing
  server string = My Samba Server  # Descriptive name for your Samba server
  security = ads  # Enable LDAP security for user authentication
  encrypt passwords = yes  # Encrypt user passwords for added security
  map to guest = bad user # No access for the unauthenticated- gives guest access as 'bad user' which doesn't exist (denied!)
  # wins server = 10.0.0.1  # WINS server IP for name resolution, logging, and specifying one interface to listen (not all)
  logging = file  # Enable file-based logging
  log level = warn  # Log warnings and more severe messages
  log file = /var/log/samba/smb.log  # Specify the log file location
  # interfaces = 192.168.1.0/24  # Example: Listen only on the 192.168.1.0/24 subnet
[SharedFolder]
  path = /home/share  # Path to the directory you want to share
  browsable = no  # Hide this share from network browsing
  writable = yes  # Allow users with access to modify files
  read only = no  # Allow both reading and writing
  valid users = @share_access  # Grant access only to users in the "share_access" group
  create mask = 0664  # Example: New files get rw-rw---- permissions (user:group:others)
  directory mask = 0775  # Example: New directories get rwxrwxr-x permissions
[AnotherShare]
  path = /var/www/html  # Path to the web server's document root (example)
  read only = yes  # Allow users to only read files in this share
  valid users = user1 user2 @web_admins  # Grant access to specific users and a group
  # locking = ...  # Options for controlling how multiple clients share access to files
  # oplocks = ...  # Options related to optimistic locking (advanced)
  # write cache (or 'read cache') = yes  # Enable caching writes/ reads for faster performance
```

The "map to guest = bad user" matches any user that fails authentication (denied)

Kerberos implementation

Log in to the Kerberos server	ssh <username>@<kerberos_server_ip>
Launch kadmin.local to enter CLI	sudo kadmin.local
Add the service principal	addprinc -randkey smb/<smb_server_hostname>
Create a keytab file for SMB server	ktadd -k /etc/krb5.keytab smb/<smb_hostname> (hit return, type quit to exit CLI)
Copy file to SMB Server	scp /etc/krb5.keytab <username>@<smb_server_ip>:/etc/krb5.keytab (hit return, then exit ssh)
Set write permissions for keytab file	sudo chmod 400 /etc/krb5.keytab; sudo chown root:root /etc/krb5.keytab
Edit /etc/krb5.conf to check config	sudo vi /etc/krb5.conf - Ensure [realms], [domain_realm] are configured
Ensure /etc/samba/krb5users is ready	The file will need entries similar to example provided below
Enable, start SMB and kerberos	sudo systemctl enable smb rpcbind && sudo systemctl restart smb
Run on client to verify it's working	sudo mount -t cifs -o sec=krb5i,username=<client_username>@<REALM> //$server_hostname/$share_name /mnt/smbshare

Create a user mapping file to translate Kerberos principals (username@REALM) to specific Samba usernames

```
# /etc/samba/krb5users
# Map Kerberos principal "user1@MYREALM" to Samba user "samba_user1"
user1@MYREALM = samba_user1   # samba_user1 would be the username on the Linux host
```

```
[Global]
  workgroup = MYWORKGROUP
  server string = My Samba Server
  security = krb5i  # Enable Kerberos integration with user mapping
  encrypt passwords = yes
  map to guest = bad user
  username map = /etc/samba/krb5users  # Specify the user mapping file location
[SharedFolder]
  path = /home/share
  browsable = no
  writable = yes
  read only = no
  # Allow access only to users mapped in the user mapping file
#  valid users = %    using % means all Kerberos users logged in!
  valid users =  samba_user1
```

More security stuff:

The file /etc/samba/smb.secrets stores encrypted passwords (machine, keys). Use alternative approaches within /etc/samba/smb.conf first. If using Kerberos keytabs, this file can be emptied.

firewalld configuration: sudo firewall-cmd --permanent --add-service=samba
iptables configuration: sudo iptables -I INPUT -p tcp --dport 137-139,445 -j ACCEPT
List SELinux stuff with semanage fcontext -l | grep -i '(smb_|samba_)' semanage boolean -l | grep -i '(smb_|samba_)'

Extending SMB: The "Samba-VFS" framework

Is not a virtual file system as named. Leverage to enhance SMB server with modules, shared libraries (.so files- think ldd, not kernel libraries like .ko) and they can be declared in /etc/samba/smb.conf using the vfs objects parameter. Some modules can be chained, allowing multiple modules to work sequentially.

```
[global]
  vfs objects = full_audit  # Load the vfs_full_audit module-  detailed logs of file operations for enhanced security
  vfs_full_audit_log_dir = /var/log/samba/audit  # Specify log directory
  vfs_full_audit_log_file = full_audit.log    # Specify log filename
  vfs_full_audit_log_rotate = 5            # Rotate logs after 5 rotations
  vfs_full_audit_log_size = 10M              # Maximum log size (10 Megabytes)
```

```
[global]
  vfs objects = acl_tdb  #  Access control through storage of Access Control Lists (ACLs)
  vfs_acl_tdb_path = /etc/samba/acl.tdb     # Specify TDB database path.
```

vfs_recycle (recycle bin to recover deleted files); vfs_usershare (give users share definitions); vfs_fruit: (Apple File System (AFP) macOS client shares); vfs_fake_chroot (make a chroot environment for each connected user); vfs_deny_hosts (restrict access to shares based on IP or hostnames); vfs_cifs_xattr (enables storing extended attributes on Samba shares)

Red Hat Identity Management (IdM) - FreeIPA Identity Policy Audit

IdM server with integrated DNS using FreeIPA which incorporates Kerberos, LDAP, TLS CA, NTP, and BIND in one install.

Installation:

Begin with getting the needed packages:

sudo dnf install freeipa freeipa-server bind bind-utils bind-dyndb-ldap krb5-server krb5-libs chrony

Next, run the installer script. This provides integrated DNS which will be relied on by other FreeIPA components

sudo ipa-server-install --enable-dns (--enable-dns is not needed in RHEL8 and beyond)

During the run of the installer script, you will be asked for the domain name (realm), you'll need to set a Directory Manager password and a primary administrator password. You will also be asked for DNS settings like type (usually choose BIND, if it asks to), forwarders (8.8.8.8 is fine for non-enterprise or testing installs), and reverse DNS (use the in-addr.arpa reverse version of our primary IP address, i.e., 192.168.1.123 would be 123.1.168.192.in-addr.arpa). You can also expect to see some choices and setting for Kerberos integration, NTP , database type (often PostgreSQL), and LDAP.

Here is what we have after that completes:
Dogtag Certificate System: Certificate Authority & Registration Authority for certificate management
LDAP Server: Employs 389 Directory Server for user and group management.
MIT KDC: Kerberos Key Distribution Center is the basis for single sign-on
Apache: IdM administration functionalities need a built-in webserver, so there it is.
NTP (chrony in RHEL 8/9): Sets up the Network Time Protocol service
BIND: Integrates the BIND DNS server with the FreeIPA environment for DNS management
SSSD - client side component employing FreeIPA as authentication & identity provider superior to NSS & PAM.

FreeIPA SystemD Services:
FreeIPA Server: freeipa-server.service, ipactl.service, freeipa-healthcheck.service
Kerberos Key Distribution Center: krb5kdc.service
Kerberos DB Administration: kadmin.service
Directory Services (LDAP): slapd.service
DNS Server: named.service

Important Configuration Files:
The primary config file - /etc/ipa/default.conf

Server Settings:

realm (Required): Defines your FreeIPA realm name (e.g., EXAMPLE.COM).
server_principal: (Optional) Specifies name for FreeIPA server. Autogenerates as host/<hostname>@<realm>
server_cert: (Optional) Path to FreeIPA server certificate file
server_key: (Optional) Path to private key file associated with the server certificate
ca_cert: (Optional) Path to CA certificate used to sign the FreeIPA server certificate
offline: (Optional) True means disable communication with other FreeIPA servers (isolated deployments)

DNS Settings:

enable_dns: (Optional) Set to true to enable the integrated FreeIPA DNS server. Defaults to false
dns_forwarder: (Optional) Comma-separated list of IPs of DNS servers to forward unresolved queries to
dns_allow_update: (Optional) IPs or networks allowed to update DNS records. Default is 127.0.0.1
disable_anonymous_bind: Restrict anonymous BIND queries, improving security
forwarder_permit: Define IPs or networks allowed for DNS forwarding requests (prevents open relays)

Security Settings:

password_minimum_length: Set a minimum password length
password_require_mixed_case: Require mixed case (uppercase/ lowercase)
password_require_numeric: Require at least one number
password_require_special: Require at least one special character
user_enable_lockout: Enable user account lockout after failed login attempts
user_lockout_duration: Define the duration (minutes) a locked account remains inaccessible
allow_unsafe_kerberos_keytypes: Leaving this disabled prevents weak Kerberos encryption types
ca_cert_subject: Defines the subject info for a custom CA certificate
server_cert_subject: Defines the subject info for a custom FreeIPA server certificate
db_type: Specifies the database backend used by FreeIPA (defaults to postgresql)
allow_weak_password: Disabling this enforces strong passwords for IPA clients (keep false).

Debug and Logging:

debug_level: Sets the debug logging level (higher values provide more detailed logs).
log_file: Path to the log file for FreeIPA server events.

[For very specialized configs, an optional /etc/ipa/server.conf can be used for server-specific overrides. it would be read first but is seldom needed and this is simply a footnote to it "being a thing"]

Enforcing standardized user authentication with /etc/ipa/userauth.conf

A general idea of entiries in a /etc/ipa/userauth.conf file. Security management could mandate this be used to emphasize and/or standardize password policy and security configurations (sometimes simple alternates to kerberos). The details for each module is a little out of scope for this writing, but module docs would have specifics and actual items to replace what's below. Many configuration options will be in a module's config file.

Options in /etc/ipa/default.conf can also be over-ridden here, and there are some new ones:
password_history_depth: Define the number of previous passwords a user cannot reuse.
user_enable_lockout: Enable account lockout after a certain number of failed login attempts.
user_lockout_duration: Define the duration (minutes) an account remains locked after failed login attempts.

```
[Service: sudo]      #  Enable RADIUS authentication for sudo service
    authtype = radius
    server = radius.example.com  # Replace with actual RADIUS server address
    shared_secret = (secret)     # Replace with actual shared secret (not recommended in plain text)
    port = 1812                  # Default RADIUS port
    #   timeout = 3                # RADIUS authentication timeout (seconds)
    #   nas_port_type = 5          # Network Access Server (NAS) port type

[Service: vpn]      #  Enable LDAP authentication for a custom VPN service
    authtype = ldap
    server = ldap.example.com    # Replace with actual LDAP server address
    basedn = dc=example,dc=com   # Replace with appropriate base DN for user search
    binddn = cn=FreeIPA_Bind_User,ou=Service Accounts,dc=example,dc=com  # Replace with bind DN
    bind_password = (secret)     # Replace with actual bind password (not recommended in plain text)
    #   search_scope = subtree       # LDAP search scope (base, onelevel, subtree)
    #   tls_cacertfile = /etc/ipa/certs/ca.crt  # Path to CA certificate for LDAP TLS

[Service: shell]      #  PAM for shell logins
    auth         pam_ServiceName.so

[Service: secureapp]      #  Enable token-based authentication for a specific application (hypothetical)
    authtype = token  # Assuming a token-based authentication module is installed

[Service: shell]      #  Disable alternative authentication for shell logins (only use Kerberos)
    alternative_authentication = false

[Service: console]      #  PIN login module.  Allows users to log in using a PIN instead of a password
    authtype = pin  # Assuming the PIN login module is installed
    #   pin_retries = 3  # Maximum allowed PIN attempts before lockout
    #   pin_length = 6   # Minimum PIN length

[Service: ssh]      #  2FA/OTP (Example: Google Authenticator- others include RSA SecurID, Duo Security, etc.
    require_otp = true  # Enforces OTP for SSH logins
    #  require_mfa = true  # Enforces MFA for SSH logins

[Service: myapp]      #  Social login module (hypothetical - Facebook for a custom web application):
    authtype = social  # Assuming a social login module is installed
    #   provider = facebook  # Specify Facebook as the social login provider
    #   client_id = your_facebook_app_client_id  # Replace with your Facebook App details
    #   client_secret = (secret)  # Replace with your Facebook App secret (avoid plain text)

[Service: sudo]      #  Certificate-based auth module using PKI for sudo service (Example: freeipa-certlogin):
    authtype = cert  # Assuming the freeipa-certlogin module is installed
    #   ca_certfile = /etc/ipa/certs/ca.crt  # Path to the Certificate Authority certificate
    #   require_crl_check = true  # Enforce Certificate Revocation List (CRL) checking

[Service: shell]      #  External database auth module (example: ipa_ldap_sync-  LDAP for shell logins):
    # Users are authenticated against FreeIPA, but user data is synchronized from LDAP server
    uri = ldaps://ldap.example.com:636
```

The IPA commands for user and resource management
ipa <category> <subcommand> [options] [arguments]
<category> is for example user, group, host, etc.) <subcommand> is the action (e.g., add, delete, show, etc.)
Most of categories typically have the subcommands add, delete, modify, show, show all (or list), and find

ipa config: Manage FreeIPA server configuration files
ipa package: Manage FreeIPA packages (installation, updates)
ipa profile: Manage FreeIPA server profiles (configurations)
ipa server: Manage the FreeIPA server itself (installation, configuration)
ipa vpnconfig: Manage VPN configuration options within FreeIPA
ipa trust: Manage trust relationships (e.g., with Active Directory)
ipa host: Manage FreeIPA hosts (machines joining the identity domain)
ipa hostgroup: Manage groups specifically for FreeIPA hosts (machines)
ipa interface: Manage network interfaces on the FreeIPA server
ipa nfsserver: Manage FreeIPA's NFS server configuration
ipa service: Manage FreeIPA services (applications requiring identity management)
ipa join: Joins a machine to a FreeIPA domain without using the client installation command.
ipa domain: Manage FreeIPA domains (logical groupings of identities)
ipa fqdn: Manage Fully Qualified Domain Names (FQDNs) associated with FreeIPA
ipa domaindns: Manage DNS domains integrated with FreeIPA
ipa dnskey: Manage DNS keys used for DNS signing (important for DNSSEC)
ipa dbbackup: Manage database backups and restores
ipa dnstable: Manage FreeIPA's internal data tables (use with caution)
ipa restore/backup: Create or load a backup of an IPA config into FreeIPA
ipa sync: Synchronize data with external directory services
ipa topology: Manage FreeIPA's server topology (replica management)
ipa vault: Manage FreeIPA vaults (secure storage for secrets)
ipa ca: Manage Certificate Authority operations (for internal PKI)
ipa cert: Manage certificates used by FreeIPA (server TLS, user certificates)
ipa tls: Manage Transport Layer Security (TLS) certificates
ipa kerberos: Manage Kerberos tickets and keytabs
ipa servicedelegationrule: Manage service delegation rules (allow services to request Kerberos tickets)
ipa servicedelegationtarget: Manage service delegation targets (used with service delegation rules)
ipa realmdomains: Manage realm domains used for Kerberos authentication
ipa diagnose: Perform diagnostic operations on the FreeIPA server
ipa monitor: Monitor the FreeIPA server's health and status
ipa find: Search for users, groups, hosts, and other FreeIPA objects
ipa user: Manage FreeIPA users - ipa userpolicy: Manage user password policies - ipa group: Manage groups
ipa passwd: Reset or change passwords for FreeIPA users and services
ipa pwpolicy (alias for userpolicy): Manage password policies (password complexity)
ipa shadow: Manage shadow password information (use with caution)
ipa permission: Manage individual permissions assigned to users or groups
ipa rightsource: Manage rights sources used for access control
ipa role: Manage FreeIPA roles (sets of permissions)
ipa relation: Manage relationships between FreeIPA objects (e.g., user-group membership)
ipa selinuxusermap: Manage SELinux user maps
ipa sshkey: Manage SSH keys for FreeIPA users and services
ipa hbacrule: For Host-Based Access Control (HBAC) - ipa hbacsvc (services) - ipa hbacsvcgroup (groups)
ipa sudocmd: Manage commands usable w/ sudo - ipa sudocmdgroup: for sudo command groups
ipa idrange: Manage ID ranges (used for ID mapping)
ipa locale: Manage locales used within the FreeIPA server

For details, you can use "ipa help <category>" for any of these.

Open ports for FreeIPA functionality:
TCP ports: 80, 443 (HTTP/S for web interface), 389, 636 (LDAP/S), 53 (DNS), 88 (Kerberos for Windows clients)
UDP ports: 88 (Kerberos), 53 (DNS), 67 and 68 (DHCP)
RPC and rstatd use random port numbers. Unless you have multiple FreeIPA servers or modules that need it, you are probably safe to not worry about opening ports- addressing this issue is outside the scope of this writing.

Important files and directories
/etc/ipa: This directory contains configuration files for FreeIPA.
/etc/ipa/client.conf: optional- for FreeIPA client on the server itself. Has location of the server and realm info
/etc/ipa/userauth.conf (Optional) - Defines authentication backends and policies for user login.
/etc/ipa/authpolicy.conf (Optional) - Configures authentication policy for FreeIPA services.
/etc/ipa/db.conf - If you have database configuration info outside of default.conf they would go in this

/var/lib/ipa: This directory contains data files for FreeIPA, including LDAP databases and Kerberos keytabs.
/etc/krb5.conf: Config for the Kerberos client, with location of Kerberos keytabs and realm information.
/etc/pki/pki-tomcat: Directory for the Dogtag CA Certificate Authority service
/etc/pki/pki-tomcat/alias: Contains the certificate database used by Dogtag.
/etc/ipa/certs/: Holds FreeIPA server user and service certificates, private keys
/root/cacert.p12: Admin access certificate - default name, PKCS#12 for Public Key Cryptography Stds #12
/etc/ipa/ca.crt: The CA certificate file used by clients to verify the FreeIPA server's identity.
/etc/ipa/nssdb: Contains the NSS (Network Security Services) database used for storing certificates and keys.
/var/lib/freeipa/dns/: Holds zone files managed by FreeIPA's internal DNS server. Each domain/subdomain gets a zone file.
The TLD zone file is the primary, containing records for users, computers, and other services.
FreeIPA manages user/group information and dictates DNS records while BIND takes directions

/var/log/ipa: This directory contains log files related to FreeIPA operations.
/var/log/ipa-server-install.log: Log file for FreeIPA server installation.
/var/log/ipa-client-install.log: Log file for FreeIPA client installation.

Applying basic system security mechanisms:

Hardening FreeIPA with SELinux
Install the package: dnf install policycoreutils-selinux-freeipa
SELinux types and contexts:
- ipa_var_lib_t: Files under /var/lib/ipa
- ipa_var_run_t: Files under /var/run/ipa
- ipa_log_t: Logs under /var/log/ipa
- ipa_tmp_t: Temporary files
- ipa_exec_t: Executable files

Applying contexts and set booleans:

Item	Context to apply	Save what was applied
/var/lib/ipa	semanage fcontext -a -t ipa_var_lib_t "/var/lib/ipa(/.*)?"	restorecon -Rv /var/lib/ipa
/var/run/ipa	semanage fcontext -a -t ipa_var_run_t "/var/run/ipa(/.*)?"	restorecon -Rv /var/run/ipa
/var/log/ipa	semanage fcontext -a -t ipa_log_t "/var/log/ipa(/.*)?"	restorecon -Rv /var/log/ipa
/tmp/ipa	semanage fcontext -a -t ipa_tmp_t "/tmp/ipa(/.*)?"	restorecon -Rv /tmp/ipa
Executables	semanage fcontext -a -t ipa_exec_t "/usr/libexec/ipa(/.*)?"	restorecon -Rv /usr/libexec/ipa
Allow LDAP over SSL (boolean)	setsebool -P allow_ipa_ldap_ssl 1	

Verifying SELinux contexts and booleans:
ls -Z /var/lib/ipa && getsebool -a | grep ipa

To identify and resolve denials:
grep "denied" /var/log/audit/audit.log | audit2allow -M mypol
semodule -i mypol.pp

Example configuration for firewalld:
```
firewall-cmd --zone=internal --add-source=10.0.10.0/24 --permanent
firewall-cmd --zone=internal --add-source=172.16.20.0/24 --permanent  # Add 2nd subnet to internal zone
firewall-cmd --zone=internal --add-port=443/tcp --permanent  # HTTPS
firewall-cmd --zone=internal --add-port=80/tcp --permanent    # Optional for web interface
firewall-cmd --zone=internal --add-port=389/tcp --permanent  # LDAP
firewall-cmd --zone=internal --add-port=636/tcp --permanent  # LDAPS
firewall-cmd --zone=internal --add-port=88/udp --permanent  # Kerberos
firewall-cmd --zone=internal --add-port=88/tcp --permanent  # Kerberos (optional for Windows clients)
firewall-cmd --zone=internal --add-port=53/udp --permanent  # DNS
firewall-cmd --zone=internal --add-port=53/tcp --permanent  # DNS
firewall-cmd --zone=internal --add-port=67/udp --permanent  # DHCP server broadcasts
firewall-cmd --zone=internal --add-port=68/udp --permanent  # DHCP clients leases
firewall-cmd --permanent --default-zone=internal    # Set internal zone as default
firewall-cmd --reload    # Reload firewall configuration
```

Example configuration for Iptables:

```
# Chain for internal subnet 1 traffic
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -p tcp --dport 80 -j ACCEPT  # for web interface
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -p tcp --dport 389 -j ACCEPT
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -p tcp --dport 636 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -s 10.0.10.0/24 --dport 88 -j ACCEPT  # Kerberos
iptables -A INPUT -i eth0 -p tcp -s 10.0.10.0/24 --dport 88 -j ACCEPT  # Kerberos (optional for Windows clients)
iptables -A INPUT -i eth0 -p udp -s 10.0.10.0/24 --dport 53 -j ACCEPT  # for DNS
iptables -A INPUT -i eth0 -p tcp -s 10.0.10.0/24 --dport 53 -j ACCEPT  # for DNS
iptables -A INPUT -i eth0 -p udp -s 10.0.10.0/24 --dport 67 -j ACCEPT   # see DHCP server broadcasts
iptables -A OUTPUT -i eth0 -p udp -s 10.0.10.0/24 --sport 67 --dport 68 -j ACCEPT  # DHCP leases, broadcasts
# Allow established connections for subnet 1
iptables -A INPUT -i eth0 -s 10.0.10.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -d 10.0.10.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT

#  Chain for internal subnet 2 traffic
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -p tcp --dport 443 -j ACCEPT
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -p tcp --dport 80 -j ACCEPT  # for web interface
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -p tcp --dport 389 -j ACCEPT
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -p tcp --dport 636 -j ACCEPT
iptables -A INPUT -i eth0 -p udp -s 172.16.20.0/24 --dport 88 -j ACCEPT  # Kerberos
iptables -A INPUT -i eth0 -p tcp -s 172.16.20.0/24 --dport 88 -j ACCEPT  # Kerberos (for Windows clients)
iptables -A INPUT -i eth0 -p udp -s 172.16.20.0/24 --dport 53 -j ACCEPT  # for DNS
iptables -A INPUT -i eth0 -p tcp -s 172.16.20.0/24 --dport 53 -j ACCEPT  # for DNS
iptables -A INPUT -i eth0 -p udp -s 172.16.20.0/24 --dport 67 -j ACCEPT   # see DHCP server broadcasts
iptables -A OUTPUT -i eth0 -p udp -s 172.16.20.0/24 --sport 67 --dport 68 -j ACCEPT  # send DHCP leases, BC
# Allow established connections for subnet 2
iptables -A INPUT -i eth0 -s 172.16.20.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -d 172.16.20.0/24 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Post-installation checklist:

- Lock down the FreeIPA system with the options provided in the hardening section
- Create admin accounts with strong passwords for managing the FreeIPA domain. Organize them into groups
- Set up other users and groups to maintain consistency and organization within your domain.
- Define settings like lifetime for certificates issued by your CA if you're using an internal PKI for authentication.
- Issue server certificates for the FreeIPA server and others for secure communication within the domain.
- If needed, integrate FreeIPA with your existing DNS infrastructure for automatic DNS record management.
- Define Kerberos realm settings within FreeIPA if you plan to use Kerberos for authentication.
- Enroll machines (clients and servers) into the FreeIPA domain using the ipa join command.
- Install and configure FreeIPA client software on domain members
- Implement monitoring/logging solutions for server activity, identify potential issues, and ensure secure operation.
- Establish a regular backup and restore strategy for server configuration and data for disaster recovery
- Create user documentation explaining logging in, password resets, and accessing resources.
- Verify that users and groups are created and configured correctly within the FreeIPA domain.
- If using Kerberos, test user authentication using Kerberos tickets to ensure functionality.
- Verify that client software on domain members is functioning correctly and users can access resources

Installing Standalone Kerberos Server (no FreeIPA)

sudo dnf install krb5-server (client is krb5-workstation, krb5-libs, krb5-user)

/etc/krb5.conf: Main configuration file; defines Kerberos realm, KDC locations, encryption types, etc.
/etc/krb5/kdc.conf: Configuration for the KDC (server-side) if you're setting up a Kerberos server.
/etc/krb5/login.conf: Defines how Kerberos is used for authentication (login).
/var/lib/krb5/krb5.keytab: Stores the master Kerberos key for the KDC.

Client-side commands:
kinit <principal> - For client machine users to get a ticket to access a Kerberos-protected service.
klist -f - Lists all available Kerberos tickets held by the user, for verifying and seeing lifetime, -f gives more info
kdestroy - Destroys a specific Kerberos ticket. For logging out of a service or freeing up resources.
Server-side commands:
kdb5_util - Manages the Kerberos database, keytabs, and principals
kadmin.local - Manages Kerberos principals and credentials: creating, modifying user accounts, resetting passwords, and managing keytabs used by the KDC.

Mentionable related commands/ items:
keyutils - general-purpose tool for managing keyrings and keys, manage Kerberos keytabs alongside other key management tasks. (it itself doesn't interact directly with the Kerberos database)
sshd_krb5_module: This isn't a standalone command, but rather a module used by the SSH daemon to enable Kerberos authentication for SSH connections. You can configure it through SSH configuration files.

Systemd Services:
krb5-kdc.service (server-side): Manages the KDC daemon.
krb5-kadmind.service (server-side): Manages the Kerberos administration daemon.

TCP/88 (default): messages between clients and KDC. TCP is more secure but Windows clients may need UDP

Important Configurations:
Realm: Unique identifier for your Kerberos domain (e.g., EXAMPLE.COM).
KDC Locations: Specify the hostname or IP address of your KDC servers.
Default Encryption Type: Choose an appropriate encryption type (e.g., aes256-cts).
Ticket Lifetime: Set the expiration time for Kerberos tickets.
Client Principal: Define the principal name for your client machine (e.g., host/hostname`).

Managing the Kerberos database, keytabs, and principals with kdb5_util

Create and initialize database and set master password	kdb5_util create -r <realm> -s <keytab_file> -P <passwd>
Create new principal in <realm> with new <password>	kdb5_util addprinc -r <realm> -p <password> <principal>
Modifiy existing principal's attributes (e.g., password, flags)	kdb5_util modifyprinc -r <realm> <principal>
Removes a principal from DB	kdb5_util deleteprinc -r <realm> <principal>
Lists all principals in <realm> with key versions (-kv)	kdb5_util listprinc -r <realm> -kv
Create Keytab	kdb5_util create -r <realm> -s <keytab_file>
Add Entries to Keytab	kdb5_util addprinc -r <realm> -p <passwd> -t <keytab_file> <principal>
Merge Keytabs	kdb5_util merge -s <target_ktab> <source_ktab1> <source_ktab2> ...
Dump Database (can expose sensitive information)	kdb5_util dump -r <realm> -f <output_file>
Verify integrity of the Kerberos database	kdb5_util verify -r <realm>

Manage Kerberos with kadmin.local
Running the command kadmin.local alone will drop you into it's own CLI

Create a new principal for KDC administration:
addprinc -randkey kdc_admin@EXAMPLE.COM
The -randkey option is to generate a random password; kdc_admin@EXAMPLE.COM to name the principal and EXAMPLE.COM representing the Kerberos realm name.

Exit kadmin.local by entering quit.

Grant the kdc_admin principal the permissions to manage the KDC:
kadmin.local -p krb5/admin@EXAMPLE.COM ktadd -k /etc/krb5.keytab kdc_admin@EXAMPLE.COM

The first part "-p krb5/admin@EXAMPLE.COM" provides the password for the krb5/admin principal (usually the root principal) that has full administrative privileges in the Kerberos database.
The second part "ktadd..." adds the key for the kdc_admin principal to the specified keytab file (/etc/krb5.keytab)

Restrict access to kadmin.local using the /etc/sudoers file:
Run "nano /etc/sudoers" and add a block like this:

```
# Allow users in the 'kdc_admin' group to run kadmin.local as kdc_admin@EXAMPLE.COM
%kdc_admin ALL = NOPASSWD: /usr/sbin/kadmin.local -p kdc_admin@EXAMPLE.COM
```

"%kdc_admin" sets the rule applies to users in the kdc_admin group (create it using "groupadd kdc_admin")
"ALL = NOPASSWD" allows group members to run kadmin.local without a password, but only when using the kdc_admin@EXAMPLE.COM principal using the -p option.
"/usr/sbin/kadmin.local -p ..." simply specifies the command with sudo privileges.

Verification:
Create a user account that belongs to the kdc_admin group you created, og in as the newly created user.
Run "sudo kadmin.local -p kdc_admin@EXAMPLE.COM"
You should be prompted for the password of the kdc_admin principal (the one generated in step 2). If successful, you'll enter kadmin.local mode impersonating the kdc_admin principal.

Other Kadmin commands

addprinc <principal>	Adds a new principal (user or service account) to the database
delprinc <principal>	Deletes a principal from the Kerberos database
modprinc <principal>	Modifies attributes of an existing principal
rename_principal <old> <new>	Renames an existing principal in the Kerberos database
change_password <principal>	Changes the password of an existing principal
cpw <principal>	Alias for change_password
listprincs	Lists all principals in the Kerberos database
getprinc <principal>	Retrieves and displays information about a specified principal
ktadd -k <keytab_file> <principal>	Adds a principal's key to a keytab file (for passwordless authentication)
ktremove -k <keytab_file> <principal>	Removes a principal's key from a keytab file
ktdestroy -k <keytab_file>	Destroys a keytab file (use with caution)
getprivs	Shows administrative privileges of current user for kadmin.local CLI
listpols	Lists all policies in database (password rules, ticket lifetimes, etc.)
addpol <policy>	Adds a new policy to the Kerberos database.
modpol <policy>	Modifies attributes of an existing policy.
delpol <policy>	Deletes a policy from the Kerberos database.
getpol <policy>	Retrieves and displays information about a specified policy.
purgekeys <principal>	Removes all keys for a principal that are not the most recent.

SELinux Booleans

allow_httpd_pkey_init	Needed if using HTTP for key distribution.
allow_kadmind_port	Access on TCP 464 for administrative access to the KDC.
allow_kerberos_dce	Needed to support DCE clients using Kerberos.
allow_kerberos_kdc_tcp_port	Enables TCP traffic for the KDC
allow_kerberos_tgt_deleg	Enables delegation of Ticket-Granting Tickets (TGTs)
allow_mit_krb5_migrate	Needed if migrating existing Kerberos principals.
allow_smbd_krb5_right	Required if using Kerberos for Samba authentication.
allow_sshd_klogin	Enables Kerberos login for SSH connections.
allow_unreserved_ports	Allow applications to bind to privileged ports (ports 1-1024)

SELinux File Contexts

/etc/krb5.conf	etc_krb5_conf_t
/var/lib/krb5	var_lib_krb5_t
/var/log/krb5	var_log_krb5_t
Keytab - /etc/krb5.keytab)	krb5_keytab_t
/usr/sbin/kadmin, /usr/sbin/krb5kdc	usr_sbin_krb5_t
/run/krb5 (if used)	var_run_krb5_t

```
firewall-cmd --permanent --add-service=krb5 # Opens default Kerberos ports (TCP 88 and UDP 88)
firewall-cmd --permanent --add-service=kadmind # Opens KDC administration port (TCP 464)

iptables -A INPUT -p tcp --dport 88 -j ACCEPT
iptables -A INPUT -p udp --dport 88 -j ACCEPT
iptables -A INPUT -p tcp --dport 464 -j ACCEPT
systemctl restart krb5kdc kadmin
```

Client configuration - /etc/krb5.conf

```
[libdefaults]
    default_realm = EXAMPLE.COM
    ticket_lifetime = 24h
    renew_lifetime = 7d
[realms]
    EXAMPLE.COM = {
        kdc = kerberos.example.com
        # Optional: Specify additional KDC servers for redundancy
        # kdc = kerberos1.example.com
        # kdc = kerberos2.example.com
    }
[domain_realm]
    .example.com = EXAMPLE.COM
```

Server configuration example /etc/krb5/kdc.conf

```
[kdcdefaults]
    # Define encryption types supported by the KDC
    permitted_enctypes = aes256-cts-hmac-sha1-96 aes128-cts-hmac-sha1-96
    default_keytab = /etc/krb5/kdc.keytab
[realms]
    EXAMPLE.COM = {
        # Master key location (use kdb5_passwd to create)
        master_key_file = /var/lib/kerberos/krb5.keytab
        # Database for storing Kerberos principals (replace with your chosen database)
        database_module = kadm5
        # Database specific options
        database_name = EXAMPLE.COM  # Database name for the realm
        # Comment out if database resides on another machine (NOT good to have exposed on the network- don't!)
        # database_server = 192.168.1.10  # Replace with server IP (Not smart! See above)
        # admin_server = kerberos.example.com
        # Restrict access to the KDC based on IP address (Administrative Access Controls are a better option)
        # access_control = {
        #     host = 192.168.1.0/24  # Allow access from this subnet only
        # }
    }
```

Standalone Enterprise-ready LDAP - 389 Directory Server

https://www.port389.org/docs/389ds/documentation.html

Installing (also installs optional Cockpit dashboard for a nice web browser frontend)
dnf install 389-ds cockpit cockpit-389-ds sssd

Installed Files:

/etc/dirsrv/	Main directory for 389-ds configuration files.
/etc/dirsrv/slapd-config.conf	Main configuration file defining the LDAP server instance.
/etc/dirsrv/slapd-*.conf	Additional key configuration files (e.g., slapd-database.conf, slapd-access.conf)
/etc/dirsrv/schema/	Contains schema files for LDAP entries (core.schema, inetorgperson.schema)
/usr/lib/dirsrv/	Directory containing libraries used by the server
/usr/lib/systemd/system/dirsrv@slapd.service	Systemd service file for managing the 389-ds server process

Executables:

/usr/sbin/ns-slapd	Main server executable for the 389 Directory Server. (more below)
/usr/bin/dsctl	Tool to control and manage instances of 389 Directory Server.
/usr/bin/dsconf	Tool for configuring the server and managing server instances.
/usr/bin/dscreate	Tool to create a new instance of the 389 Directory Server.
/usr/bin/dsidm	Tool for managing identities (users, groups) in 389 Directory Server.
/usr/bin/dsconf-import	Tool for importing LDIF data into the directory server.
/usr/bin/dsconf-export	Tool for exporting directory server data to an LDIF file.
/usr/bin/dslogpipe.py	Tool for processing and managing server logs.
/usr/bin/ldapsearch	Command-line tool for searching LDAP directories.
/usr/bin/ldapmodify	Command-line tool for modifying LDAP entries.
/usr/bin/slaptest	Tool for testing the syntax of slapd configuration files
/usr/bin/slappasswd	Tool for generating hashed passwords for LDAP authentication.

Configuration Files:

/etc/dirsrv/slapd-instance/dse.ldif	Main configuration file for the 389 Directory Server instance.
/etc/dirsrv/slapd-instance/schema/99user.ldif	User-defined schema file for customizing directory attributes and object classes.
/etc/dirsrv/slapd-instance/ldif/template.ldif	Template LDIF file used during the initial server setup.
/etc/dirsrv/slapd-instance/password.conf	Configuration for password policies and storage.
/etc/dirsrv/slapd-instance/certmap.conf	Configuration for certificate mappings and SSL/TLS settings.
/etc/dirsrv/admin-serv/adm.conf	Configuration file for the Directory Server Admin Service.
/etc/dirsrv/admin-serv/console.conf	Admin console configuration file.
/etc/sysconfig/dirsrv	Environment variables for the Directory Server services.

Configuration Directories (files and scripts)

/etc/dirsrv/slapd-instance/schema	Schema files that define the structure and data types stored in the directory.
/etc/dirsrv/slapd-instance/ldif	Directory for LDIF files used for data import/export and initial configuration.
/etc/dirsrv/admin-serv	Configuration files for the administration server, managing the web-based console.
/etc/dirsrv/config	Contains configuration files and scripts for the server's operational settings.

Libraries and Systemd Services:

/usr/lib/dirsrv	Main directory for 389 Directory Server libraries and plugins.
dirsrv@instance.service	Manages a specific instance of the 389 Directory Server.
dirsrv-admin.service	Manages the administration service for the 389 Directory Server.

Commands and Options:

dsctl:		
dsctl instance start	Start the server instance	--debug (Run in debug mode)
dsctl instance stop	Stop the server instance	--force (Force stop the server)
dsctl instance status	Status of the server instance	--verbose (Provide detailed output)
dsconf:		
dsconf instance backend create	Create new backend for the server	--suffix (Specify the suffix DN for the backend)
dsconf instance replication enable	Enable replication for server	--role (Specify role i.e., master, consumer)
dsconf instance config replace	Modify configuration settings	--attr (Specify the attribute to replace)
dscreate:		
dscreate from-file config.inf	Create new server instance using config file	--force (Overwrite existing instance)
dscreate interactive	Create a new server instance interactively	--accept-license (Auto-accept license)
dsidm:		
dsidm instance user create	Create a new user in the directory	--uid (Specify the user ID)
dsidm instance group add-member	Add a user to a group.	--uid (Specify the user ID to add)
dsidm instance account status	Check the status of a user account.	--uid (Specify the user ID)

ns-slapd

ns-slapd is the primary executable used to start and manage the 389 Directory Server. Those familiar with other 'flavors' of LDAP servers (OpenLDAP) should be familiar with slapd. In 389 Directory Server, ns-slapd essentially serves the same role. It serves as both a server process daemon and can be used as an executable utility to do administrative tasks. Below, some options/flags are similar for both modes but have a different meaning for each- in this case ns-slapd's determination of which mode to execute them in is based on context.

Running ns-slapd as a process/daemon:

Quick example: /usr/sbin/ns-slapd -d 1 -r /var/lib/dirsrv/slapd-instance

-D, --daemon	Run as a background daemon (default).	/usr/sbin/ns-slapd -D
-d or --debug LEVEL	Run in foreground, set 1-9, (higher gives more detail)	/usr/sbin/ns-slapd -d 1
-f or --config DIR	Specify the directory containing the server's config files.	/usr/sbin/ns-slapd -f /path/to/config
-r, --read-only	Start the server in read-only mode (for maintenance)	/usr/sbin/ns-slapd -r
-w, --writenolog	Disable writing to the changelog	/usr/sbin/ns-slapd -w
-n, --no-clean	Do not remove temporary files on exit (for debugging).	/usr/sbin/ns-slapd -n
-F, --no-fsync	Disable fsync for performance (can risk data loss).	/usr/sbin/ns-slapd -F
-i or --instancename NAME	Give name of the server instance to manage.	/usr/sbin/ns-slapd -i instance
-p or --port PORT	TCP port the server listens on (default 389)	/usr/sbin/ns-slapd -p 1389

Running ns-slapd as a executable utility:

Quick example: ns-slapd -D "cn=admin,dc=example,dc=com" -W -a -f users.ldif -r -f delete_entries.ldif -n

-D or --binddn DN	Specify the DN to login with for LDAP/LDIF tasks	ns-slapd -D "cn=admin,dc=example,dc=com"
-W, --prompt	Prompt for password for the bind DN.	ns-slapd -W
-x, --simple	Use simple authentication instead of SASL.	ns-slapd -x
-a, --add	Add entries (useful for importing initial data).	ns-slapd -a
-r, --remove	Remove entries from the directory.	ns-slapd -r
-c, --continue	Continue processing despite errors.	ns-slapd -c
-n, --dry-run	Simulate task without making th task's changes.	ns-slapd -n

Using ldapmodify and ldapsearch

Common to Both ldapsearch and ldapmodify

-x	Use simple authentication (not SASL).	-x
-Y	Specify the SASL mechanism to use.	-Y GSSAPI
-D	Bind DN. Specifies the (admin) DN to login with to make changes, etc.	-D "cn=admin,dc=example,dc=com"
-W	Prompt for password.	-W
-H	Specify the LDAP URI to connect to.	-H "ldap://ldap.example.com"
-b	Base DN. The starting point for the search or modification	-b "dc=example,dc=com"
-LLL	Remove LDIF version lines/comments (e.g., trims off anything but the query answer in search results, etc.)	

Exclusive to ldapsearch

-s	Search scope: base, one, sub.	-s sub
-l	Time limit for the search in seconds.	-l 10
-z	Size limit for the number of entries returned.	-z 500
-E	Enable LDAP extensions.	-E pr=1000/noprompt
-A	Return attribute names only, not values.	-A
-T	Write results to a specified file.	-T /tmp/results.ldif
filter	Search filter expression to match entries.	(uid=jdoe)
attributes	Specify which attributes to return.	cn mail uid
subtree	Scope for subtree search (matches at and below the base DN).	subtree
one	Scope for one-level search (matches one level below the base DN).	one
base	Scope for base object search (limit matches to the base DN).	base

---- *LDAP Search Filter Operators*

`	OR operator for combining multiple search conditions.	
&	AND operator for combining multiple search conditions.	(&(objectClass=posixAccount)(uid=jdoe))
!	NOT operator for negating search conditions.	(!(objectClass=posixAccount))
=	Equality operator for matching attribute values.	(uid=jdoe)
>=	Greater than or equal to for numeric or ordered attribute values.	(uidNumber>=1000)
<=	Less than or equal to for numeric or ordered attribute values.	(uidNumber<=500)
~=	Approximate match operator for attribute values.	(cn~=John)
*	Wildcard operator for matching any attribute value.	(mail=*)

Exclusive to ldapmodify

-a	Add new entries to the directory.	ldapmodify -a -f new_entries.ldif
-c	Continue processing even if errors are encountered.	ldapmodify -c -f update.ldif
-r	Remove entries specified in the LDIF file.	ldapmodify -r -f delete_entries.ldif
-n	Show proposed changes, without applying them.	ldapmodify -n -f changes.ldif
changetype	Specifies type of change: add, modify, delete, or modrdn	changetype: modify
add	Add a new attribute value.	add: mail
delete	Delete an attribute or value.	delete: description
replace	Replace an attribute value.	replace: cn
modrdn	Modify (rename) the relative distinguished name (RDN).	modrdn: newcn
newrdn	New RDN for an entry.	newrdn: cn=Jane Doe
newsuperior	New superior (parent) entry for moving an entry	newsuperior: ou=newdept,dc=example,dc=com
deleteoldrdn	Flag to delete the old RDN value after renaming.	deleteoldrdn: 1

Example of ldapmodify

> ldapmodify -x -D "cn=admin,dc=example,dc=com" -W << EOF dn: uid=user,dc=example,dc=com changetype: modify replace: mail mail: new-email@example.com EOF

This can be better explained breaking it down like this:

ldapmodify -x -D "cn=admin,dc=example,dc=com" -W << EOF

The -D to 'bind' a DN (cn=admin,dc=example,dc=com) of the admin starting the session to do the rest

The -W says to ask for the admin password interactively, the -x says to just use simple authentication (not SASL for this)

The "<< EOF" is simply standard Linux "here document" syntax to send input until the terminating EOF at the end.

Next is the LDIF content:

dn: uid=user,dc=example,dc=com

the DN of the entry to be modified (user entry with uid=user)

changetype: modify

replace: mail

Says we are modifying an existing record, specifically replacing the mail attribute

mail: new-email@example.com

EOF

Lastly we give the replacement value for the mail attribute and close the here doc block with an EOF

Other Commands in 389 DS

slaptest -f	Test server configuration file for errors	slaptest -f /etc/dirsrv/slapd-instance/slapd.conf Use -F to force
slappasswd -s	Generate hashed passwords for server	slappasswd -s secret
dsconf-import	Import LDIF data into the server	dsconf-import -c data.ldif -c Continue on error/ skip bad entries, --dry-run: trial run with no changes.
dsconf-export	Export server data to an LDIF file	dsconf-export --base-dn "dc=example,dc=com" -f export.ldif -b, --base_dn: Specify base DN to export. -f, --file: filename
dslogpipe.py	Process and manage server logs	dslogpipe.py -i /var/log/dirsrv/slapd-instance/access -i, --input filename. -f, --filter: Apply filters to log entries

Finally, just about all commands support

-h, --help	Show help message with available options and usage.
-v, --version	Print the server version and exit.

LDAP Attributes and objectClasses

uid	User ID	uid=jdoe
sn	Surname (Last Name)	sn=Doe
givenname	Given Name (First Name)	givenname=John
cn	Common Name	cn=jdoe,ou=Users,dc=example,dc=com
dn	Distinguished Name	dn=cn=jdoe,ou=Users,dc=example,dc=com
mail	Mail	mail=jdoe@example.com
ou	Organizational Unit	ou=Programmers
department	departmentName	department=IT
title	title	title=Software Engineer
telephonenumber	Telephone Number	telephonenumber=555-123-4567
mobile	mobile	mobile=123-456-7890
o	Organization Name	o=Roxxon
postaladdress	postalAddress	postaladdress=123 Main St, Los Angeles, CA 90028
postalcode	postalCode	postalcode=90028
st	localityName	st=CA
description	description	description=Java and .Net Programming
dc	Domain Component	dc=com
dnsHostName	DNS Host Name	dnsHostName=dns1.example.com

ipHostNumber	IP Address	ipHostNumber=192.168.1.10
macAddress	MAC Address	macAddress=00:11:22:33:44:55
createTimestamp	Creation Timestamp	createTimestamp=20240625120000Z
modifyTimestamp	Modification Timestamp	modifyTimestamp=20240625120000Z
objectclass	Object Class	objectclass=inetOrgPerson,posixAccount,top
domainServer	hypothetical dn for a server	cn=server1,ou=Servers,dc=example,dc=com
inetOrgPerson	Represents a person within an organization	cn=jdoe,ou=Users,dc=example,dc=com mail=jdoe@example.com department=IT
groupOfNames	Container for group members	cn=Network Admins,ou=Groups,dc=example,dc=com member=uid=jdoe,ou=Users,dc=example,dc=com member=uid=jsmith,ou=Users,dc=example,dc=com

The objectClasses are grouping of other attributes and objectClasses, many are premade to chose from, or you can make your own

389 Directory Service quick-setup

Install preliminary packages:
dnf install 389-ds cockpit cockpit-389-ds sssd

Open Firewall Ports
sudo firewall-cmd --permanent --add-port={389/tcp,636/tcp}
sudo firewall-cmd --reload
sudo dnf install cockpit cockpit-389-ds sssd

Enable and start Cockpit (optional, for web-based management):
sudo systemctl enable --now cockpit.socket
sudo firewall-cmd --add-service=cockpit --permanent
sudo firewall-cmd --reload
    # You can access Cockpit at https://<server_ip>:9090.

Make the first directory server instance
- Option 1: Interactive setup, follow prompts to help configure your instance: run "sudo dscreate interactive"
- Option 2: Non-interactive instance creation:
    # First, build a configuration file (.inf) for the instance, at minimum containing this info (replace with yours):

```
---
# /path/to/instance_name.inf

# Specify the desired instance name (leave cn=directory_servers,cn=config as they are)
dn = cn=INSERT_INSTANCE NAME,cn=directory_servers,cn=config

# Define the base DN for your directory data, the root of your directory hierarchy.
#  For example, replace 'your_domain' with actual domain name, then dc=com or net or edu, etc.
directory = dc=YOUR_DOMAIN,dc=com

# Set the administrator password
adminPassword = your_strong_password
---
```

    #Next run this pointing to your new inf file:
sudo dscreate from-file /path/to/instance_name.inf

Prepare for next section "Securing 389 Directory Server"
    # Install these to move onto configuring authentification, authorization, and secure communication for the server.
sudo dnf install openssl openssl-libs cyrus-sasl cyrus-sasl-lib cyrus-sasl-gssapi cyrus-sasl-md5 krb5-workstation

Securing 389-DS with OpenSSL, Cyrus SASL, Kerberos

After creating your first instance in 389-DS, you can install and set up these, then apply to others as added.
Install: sudo dnf install openssl openssl-libs cyrus-sasl cyrus-sasl-lib cyrus-sasl-gssapi cyrus-sasl-md5 krb5-workstation

Configuring SSL/TLS for 389 DS
Obtain/ generate SSL/TLS Certificates:
If you don't have one, you can generate a self-signed certificate for testing purposes or get one from a trusted CA (Let's Encrypt)

To generate a self-signed certificate:
openssl req -new -x509 -days 365 -nodes -out /etc/dirsrv/slapd-INSTANCE1/ca.crt -keyout /etc/dirsrv/slapd-INSTANCE1/ca.key
Replace INSTANCE1 with your instance name. This creates a self-signed certificate and private key valid for 365 days.
Place your certificates in the appropriate directory for your 389 DS instance, usually under /etc/dirsrv/slapd-INSTANCE1/
[Note there is another option, but you have less granular control about the certificate: "sudo dsctl example tls generate-self-signed-cert --subject "/CN=example.com"]

Configure 389 Directory Server for SSL/TLS:
dsconf -D "cn=Directory Manager" ldaps://localhost ssl set --enable true
dsconf -D "cn=Directory Manager" ldaps://localhost ssl cert --import --file /etc/dirsrv/slapd-INSTANCE1/ca.crt
dsconf -D "cn=Directory Manager" ldaps://localhost ssl key --import --file /etc/dirsrv/slapd-INSTANCE1/ca.key --password PASSWD

Update the Directory Server Configuration:
Edit the dse.ldif configuration file to enable SSL with 'sudo nano /etc/dirsrv/slapd-INSTANCE1/dse.ldif'
Add or update the following entries, set the correct certificate paths:

```
dn: cn=encryption,cn=config
nsslapd-security: on
nsslapd-securePort: 636
nsslapd-ssl-check-hostname: on
nsslapd-certdir: /etc/dirsrv/slapd-INSTANCE1
nsslapd-certname: Server-Cert
nsslapd-certfile: ca.crt
nsslapd-keyfile: ca.key
```

Save and close the file, then restart the server with 'sudo systemctl restart dirsrv@INSTANCE1'

Configuring SASL for 389 Directory Server
Enable SASL in the 389 Directory Server - edit the dse.ldif file with 'sudo nano /etc/dirsrv/slapd-INSTANCE1/dse.ldif'
Add or modify the following entries:

```
dn: cn=config
nsslapd-sasl-maps: on
nsslapd-sasl-max-buffer-size: 65536
nsslapd-sasl-secprops: noanonymous,noplain,novalidate
```

Configure SASL Mechanisms:
Create/edit the /etc/sasl2/slapd.conf file to specify SASL options. Note the keytab location for Kerberos setup (the next step)

```
pwcheck_method: saslauthd
mech_list: GSSAPI DIGEST-MD5 CRAM-MD5
keytab: /etc/dirsrv/slapd-INSTANCE1/ldap.servername.keytab
```

Setup Kerberos for SASL/GSSAPI:
Edit /etc/krb5.conf and replace the following with your kerberos server's information:

```
[libdefaults]
   default_realm = YOUR.REALM
   dns_lookup_realm = false
   dns_lookup_kdc = true
[realms]
   YOUR.REALM = {
      kdc = kdc.example.com
      admin_server = kdc.example.com
   }
[domain_realm]
   .example.com = YOUR.REALM
   example.com = YOUR.REALM
```

Create a service principal for 389 DS by running ktpasswd (generaties key and stores the keytab where directed)

```
Be sure pathname to keytab is matched in /etc/sasl2/slapd.conf (noted above)
ktpasswd -q -h /etc/dirsrv/slapd-INSTANCE1/ldap.servername.keytab ldap/servername@example.com
```

For this next step, make sure there is a username on the Kerberos server called 389-LDAPsetup (for this example) or something to be a admin placeholder username for the ongoing server key usage and initial setup of LDAP admin users.
Use kinit to obtain a ticket (basically kerberos-username-for-server@REALM)

kinit 389-LDAPsetup@example.com

Testing the Configuration

Verify SSL: ldapsearch -x -H ldaps://localhost -b "dc=example,dc=com"
Test SASL: ldapsearch -Y GSSAPI -H ldap://localhost -b "dc=example,dc=com" "(objectclass=*)"

More items to improve security of the 389 Directory Server

Restrict LDAP access from only the secure port (LDAPS)

```
#Put in slapd.conf
dn: cn=config
nsslapd-listenhost: localhost
nsslapd-port: 636  # Use port 636 for LDAPS
```

Save and restart the directory server instance:

```
sudo systemctl restart dirsrv@<instance_name>
```

Configure Access Control Lists (ACLs)

```
# Define ACLs to control access to the directory data, maintain data integrity and security.
# Example ACL to allow read access to all users but restrict write access to admins:
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr != "userPassword")(version 3.0; acl "Allow read"; allow (read, search, compare) userdn="ldap:///self";)
aci: (targetattr="*")(version 3.0; acl "Admin write"; allow (all) groupdn="ldap:///cn=admins,dc=example,dc=com";)

# Save this to a file acl.ldif, then, apply the ACL using:
ldapmodify -x -D "cn=Directory Manager" -W -f acl.ldif
# The content of the ACL is saved in attributes of the target dn, in this case  dn: dc=example,dc=com
```

Enable password policies to enforce complexity, expiration, and lockout rules.

Create a file password_policy.ldif with the desired policies:

```
dn: cn=config
changetype: modify
replace: nsslapd-pwpolicy
nsslapd-pwpolicy: on

dn: cn=default,ou=pwpolicies,dc=example,dc=com
objectClass: top
objectClass: pwdPolicy
pwdAttribute: userPassword
pwdMaxAge: 7776000  # 90 days
pwdMinLength: 8
pwdCheckSyntax: 1  # Enforce complexity rules
pwdInHistory: 5  # Remember past passwords
pwdLockout: TRUE
pwdLockoutDuration: 900  # 15 minutes lockout
```

Apply the policy:

```
ldapmodify -x -D "cn=Directory Manager" -W -f password_policy.ldif
```

Enable and configure logging for access and errors (edit dse.ldif):

```
dn: cn=config
nsslapd-accesslog: /var/log/dirsrv/slapd-example/access
nsslapd-errorlog: /var/log/dirsrv/slapd-example/errors
# Ensure logs are rotated and reviewed regularly:
sudo logrotate /etc/logrotate.d/dirsrv
```

Enable SNMP monitoring using dsconf:

```
sudo dsconf example config replace nsslapd-schemachecking=on
sudo dsconf example config replace snmp-port=199
```

Implement disaster recovery using dsconf to perform a backup

```
sudo dsconf example backend export --base-dn "dc=example,dc=com" /path/to/backup
```

Core Configuration and Settings for 389-Directory Server

Main Configuration (slapd.conf):

```
        # General settings
instance-name my-ldap-server  # Replace with your desired instance name
suffix "dc=example,dc=com"  # Replace with your domain name
        # Directory database configuration
database: mdb_directory  # Database type (Modify based on your setup)
index objectClass, cn  # Attributes to be indexed
        # Network settings
listen-address 0.0.0.0  # Listen on all interfaces
listen-port 389  # Standard LDAP port
        # Include additional configuration files
include "user_management.conf"
include "security.conf"
include "backup.conf"
```

User Management (user_management.conf):

```
        # User account settings
baseDn "ou=People,dc=example,dc=com"  # Base DN for user accounts
password-min-length 8
password-require-mixed-case TRUE
password-history-length 5
account-lockout-threshold 5
account-lockout-duration 30m
groupDn "ou=Groups,dc=example,dc=com"  # Base DN for groups
# password-expire: #days  # Enable password expiry after a set number of days
# user-quota: #bytes  # Set quota on the size of user entries
# dynamic-groups:  # Configure rules for dynamic group membership
```

Security Settings (security.conf):

```
        # Password complexity rules
require-mixed-case TRUE
password-minimum-length 12
require-numbers TRUE
require-special-characters TRUE
        # TLS/SSL encryption
tls-enabled TRUE
tls-certificate-file /path/to/server.crt
tls-key-file /path/to/server.key
        # Access control (modify based on your needs)
access to by dn "cn=Directory Manager,ou=People,dc=example,dc=com" read
        # Additional security options (uncomment and configure as needed)
# access-control-list:  # Define granular access control for specific DNs
# security-auditing:  # Enable security auditing for directory operations
```

Backup Configuration (backup.conf):

```
        # Backup settings
backup-frequency daily
backup-directory /var/lib/389-ds/backups
backup-retention 7
        # Backup tool configuration (replace 'tool' with your chosen tool)
backup-tool "tar"
backup-arguments "-cvzf"
```

Logging Configuration (logging.conf): (Optional, can be included in the main conf)

```
        # General logging level
log-level info
        # Specific log levels for components (uncomment and adjust as needed)
# slapd-level debug
# slapd-modules-level warn
        # Log rotation settings
log-rotate-size 10m  # Rotate logs when they reach 10MB
log-rotate-count 5  # Keep the last 5 rotated logs
        # Additional logging options (uncomment and configure as needed)
# remote-logging: # Configure remote logging to a central server
```

Tips on improving the 389 Directory Server implementation moving forward

Automated Instance Creation with a Standardized Configuration File
Streamline setup - Make clones from a config instance.inf file with the following content:

```
[general]
config_version = 2

[slapd]
instance_name = auto-made-instance
root_dn = cn=Directory Manager
root_password = <STRONG_PASSWORD>
server_port = 389
suffix = dc=example,dc=com
self_sign_cert = True  # Automatically generate a self-signed certificate

[backend-userroot]
database_name = userroot
suffix = dc=example,dc=com
```

```
        # Create the directory server instance on-demand and ensure the instance is running and verify the status:
sudo dscreate from-file instance.inf
sudo dsctl auto-made-instance status
```

Configure Backend Databases
Splitting data into separate backend databases improves load management and scalability. Add a new backend for organizational units. Add this to dse.ldif or use dsconf to configure dynamically.

```
[backend-newunit]
database_name = newunit
suffix = ou=newunit,dc=example,dc=com
```

Configure Indexing for Enhansing Performance
Use the following dsconf commands to index commonly searched attributes without the need to manually edit dse.ldif

```
        # Index uid for equality searches:
sudo dsconf example backend index create --attr-name uid --types eq
        # Index mail for equality and presence searches:
sudo dsconf example backend index create --attr-name mail --types eq pres
        # Index sn (Surname) for equality and presence searches:
sudo dsconf example backend index create --attr-name sn --types eq pres
```

Optimize Database Cache Settings - Enhance performance by tuning the database cache

```
sudo dsconf example backend config set --db-cache-size 512MB
sudo dsconf example backend config set --entry-cache-size 512MB
```

Replication for High Availability - Multi-master replication ensures data redundancy and HA

```
        # Primary Server Configuration (edit dse.ldif or use dsconf):
dn: cn=replica,cn="dc=example,dc=com",cn=mapping tree,cn=config
objectClass: top
objectClass: nsDS5Replica
nsDS5ReplicaRoot: dc=example,dc=com
nsDS5ReplicaId: 1
nsDS5ReplicaType: 3
nsDS5Flags: 1
nsDS5ReplicaBindDN: cn=Replication Manager,cn=config
```

Secondary Server Configuration:
Configure similarly, but with a unique nsDS5ReplicaId (e.g., 2).

```
        # Establish replication relationships:
sudo dsconf example repl-agmt create --suffix="dc=example,dc=com" --host=<secondary-server-ip> --port=389 --bind-
dn="cn=Replication Manager,cn=config" --bind-passwd=<bind_password> --conn-protocol=LDAP --init
```

Repeat on the secondary server pointing back to the primary server.

BIND9 for DNS Management on RHEL

BIND (Berkeley Internet Name Domain) is typically provided by the bind package along with several supporting packages:
sudo dnf install bind bind-utils bind-libs bind-chroot bind-dyndb-ldap bind-ldap-schema

- bind: The main BIND DNS server package.
- bind-utils: Utilities for querying DNS servers.
- bind-libs: Libraries for BIND.
- bind-chroot: Allows running BIND in a chroot jail for added security.
- bind-dyndb-ldap: A dynamic plug-in to use LDAP as a backend for BIND.
- bind-ldap-schema: LDAP schema extensions for BIND to integrate with LDAP

Configuration locations and notable files

/etc/named - A directory that typically holds zone files and additional configuration files.
/etc/rndc.key - The key file for secure communication between rndc and named.

/etc/named.conf	Central config file for the DNS server (global options, logging config, zone declarations
/var/named/*.zone	Zone files, domain-based name (example.com.zone); define a domain's DNS records
/etc/named/keys/	Directory for DNSSEC keys
/etc/named.rfc1912.zones	Used to include addl. zone files and settings
/etc/rndc.conf and /etc/rndc.key	Configuration and key files for the rndc utility, to securely control the BIND daemon
bind-dyndb-ldap	LDAP back-end for BIND, to enable storage of DNS data in LDAP

When using the bind-chroot package, the configuration files are located in the /var/named/chroot/etc/ directory.

Logs:
Syslog: BIND can be configured to send its logs to the system syslog (/var/log/messages or /var/log/syslog
Log Files: If configured explicitly in the BIND configuration file (named.conf),
A few pages into this, I put instructions on setting up SNMP for logs and network monitoring tool

DNS Configuration and File Locations: Debian vs. RHEL 9

File/Directory Type	Debian Default Location	RHEL 9 Default Location
BIND Main Config File	/etc/bind/named.conf	/etc/named.conf
BIND Addl Config Files	/etc/bind/named/	/etc/named/
Zone Files	/etc/bind/zones/ or /var/lib/bind/	/var/named/
Root Hints File	/usr/share/dns/root.hints	/var/named/named.ca
RNDC Configuration File	/etc/bind/rndc.conf	/etc/rndc.conf
RNDC Key File	/etc/bind/rndc.key	/etc/rndc.key
BIND Log Directory	/var/log/bind/	/var/log/named/
DNSSEC Key Files	/etc/bind/keys/ (if specified) or /var/lib/bind/	/etc/named/keys/ or /var/named/
DNSSEC KSK and ZSK Files	Typically in /var/lib/bind/	Typically in /var/named/
DNSSEC Signing Policies	/etc/bind/keys/ or defined in the named.conf	/etc/named/keys/ or defined in named.conf
TSIG Key Files	/etc/bind/ or /etc/bind/tsig/	/etc/named/ or /etc/named/tsig/
Cache Directory	/var/cache/bind/	/var/named/data/
Pid File	/run/named/named.pid	/run/named/named.pid

Zone Files: Can be stored where specified in named.conf but usually /etc/bind/ or /var/lib/bind/ (Debian) or /var/named/ (RHEL).
DNSSEC Files: KSK and ZSK keys sometimes put with zone files, but more common is /etc/bind/keys/ or /etc/named/keys/.
RNDC Key Files: By default, the RNDC key is placed in /etc/rndc.key, but can be overridden in rndc.conf or named.conf.
Config Mgmt: Debian often separates config files into smaller chunks in /etc/bind/, while RHEL puts them in /etc/named/.

DNS Records Review

A	IPv4 Address	Maps a domain name to an IPv4 address	example.com IN A 93.184.216.34
AAAA	IPv6 Address	Maps a domain name to an IPv6 address	example.com IN AAAA 2001:0db8::abcd
CNAME	Canonical Name	Creates an alias for a domain name	www.example.com IN CNAME example.com
MX	Mail Exchange	Specifies the mail server for a domain	example.com IN MX 10 mail.example.com
NS	Name Server	Indicates authoritative DNS servers for a domain	example.com IN NS ns1.example.com
PTR	Pointer	Map IP address to domain name (reverse DNS)	34.216.184.93.in-addr.arpa IN PTR example.com
SRV	Service Locator	Specifies the location of services within a domain	_sip._tcp.example.com IN SRV 10 60 5060 sipserver.example.com
TXT	Text	Stores arbitrary text strings, often for validation	example.com IN TXT "Client ID = 436HJK54J"
DNSKEY	DNS Public Key	Publishes the public key for DNSSEC	example.com IN DNSKEY 256 3 8 AwEAA...
RRSIG	Resource Record Sig	Contains the digital signature for DNS records	example.com IN RRSIG A 5 2 3600 202406270 2023062701 12345 example.com. AwEAA...
DNAME	Delegation Name	Redirects subtree of DNS NS to another domain	sub.example.com IN DNAME newsub example.com
SPF	Sender Policy Framewk	Specifies permitted mail servers for a domain	example.com IN SPF "v=spf1 include:_spf.example.com"
NSEC	Next Secure	Proof of non-existence for DNSSEC records	example.com IN NSEC a.example.com A RRSIG NSEC
NSEC3	Next Secure Version 3	Proof w/ hashed names - see also NSEC3PARAM	3a2a3b4d3b3a4d5b.example.com IN NSEC3 1 0 1
CDNSKEY	Child DS	DNSSEC, Securely delegate child zones	Example.com. IN CDNSKEY 256 3 8 (truncated pub key)
DS	Delegation Signer	Links a DNSSEC-signed child zone to its parent	example.com N DS 12345 8 2 49FD46E6C...
CDS	Child DS	Digest of a CDNSKEY	Example.com IN CDS 256 3 8 (truncated SHA)
CAA	CA Authorization	Specifies allowed CAs for issuing certificates	example.com IN CAA 0 issue "letsencrypt.org"

Installed Executables [/usr/sbin/ and /usr/bin/]

named — Main BIND DNS server daemon and executables

named-pkcs11 — Extended version of named; adds PKCS#11 support for HSM hardware-backed keys for DNSSEC

-c <config-file> Use the specified configuration file
-g Run in the foreground and print logs to stdout
-u <user> Run named as a specified user.
-t <directory> Specify a directory for chroot operation
-p <port> Listen on the specified port (default is 53)
-m <policy> Memory allocation policy
-4 / -6 Use IPv4 or IPv6 exclusively
-S Use a single processing thread
-T <number> Set the number of worker threads
-n <number> Set the number of UDP listeners per interface
-D <directory> Set the working directory for the server
-d <level> Set the debug level (higher levels increase verbosity)
-H Use HSM for cryptographic operations (named-pkcs11 only)
-l Log to syslog instead of stderr (named-pkcs11 only)

```
sudo named -c /etc/named.conf -g
sudo named-pkcs11 -c /etc/named-pkcs11.conf -u named -g -p 53 -t /var/named -k /etc/pki/keys/dnssec -H pkcs11 -l
```

rndc — Remote Name Daemon Control utility to manage named

start, stop, status; reload for reloading config and zones; reconfig to reload config but not zones
flush: Flush all caches
addzone and delzone <zone> Add or delete a zone

named-checkconf — Check syntax of named.conf; use -z to also check the validity of the zone files

```
sudo named-checkconf -z /etc/named.conf
```

named-checkzone — Check the syntax and consistency of a zone file

```
sudo named-checkzone <zone> <zone-file>
sudo named-checkzone example.com /var/named/example.com.zone
```

named-compilezone — Compile a zone file into a more efficient binary format.

<zone>: The zone name.
<input-file>: The input zone file.
<output-file>: The output file to write the compiled zone.

```
sudo named-compilezone -f <format> -o <output-file> <zone> <input-file>
sudo named-compilezone -f raw -o /var/named/example.com.zone.db example.com /var/named/example.com.zone
```

dnssec-keygen — Generate DNSSEC keys

-a <algorithm>: Specify the key algorithm
-b <keysize>: Specify the key size
-n <nametype>: Specify the type of owner name (zone, host, etc.)

```
sudo dnssec-keygen -a RSASHA256 -b 2048 -n ZONE example.com
```

dnssec-signzone — Sign a zone file with DNSSEC

-o <zone>: Specify the zone name
-k <key>: Specify a key for signing

```
sudo dnssec-signzone -o example.com -k Kexample.com.+008+12345 /var/named/example.com.zone
```

Common options for dnssec-dsfromkey, dnssec-importkey, and dnssec-verify:

-K <directory> Directory where key files are stored.
-l <ttl> Set TTL for the DS record (not in dnssec-verify)
-f <format> Output format (text, full)
-v <level> Set the verbosity level
-c <class> DNS class- default is IN (not in dnssec-dsfromkey)

dnssec-dsfromkey — Creates Delegation Signer (DS) records from DNSKEY records

-2: Use SHA-256 as the hash algorithm.
-a <algorithm>: Specify the hash algorithm (SHA-1, SHA-256, SHA-384).
-f <format>: Output format (text, full).
-T <type> Specify the DNS record type (default is DS)

```
dnssec-dsfromkey -a SHA-256 -f text example.com.dnskey
```

dnssec-importkey — Imports DNSSEC keys into BIND's DNSSEC mgmt system

-c <class>: Specify the DNS class (default is IN).
-e <epoch>: Specify the end of the key's validity period.
-t <type>: Specify the key type (KSK, ZSK).

```
dnssec-importkey -K /var/named/keys -t ZSK example.com example.com.key
```

dnssec-verify — Verifies the signatures in a zone, ensuring zone is correctly signed and validating integrity.

-k <key-file>: Specify a key file to use for verification.
-o <origin>: Specify the zone origin (domain name).
-t <directory>: Specify a directory for temporary files.
-x: Perform a DNSSEC signature expiry check.

```
dnssec-verify -o example.com -K /var/named/keys example.com.zone
```

Review: Zone Transfer Types
AXFR (Authority Full Zone Transfer): Transfers the entire zone file, containing all DNS records for a domain, from the primary authoritative server to a secondary server. This is typically done periodically or after significant changes to the zone file. Due to its size, full transfers can be resource-intensive, especially for large domains.

IXFR (Incremental Zone Transfer): Transfers only the updated portions of the zone file since the last successful transfer. This is the preferred method for frequent updates as it reduces bandwidth consumption and server load. Signed IXFR utilizes DNSSEC to cryptographically sign the transferred data, ensuring its authenticity and preventing tampering.

Review: Root Servers, TLD Servers, Etc
Internet Assigned Numbers Authority (IANA) manages the root zone, that is comprised of 13 sets of root servers globally. They don't perform zone transfers due to the immense load they manage- instead relying on efficient replication. They delegate responsibilities of top-level domains (TLDs) like .com and .org to specific registries such as Verisign (.com, .net, and .name TLDs); ICANN (.gov, .edu, and .mil); Public Interest Registry (PIR) oversees the .org TLD, and IANA (for ICANN) internet infrastructure TLDs like .arpa. The root zone data contains information about the authoritative name servers for each TLD. When a DNS resolver initiates a query for a domain name, the root servers simply point the resolver in the direction of the appropriate TLD servers

It is the TLD server layer in the hierarchy where zone transfers start being relevant. A primary TLD server holds and distributes the authoritative zone file for the TLD, while secondary TLD servers regularly perform zone transfers to obtain the latest zone data from the promary. When a change is made to the zone file on the primary TLD server, a notification is sent to the secondary servers using a mechanism like SIGNOTIFY (part of DNSSEC). Secondary servers receive the notification and examine its content to understand the nature of the change, like details about the specific resource that was modified (e.g., a new domain added or an existing record updated). Based on the notification details and potentially a configured time interval, the secondary server initiates a zone transfer to retrieve the update from the primary server.

If changes to the TLD zone file occur frequently, updates might be more frequent (potentially every few minutes). Each TLD registry might have its own policies regarding update intervals, since there is also a trade-off between speed and efficiency: frequent updates can create more traffic, less frequent updates might introduce a propagation delay.

TLD servers actually experience lower traffic compared to root servers, making zone transfers more manageable. Techniques like zone change incrementalism (ZCI) can further optimize zone transfers by minimizing the amount of data transferred based on the specific changes made. Secondary servers play a crucial role in efficiently answering DNS queries, often handle a large portion of the overall traffic volume, reducing the load on the primary server and root servers.

Review: DNS behavior - a seldom-visited webpage in a new browser window:
You type the domain name into your web browser's address bar or click a hyperlink
Your PC's DNS cache is checked - Does the OS have the IP or domain name cached?
Home router: have the IP or domain name cached?
Recursive resolver at ISP takes responsibility for finding the answer to your query, including contacting other name servers.
- Iterative Resolution Process:
  - Checks its own cache to see if it has the info; if not, ask the appropriate TLD servers
    - If not found, it asks root nameservers for the right TLD servers.
      - Root nameservers point the recursive resolver to the proper TLD nameservers
    - The recursive resolver then queries the TLD servers it was referred to
    - TLD servers will know the authoritative nameservers for this domain
  - The recursive resolver contacts the authoritative nameservers for domain name
- An authoritative nameserver checks its zone file containing the domain's IP address.

Once the recursive resolver receives the IP address, it sends it back to your web browser.
The recursive resolver likely will cache this information for future queries.
Finished- website connection established.

Sample named.conf for Red Hat BIND

The original of this in that shipped in RHEL was over-commented. It was been trimed to save space)

```
*/
options
{
// Directory for writable files
        directory "/var/named";
        dump-file "data/cache_dump.db";
        statistics-file "data/named_stats.txt";
        memstatistics-file "data/named_mem_stats.txt";
        secroots-file "data/named.secroots";
        recursing-file "data/named.recursing";

// Listen on these interfaces
        listen-on port 53 { 127.0.0.1; };
        listen-on-v6 port 53 { ::1; };

// Restrict access
        allow-query { localhost; };
        allow-query-cache { localhost; };

// Recursion settings
        recursion yes;

// DNSSEC validation
        dnssec-validation yes;

// File paths for system specifics
        pid-file "/run/named/named.pid";
        session-keyfile "/run/named/session.key";
        managed-keys-directory "/var/named/dynamic";

// Use system-wide Crypto Policy
        include "/etc/crypto-policies/back-ends/bind.config";
};
logging
{
// Debugging log
        channel default_debug {
            file "data/named.run";
            severity dynamic;
        };
};
// Views for different client types
view "localhost_resolver"
{
// Localhost resolver (caching only)
        match-clients { localhost; };
        recursion yes;

// Root hints zone
        zone "." IN {
            type hint;
            file "/var/named/named.ca";
        };

// Zones for localhost
        include "/etc/named.rfc1912.zones";

// Root hints zone
        zone "." IN {
            type hint;
            file "/var/named/named.ca";
        };
};
view "internal"
{
// Zones for internal clients (localnets)
        match-clients { localnets; };
        recursion yes;

// Zones for localhost
        include "/etc/named.rfc1912.zones";

// Authoritative internal zones
        zone "my.internal.zone" {
            type primary;
            file "my.internal.zone.db";
        };
        zone "my.slave.internal.zone" {
            type secondary;
            file "slaves/my.slave.internal.zone.db";
            masters { 127.0.0.1; };
        };
        zone "my.ddns.internal.zone" {
            type primary;
            allow-update { key ddns_key; };
            file "dynamic/my.ddns.internal.zone.db";
        };
};
key ddns_key
{
        algorithm hmac-sha256;
        secret "use /usr/sbin/ddns-confgen to make TSIG keys";
};
view "external"
{
// Zones for external clients
        match-clients { any; };
        recursion no;

// Root hints zone
        zone "." IN {
            type hint;
            file "/var/named/named.ca";
        };
// Authoritative external zones
        zone "my.external.zone" {
            type primary;
            file "my.external.zone.db";
        };
};
/* DNSSEC keys (trusted anchors) */
trust-anchors {
 // Root Key
  . initial-key 257 3 8
"AwEAAaz/tAm8yTn4Mfeh5eyI96WSVexTBAvkMgJzkKTOiW1vkI
bzxeF3   [truncated]     R1AkUTV74bU=";

 // Key for forward zone
  example.com. static-key 257 3 8
"AwEAAZ0aqu1rJ6orJynrRfNpPmayJZoAx9Ic2/Rl9VQW
            [truncated]  NWUla4fWZbbaYQzA93mLdrng+M=";

 // Key for reverse zone.
  2.0.192.IN-ADDRPA.NET. initial-ds 31406 8 2 "F78CF3344F72
[truncated]  6D";
};
*/
```

Simple quick solutions for smaller DNS setups

IPv6 and Dual-Stack Support

```
options { listen-on { 192.168.0.1; }; listen-on-v6 { 2001:db8::1; }; allow-query { any; }; allow-query-v6 { any; };  };
        # listen-on are the IP addresses of the interfaces to get info from, allow-query opens availability
```

Blackhole Lists - Blocking Queries

```
options { blackhole { 192.0.2.0/24; 203.0.113.0/24; };  };
        # putting contents of a bracketed block on one line is frowned upon but I need the space to fill the page neatly!
```

Granular Access Control Lists (ACLs)

```
acl "trusted" {
   192.168.1.0/24;  // Local network
   10.0.0.0/16;     // Another trusted network
   localhost;       // Localhost
};
options {
   allow-query { trusted; };  // Restrict queries to trusted networks
   allow-recursion { trusted; };  // Restrict recursion to trusted networks
};
```

Query Rate Limiting to Mitigate DoS Attacks

```
options {
   rate-limit {
      responses-per-second 10;  // Limit to 10 responses per second per client
      window 5;  // Time window in seconds for rate limiting
      log-only yes;  // Log but don't drop excess responses (for monitoring)
   };
};
```

Basic High Availability and Replication (provision secondary DNS server)

On Primary DNS Server (named.conf):

```
zone "example.com" {
   type primary;
   file "zones/example.com.db";
   allow-transfer { secondary_dns_ips; };  // Allow transfers to secondary servers
   also-notify { secondary_dns_ips; };     // Notify secondary servers of zone updates
};
```

On Secondary DNS Server (named.conf):

```
zone "example.com" {
   type secondary;
   file "slaves/example.com.db";
   masters { primary_dns_ip; };  // IP of the primary DNS server
};
```

Secure Internal DNS Forwarding (internal.corp)

```
zone "internal.corp" {              # Configure forwarding for "internal.corp" domain
   type forward;                    # Zone type to forward- forward queries for this domain to specified servers.
   forwarders {                     # Internal DNS servers that will handle queries for "internal.corp" domain
      192.168.0.1;                  # Primary internal DNS server IP
      192.168.0.2;                  # Secondary
   };
                                    # Define client networks allowed to forward queries to internal DNS servers.
   allow-query { internal-net; };   # Only allow queries from the defined ACL (internal-net).
   forward only;                    # Use only the forwarders- do NOT try to resolve the queries if forwarders fail
};
acl "internal-net" {                # Has to be defined outside the zone { } block
   192.168.1.0/24;                  # Example of an internal subnet
   10.0.0.0/16;                     # Another internal subnet
};
```

Chroot Configuration

```
options {
   directory "/var/named/chroot";  // Chroot directory
   pid-file "/var/named/chroot/run/named.pid";  // Adjust paths for chroot environment
   session-keyfile "/var/named/chroot/run/session.key";
};
```

Geo-Location Based Routing

```
view "us_clients" {
   match-clients { 192.0.2.0/24; 198.51.100.0/24; };  // US clients
   zone "example.com" {
      type primary;
      file "zones/us.example.com.db";  // Zone file for US clients
   };
};
view "eu_clients" {
   match-clients { 203.0.113.0/24; 203.0.114.0/24; };  // EU clients
   zone "example.com" {
      type primary;
      file "zones/eu.example.com.db";  // Zone file for EU clients
   };
};
```

Dynamic DNS Updates Across Multiple Views (with TSIG Key)

```
zone "dynamic.example.com" {                          # DNS zone where dynamic updates are allowed
   type primary;                                      # Primary (authoritative) source for this zone
   file "zones/dynamic.example.com.db";               #  File where the zone data is stored
   update-policy {                                    #Configure the update policy for this zone.
      // Grant permissions to the TSIG key (ddns_key)
      grant ddns_key wildcard *.dynamic.example.com. A;   # Any A record (map hostnames to IP) in zone can be updated
   };
};
key ddns_key {                                        #  Define TSIG key used for securing dynamic DNS updates
   algorithm hmac-sha256;                             # Hashing algorithm used to create the MAC
   secret "base64-encoded-secret-key";                # Actual base64-encoded key for making HMAC signature
};
```

Advanced Logging and Monitoring

```
logging {                          # BIND's part: In named.conf,
   channel query_log {
      file "/var/log/named/queries.log" versions 10 size 100M;
      severity info;
      print-time yes;
   };
   channel security_log {
      file "/var/log/named/security.log" versions 10 size 50M;
      severity notice;
      print-time yes;
   };
   category queries { query_log; };
   category security { security_log; };
};

agentAddress  udp:161          # SNMP's part - in /etc/snmp/bind-snmp.conf)
rocommunity public
view all    included  .1 80
group MyROGroup v1    all
group MyROGroup v2c   all
group MyROGroup usm   all
access MyROGroup ""      any       noauth    exact  all    none   none
```

Minimizing Unnecessary Transfers

Allow-Transfer: This directive specifies which hosts or networks are allowed to initiate zone transfers. By default, it might be set to any, allowing anyone to request a transfer. Here's an example to restrict transfers to specific IP addresses:

```
zone "yourdomain.com" {
  allow-transfer { 192.168.1.10; 10.0.0.2; };  # Replace with authorized IP addresses
};
```

Also-Notify: This directive informs secondary servers when the zone file is updated. This can help automate transfer requests from authorized secondary servers, reducing unnecessary manual transfers.

```
zone "yourdomain.com" {
  also-notify { 192.168.1.20; };  # Replace with secondary server IP
};
```

Schedule Transfers During Off-Peak Hours
While named.conf doesn't directly schedule transfers initiated by secondary servers, you can achieve a similar effect on the primary server. The transfer-source directive specifies the IP address the server uses to initiate outgoing zone transfers. You can combine this with firewall rules to restrict outbound traffic during peak hours, indirectly influencing transfer timing.

```
zone "yourdomain.com" {
  transfer-source { 10.0.0.1; };  # Replace with appropriate IP for outbound transfers
};
```

A safer and easier method would be to leverage cron to schedule it
This example usies a cron job to reload the BIND rndc service to initiate a zone transfer during off-peak hours:

```
        # Edit the cron job file for the BIND user (usually named or bind)
sudo crontab -e -u bind
# Add the following line to schedule a transfer at 2:00 AM
0 2 * * * /usr/sbin/rndc reload example.com
```

DNSSEC Key Pairs, ZSK and KSK, Secure Zone Transfers
Key Creation: dnssec-keygen, tsig-keygen

```
        # Generate a Zone Signing Key (ZSK)
dnssec-keygen -a RSASHA256 -b 2048 -n ZONE example.com
        # Generate a Key Signing Key (KSK) helps validating DNSKEY records
dnssec-keygen -a RSASHA256 -b 4096 -n ZONE -f KSK example.com
        # This will create key files with .key and .private extensions. These files are used for signing the zone.
tsig-keygen -a hmac-sha256 tsig_key > /etc/bind/tsig_key.conf
        # This will create the TSIG file for actual zone transfers
```

Signing the zone file, updating named.conf

```
dnssec-signzone -o example.com -k Kexample.com.+008+12345 example.com.db
```

-o example.com (zone domain), -k Kexample.com.+008+12345 (the KSK) example.com.db (zone file to sign)

```
        # Put keys and signed zone file in named.conf
zone "example.com" {
   type master;
   file "zones/example.com.db.signed";         // Use the signed zone file
   allow-transfer { key /etc/bind/tsig_key.conf; };          // Secure transfers - reference key defined in tsig_key.conf
   notify yes;        // Enable notifications
   also-notify { 192.168.1.10; };        // Secondary server to notify
   inline-signing yes;          // Enable inline signing for automatic DNSSEC signing
   auto-dnssec maintain;   // Automatically maintain DNSSEC records
   max-transfer-time-in 60;// Limit the maximum transfer time to 60 seconds
   transfer-format many-answers;    // Optimize transfer format for efficiency
};
        # Setting up secondary server
Copy keys and signed zone(s) from the primary to secondary
scp /var/named/zones/example.com.db.signed secondary:/var/named/zones/example.com.db.signed
scp /var/named/keys/Kexample.com.+008+12345.key secondary:/var/named/keys/Kexample.com.+008+12345.key
scp /etc/named/tsig_key.conf secondary:/etc/named/tsig_key.conf
scp /var/named/keys/example.com.db.signed secondary:/var/named/keys/example.com.db.signed
```

Above I am copying all these manually to start off fresh without waiting (copy then restart named on secondary). ZSK and KSK are managed on the primary, the secondary server only gets public keys to verify signatures, are included in DNSKEY records sent during zone transfers. Auto-dnssec and inline-signing features simplifies this by managing keys and signatures automatically.

Add matching secondary server configuration (named.conf)

```
zone "example.com" {
   type slave;
   file "zones/example.com.db";
   masters { 192.168.0.1; };  // IP of the primary server
   allow-notify { 192.168.0.1; };  // Allow notifications from the primary server
   key "/etc/bind/tsig_key.conf";  // Referencing a defined TSIG key for secure communication
};
```

Above the secondary server also has been given the allow-notify line to get notifications from the primary. The lines for the primary were already added here, but you'll need these three lines in the primary for notifications:

```
   allow-transfer { key /etc/named/tsig_key.conf; };      // Secure transfers - reference key defined in tsig_key.conf
   notify yes;
   also-notify { 192.168.0.2; };          // Secondary server IP
```

Enable RNDC for Secure Remote Commands
RNDC (Remote Name Daemon Control) keys allow a secure secure control channel for commands to be issued to BIND (e.g., for restarting or reloading configurations).

Generating RNDC Keys
Use the rndc-confgen tool to generate an RNDC key. This command generates a key with 512 bits
rndc-confgen -a -b 512 -c /etc/rndc.key

Primary server congiuration
Configure rndc.conf and named.conf for RNDC:
rndc.conf:

```
key "rndc-key" {
    algorithm hmac-sha256;
    secret "base64-encoded-secret-key";  // Your RNDC secret key
};
options {
    default-key "rndc-key";
    default-server 127.0.0.1;  // RNDC server
    default-port 953;  // RNDC port
};
```

named.conf:

```
include "/etc/rndc.key";
controls {
    inet 127.0.0.1 port 953 {
        allow { 127.0.0.1, <secondary_serv_IP>, <workstation_IPs>; };   // Add IPs for machines that should have RNDC access
        keys { "rndc-key"; };
    };
};
```

Configuration on secondary server:
The rndc.conf file is generally identical so it can also be copied over. Changes to named.conf are the same as the primary.
Both the primary and secondary servers should have the same RNDC key if RNDC commands should control both servers.
Transfer RNDC key: scp /etc/rndc.key user@secondary-server:/etc/bind/rndc.key
Transfer RNDC config file: scp /etc/rndc.conf user@secondary-server:/etc/rndc.conf

Update named.conf to include the RNDC key:

```
include "/etc/rndc.key";
controls {
    inet 127.0.0.1 port 953 {
        allow { 127.0.0.1, <primary_serv_IP>, <workstation_IPs>; };
        keys { "rndc-key"; };
    };
};
```

Any machines that should be able to send RNDC commands to the servers will need BIND configured on them, have the key and have rndc conf files configured to do so. When running RNDC commands, specify the IP of the server:

```
rndc -s 192.168.1.10 -k /etc/rndc.key status
```

Centralized Management with LDAP

1. Adding LDAP in named.conf

```
dynamic-db "example" {
  library "ldap.so";
  arg "uri ldapi:///";  # No trailing slash here
  arg "base cn=dns,dc=example,dc=com";
  arg "auth_method sasl";
  arg "security sasl";  # Add this line for improved security
};
```

2. On the LDAP server, create a schema file, (e.g., /etc/dirsrv/slapd-instance_name/schema/99-dns.ldif) with this in it:

```
dn: cn=schema
objectClass: top
objectClass: ldapSubentry
objectClass: subschema
cn: schema
attributeTypes: ( 2.16.840.1.113730.3.1.16 NAME 'dNSDomain' EQUALITY caseIgnoreIA5Match ORDERING
caseIgnoreIA5OrderingMatch SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX '1.3.6.1.4.1.1466.115.121.1.26' USAGE
userApplications X-ORIGIN 'LDAP schema for DNS' )
attributeTypes: ( 2.16.840.1.113730.3.1.17 NAME 'dNSTTL' EQUALITY integerMatch SYNTAX '1.3.6.1.4.1.1466.115.121.1.27'
USAGE userApplications X-ORIGIN 'LDAP schema for DNS' )
attributeTypes: ( 2.16.840.1.113730.3.1.18 NAME 'dNSRecord' EQUALITY caseIgnoreIA5Match ORDERING
caseIgnoreIA5OrderingMatch SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX '1.3.6.1.4.1.1466.115.121.1.26' USAGE
userApplications X-ORIGIN 'LDAP schema for DNS' )
objectClasses: ( 2.16.840.1.113730.3.2.3 NAME 'dNSZone' SUP top STRUCTURAL MUST dNSDomain MAY ( dNSTTL $
dNSRecord ) X-ORIGIN 'LDAP schema for DNS' )
```

3. Create files for zones you want LDAP to handle

This file is for 2 zones. Add another identical block under these to add more. Save this in multiple-dns-zones.ldif

```
# Zone for xyz321.com
dn: dc=xyz321,dc=com
objectClass: top
objectClass: dNSZone
dNSDomain: xyz321.com
dNSTTL: 86400
dNSRecord: SOA ns1.xyz321.com. hostmaster.xyz321.com. (
        2024062601 ; Serial
        3600       ; Refresh
        1800       ; Retry
        1209600    ; Expire
        86400 )    ; Minimum TTL
dNSRecord: NS ns1.xyz321.com.
dNSRecord: NS ns2.xyz321.com.

# www.xyz321.com record
dn: cn=www,dc=xyz321,dc=com
objectClass: top
objectClass: dNSZone
dNSDomain: www.xyz321.com
dNSTTL: 86400
dNSRecord: A 192.0.2.1

# Zone for abc123.com
dn: dc=123abc,dc=com  [etc., just like the blocks above- do that for each one you want to add]
```

4. When completed, run this line of code (replace "cn=admin,dc=example,dc=com" with your actual LDAP admin DN)

```
ldapadd -x -D "cn=admin,dc=example,dc=com" -W -f multiple-dns-zones.ldif
```

You can verify the successful addition of these entries by using the ldapsearch command

```
ldapsearch -x -b "dc=xyz321,dc=com" -D "cn=admin,dc=example,dc=com" -W
```

Later you can update that same ldif file with more zones (or changes), run ldapadd again to update the database

This deletes the www record, not the zone

"ldapmodify -D "cn=admin,dc=example,dc=com" -W << EOF dn: cn=www,dc=xyz321,dc=com changetype: delete EOF"

I

f you delete a zone it will delete that zone's records too, so, to delete both you could just run the ldapmodify delete on "dn: dc=xyz321,dc=com" (the zone) and it will zap both on one shot!

Multi-Master DNS with Anycast:Load Balancing, High Availability, and Scalability
DNS Failover for fallback for continuous service

Multi-Master DNS ensures that DNS queries can be answered by any of the master servers, providing redundancy. If one server fails, others can continue to serve the zone data. Anycast allows multiple servers to share the same IP address, routing client requests to the nearest or best-performing server (for load balancing, distributing the query load evenly across servers. High Availability is enhanced because the system is resilient to individual server failures. Anycast addresses also improve performance by reducing latency, directing users to the closest server. DNS Failover involves monitoring the availability of services and rerouting traffic to backup servers when a primary server becomes unavailable.

Assume you have three servers:
DNS Server 1: 192.168.0.1 - DNS Server 2: 192.168.0.2 - Anycast IP: 203.0.113.10

On Server 1:

```
// named.conf on 192.168.0.1
options {
   directory "/var/named";                          // Default directory for zone files
   allow-transfer { 192.168.0.2; 203.0.113.10; };   // Allow zone transfers to other masters and Anycast
   also-notify { 192.168.0.2; 203.0.113.10; };      // Notify other masters and Anycast server of zone updates
   listen-on port 53 { 192.168.0.1; };              // Listen on the specific IP address for DNS queries
   listen-on-v6 { none; };                          // Disable IPv6 if not needed
};
logging {
   channel default_log {
      file "/var/log/named/default.log" versions 3 size 10M;
      severity info;
      print-time yes;
      print-severity yes;
   };
   category default { default_log; };
};
zone "example.com" {
   type primary;                                    // Master server type
   file "zones/db.example.com";                     // Location of the zone file
   allow-transfer { 192.168.0.2; 203.0.113.10; };   // Allow transfers to other masters and Anycast
   also-notify { 192.168.0.2; 203.0.113.10; };      // Notify other masters and Anycast server of changes
```

On DNS Server 2 (192.168.0.2):

```
// named.conf - all except what is below is identical to Server 1
options {
   allow-transfer { 192.168.0.1; 203.0.113.10; };
   also-notify { 192.168.0.1; 203.0.113.10; };
   listen-on port 53 { 192.168.0.2; };
};
zone "example.com" {
   allow-transfer { 192.168.0.1; 203.0.113.10; };
   also-notify { 192.168.0.1; 203.0.113.10; };
};
```

On Anycast DNS Server (203.0.113.10):

```
// named.conf - all except what is below is identical to Server 1
options {
   allow-transfer { 192.168.0.1; 192.168.0.2; };
   also-notify { 192.168.0.1; 192.168.0.2; };
   listen-on port 53 { 203.0.113.10; };
};
zone "example.com" {
   allow-transfer { 192.168.0.1; 192.168.0.2; };
   also-notify { 192.168.0.1; 192.168.0.2; };
};
```

Create and Configure the Zone Files
Ensure the zone file db.example.com is synchronized across all servers.

```
$TTL 86400 ; 1 day
@       IN SOA  ns1.example.com.  admin.example.com. (
                                  2024062901   ; serial
                                  3600         ; refresh (1 hour)
                                  1800         ; retry (30 minutes)
                                  1209600      ; expire (2 weeks)
                                  3600         ; minimum (1 hour)
                                  )
        NS      ns1.example.com.
        NS      ns2.example.com.
ns1     A       192.168.0.1
ns2     A       192.168.0.2
anycast A       203.0.113.10
www     A       192.168.0.3
```

Scalability and Performance Considerations
Allow-Transfer, Also-Notify: Expand these with additional IPs for new servers or Anycast instances as things scale.
Logging: Adjust log file sizes and the number of retained versions based on monitoring needs and storage capacity.
Listen-on, Listen-on-V6: Update these to include additional IPs or enable IPv6 (no IPv6 here just to save page space)

Testing and Validation
- DNS Query Testing:
  - Use dig to test DNS resolution from each server, including the Anycast IP
    If one isn't working you will likely get an NXDOMAIN instead of an IP address.
    dig @192.168.0.1 example.com +short; dig @192.168.0.2 example.com+short; dig @203.0.113.10 example.com
- Zone File Synchronization:
  - Update the zone file on one server and verify that the changes propagate to the others.
- Anycast Routing:
  - Confirm that queries to the Anycast IP are correctly distributed to the nearest or least-loaded server.

Response Policy Zones (RPZ): Policy-based managing and modifying DNS responses

Blocking, redirecting, or altering DNS queries to enforce security policies, control content, and manage network behavior dynamically.

RPZ Components
RPZ Zone Definition:
-- A DNS zone specifically created to hold RPZ data, generally named with a .rpz suffix. Can be primary or secondary

RPZ Policy Association:
-- The response-policy directive links a DNS zone to RPZ, specifying which zones' data will be used to apply policies.

RPZ Data File:
-- Contains SOA records and policy rules that dictate how to handle DNS queries based on matching criteria. They should not be placed with other zone files, but should be in the bind/named directories. On RHEL, I am using /etc/bind/rpz/

Basic RPZ Configuration
-- Define the RPZ zone (named.conf)

```
   # Create a DNS zone with an appropriate .rpz suffix
zone "example.rpz" {
   type master;                    # Define as master (primary)
   file "/etc/bind/db.example.rpz";   # Specify the file containing RPZ data
};
```

-- Associate the RPZ policy (named.conf)

```
   # Link the RPZ zone to BIND's response policy using the response-policy directive.
options {
   response-policy {
      zone "example.rpz";
   };
};
```

-- Populate the RPZ data file (example.rpz)

```
   # Define rules and actions in the RPZ data file, typically starting with SOA records followed by specific policy rules.

$TTL 60
@   IN SOA   ns1.example.com.  admin.example.com. (
                  2024062901          ; serial
                  3600                ; refresh
                  1800                ; retry
                  1209600             ; expire
                  60                  ; minimum TTL
                  )
      NS         ns1.example.com.

*.malicious-domain.com.   60   IN   CNAME   rpz-nxdomain.
*.old-service.com.        60   IN   CNAME   new-service.com.
```

Expanded RPZ Rules Table

response-policy	Specifies the policy zone(s) used for RPZ. Lists the zones and the order in which they apply.	Priority: Determines the order of zone application.
policy-zone	Defines a named policy zone that holds the RPZ rules.	Zone Source: File or feed from which the zone data is loaded.
match-clients	Matches client IP addresses or address ranges. Used to apply policies based on the source of the DNS query.	ACLs: Predefined Access Control Lists for matching clients.
match-destination	Matches based on the destination IP addresses in the DNS queries.	IP Range: Range or specific IP to match against.
match-subdomain	Matches queries based on subdomains. Useful for wildcard matching within a domain.	Wildcards: Supports wildcard entries for flexible matching.
match-type	Matches DNS queries based on their type (e.g., A, AAAA, MX). Allows policies to be applied to specific query types.	Query Type: List specific DNS query types (e.g., A, AAAA).
match-regex	Uses regular expressions to match DNS queries. Provides advanced matching criteria for complex policies.	Regex Pattern: Specify the regex pattern for query matching.
policy	Defines the action to be taken when a rule matches (e.g., NXDOMAIN, NODATA, CNAME). Actions dictate how matched queries are handled.	Actions List: Possible actions include NXDOMAIN, CNAME, etc.
qname-wait-recurse	Configures whether the RPZ should wait for recursion to complete before applying policies.	Boolean: True or False to enable/disable waiting.
qname-wait-time	Sets the maximum time to wait for a recursive query before applying the RPZ policies.	Time Value: Duration to wait for recursion (e.g., 2s).
break-dnssec	Specifies if DNSSEC validation should be disabled for RPZ responses.	Boolean: True (disable DNSSEC) or False (keep DNSSEC).
log	Enables or configures logging for RPZ matches, providing visibility into policy enforcement and aiding in troubleshooting.	Logging Level: Defines verbosity or specific logging actions.
rate-limit	Applies rate limiting to queries matching the RPZ rules, useful for mitigating abuse or attacks.	Rate Parameters: Max queries per second, burst sizes, etc.
feed-update	Specifies how often to update the RPZ zone from external feeds.	Update Frequency: Interval for feed updates (e.g., hourly).
source-policy	Defines policies based on the origin of the RPZ data, allowing differentiation of policies from different sources.	Source Specification: Identify sources and their policies.
inform-action	Configures additional actions (e.g., notify) when an inform action is taken, useful for alerting systems or administrators about policy matches.	Notification Methods: Email, SNMP traps, etc.

Expanded RPZ Actions Table

NXDOMAIN	Returns a "no such domain" response, effectively blocking the domain.	Default Action: Common for blocking malicious domains.
NODATA	Returns no data for the requested domain, blocking specific records while allowing others.	Use Case: Selective blocking of record types.
CNAME	Redirects the query to another domain by providing a canonical name.	Redirection Target: Specify the target domain for redirection.
PASSTHRU	Allows the query to bypass RPZ processing and be resolved normally.	Exception Handling: Useful for whitelisting specific queries.
DROP	Silently drops the query, providing no response, effectively blackholing it.	Application: Stops traffic to/from known bad actors.
TCP-Only	Forces the DNS query to be resolved over TCP instead of UDP.	Performance Impact: Adds overhead due to TCP's nature.
Redirect	Responds with a specific IP address, redirecting the traffic.	Redirection IP: Target IP for redirection.
Inform	Logs the query without modifying the response, used for monitoring and auditing DNS traffic.	Logging Level: Determines the verbosity of the logs.
Alter	Modifies the DNS response data, such as changing the TTL or other record attributes.	Modification Details: Specify the changes to be applied.
Override	Replaces the response with a predefined answer, useful for internal redirections or custom responses.	Override Content: The custom response data.
Fake-IP	Responds with a false IP address, typically used to redirect traffic to a controlled or null destination.	Fake IP Address: The IP to be returned.
Geo-Location	Uses geographical information to tailor the response, often used for content localization or restriction.	Geo-Parameters: Regions or countries to target.
Notify	Sends notifications or triggers actions when specific RPZ rules are matched.	Notification Methods: Email, webhook, etc.
Rewrite	Alters parts of the query or response, such as changing the domain name or resource record being queried.	Rewrite Rules: Specify how the query or response is rewritten.
Throttle	Limits the rate of responses to certain queries to mitigate abuse or DDoS attacks.	Throttle Parameters: Max queries per second, burst sizes, etc.

Advanced Use Cases for RPZ

Using External Blacklist Data
Community-maintained RPZ feeds can be used to block known malicious domains. DNSBLs can use thesame method
Many sites with RPZs want you to sign up for a trial of thier product- this one didn't - https://urlhaus.abuse.ch/downloads/rpz/
sudo wget -O /etc/bind/rpz/urlhaus.rpz https://urlhaus.abuse.ch/downloads/rpz/

```
        # Configure RPZ zone
zone "urlhaus.rpz" {
   type slave;
   file "/etc/bind/rpz/urlhaus.rpz";
   masters { 192.0.2.1 key "local_keyfile.key"; };       // Defines the master server and the key for secure zone transfers
};
        # Policy directive
options {
   recursion yes;
   response-policy { zone "urlhaus.rpz" policy nxdomain; };   // Matches get a NXDOMAIN response, effectively blocking them.
};
        # RPZ Data - /etc/bind/rpz/urlhaus.rpz
$TTL 30
@ SOA rpz.urlhaus.abuse.ch. hostmaster.urlhaus.abuse.ch. 2407010019 300 1800 604800 30
 NS localhost.
; abuse.ch URLhaus Response Policy Zones (RPZ)
; Last updated: 2024-07-01 00:19:23 (UTC)
; Terms Of Use: https://urlhaus.abuse.ch/api/
; For questions please contact urlhaus [at] abuse.ch
testentry.rpz.urlhaus.abuse.ch CNAME . ; Test entry for testing URLhaus RPZ
1.bdl99down.kukulaa.cn CNAME . ; Malware download (2024-05-30), see https://urlhaus.abuse.ch/host/1.bdl99down.kukulaa.cn/
139520.aioc.qbgxl.com CNAME . ; Malware download (2024-05-06), see https://urlhaus.abuse.ch/host/139520.aioc.qbgxl.com/
```

Internal DNS Redirection
Redirect queries for internal domain names to appropriate external addresses.

```
        # Configure RPZ zone
zone "internal.rpz" {
   type master;
   file "/etc/bind/rpz/db.internal.rpz";
};
        # Policy directive
options {
   response-policy {
      zone "internal.rpz";
   };
};
        # RPZ Data - /etc/bind/rpz/db.internal.rpz
```

Define redirection rules in the RPZ data file. These assume internal private DNS names inside company are *.corp

```
$TTL 60
@   SOA  ns1.example.com. admin.example.com. (
      2024062901; serial
      3600        ; refresh
      1800        ; retry
      604800      ; expire
      60 )        ; TTL
;
server1.corp  60  IN  CNAME  server1.example.com.
server2.corp  60  IN  CNAME  server2.example.com.
```

Rate Limiting DNS Queries
This configuration limits responses to 10 per second for queries matching the abuse domain.

```
        # Configure RPZ zone
zone "rtlimit-ddos.rpz" {
   type master;
   file "/etc/bind/rpz/rtlimit-ddos.rpz";
};
```

```
            # Policy directive
options {
   response-policy {
            zone "/etc/bind/rpz/rtlimit-ddos.rpz" policy passthru;
            rate-limit { responses-per-second 10; };
        };
            # RPZ Data -/etc/bind/rpz/rtlimit-ddos.rpz
$TTL 60
@    SOA  ns.example.com. admin.example.com. (
       2023062503; serial
       3600           ; refresh
       1800           ; retry
       604800         ; expire
       60 )           ; TTL
         ;
abuse.example.com  A  127.0.0.1
```

Geolocation-Based Content Control
Redirect queries based on the geographical location of the request.

```
            # Configure RPZ zone
zone "geo.rpz" {
   type master;
   file "/etc/bind/rpz/db.geo.rpz";
};
            # Policy directive
options {
   response-policy {
      zone "geo.rpz";
   };
};
            # RPZ Data - /etc/bind/rpz/db.geo.rpz
```
Route traffic to different servers based on region-specific subdomains.
```
$TTL 60
@   SOA  ns1.example.com. admin.example.com. (
       2024062901; serial
       3600           ; refresh
       1800           ; retry
       604800         ; expire
       60 )           ; TTL
;
us.region.example.com  60  IN  CNAME  us-content.example.com.
eu.region.example.com  60  IN  CNAME  eu-content.example.com.
```

Redirecting Traffic
Redirect traffic from a deprecated service to a new domain.

```
zone "redirects.rpz" {
   type master;
   file "/etc/bind/db.redirects.rpz";
};
            # Policy directive
response-policy {
   zone "redirects.rpz";          // internal policy rules
};
            # RPZ Data /etc/bind/rpz/redirects.rpz
$TTL 60
@    IN SOA ns1.example.com. hostmaster.example.com. (
       2024062901; serial
       3600           ; refresh
       1800           ; retry
       604800         ; expire
       60 )           ; TTL
   )
   IN NS ns1.example.com.

*.old-service.com. 60 IN CNAME new-service.com.
```

DNS Rcodes (Return Codes)

NOERROR	No Error. Query successful; valid response.
FORMERR	Format Error (unable to interpret query format.)
SERVFAIL	Server error processing request;
NXDOMAIN	FQDN doesn’t exist; "NXDOMAIN"
NOTIMPL	Server doesn't support query type
REFUSED	Server refused query; likely a policy, zone transfer
NOTAUTH	Server not authoritative for the zone.
NOTZONE	Name not found in the zone.
PREREQ	Failed prerequisites: YXDomain, YXRRSet, NXRRSet.

Domain Statuses (reported in whois queries)

Depending on the Top Level Domain (TLD), there can be different status names used by the underlying registry.

.COM and .NET domain names

ACTIVE	Domain registered; functioning for websites or email.
REGISTRAR-HOLD	On hold by registrar; contact registrar.
REGISTRY-HOLD	On hold by registry; contact registrar.
REGISTRAR-LOCK	Registrar locked domain; settings can't change.
REGISTRY-LOCK	Registry locked domain; settings can't change.
REDEMPTIONPERIOD	Domain expired; 30-day hold before release.
PENDINGRESTORE	Expired domain being restored to ACTIVE.
PENDINGDELETE	Domain expired 75 days ago; deleting soon.

.ORG, .BIZ, and .INFO domain names

OK	Domain active; usable for websites, email, or name servers.
CLIENT_DELETE_PROHIBITED	Registrar locked domain; cannot delete.
SERVER_DELETE_PROHIBITED	Registry locked domain; cannot delete.
CLIENT_HOLD	Registrar on hold; domain unusable.
SERVER_HOLD	Registry on hold; domain unusable.
CLIENT_RENEW_PROHIBITED	Registrar locked domain; cannot renew.
SERVER_RENEW_PROHIBITED	Registry locked domain; cannot renew.
CLIENT_TRANSFER_PROHIBITED	Registrar locked domain; cannot transfer.
SERVER_TRANSFER_PROHIBITED	Registry locked domain; cannot transfer.
CLIENT_UPDATE_PROHIBITED	Registrar locked domain; settings can't change.
SERVER_UPDATE_PROHIBITED	Registry locked domain; settings can't change.
INACTIVE	Domain unusable; name server issues.
PENDING_DELETE	Domain registration about to delete.
PENDING_TRANSFER	Domain transferring; no modifications allowed.
PENDING_VERIFICATION	Registry creating domain record.

DNS message format

Message ID									
QR	OPCODE	AA	TC	RD	RA	RE	AD	CD	RCODE
QDCOUNT									
ANCOUNT									
NSCOUNT									
ARCOUNT									

QR	Query () or response ()	Is the message is a query (0) or a response (1)?
OPCODE	Type	Type of message being sent: Query (0), Inverse Query (1), Status Request (2)
AA	Authoritative answer	Indicates the name server is authoritative for queried domain
TC	Truncation	Response was truncated due to exceeding the size limit (UDP)
RD	Recursion desired	The client requests recursive resolution
RA	Recursion available	The DNS server supports recursive queries
RE	Reserved for future use	
AD	Authenticated data (DNSSEC)	Answer and authority sections were authenticated
CD	Checking disabled (DNSSEC)	Client requests the DNS server to disable DNSSEC validation
RCODE	Response type	[NoError (0), FormatError (1), ServerFailure (2), NameError (3), etc.]
QDCOUNT	Question Count	Number of client queries to server in the Question section
ANCOUNT	Answer Count	Number of resource records in the Answer (actual answers)
NSCOUNT	Authority Count	Number of resource records (authoritative name servers) in Authority section
ARCOUNT	Additional Count	Number of resource records in the Additional section

Utilities in the bind-utils package: dig, nslookup, host

The **host** command for DNS lookup (using OS resolver libraries for queries.
-a: Displays all records for a domain (same as -v). -l: Lists all hosts in the specified zone (zone transfer).
-C: Checks DNS configuration and zone files. -t type: DNS record to query (e.g., A, MX, NS, etc.).
-W <seconds> (timeout), -R retries, -T: Forces TCP instead of UDP, -4: IPv4 only, -6: IPv6 onl, -v: verbose

The **nslookup** command for DNS queries directly to DNS servers
-type or -query: DNS record to query (e.g., A, MX, TXT). -retry: number of retries.
-server: DNS server to query -timeout <seconds> (query timeout)
-vc: Use TCP for queries (a virtual circuit) -debug: Debugging mode for detailed output.

The **whois** command for info about domain registrations and IP address allocations (not part of bind-utils)
-h: Connects to a specified WHOIS server instead of the default. -l: Recursive domain registration lookup
-p: Connects to a specified port on the WHOIS server. -r: Does a raw WHOIS query (no special handling of data).
-I: Searches for IP networks. -H: Disables the display of the legal disclaimers in the output.

The **dig** command to query domain records, name servers, and troubleshooting DNS

dig example.com ANY	Fetches all available DNS records (A, MX, NS, TXT, etc.)
dig example.com +trace	Show full path of servers in resolving domain from root servers to the authoritative DNS servers
dig example.com +dnssec	Get DNSSEC-related information for example.com (DNSKEY, RRSIG, etc.)
dig -x 192.0.2.1	Reverse DNS lookup of 192.0.2.1, getting associated domain name
dig @8.8.8.8 example.com +noall +answer +stats	Query 8.8.8.8 for example.com, show only the answer section, stats

Dig Query Options

+[no]additional	Displays or omits the additional section data in responses. Default is to display it.
+aaflag	Sets the AA (Authoritative Answer) flag in the query.
+adflag	Sets the AD (Authentic Data) bit in the query to indicate DNSSEC validation status.
+[no]all	Sets or clears all display flags. Default is not to clear any flags.
+[no]authority	Displays or omits the authority section of a reply. Default is to display it.
+[no]cmd	Toggles the printing of the initial comment in the output with version and query options info.
+[no]comments	Toggles the display of comments in the output, (packet headers and OPT pseudosection info).
+[no]crypto	Toggles the display of cryptographic fields in DNSSEC records.
+[no]dns64prefix	Looks up IPV4ONLY.ARPA AAAA and prints any DNS64 prefixes found.
+dnssec	Requests DNSSEC records and sets the DO (DNSSEC OK) bit in the query.
+domain=NAME	Sets the search list to contain a single domain NAME.
+edns[=#]	Specifies the EDNS version to query with. Default is 0.
+[no]ednsflags[=#]	Sets or clears the EDNS flags (Z bits) in the query.
+[no]ednsnegotiation	Enables or disables EDNS version negotiation. Default is enabled.
+ednsopt[=code[ ]]	Specifies an EDNS option with code and optional value in hexadecimal.
+[no]expire	Sends/ does not send an EDNS Expire option.
+[no]fail	Indicates to retry or not retry with another server if a SERVFAIL is received. Default is not to retry.
+[no]identify	Shows or hides IP addr and port of the responding server when +short is enabled. Default is to hide.
+[no]idnout	Converts/ does not convert punycode on output. Default is to process when output is a tty.
+[no]multiline	Prints records in a verbose, multi-line format or not. Default is not to print in multiline.
+ndots=D	Number of dots that must be in name before is considered absolute. Default is in /etc/resolv.conf or 1.
+[no]nssearch	Finds/ does not find authoritative name servers for the zone and displays their SOA records.
+opcode=value	Sets the DNS message opcode to the specified value. Default is QUERY (0).
+padding=value	Pads the size of the query packet to blocks of value bytes. Default is 0 (no padding).
+qid=value	Specifies the query ID to use for sending queries.
+recurse	Enables recursion in the query. This bit is set by default.
+short	Provides terse answer format or not. Default is verbose.
+[no]tcp	Uses/ does not use TCP for querying name servers. Default is to use UDP unless required otherwise.
+[no]trace	Traces/ does not trace the delegation path from the root name servers. Default is not to trace.
+time=T	Sets the query timeout to T seconds. Default is 5 seconds.
+[no]tls	Uses/ does not use DNS over TLS (DoT) for querying. Default port is 853.
+tls-ca[=file-name]	Validates server TLS certificates using the specified CA file. Uses default CA store if not specified.
+tls-hostname=hostname	Uses the specified hostname for TLS certificate verification.
+[no]tries=T	Sets the number of times to try UDP and TCP queries. Default is 3.
+[no]split=W	Splits long hex- or base64-formatted fields into chunks of W characters. Default is 56 or 44 in multiline.
+[no]stats	Toggles the printing of statistics. Default is to print statistics.
+[no]subnet=addr/prefix	Uses the EDNS Client Subnet option with the specified address and prefix.
+subnet=addr[@src...	+subnet=addr[@src-prefix]/prefix Specifies EDNS Client Subnet addr, source and destination prefix.
+tls-only	Forces the use of DNS over TLS (DoT) only for the query.

Mail Server Provisioning on RHEL 8/9 (Enterprise-ready)

Installing Postfix, Dovecot with CRAM-MD5 and TLS/SSL (STARTTLS) via SASL with Kerberos integration and SSL

Install Postfix core component for handling SMTP mail delivery. Dovecot for IMAP and POP3 to end user mail clients

sudo dnf install postfix dovecot-imap dovecot-pop3d dovecot-sieve

Provide SASL and libraries for authentication- plus openSSL's TLS/STARTTLS libraries for encryption.

sudo dnf install openssl cyrus-sasl cyrus-sasl-lib cyrus-sasl-devel cyrus-sasl-gssapi cyrus-sasl-ldap cyrus-sasl-scram cyrus-sasl-md5 cyrus-sasl-plain openssl krb5-workstation

What is installed:

Postfix (postfix):

/usr/sbin/postfix: The Postfix control program.
/usr/bin/postconf: Configuration utility for simplifying editing /etc/postfix/main.cf (main config file)
/usr/sbin/master: The master daemon that manages Postfix processes.
/usr/sbin/smtpd: The Postfix SMTP server.
/etc/postfix/main.cf: Main Postfix configuration file.
/etc/postfix/master.cf: The configuration file for Postfix daemon processes.
/etc/postfix/sasl_passwd: Stores credentials for Postfix to authenticate to external SMTP servers for relaying outbound mail.
/var/spool/postfix: Mail queue directory.

Dovecot:

/usr/sbin/dovecot: The Dovecot main daemon.
/etc/dovecot: Contains all Dovecot configuration files.
/etc/dovecot/conf.d/*.conf: Individual configuration files for different Dovecot features (e.g., 10-mail.conf, 10-auth.conf).
/var/lib/dovecot: Stores mail user data and mailbox information.
/var/run/dovecot: Location for Dovecot socket files used for communication.

Cyrus SASL:

/usr/bin/saslpasswd2: Manages SASL authentication credentials (if using FreeIPA).
/usr/sbin/saslauthd: The SASL authentication daemon.
/usr/lib/libsasl2.so: Main shared library for SASL.
/usr/lib/sasl2/: This directory holds libraries and plugins for all the SASL mechanisms (see section for those).
/etc/default/saslauthd: Configuration for saslauthd daemon.
/etc/sasl2/smtpd.conf: Common mechanisms and options for SMTP servers
/etc/sasl2/imapd.conf: For configuring SASL for IMAP applications.
/usr/lib/sasl2/libsasl2.conf: Global SASL library settings (rarely changed).
/etc/sasl2/sampleapp.conf: Example configurations for alternate or custom authentication backends like SQL, etc.

OpenSSL:

/usr/bin/openssl: The command-line tool for using OpenSSL's features.
/etc/pki/tls/openssl.cnf: The OpenSSL configuration file.

Commands and Subcommands Used in Configuration and Management

-- <u>Postfix:</u>	
postfix start | stop | reload	Starts and stops the server, or reload from config file without restarting
postfix check	Checks the Postfix configuration for errors.
postconf -p	View current Postfix configuration settings.
postconf -e 'parameter=value'	Set a specific Postfix parameter.
postqueue -f	Show details of a specific mail message in the queue.
postfix flush	Flush the mail queue.
-- <u>Dovecot:</u>	
systemctl stop | start | restart dovecot	Starts, stops, or reloads the Dovecot service.
dovecot -n	Shows the running Dovecot processes and their PIDs.
dovecot -d	Enables debugging modegenerating more detailed logs.
-- <u>SASL Config:</u>	
systemctl start | stop | reload saslauthd	Starts, stops, or reloads the SASL service.
saslpasswd2	Manages SASL authentication credentials.
saslpasswd2 -a mechanism user -p	Creates user password for a specific mechanism (e.g., ldap, etc).
saslauthd -a <daemon>	Starts the SASL auth daemon (e.g., pam, ldap, shadow, sasldb)
-- <u>OpenSSL for Cert Management:</u>	
openssl genrsa -out <key_filename>.key 2048	Generates a private RSA key for server use.
openssl verify <cert_filename>.crt	Verifies the validity of your server certificate.
openssl req -new -key <key_file>.key -out <csr_file>.csr	Make Cert Signing Request (CSR) using private key.
openssl x509 -req -days 365 -in <csr_file>.csr -CA cert.pem -CAkey key.pem -CAcreateserial -out <cert_file>.crt	Signs the CSR to generate a server certificate using a Certificate Authority (CA).

Important Files and Directories:

-- Postfix:	
/etc/postfix	Contains all Postfix configuration files.
/etc/postfix/main.cf	Main Postfix configuration file.
/etc/postfix/master.cf	Configuration file for Postfix daemon processes.
/var/spool/postfix	Mail queue directory.
-- Dovecot:	
/etc/dovecot	Contains all Dovecot configuration files.
/etc/dovecot/conf.d/*.conf	Configs for features (e.g., 10-auth.conf is authentication, 10-mail.conf is mail storage).
/var/lib/dovecot	Stores mail user data and mailbox information.
/var/run/dovecot	Location for Dovecot socket files used for communication.
Cyrus SASL - /etc/sasl2/smtpd.conf	Configuration for SASL mechanisms and options for SMTP authentication.
OpenSSL - /etc/pki/tls/openssl.cnf	The OpenSSL configuration file.

SASL Mechanism Plugin Libraries Installed by cyrus-sasl Packages

By installing these SASL plugins, you can configure Postfix (or other services using SASL like LDAP) to support a broad range of authentication mechanisms suitable for different security requirements and environments. They get installed in /usr/lib/sasl2/

GSSAPI (Kerberos)	libgssapiv2.so	Use GSSAPI, commonly used with Kerberos for authentication (SSO).
LDAP	libldapdb.so	Authenticate via one communication method but using LDAP to check credentials
SCRAM-SHA	libscram.so	Salted Challenge Response Auth with hashing, a CRAM-MD5 replacement
CRAM/DIGEST-MD5	libcrammd5.so	Challenge-response plus hashing, DIGEST-MD5 for mutual auth, etc.
PLAIN	libplain.so	Plaintext password sent. Use when traffic already encrypted (e.g., TLS) or testing
ANONYMOUS	libanonymous.so	Allows clients to authenticate anonymously, no credentials
LOGIN	liblogin.so	Like PLAIN, plaintext credentials, for legacy systems or basic setups with TLS/SSL

Postfix/Dovecot/SASL Installation and Configuration Guide

These steps provide functional email server setup, which can scale to 500+ users with enough hardware and network

Although mentioned at the very beginning, here again are the items you need to install in one big chunk

Install Postfix core component for handling SMTP mail delivery. Dovecot for IMAP and POP3 to end user mail clients

```
sudo dnf install postfix dovecot-imap dovecot-pop3d dovecot-sieve
```

Install SASL and libraries for authentication- plus openSSL's TLS/STARTTLS libraries for encryption and kerberos client.

```
sudo dnf install openssl cyrus-sasl cyrus-sasl-lib cyrus-sasl-devel cyrus-sasl-gssapi cyrus-sasl-ldap cyrus-sasl-scram cyrus-sasl-md5 cyrus-sasl-plain krb5-workstation
```

Generate Server Private Key with OpenSSL

```
# Generate the server's private key in  /etc/ssl/private/server.key. This way won't ask for passphrase after rebooting
sudo openssl genpkey -algorithm RSA -out /etc/ssl/private/server.key 2048
# Generate a Certificate Signing Request (CSR) used to generate the certificate:
sudo openssl req -new -key /etc/ssl/private/server.key -out /etc/ssl/certs/server.csr
# Create a self-signed certificate. For production use, it's recommended to get a cert from a trusted CA.
sudo openssl x509 -req -days 365 -in /etc/ssl/certs/server.csr -signkey /etc/ssl/private/server.key -out /etc/ssl/certs/server.crt
# That spit out /etc/ssl/certs/server.crt.  Finally, secure the private key file access is limited to the root user
sudo chmod 600 /etc/ssl/private/server.key
sudo chown root:root /etc/ssl/private/server.key
```

Kerberos Client Configuration

Edit /etc/krb5.conf and replace the following with your kerberos server's information:

```
[libdefaults]
   default_realm = YOUR.REALM
   dns_lookup_realm = false
   dns_lookup_kdc = true
[realms]
   YOUR.REALM = {
      kdc = kdc.example.com
      admin_server = kdc.example.com
   }
[domain_realm]
   .example.com = YOUR.REALM
   example.com = YOUR.REALM
```

Save the config, then create a service principal for Postfix by running ktpasswd (generaties key):

```
ktpasswd -q -h /etc/postfix/sasl/postfix.keytab postfix@EXAMPLE.COM
```

SASL Base Configuration: saslauthd
This example prioritizes using Kerberos first to check credentials, then LDAP if kerberos isn't working

Edit /etc/sysconfig/saslauthd:
```
        # Enable saslauthd on startup
START=yes
        # Specify the authentication mechanisms
MECHANISMS="kerberos5:ldap"
        # Often this will be set to "ldap" which is fine. I want my config to try kerberos first then use ldap if it can't.
OPTIONS="-c -m /var/run/saslauthd -r"
# Here, the -c enables cache, -m sets the directory for the mux, and -r includes realm in username
```

Mux in this context refers to the Unix domain socket directory that is used to multiplex authentication requests from different client applications. This allows for secure and efficient IPC (Inter-Process Communication) for authentication purposes.
```
        # LDAP settings for saslauthd
ldap_servers: ldap://ldap.example.com
ldap_search_base: ou=users,dc=example,dc=com
ldap_filter: (uid=%u)
ldap_bind_dn: cn=admin,dc=example,dc=com
ldap_password: your_password
```

Configure Postfix [Mail Transfer Agent (MTA) responsible for receiving and sending emails]

Edit the main Postfix configuration file /etc/postfix/main.cf
Add or update the following settings:
```
        # Set the hostname for the mail server
myhostname = mail.example.com
        # Define the domain name
mydomain = example.com
        # Set the origin domain
myorigin = $mydomain
        # Specify the network interfaces Postfix will listen on (localhost and all IPv4 addresses)
inet_interfaces = all
        # Restrict to the domain and subdomains specified in mydestination
mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
        # Define the mailbox location for users
home_mailbox = Maildir/
        # Virtual aliases mapping
virtual_alias_maps = hash:/etc/postfix/virtual
        # Enable SMTP TLS
smtpd_tls_cert_file = /etc/ssl/certs/server.crt
smtpd_tls_key_file = /etc/ssl/private/server.key
        # Use TLS for incoming connections
smtpd_use_tls = yes
smtpd_tls_security_level = encrypt  # Changing to 'may' makes TLS optional
smtpd_tls_auth_only = yes
smtpd_tls_loglevel = 2       # Logging level for TLS transactions
smtpd_tls_received_header = yes   # Add TLS status to the received header
        # Session caching to improve performance
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
        # Disable outdated SSL protocols
smtpd_tls_mandatory_protocols = !SSLv2, !SSLv3
smtpd_tls_protocols = !SSLv2, !SSLv3
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination
```

Configure support for SASL
Edit /etc/postfix/sasl/smtpd.conf and put in the following
```
        # Specify SASL mechanisms Postfix should support
pwcheck_method: saslauthd
mech_list: GSSAPI SCRAM-SHA-256 SCRAM-SHA-1 DIGEST-MD5 CRAM-MD5
        #  mechlist order for comm methods- saslauthd says try kerberos for checking credentials, use LDAP if it fails
        # GSSAPI (for kerberos) Configuration, tell it where the server key is
keytab: /etc/postfix/sasl/postfix.keytab
```

Set up virtual alias mapping:
Create or edit the /etc/postfix/virtual file to define email address mappings:
sudo nano /etc/postfix/virtual
```
        # Add entries to map email addresses to local usernames
user1@example.com user1
user2@example.com user2
        # Save it, then update the Postfix virtual alias database:
sudo postmap /etc/postfix/virtual
```

Configure Dovecot [IMAP and POP3 services for retrieving emails]

Edit the main Dovecot configuration file /etc/dovecot/dovecot.conf
Add or update the following settings:
```
        # Enable IMAP and POP3 protocols
protocols = imap pop3
        # Specify the location for mailbox data
mail_location = maildir:~/Maildir
        # Set the hostname for IMAP and POP3 services
hostname = mail.example.com
        # Enable the use of SSL/TLS
ssl = yes
ssl_cert = </etc/ssl/certs/server.crt
ssl_key = </etc/ssl/private/server.key
        # Allow all users to access their mail
userdb {
  driver = passwd
}
```

Specify location for mail storage
Edit /etc/dovecot/conf.d/10-mail.conf
```
        # Add this:
mail_location = maildir:~/Maildir
```

Configure Dovecot's authentication options
Open /etc/dovecot/conf.d/10-auth.conf:
```
        # Ensure the following settings are enabled:
disable_plaintext_auth =  yes
        # Enable specified authentication mechanisms
auth_mechanisms = gssapi scram-sha-256 scram-sha-1 digest-md5 cram-md5
        # Why no LDAP above? auth_mechanisms are communication methods used- not to check or store credentials

        # LDAP Authentication Settings
passdb {
  driver = ldap
  args = /etc/dovecot/dovecot-ldap.conf.ext
}
userdb {
  driver = ldap
  args = /etc/dovecot/dovecot-ldap.conf.ext
}
        # Add this if Dovecot needs to directly talk to kerberos without SASL (or using Postfix as a proxy)
auth_gssapi_hostname = yourhostname.example.com
```

Make or edit config extention file for basic LDAP info
Edit /etc/dovecot/dovecot-ldap.conf.ext to specify the LDAP server details for verifying credentials
```
        # Add the following items:
hosts = ldap.example.com
dn = cn=admin,dc=example,dc=com
dnpass = your_password
base = ou=users,dc=example,dc=com
scope = subtree
user_attrs = uid=user,homeDirectory=home,uidNumber=uid,gidNumber=gid
user_filter = (&(objectClass=posixAccount)(uid=%u))
pass_filter = (&(objectClass=posixAccount)(uid=%u))
```

Dovecot Alternate Config (Use PAM to use kerberos, then try LDAP)

I wanted to prioritize kerberos over LDAP, and then try other methods after trying those two first. This meant invoking /etc/sysconfig/saslauthd to handle the order of things. A second solution was to pass things to a PAM config for Dovecot which would then handle things in the order I wanted them.

-- /etc/dovecot/conf.d/10-auth.conf

```
# Enable specified authentication mechanisms for Dovecot
auth_mechanisms = gssapi scram-sha-256 scram-sha-1 digest-md5 cram-md5
# Set up password database to use pam, which refers to saslauthd
passdb {
  driver = pam
  args = session=yes dovecot
}
# User database using system's passwd file (could also be ldap if needed)
userdb {
  driver = passwd
}
```

[The file /etc/dovecot/dovecot-ldap.conf.ext is still relevant, but identical and removed here since it is redundant]

-- Configure PAM to for Dovecot to delegate order of operations

PAM configuration files (stored in /etc/pam.d) generally have three domains of a programs operations they address:
- Authentication management specifies how to verify the user's identity- to authenticate the user based on local system credentials, kerberos, LDAP, biometric input, SSO, or MFA, etc
- Account Management verifies if the user's account is in good standing and can be used for login (is not disabled or expired)
- Session Management configures settings for the session after successful login, to manage session environment, apply system resource limits, set environment variables, configure user namespaces for the session, or configure other session-related aspects.

Create or modify the PAM configuration file for Dovecot, /etc/pam.d/dovecot:

-- /etc/pam.d/dovecot:

```
#%PAM-1.0
# Authentication management (these are listed in order they are to be attempted)
auth       required     pam_unix.so nullok
auth       sufficient   pam_krb5.so use_first_pass
auth       sufficient   pam_ldap.so use_first_pass
auth       requisite    pam_succeed_if.so uid >= 1000 quiet_success
auth       required     pam_deny.so

# Account management
account    required     pam_unix.so
account    sufficient   pam_krb5.so
account    sufficient   pam_ldap.so

# Session management
session    required     pam_limits.so
session    sufficient   pam_krb5.so
session    optional     pam_ldap.so
```

 # pam_unix.so: Checks the local /etc/passwd and /etc/shadow files.
 # pam_krb5.so: Integrates Kerberos for authentication.
 # pam_ldap.so: Integrates LDAP for authentication.
 # use_first_pass: Passes the password from the first module to subsequent ones to avoid prompting the user multiple times.

-- Restart Services:

After editing the PAM configuration, restart the relevant services to apply the changes.

```
systemctl restart saslauthd
systemctl restart dovecot
```

You can test PAM functionality with tools like `dovecot auth test <username>`

Configuration Tips for Large Scale Deployments

Increase Connection and Message Limits (Postfix):

Adjust parameters in /etc/postfix/main.cf to handle more connections and larger message volumes:

```
default_process_limit = 100
smtpd_recipient_limit = 1000
smtpd_client_connection_count_limit = 50
smtpd_client_message_rate_limit = 100
```

Optimize queue management to handle large volumes of email efficiently (Postfix):

```
queue_minfree = 10000000
```

Optimize to handle a high number of simultaneous IMAP/POP3 connections (Dovecot):

```
service imap-login {
   process_min_avail = 16
   service_count = 0
   client_limit = 4096
}
service pop3-login {
   process_min_avail = 16
   service_count = 0
   client_limit = 4096
}
```

Enable mailbox indexing to speed up mailbox operations (Dovecot):

```
mail_plugins = $mail_plugins imap_quota
```

Leveraging iRedMail for Administrative Tasks

- Ease of use streamlines managing a robust mail server
- Simplifies implementing measures such as SPF, DKIM, and DMARC; spam filtering and user management
- A web-based administration panel (iRedAdmin) which simplifies the management of domains, user accounts, and mail server settings. Can integrate webmail clients like Roundcube and SOGo, which provide users with a feature-rich email interface.

Installation

```
wget https://github.com/iredmail/iRedMail/archive/refs/tags/1.5.2.tar.gz
tar zxvf 1.5.2.tar.gz
cd iRedMail-1.5.2
```

Run the Installation Script: 'sudo bash iRedMail.sh'

The interactive installation wizard will prompt for various configurations, such as mail storage path. web server selection, database selection for user and domain management (various SQL and LDAP options), domain and admin email settings. Post-installation, iRedMail will provide you with URLs for accessing the web admin interface (iRedAdmin) and webmail. Save the generated credentials for the admin user. Restart the server if necessary to apply all changes.

Integrating iRedMail Security Features

SPF (Sender Policy Framework) configures SPF automatically. This record specifies that only the mail servers specified in the MX records are allowed to send email for your domain. Ensure your DNS records include SPF settings:

```
example.com. IN TXT "v=spf1 mx -all"
```

DKIM (DomainKeys Identified Mail):

During the iRedMail installation, DKIM keys are generated and configured.

Add the DKIM public key to your DNS records as instructed in the post-installation steps it will be something like this:

```
default._domainkey.example.com IN TXT "v=DKIM1; k=rsa;
p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDYmG... (rest of the public key)"
```

The part of that that says default can be any random string... unique is prefered but not manditory, allows for multiple keys so you can give them different selector names to help differenciate them

DMARC (Domain-based Message Authentication, Reporting, and Conformance) This record specifies that emails failing DMARC checks should be quarantined and reports sent to the specified address.

Create a DMARC record in your DNS to enforce policies on email sending:

```
_dmarc.example.com. IN TXT "v=DMARC1; p=quarantine; rua=mailto:dmarc-reports@example.com"
```

Spam Filtering:

iRedMail uses tools like Amavisd, SpamAssassin, and ClamAV. These are pre-configured, fine-tuning can be done

Adjusting Postfix and Dovecot Security Configs

Iredmail can handle generating SSL keys and certificates as part of its setup process, with the option to use self-signed, Let's Encrypt, or a commercial CA-provided cert. These should be added to your mail server configs. Be sure to comment the old lines out so they don't get lost

Postfix TLS/SSL configuration edit /etc/postfix/main.cf

```
smtpd_tls_cert_file = /etc/ssl/certs/iRedMail.crt
smtpd_tls_key_file = /etc/ssl/private/iRedMail.key
smtpd_tls_CAfile = /etc/ssl/certs/iRedMail_CA.crt
smtpd_use_tls = yes
smtpd_tls_session_cache_database = btree:${data_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache
smtpd_tls_security_level = encrypt
smtp_tls_security_level = encrypt
```

Dovecot SSL/TLS configuration:

It says to put this in /etc/dovecot/conf.d/10-ssl.conf but you can just put them in /etc/dovecot/dovecot.conf

```
ssl = required
ssl_cert = </etc/ssl/certs/iRedMail.crt
ssl_key = </etc/ssl/private/iRedMail.key
ssl_ca = </etc/ssl/certs/iRedMail_CA.crt
```

Restart the services to apply the changes:

```
sudo systemctl restart postfix dovecot
```

Other mail server-related topics of interest

DKIM (DomainKeys Identified Mail):

DKIM is a crucial email authentication standard that helps prevent email spoofing and phishing attacks. It uses digital signatures to verify the sender's identity. When you receive an email, the DKIM signature is checked against the sender's domain to ensure it wasn't forged. A failing DKIM signature might indicate a spoofing attempt (someone else trying to impersonate the sender's email address).

DMARC (Domain-based Message Authentication, Reporting, and Conformance):

DMARC is a powerful tool that builds on SPF and DKIM to provide email authentication and reporting. DMARC reports tell you how email receivers handled your emails (passed authentication, rejected, etc.). This allows you to identify potential spoofing attempts or delivery issues. DMARC reports provide lots of information during an email security investigation. They can reveal authentication failures, unauthorized use of your domain for sending emails, and potential phishing attempts impersonating your domain.

Mail Header Analysis for Advanced Troubleshooting:

Email headers contain a wealth of information about the email's journey. Analyze fields to diagnose delivery issues, identify spam characteristics, and verify message integrity.

Some of these include:

- Received (multiple times): Traces the email's path through mail servers, including timestamps and authentication methods used at each hop. This helps identify delays, routing issues, and potential spam characteristics (e.g., excessive hops through unknown servers).
- DKIM-Signature: Contains the digital signature added by the sender using their DKIM key for message verification. A failing DKIM signature might indicate a spoofing attempt.
- SPF (visible through Received headers): Summarizes the results of the Sender Policy Framework (SPF) authentication check. This helps verify if the sender's domain authorized sending the message.
- DMARC (visible through Received headers): If DMARC is implemented, these headers might indicate how the receiver handled the message based on authentication results (e.g., passed, rejected).
- Authentication-Results: Summarizes the authentication checks performed on the email by different mail servers. Provides a comprehensive overview of the email's authentication status.
- Received-SPF: Specifically shows the results of the SPF authentication check.
- X- Headers: These custom headers can provide additional insights depending on the service or application used.

MX Toolbox: https://mxtoolbox.com/
Spamhaus Project: https://www.spamhaus.org/
DNSBL (DNS BlackList/ RBL) lookup and FCrDNS: https://multirbl.valli.org/
DMARC.org Tools: https://dmarc.org/resources/deployment-tools/
The SPF Project: http://www.open-spf.org/Sitemap/

Git Commands

Running config creates the appropriate config file:
Project configs are only available for the current project and stored in .git/config in the project's directory.
Global configs are available for all projects for the current user and stored in ~/.gitconfig.
System configs are available for all the users/projects and stored in /etc/gitconfig.
Use --global for global config, --system for system config, and neither for project config (run inside project directory).

$ git config --global user.name “Your Name” --global user.email “you@example.com”	
$ git config --global core.excludesfile [file]	System-wide ignore pattern for all local repos (like .gitignore)
$ git config --global –edit	Edit config file in editor
$ git init [project name]	If no name given, create a new repo in the current directory.
$ git add [file]	Add a file to the staging area.
$ git rm [file]	Remove file from working directory and staging area.
$ git clone ssh://user@domain.com/repo.git <OR> ~/existing/repo ~/new/repo	
$ git commit -m [message] (-a for all)	Create a new commit from changes added to the staging area
$ git fetch [remote]	Fetch changes from remote, don‘t merge into HEAD or tracking branches.
$ git fetch --prune [remote]	Delete remote refs that were removed from the remote repo.
$ git pull [remote] [branch]	Get changes from the remote and merge current branch with its upstream.
$ git pull --rebase <remote>	Same as above, but uses git rebase instead of merge
$ git merge [branch_name]	Join specified remote branch into your current HEAD
$ git branch [branch_name]	Create new branch, referencing the current HEAD.
$ git branch -d [branch_name]	Remove branch, if it is already merged into any other. -D forces
$ git checkout [-b][branch_name]	Switch working dir to branch; -b: create branch if it does not exist.
$ git checkout -b $new_branch $other	Create $new_branch based on branch $other and switch to it
$ git checkout --track [remote/branch]	Create new tracking branch based on a remote branch
$ git push [remote] [branch]	Publish local commits to a remote; --tags to push tags; -u as an upstream
$ git push --all (means all branches), --tags (means all tags, aren’t normally pushed) --force (careful!)	
$ git checkout $id $file	Checkout the $id version of a file
$ git diff [file]	Show changes between working directory and staging area.
$ git diff --staged [file]	Diff of what is staged but not yet commited
$ git diff HEAD	Show difference between working directory and last commit.
$ git diff --cached	Show difference between staged changes and last commit
$ git log --follow [file]	how the commits that changed file, even across renames
$ git diff $id1 $id2	What changed between $ID1 and $ID2
$ git checkout [file]	Discard local changes in a specific file (replace with remote commit)
$ git revert [commit]	Create new commit that undoes all of the changes made in [commit]
$ git revert HEAD	Revert the last commit
$ git revert $id	Revert specific commit
$ git revert [commit sha]	Create a new commit, reverting changes from the specified commit.
$ git reset [file]	Unstage a file while retaining the changes in working dir
$ git reset [commit, file, tag]	Reset HEAD pointer to previous commit, preserve all changes as unstaged
$ git reset --hard [commit]	Reset HEAD pointer to previous commit & discard all changes since
$ git reset --keep [commit]	If difference between <commit> and HEAD, has local changes, reset aborted.
$ git clean -n	Shows files that clean would remove. The -f flag executes the clean.
$ git stash	Put current changes in your working directory into stash for later use.
$ git stash list	List stack-order of stashed file changes
$ git stash pop	Apply stored stash content into working directory, and clear stash.
$ git stash drop	Delete a specific stash from all your previous stashes.
$ git tag -a [name] [commit sha]	Create a tag object named name for current commit. (use -d to remove tag)
$ git tag v1.0	Mark a version / milestone
$ git tag	List all tags.
$ git remote add [remote] [url]	Add new remote repository, named [remote]
$ git branch -dr [remote/branch]	Delete a branch on the remote

$ git rebase [branch]	Apply commits in branch ahead of remote branch. Don‘t rebase published commits!
$ git rebase --abort	Abort a rebase <AND> git rebase --continue Continue a rebase
$ git rebase -i	Interactively rebase current branch onto <base> specify how to handle each commit
$ git mergetool	Use your configured merge tool to solve conflicts
$ git commit –amend	Replace the last commit with combined staged changes and last commit. Use with nothing staged to edit the last commit’s message. Don’t amend published commits.

$ git remote -v	List all currently configured remotes
$ git remote show [remote]	Show information about a remote
$ git branch [-a]	List branches. A * notes the currently active branch; -a: show all incl. remote).
$ git show [SHA]	Show any object in Git in human-readable format
$ git show $id:$file	A specific file from a specific $ID
$ git blame $file	Who changed what and when in a file
$ git log [-n count]	List commit history of current branch
$ git log --oneline --graph --decorate	Overview with reference labels and history graph
$ git log refA..refB	Show commits on between branchA and branchB (ref can be branch, tag, etc)
$ git log --follow [file]	Show the commits that changed file, even across renames
$ git log –author= "<pattern>"	Search for commits by a particular author.
$ git log --grep= "<pattern>"	Search for commits with a commit message that matches
$ git log -p $file $dir/ec/tory/	History of changes for file with diffs
$ git log --stat -M	Show all commit logs with indication of any paths that moved
$ git reflog	List operations (e.g. checkouts or commits) made on local repo. --relative-date to show date info, --all to show all refs

To view the merge conflicts

$ git diff --base $file	(against base file)
$ git diff --ours $file	(against your changes)
$ git diff --theirs $file	(against other changes)

To discard conflicting patch

```
$ git reset --hard
$ git rebase --skip
```

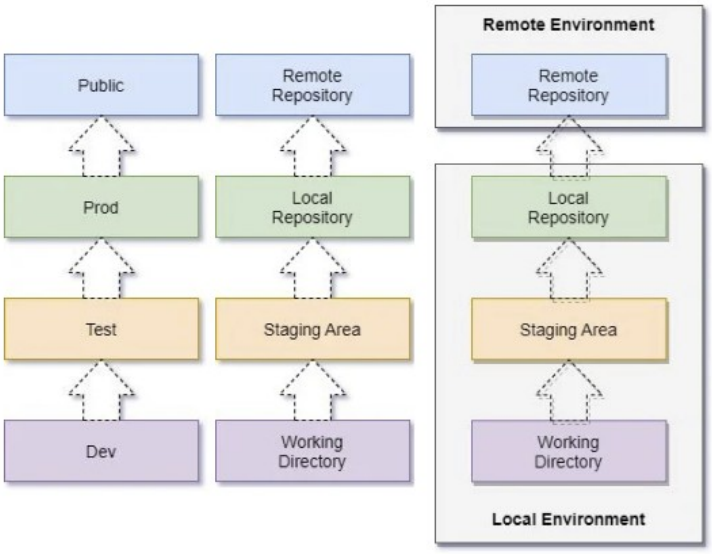
After resolving conflicts, merge with:

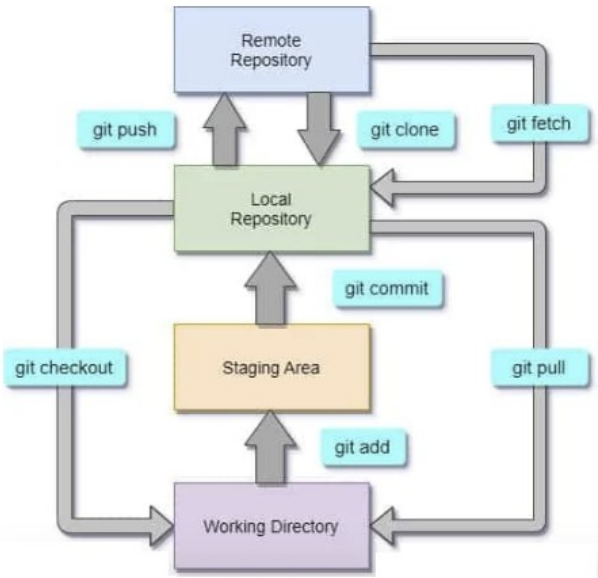
$ git add $conflicting_file	Do for all resolved files
$ git rebase –continue	

$ git fsck	Check for errors
$ git gc –prune	Cleanup repository
$ git grep "foo()"	Search working directory for foo()

Finding regressions

$ git bisect start	To start
$ git bisect good $id	$id is the last working version
$ git bisect bad $id	$id is a broken version
$ git bisect bad | good	To mark it as bad or good
$ git bisect visualize	Once you're done
$ git bisect reset	To launch gitk and mark it





MASTER = default development branch; ORIGIN = default upstream repository; HEAD = current branch; HEAD^ = parent of HEAD; HEAD~4 = the great-great grandparent of HEAD

Git Aliases

Git Please
$ git config --global alias.please 'push --force-with-lease'

Team leads warn about not force pushing to a shared branch. Rebasing, amending, and squashing can rewrite some shared history and spill duplicate commits all over the repo. Force stomps the upstream branch with your local version, and any changes that you hadn't already fetched are erased from history. Git's --force-with-lease checks that your local copy of the ref that you're overwriting is up-to-date first; that you've at least fetched the changes you're about to stomp. Here you only have to type "git please"

Git Commend
$ git config --global alias.commend 'commit --amend --no-edit'

Commit and then realize you'd forgotten to stage a file? Quietly amend any staged files onto the last commit you created, re-using your existing commit message. So as long as you haven't pushed yet, no-one will be the wiser. Don't amend published commits.

Git It
$ git config --global alias.it '!git init && git commit -m "root" –allow-empty'

The first commit of a repo can not be rebased like regular commits, so it's good practice to create an empty commit as your repo root. "git it" both initializes and creates an empty root commit in one quick step.
$ cd shiny-new-thing
$ git it
Initialized empty Git repo in /shiny-new-thing/.git/

Git Staaash
$ git config --global alias.stsh 'stash --keep-index'
$ git config --global alias.staash 'stash --include-untracked'
$ git config --global alias.staaash 'stash --all'

Takes any changes to tracked files in your work tree and stashes them away for later use, leaving you with a clean work tree to start hacking on something else. However if you've created any new files and haven't yet staged them, git stash won't touch them by default, leaving you with a dirty work tree, use staash. Similarly, the contents of untracked or ignored files are not stashed by default; use staaash. If in doubt, the long one (git staaash) will always restore your worktree to what looks like a fresh clone of your repo.
git stsh # stash only unstaged changes to tracked files
git stash # stash any changes to tracked files
git staash # stash untracked and tracked files
git staaash # stash ignored, untracked, and tracked files

Git Shortstat
$ git config --global alias.shortstat 'status --short --branch'

Git status inline help has improved, but the output is overly verbose for those more familiar with Git. For example, git status emits 18 lines to tell me that I have a couple of staged, unstaged, and untracked changes. Git shortstat tells me the same thing in three lines:
$ git shortstat
master
AM test
?? .gitignore

Git Merc
$ git config --global alias.merc 'merge --no-ff'

On non-rebasing branching workflows running a git merge to combine feature branches with the master is not ideal. With no options, git merge uses the --ff merge, which creates a merge commit only if no new changes are on the master branch, otherwise it "fast forwards" your master branch to point at the latest commit on your feature branch. Without a merge commit it's tricky to tell which code was developed on which branches in the git history. The --no-ff strategy, to always create a merge commit.

Git Grog (or "graphical log")
$ git config --global alias.grog 'log --graph --abbrev-commit --decorate --all --format=format:"%C(bold blue)%h%C(reset) - %C(bold cyan)%aD%C(dim white) - %an%C(reset) %C(bold green)(%ar)%C(reset)%C(bold yellow)%d%C(reset)%n %C(white)%s%C(reset)"'

Docker Primer / Command Overview

After 2017, most commands take the form of docker [command] [subcommand] [options], being more specific to the intended item, (i.e., 'docker container _____', or 'docker image ____'. Prior to this, commands such as 'docker [command] [identifier]' were used, often pertaining to containers: docker attach, build, commit, cp, create, diff, events, exec, export, history, images, import, info, inspect, kill, load, login, logout, logs, pause, port, ps, pull, push, rename, restart, rm, rmi, run, save, search, start, stats, stop, tag, top, unpause, update, version, wait, etc.

Below, subcommands with names similar to Unix/ Linux commands with that name will not be explained (unless needed), their actions are predictable in context. Prune always means remove unused.

System Information
docker system events | info | df | prune | inspect | version Info displays system-wide info, events show in realtime.
docker volume create | inspect | ls | rm | prune | update Manage volumes containers can use for data (persistent storage)
docker trust inspect | revoke | sign | key [generate | load] | signer [add | remove]
docker scout cves | version Checks container layers for CVEs and gives countermeasures (or known secure images)
docker search Search Docker Hub for images

Managing networks.
docker network create | ls | rm | prune | inspect | connect | disconnect
docker container port

Container-specific commands
docker container start | stop| restart | create | rm | rename | run | kill | prune
docker container top | ps | stats | logs | port | ls
docker container pause | unpause Refers to all processes within one or more containers
docker container exec Execute a command in a running container
docker container cp Copy files/folders between a container and the local filesystem
docker container diff Inspect changes to files or directories on a container's filesystem
docker container attach Attach local standard input/output/error streams to a running container
docker container export Export a container's filesystem as a tar archive
docker container commit Create a new image from a container's changes
docker container update Update configuration of one or more containers
docker container inspect
docker container wait Blocks container(s) until they stop, then print their codes

Image-specific commands
docker image build Builds image from a Dockerfile
docker image inspect | history
docker image ls | rm | prune | tag Tag adds a label
docker image pull | push Actions with local registry/ repo
docker image load | save | import Load/save from/to tarball or stdin/out. See docs for diff of import and load
docker image tag Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
Note: for push and pull, use 'docker login | logout' to log in/out from a registry

Docker Compose commands (define and run multi-container applications)
docker compose build Build or rebuild services
docker compose config Parse, resolve and render compose file
docker compose start | stop| restart | create | rm | rename | run | kill | prune
docker compose cp | exec | port | ps | top | version | events | log | push | pull Same as the container equivalents
docker compose images | ls With Compose, ls lists compose projects and ls lists images containers use
docker compose up | down Pertains to starting and to stop/remove containers, networks
docker compose pause | unpause Pertains to the running containers of a service

Docker Build and Builder-X (BuildKit)
docker builder build | prune Build an image from a Dockerfile, prune removes the build cache
docker buildx bake | build Bake orders to build from a file, build directs to start a build
docker buildx imagetools create | inspect Create a new image based on source images; show details of an image
docker buildx use | create | ls | rm | stop | inspect |du | prune Apply to a builder instances; prune removes cache

---
docker plugin create | enable | disable | install | ls | rm | inspect | upgrade | push
docker plugin set -change settings (not a typo)
Plugins extend Docker's functionality and help users connect with other popular services
Create makes a plugin from a rootfs and configuration. Plugin data directory must contain config.json and rootfs directory.

docker context create | import | export | ls | rm | use | show | inspect | update Import/export works with zip or tar data
Context info is metadata specifying a name, endpoint configuration(s), TLS info, orchestrator(s), usually json files

docker checkpoint create | ls | rm
Checkpoint and Restore allows you to freeze a running container by checkpointing it, which turns its state into a collection of files on disk. Later, the container can be restored from the point it was frozen.

docker manifest create | annotate | rm | inspect | push
A manifest holds info on one image- layers, size, and digest. The command returns os and arch it was built for. A manifest list holds several image names, is intended for images identical in function for different os/arch combinations, so are often referred to as “multi-arch images”. Docker calls the command ‘experimental’ but has since 2017.

Docker Swarm/ Cluster Management Commands
Should be executed on a swarm manager node.

docker config create | inspect | ls | rm	Manage Swarm configs
docker secret create | inspect | ls | rm	Passcode for Swarm management
docker stack config | deploy | ps | ls | rm | services	List stacks with ls, services lists services, config outputs config file
docker node promote | demote | ps | ls | rm | inspect | update	Manage Swarm nodes
docker service create | rollback | scale | ps | ls | rm | inspect | logs | update	
docker swarm ca | init | update | join-token | unlock-key | join | leave | unlock	

Docker Run Options

docker run --name mycontainer3 -it [IMGNAME] [CMD]	Names the container, -i keeps STDIN open, -t gives a pseudo-tty
docker run -a stdin -a stdout -it ubuntu /bin/bash	-a specifies terminal access; ubuntu is the image, cmd is bash
docker run -p 80:8080/tcp -it ubuntu /bin/bash	-p is port-map container's 8080 to host's 80 (can add host IP too)

Detached mode:
- Running -d for detached runs container in background. Allows closing the terminal session without stopping the container
- Containers exit when the root process starting it exits; using -d with --rm, it’s removed on exit or when the daemon exits.
- Don’t send a command like ‘service nginx start’ to a detached container. Use this syntax: nginx -g 'daemon off;'
- To do input/output with a detached container use network connections or shared volumes. These are required because
- The container stops listening to the terminal where ‘run’ was executed. Net connections or shared volumes are needed for I/O and this is why the -it option is needed (provides TTY)

Foreground mode:
- Default mode when -d isn’t specified. The streams stdout and stderr are attached if you dont use the -a option (no sdtin)
- Using the options "-it" is still common to provide TTY access, but you can also say -a stdin -a stdout -a stderr

Names are more user-friendly than UUID long and short identifiers assigned by the Docker daemon. When networking, containers on the default bridge network must be linked to communicate by name. There may also be some caveats with custom names, but there is a containerID/PID file option to remedy automation, etc., designated using --cidfile="___"
[Namespace designation options get out of scope of this document. See https://docs.docker.com/engine/reference/run/]

Container images can be accessed more specifically. Often a specific image version will be the tag added, such as in "ubuntu:22.04". Getting even more specific by supplying a hash (digest) value as in "nginx@sha256:9cacb71397...."

Docker Run Network Options

--network=" "	'bridge'	Network stack on default Docker bridge
	'container'	<name | id>': reuse another container's network stack
	'host'	Use the host network stack
	'<network-name> | <network-id>'	Connect to a user-defined network
	'none'	None
--network-alias=[ ]		Add network-scoped alias for the container
--add-host=" "		Add a line to /etc/hosts (host:IP)
--mac-address=" ", --ip=" ", --ip6=" "		Specifically as needed, set container's ethernet MAC, IPv4, or IPv6 addresses
--link-local-ip=[ ]		Set 1+ container's Eth link local IPv4/IPv6 addresses
--dns=[ ]		DNS servers

By default, networking is enabled on all containers unless disabled (with "none"); can make any outgoing connections- yet mapping ports as previously seen here and linking to other containers only works with the same default bridge. These are legacy Docker methods which have evolved. It's obvious from the command list above things are now more granular and natural to what we expect from actual hardware.

This is part I of a Docker overview currently being re-written from scratch. Part II will finish networking, cover security topics, build files, and wrap up the general topics

Ansible: Keywords For General Playbook Objects

Common (to play, role, block and task objects)

any_errors_fatal	Force any un-handled task errors on any host to propagate to all hosts and end the play.
become	Boolean that controls if privilege escalation is used or not on Task execution.
become_flags	A string of flag(s) to pass to the privilege escalation program when become is True.
become_method	Which method of privilege escalation to use (such as sudo or su).
become_user	User that you 'become' after using privilege escalation (as permissions allow) Default is root.
ansible_become_pass	Provides the password for the become directives
check_mode	A boolean that controls if a task is executed in 'check' mode
collections	UNDOCUMENTED!!
connection	Allows you to change the connection plugin used for tasks to execute on the target.
debugger	Enable debugging tasks based on state of the task result. See Playbook Debugger
diff	Toggle to make tasks return 'diff' information or not.
environment	A dictionary converted into env vars for the task upon execution. Does not affect Ansible configuration, it just sets the variables for the code responsible for executing the task.
ignore_errors	Boolean option to ignore task failures and continue with play. Does not affect connection errors.
ignore_unreachable	Boolean that allows you to ignore unreachable hosts and continue with play. This does not affect other task errors (see ignore_errors) but is useful for groups of volatile/ephemeral hosts.
module_defaults	Specifies default parameter values for modules.
name	Identifier. Can be used for documentation, in or tasks/handlers.
no_log	Boolean that controls information disclosure.
port	Used to override the default port used in a connection.
remote_user	User used to log into the target via the connection plugin.
run_once	Boolean that will bypass the host loop, forcing the task to attempt to execute on the first host available and afterwards apply any results and facts to all active hosts in the same batch.
tags	Tags applied to the task or included tasks, this allows selecting subsets of tasks from CLI
vars	Dictionary/map of variables

Play object-specific

fact_path	Set the fact path option for the fact gathering plugin controlled by gather_facts.
force_handlers	Will force notified handler execution for hosts even if they failed during the play. Will not trigger if the play itself fails.
gather_facts	A boolean that controls if the play will automatically run the 'setup' task to gather facts for the hosts.
gather_subset	Allows you to pass subset options to the fact gathering plugin controlled by gather_facts.
gather_timeout	Allows you to set the timeout for the fact gathering plugin controlled by gather_facts.
handlers	A section with tasks that are treated as handlers, these won't get executed normally, only when notified after each section of tasks is complete. A handler's listen field is not templatable.
hosts	A list of groups, hosts or host pattern that translates into a list of hosts that are the play's target.
max_fail_percentage	can be used to abort the run after a given percentage of hosts in the current batch has failed.
order	Controls the sorting of hosts as they are used for executing the play. Possible values are inventory (default), sorted, reverse_sorted, reverse_inventory and shuffle.
post_tasks	A list of tasks to execute after the tasks section.
pre_tasks	A list of tasks to execute before roles.
roles	List of roles to be imported into the play
serial	Explicitly define how Ansible batches the execution of the current play on the play's target
strategy	Allows you to choose the connection plugin to use for the play.
tasks	Main list of tasks to execute in the play, they run after roles and before post_tasks.
vars_files	List of files that contain vars to include in the play.
vars_prompt	list of variables to prompt for.

Role object-specific

delegate_facts	Boolean that allows you to apply facts to a delegated host instead of inventory_hostname.
delegate_to	Host to execute task instead of the target (inventory_hostname); use that host's connection vars.
when	Conditional expression, determines if an iteration of a task is run or not.

Block object-specific

always	List of tasks, in a block, that execute no matter if there is an error in the block or not.
block	List of tasks in a block.
delegate_facts	Boolean that allows you to apply facts to a delegated host instead of inventory_hostname.
delegate_to	Host to execute task instead of the target (inventory_hostname); use that host's connection vars.
rescue	List of tasks in a block that run if there is a task error in the main block list.
when	Conditional expression, determines if an iteration of a task is run or not.

Task object-specific

action	The 'action' to execute for a task, it normally translates into a C(module) or action plugin.
args	Another way to add arguments into a task. Takes a dictionary; keys map to options and values.
async	Run a task asynchronously if the C(action) supports this; value is maximum runtime in seconds.
changed_when	Conditional expression that overrides the task's normal 'changed' status.
delay	Number of seconds to delay between retries. This setting is only used in combination with until.
delegate_facts	Boolean that allows you to apply facts to a delegated host instead of inventory_hostname.
delegate_to	Host to execute task instead of the target (inventory_hostname); use that host's connection vars.
failed_when	Conditional expression that overrides the task's normal 'failed' status.
local_action	Same as action but also implies delegate_to: localhost
loop	List for the task to iterate over, saving each list element into the item variable (set via loop_control)
loop_control	Several keys here allow you to modify/set loop behaviour in a task.
notify	List of handlers to notify when the task returns a 'changed=True' status.
poll	Sets the polling interval in seconds for async tasks (default 10s). Value of 0 is 'fire and forget'
register	Name of variable that will contain task status and module return data.
retries	Number of retries before giving up in a until loop. This setting is only used in combination with until.
until	Enables a 'retries loop' that will go on until the condition supplied here is met or we hit the retries limit.
when	Conditional expression, determines if an iteration of a task is run or not.
with_<lookup_plugin>	The same as loop but magically adds the output of any lookup plugin to generate the item list.

Ansible Commands - Overview

ansible <host-pattern> [options]

-i, --inventory, --inventory-file	Specify inventory host path or comma separated host list. –inventory-file is deprecated
-m OR ---module-name <MODULE_NAME>	Module name to execute (default=command)
-b, --become	Run operations with become (does not imply password prompting)
--become-method <BECOME_METHOD>	Privilege escalation method (default=%default), use ansible-doc -t become -l to list valid choices.
--become-user <BECOME_USER>	Run operations as this user (default=root)
-K, --ask-become-pass	Ask for privilege escalation password- depends on configuration of ssh etc. items in ansible.cfg
--list-hosts	Outputs a list of matching hosts; does not execute anything else
--playbook-dir <BASEDIR>	Since this tool does not use playbooks, use this as a substitute playbook dir, relative path for many features including roles/ group_vars/ etc.
-M, --module-path	Prepend colon-separated path(s) to module library (default=~/.ansible/plugins/modules:/usr/share/ansible/plugins/modules)
-u OR --user <REMOTE_USER>	Connect as this user (default=none)
-k, --ask-pass	Ask for connection password
--private-key, --key-file	Use this file to authenticate the connection
-c OR ---connection <CONNECTION>	Connection type to use (default=smart)
-T OR ---timeout <TIMEOUT>	Override the connection timeout in seconds (default=10)
--ssh-common-args <SSH_COMMON_ARGS>	Specify common arguments to pass to sftp/scp/ssh (e.g. Proxycommand)
--ssh-extra-args <SSH_EXTRA_ARGS>	Specify extra arguments to pass to ssh only (e.g. -r)
--scp-extra-args <SCP_EXTRA_ARGS>	Specify extra arguments to pass to scp only (e.g. -l)
--sftp-extra-args <SFTP_EXTRA_ARGS>	Specify extra arguments to pass to sftp only (e.g. -f, -l)
--ask-vault-pass	Ask for vault password
--vault-id	The vault identity to use
--vault-password-file	Vault password file
--syntax-check	Perform a syntax check on the playbook, but do not execute it
-C, --check	Don't make any changes; instead, try to predict some of the changes that may occur
-D, --diff	When changing files and templates, show differences in those files; works with –check
-B OR ---background <SECONDS>	Run asynchronously, failing after x seconds (default=n/a)
-P OR ---poll <POLL_INTERVAL>	Set the poll interval if using -b (default=15) 'P 0" is 'fire and forget'
-a OR ---args <MODULE_ARGS>	Module arguments
-e, --extra-vars	Set additional variables as key=value or yaml/json, if filename prepend with @
-f OR ---forks <FORKS>	Specify number of parallel processes to use (default=5)
-l OR ---limit <SUBSET>	Further limit selected hosts to an additional pattern
-v, --verbose	Verbose mode (-vvv for more, -vvvv enable connection debugging)
-o, --one-line	Condense output
-t OR ---tree <TREE>	Log output to this directory
-h, --help	Show this help message and exit

ansible-playbook [options] playbook.yml [playbook2 ...]

Focused at simply the running of playbooks. Most of the CLI options in the main Ansible executable also work here.

--flush-cache	clear the fact cache for every host in inventory
--force-handlers	run handlers even if a task fails
--list-hosts	outputs a list of matching hosts; does not execute anything else
--list-tags	list all available tags
--list-tasks	list all tasks that would be executed
--skip-tags	only run plays and tasks whose tags do not match these values
--start-at-task <START_AT_TASK>	start the playbook at the task matching this name
--step	one-step-at-a-time: confirm each task before running

ansible-console [<host-pattern>] [options] - A REPL that allows for running ad-hoc tasks against an inventory
ansible-pull -U <repository-url> [options] [<playbook.yml>] - Pull playbooks and supporting items from a Git repo and spin them up.
ansible-config [dump|list|view] [--help] [options] [ansible.cfg]
ansible-inventory [options] [host|group] - display or dump the currently configured inventory as

Environmental variables
ANSIBLE_CONFIG can be specified to override the default ansible config file. The default config file is at /etc/ansible/ansible.cfg The file at ~/.ansible.cfg is a user config file which overrides the default config if present. Inside of this file are all the environmental variables that can be specified/ altered.

ansible-galaxy [action] [--help] [options]

Manage Ansible roles in shared repositories, the default of which is Ansible Galaxy [https://galaxy.ansible.com]

Actions **(delete | import | info | init | install | list | login | remove | search | setup)**

Actions: info	Prints out detailed information about an installed role as well as info available from the galaxy api.
--offline	Don't query the galaxy api when creating roles
-p, --roles-path	Path to directory containing your roles. Default is roles_path (ansible.cfg) or /etc/ansible/roles
Actions: login	Verify user's identify via github and retrieve an auth token from ansible galaxy.
--github-token <TOKEN>	Identify with github token rather than username and password.
Actions: delete	Delete a role from ansible galaxy.
Actions: init	Creates the skeleton framework of a role that complies with the galaxy metadata format.
--init-path <INIT_PATH>	The path in which the skeleton role will be created. The default is the current working directory.
--offline	Don't query the galaxy api when creating roles
--role-skeleton <ROLE_SKELETON>	The path to a role skeleton that the new role should be based upon.
--type <ROLE_TYPE>	Initialize using an alternate role type. Valid types include: 'container', 'apb' and 'network'.
-f, --force	Force overwriting an existing role
Actions: install	Uses args list of roles to be installed, unless -f was specified. Can be a name or local .tar.gz file.
-f, --force	Force overwriting an existing role
--force-with-deps	Force overwriting an existing role and it's dependencies
-g, --keep-scm-meta	Use tar instead of the scm archive option when packaging the role
-i, --ignore-errors	Ignore errors and continue with the next specified role.
-n, --no-deps	Don't download roles listed as dependencies
-p, --roles-path	Path to directory containing your roles. Default is roles_path (ansible.cfg) or /etc/ansible/roles
-r OR --role-file <ROLE_FILE>	A file containing a list of roles to be imported
Actions: list	Lists the roles installed on the local system or matches a single role passed as an argument.
-p, --roles-path	Path to directory containing your roles. Default is roles_path (ansible.cfg) or /etc/ansible/roles
Actions: remove	Removes the list of roles passed as arguments from the local system.
-p, --roles-path	Path to directory containing your roles. Default is roles_path (ansible.cfg) or /etc/ansible/roles
Actions: import	Used to import a role into ansible galaxy
--branch <REFERENCE>	The name of a branch to import. Defaults to the repository's default branch (usually master)
--no-wait	Don't wait for import results.
--role-name <ROLE_NAME>	The name the role should have, if different than the repo name
--status	Check the status of the most recent import request for given github_user/github_repo.
Actions: setup	Setup an integration from github or travis for ansible galaxy roles
--list	List all of your integrations.
--remove <REMOVE_ID>	Remove the integration matching the provided id value. Use –list to see id values.
Actions: search	Searches for roles on the ansible galaxy server
--author <AUTHOR>	Github username
--galaxy-tags <GALAXY_TAGS>	List of galaxy tags to filter by
--platforms <PLATFORMS>	List of os platforms to filter by
-p, --roles-path	Path to directory containing your roles. Default is roles_path (ansible.cfg) or /etc/ansible/roles

Common Options:

--author <AUTHOR>	Github username
--galaxy-tags <GALAXY_TAGS>	List of galaxy tags to filter by
--platforms <PLATFORMS>	List of os platforms to filter by
--version	Show program's version, config file, module search path, location, executable location and exit
-c, --ignore-certs	Ignore ssl certificate validation errors.
-h, --help	Show this help message and exit
-p, --roles-path	Path to directory containing your roles. Default is roles_path (ansible.cfg) or /etc/ansible/roles
-s OR --server <API_SERVER>	The api server destination
-v, --verbose	Verbose mode (-vvv for more, -vvvv to enable connection debugging)

ansible-doc [-l|-F|-s] [options] [-t <plugin type>] [plugin]

Displays information on modules installed in Ansible libraries; plugins and their short descriptions, helpful details
https://docs.ansible.com/ansible/latest/collections/all_plugins.html

ansible-doc -l	List all plugins
ansible-doc -t connection -l	List all of the type 'connection'
ansible-doc -t connection -s ssh	Show the ssh plugin usage and more details
-l, --list	List available plugins
-t OR --type <TYPE>	Plugin type (default is module). Includes: become, cache, callback, cliconf, connection, filter, httpapi, inventory, lookup, modules, netconf, roles, shell, strategy, test, vars
-s, --snippet	Show brief playbook snippet for specified plugin(s) to paste and customize
-F, --list_files	Show plugin names and their source files without summaries
-j, --json	Dump json metadata
-M, --module-path	Prepend colon-separated path(s) to module library (default=~/.ansible/plugins/modules:/usr/share/ansible/plugins/modules)

Ad Hoc Commands Examples

ansible <host-pattern> [options]
ansible <hostgroup> -m <modulename> -a <arguments to the module>
https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html
The shell module is the default module, but if you are doing some things like chaining commands, you must specify it

Check the disk space: ansible multi -a "df -h"
Get free memory: ansible multi -m shell -a "cat /proc/meminfo | head -2" - ansible multi -a "free -m" -i ansible_hosts
Check uptime (all return same info): "ansible multi -m command -a uptime" - "ansible multi -m shell -a uptime" - "ansible multi -a uptime"
Execute a command as root user: "ansible multi -m shell -a "cat /etc/passwd|grep -i vagrant" -b -K" -K will need to be verified depending on ansible.cfg
Creating user groups: "ansible app -s -m group -a "name=devops3 state=present" (or state=absent to remove)
Creating user: "ansible app -m user -a "name=devops3-user2 group=devops3 createhome=yes" -b
Change file ownership info: ansible app -m file -a "path=/opt/mydir group=devops3 owner=devops3-user2" -i ansible_hosts -b"
Creating directory with permissions: "ansible app -m file -a "path=/opt/mydir state=directory mode=0755" -b" (delete with state=absent)
Same for a file: "ansible app -m file -a "path=/tmp/testfile state=touch mode=0755"
To copy (scp) files: "ansible 172.6.7.10 -m copy -a "src=~/Downloads/index.html dest=/var/www/html owner=apache group=apache mode=0644"
Start (or stop) service on hosts: "ansible multi -s -m service -a "name=httpd state=started enabled=yes"
To check the a service's status info: "ansible 172.6.7.10 -m service -a "name=httpd" -i ansible_hosts -u vagrant"
Using systemd module to start/stop/restart/reload services: ansible webservers -m systemd -a "name=nginx state=reloaded" -i prod-ansible-hosts
Get file/directory info: "ansible multi -m stat -a "path=/etc/environment"
Download file from URL: ansible 172.6.7.10 -m get_url -a "url=https://nodejs.org/myfile.tar.gz dest=/tmp mode=0755" -i prod-ansible-hosts
Check the open ports: ansible 172.6.7.10 -m listen_ports_facts -i prod-ansible-hosts
Get some logging info: ansible multi -b -a "tail /var/log/messages"
Reboot servers in group, 12 at a time: "ansible apps -a "/sbin/reboot" -f 12"
Install on RH-compatible system: "ansible rh-hosts -s -m yum -a "name=httpd state=installed"
Install on Debian-compatible system: "ansible ubuntu-hosts -m apt -a "name=vsftpd-3.0.2 state=present"
- Both apt and yum have almost identical state options (present, absent, latest); repo-related stuff differs.
Run a cron job every 4 hours: "ansible multi -s -m cron -a "name='daily-cron-all-servers' hour=4 job='/path/to/hour-script.sh'"
- With the cron module, 15 minutes is specified with minute=*/15, there is also and special_time=reboot or daily or weekly for those

• Using the limit option lets you get very specific: ansible app -b -a "systemctl status ntpd" --limit "172.6.7.10" to limit running this command module op on this one ip node... can also be wildcarded *.4 for all IPs ending in .4 you can use the style --limit !172.6.7.10 to exclude with "!", and --limit "app:&multi" for all members of several groups, and even combine them such as --limit "app:!172.6.7.80" which specifies one IP in a specific group

• The -B option mentioned here is most helpful to run an adhoc command that may take some time to run in the background so you can continue to work in your terminal without interruption.

• The setup module access ansible_facts variables, which as seen below can be filtered. It is similar to info found using the facter utility:
ansible appservers -m setup -i ansible_hosts -a 'filter=ansible_distribution,ansible_distribution_version,ansible_memfree_mb, ansible_memtotal_mb, ansible_processor_cores*,ansible_architecture' 2>/dev/null
In playbooks, it is more suitable to use the gather_facts option instead, which you do not need to call explicitly, as it is automatically run at the beginning of each playbook execution. Setup would then be used for more customized, on-demand queries (such as to update facts after running plays)

Ansible Directory Layout

```
production              # inventory file for production servers
staging                 # inventory file for staging environment

group_vars/
   group1.yml           # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml        # here we assign variables to particular systems
   hostname2.yml

library/                # if any custom modules, put them here (optional)
module_utils/           # if any custom module_utils to support modules, put them here (optional)
filter_plugins/         # if any custom filter plugins, put them here (optional)

site.yml                # master playbook
webservers.yml          # playbook for webserver tier
dbservers.yml           # playbook for dbserver tier

roles/
    common/             # this hierarchy represents a "role"
        tasks/          #
            main.yml    #  <-- tasks file can include smaller files if warranted
        handlers/       #
            main.yml    #  <-- handlers file
        templates/      #  <-- files for use with the template resource
            ntp.conf.j2 #  <------- templates end in .j2
        files/          #
            bar.txt     #  <-- files for use with the copy resource
            foo.sh      #  <-- script files for use with the script resource
        vars/           #
            main.yml    #  <-- variables associated with this role
        defaults/       #
            main.yml    #  <-- default lower priority variables for this role
        meta/           #
            main.yml    #  <-- role dependencies
        library/        # roles can also include custom modules
        module_utils/   # roles can also include custom module_utils
        lookup_plugins/ # or other types of plugins, like lookup in this case

    webtier/            # same kind of structure as "common" was above, done for the webtier role
    monitoring/         # " "
```

Alternate layout:

Putting each inventory with its group_vars/host_vars in a separate directory. Useful if these don't have much in common in different environments

```
inventories/
   production/
      hosts             # inventory file for production servers
      group_vars/
         group1.yml     # here we assign variables to particular groups
         group2.yml
      host_vars/
         hostname1.yml  # here we assign variables to particular systems
         hostname2.yml

   staging/
      hosts             # inventory file for staging environment
      group_vars/
         group1.yml     # here we assign variables to particular groups
         group2.yml
      host_vars/
         stagehost1.yml # here we assign variables to particular systems
         stagehost2.yml

library/
module_utils/
filter_plugins/

site.yml
webservers.yml
dbservers.yml

roles/
    common/
    webtier/
    monitoring/
    fooapp/
```

Static inventory example: production

```
# file: production.  Define groups based on purpose of host (roles) and also city or datacenter location (if applicable)

[houston_webservers]
www-atl-1.example.com
www-atl-2.example.com

[richmond_webservers]
www-bos-1.example.com
www-bos-2.example.com

[houston_dbservers]
db-atl-1.example.com
db-atl-2.example.com

[richmond_dbservers]
db-bos-1.example.com

# webservers in all cities
[webservers:children]
houston_webservers
richmond_webservers

# dbservers in all cities
[dbservers:children]
houston_dbservers
richmond_dbservers

# everything in the houston
[houston:children]
houston_webservers
houston_dbservers

# everything in the richmond
[richmond:children]
richmond_webservers
richmond_dbservers
```

Group And Host Variables; Assign variables to groups:

Houston has its own NTP servers, so when setting up ntp.conf, we should use them:

```
# file: group_vars/houston.yml
ntp: ntp-houston.example.com
backup: backup-houston.example.com
```

Webservers have some specific configuration that doesn't make sense for the database servers:

```
# file: group_vars/webservers.yml
apacheMaxRequestsPerChild: 3000
apacheMaxClients: 900
```

Default/universal values, put them in a file called group_vars/all:

```
# file: group_vars/all.yml
ntp: ntp-richmond.example.com
backup: backup-richmond.example.com
```

If needed, define specific hardware variance in systems in a host_vars file, but avoid doing this unless you need to:

```
# file: host_vars/db-bos-1.example.com.yml
foo_agent_port: 86
bar_agent_port: 99
```

Again, if we are using dynamic inventory sources, many dynamic groups are automatically created. So a tag like "class:webserver" would load in variables from the file "group_vars/ec2_tag_class_webserver" automatically.

Top Level Playbooks Are Separated By Role

In site.yml, we import playbooks that defines our entire infrastructure

```
# file: site.yml
- import_playbook: webservers.yml
- import_playbook: dbservers.yml
```

In a file like webservers.yml (also at the top level), map the configuration of webservers group to the roles it performs:

```
# file: webservers.yml
- hosts: webservers
  roles:
    - common
    - webtier
```

You can either choose to configure our whole infrastructure by "running" site.yml or just run a subset (webservers.yml). This is analogous to the "–limit" parameter to ansible but a little more explicit: "ansible-playbook site.yml --limit webservers" is the same as "ansible-playbook webservers.yml"

Task And Handler Organization For A Role

Below is an example tasks file that explains how a role works. Our common role here just sets up NTP, but it could do more if we wanted:

```
# file: roles/common/tasks/main.yml
- name: be sure ntp is installed
  yum:
    name: ntp
    state: present
  tags: ntp

- name: be sure ntp is configured
  template:
    src: ntp.conf.j2
    dest: /etc/ntp.conf
  notify:
    - restart ntpd                # notify handler (see below
  tags: ntp

- name: be sure ntpd is running and enabled
  service:
    name: ntpd
    state: started
    enabled: yes
  tags: ntp
```

Handlers are only fired when certain tasks report changes, and are run at the end of each play:

```
# file: roles/common/handlers/main.yml
- name: restart ntpd
  service:
    name: ntpd
    state: restarted
```

What This Organization Enables (Examples)

Reconfigure my whole infrastructure: ansible-playbook -i production site.yml
To reconfigure NTP on everything: ansible-playbook -i production site.yml --tags ntp
To reconfigure just my webservers: ansible-playbook -i production webservers.yml
For just my webservers in Richmond: ansible-playbook -i production webservers.yml --limit richmond

For just the first 10, and then the next 10:
ansible-playbook -i production webservers.yml --limit richmond[0:9]
ansible-playbook -i production webservers.yml --limit richmond[10:19]

Confirm what task names would be run if I ran this command and said "just ntp tasks"
ansible-playbook -i production webservers.yml --tags ntp --list-tasks

Confirm what hostnames might be communicated with if I said "limit to richmond"
ansible-playbook -i production webservers.yml --limit richmond --list-hosts

Rolling Update Batch Size and Maximum Failure Percentage

By default, Ansible will try to manage all of the machines referenced in a play in parallel. For rolling update use "serial" to limit this. Example, if we had 4 hosts in the group ‘webservers’, 2 would complete the play before moving on to the next 2 hosts- can also use a percentage (serial: "30%")

```
- name: test play
  hosts: webservers
  serial: 2
  gather_facts: False
  tasks:
  - name: task one
    command: hostname
  - name: task two
    command: hostname
```

Below, the first batch would contain a single host, the next would contain 5 hosts, every following batch would contain 10 hosts

```
- name: test play
  hosts: webservers
  serial:
  - 1
  - 5
  - 10
```

If you need to stop things due to errors- if more than 3 of the 10 servers in the group were to fail, the rest of the play would be aborted:

```
- hosts: webservers
  max_fail_percentage: 30
  serial: 10
```

Quick descriptions of modules to import for specific platforms: Kubernetes, VMWare, AWS, GCP, Azure, Juniper, Cisco

Kubernetes.Core
Collection version 2.4.0
https://docs.ansible.com/ansible/latest/collections/kubernetes/core/index.html#plugins-in-kubernetes-core
helm module – Manages Kubernetes packages with the Helm package manager
helm_info module – Get information from Helm package deployed inside the cluster
helm_plugin module – Manage Helm plugins
helm_plugin_info module – Gather information about Helm plugins
helm_pull module – download a chart from a repository and (optionally) unpack it in local directory.
helm_repository module – Manage Helm repositories.
helm_template module – Render chart templates
k8s module – Manage Kubernetes (K8s) objects
k8s_cluster_info module – Describe Kubernetes (K8s) cluster, APIs available and their respective versions
k8s_cp module – Copy files and directories to and from pod.
k8s_drain module – Drain, Cordon, or Uncordon node in k8s cluster
k8s_exec module – Execute command in Pod
k8s_info module – Describe Kubernetes (K8s) objects
k8s_json_patch module – Apply JSON patch operations to existing objects
k8s_log module – Fetch logs from Kubernetes resources
k8s_rollback module – Rollback Kubernetes (K8S) Deployments and DaemonSets
k8s_scale module – Set a new size for a Deployment, ReplicaSet, Replication Controller, or Job.
k8s_service module – Manage Services on Kubernetes
k8s_taint module – Taint a node in a Kubernetes/OpenShift cluster

Connection Plugins
kubectl connection – Execute tasks in pods running on Kubernetes.

Filter Plugins
k8s_config_resource_name filter – Generate resource name for the given resource of type ConfigMap, Secret

Inventory Plugins
k8s inventory – Kubernetes (K8s) inventory source

Lookup Plugins
k8s lookup – Query the K8s API
kustomize lookup – Build a set of kubernetes resources using a ‘kustomization.yaml’ file.

Vmware.Vmware_Rest
Collection version 2.3.1
https://docs.ansible.com/ansible/latest/collections/vmware/vmware_rest/index.html#plugins-in-vmware-vmware-rest
appliance_access_consolecli module – Set enabled state of the console-based controlled CLI (TTY1).
appliance_access_consolecli_info module – Get enabled state of the console-based controlled CLI (TTY1).
appliance_access_dcui module – Set enabled state of Direct Console User Interface (DCUI TTY2).
appliance_access_dcui_info module – Get enabled state of Direct Console User Interface (DCUI TTY2).
appliance_access_shell module – Set enabled state of BASH, that is, access to BASH from within the controlled CLI.
appliance_access_shell_info module – Get enabled state of BASH, that is, access to BASH from within the controlled CLI.
appliance_access_ssh module – Set enabled state of the SSH-based controlled CLI.
appliance_access_ssh_info module – Get enabled state of the SSH-based controlled CLI.
appliance_health_applmgmt_info module – Get health status of applmgmt services.
appliance_health_database_info module – Returns the health status of the database.
appliance_health_databasestorage_info module – Get database storage health.
appliance_health_load_info module – Get load health.
appliance_health_mem_info module – Get memory health.
appliance_health_softwarepackages_info module – Get information on available software updates available in the remote vSphere Update Manager repository
appliance_health_storage_info module – Get storage health.
appliance_health_swap_info module – Get swap health.
appliance_health_system_info module – Get overall health of system.
appliance_infraprofile_configs module – Exports the desired profile specification.
appliance_infraprofile_configs_info module – List all the profiles which are registered.
appliance_localaccounts_globalpolicy module – Set the global password policy.
appliance_localaccounts_globalpolicy_info module – Get the global password policy.
appliance_localaccounts_info module – Get the local user account information.
appliance_monitoring_info module – Get monitored item info
appliance_monitoring_query module – Get monitoring data.
appliance_networking module – Reset and restarts network configuration on all interfaces, also this will renew the DHCP lease for DHCP IP address.
appliance_networking_dns_domains module – Set DNS search domains.
appliance_networking_dns_domains_info module – Get list of DNS search domains.
appliance_networking_dns_hostname module – Set the Fully Qualified Domain Name.
appliance_networking_dns_hostname_info module – Get the Fully Qualified Doman Name.
appliance_networking_dns_servers module – Set the DNS server configuration
appliance_networking_dns_servers_info module – Get DNS server configuration.
appliance_networking_firewall_inbound module – Set the ordered list of firewall rules to allow or deny traffic from one or more incoming IP addresses
appliance_networking_firewall_inbound_info module – Get the ordered list of firewall rules
appliance_networking_info module – Get Networking information for all configured interfaces.
appliance_networking_interfaces_info module – Get information about a particular network interface.
appliance_networking_interfaces_ipv4 module – Set IPv4 network configuration for specific network interface.
appliance_networking_interfaces_ipv4_info module – Get IPv4 network configuration for specific NIC.
appliance_networking_interfaces_ipv6 module – Set IPv6 network configuration for specific interface.
appliance_networking_interfaces_ipv6_info module – Get IPv6 network configuration for specific interface.
appliance_networking_noproxy module – Sets servers for which no proxy configuration should be applied
appliance_networking_noproxy_info module – Returns servers for which no proxy configuration will be applied.
appliance_networking_proxy module – Configures which proxy server to use for the specified protocol
appliance_networking_proxy_info module – Gets the proxy configuration for a specific protocol.

appliance_ntp module – Set NTP servers
appliance_ntp_info module – Get the NTP configuration status
appliance_services module – Restarts a service
appliance_services_info module – Returns the state of a service.
appliance_shutdown module – Cancel pending shutdown action.
appliance_shutdown_info module – Get details about the pending shutdown action.
appliance_system_globalfips module – Enable/Disable Global FIPS mode for the appliance
appliance_system_globalfips_info module – Get current appliance FIPS settings.
appliance_system_storage module – Resize all partitions to 100 percent of disk size.
appliance_system_storage_info module – Get disk to partition mapping.
appliance_system_time_info module – Get system time.
appliance_system_time_timezone module – Set time zone.
appliance_system_time_timezone_info module – Get time zone.
appliance_system_version_info module – Get the version.
appliance_timesync module – Set time synchronization mode.
appliance_timesync_info module – Get time synchronization mode.
appliance_update_info module – Gets the current status of the appliance update.
appliance_vmon_service module – Lists details of services managed by vMon.
appliance_vmon_service_info module – Returns the state of a service.
content_configuration module – Updates the configuration
content_configuration_info module – Retrieves the current configuration values.
content_library_item_info module – Returns the {@link ItemModel} with the given identifier.
content_locallibrary module – Creates a new local library.
content_locallibrary_info module – Returns a given local library.
content_subscribedlibrary module – Creates a new subscribed library
content_subscribedlibrary_info module – Returns a given subscribed library.
vcenter_cluster_info module – Retrieves information about the cluster corresponding to {@param.name cluster}.
vcenter_datacenter module – Create a new datacenter in the vCenter inventory
vcenter_datacenter_info module – Retrieves information about the datacenter corresponding to {@param.name datacenter}.
vcenter_datastore_info module – Retrieves information about the datastore indicated by {@param.name datastore}.
vcenter_folder_info module – Returns information about at most 1000 visible (subject to permission checks) folders in vCenter matching the {@link FilterSpec}.
vcenter_host module – Add a new standalone host in the vCenter inventory
vcenter_host_info module – Returns information about at most 2500 visible (subject to permission checks) hosts in vCenter matching the {@link FilterSpec}.
vcenter_network_info module – Returns information about at most 1000 visible (subject to permission checks) networks in vCenter matching the {@link FilterSpec}.
vcenter_ovf_libraryitem module – Creates a library item in content library from a virtual machine or virtual appliance
vcenter_resourcepool module – Creates a resource pool.
vcenter_resourcepool_info module – Retrieves information about the resource pool indicated by {@param.name resourcePool}.
vcenter_storage_policies_info module – Returns information about at most 1024 visible (subject to permission checks) storage solicies availabe in vCenter
vcenter_vm module – Creates a virtual machine.
vcenter_vm_guest_customization module – Applies a customization specification on the virtual machine
vcenter_vm_guest_filesystem_directories module – Creates a directory in the guest operating system
vcenter_vm_guest_identity_info module – Return information about the guest.
vcenter_vm_guest_localfilesystem_info module – Returns details of the local file systems in the guest operating system.
vcenter_vm_guest_networking_info module – Returns information about the network configuration in the guest operating system.
vcenter_vm_guest_networking_interfaces_info module – Returns information about the networking interfaces in the guest operating system.
vcenter_vm_guest_networking_routes_info module – Returns information about network routing in the guest operating system.
vcenter_vm_guest_operations_info module – Get information about the guest operation status.
vcenter_vm_guest_power module – Issues a request to the guest operating system asking it to perform a soft shutdown, standby (suspend) or soft reboot
vcenter_vm_guest_power_info module – Returns information about the guest operating system power state.
vcenter_vm_hardware module – Updates the virtual hardware settings of a virtual machine.
vcenter_vm_hardware_adapter_sata module – Adds a virtual SATA adapter to the virtual machine.
vcenter_vm_hardware_adapter_sata_info module – Returns information about a virtual SATA adapter.
vcenter_vm_hardware_adapter_scsi module – Adds a virtual SCSI adapter to the virtual machine.
vcenter_vm_hardware_adapter_scsi_info module – Returns information about a virtual SCSI adapter.
vcenter_vm_hardware_boot module – Updates the boot-related settings of a virtual machine.
vcenter_vm_hardware_boot_device module – Sets the virtual devices that will be used to boot the virtual machine
vcenter_vm_hardware_boot_device_info module – Returns an ordered list of boot devices for the virtual machine
vcenter_vm_hardware_boot_info module – Returns the boot-related settings of a virtual machine.
vcenter_vm_hardware_cdrom module – Adds a virtual CD-ROM device to the virtual machine.
vcenter_vm_hardware_cdrom_info module – Returns information about a virtual CD-ROM device.
vcenter_vm_hardware_cpu module – Updates the CPU-related settings of a virtual machine.
vcenter_vm_hardware_cpu_info module – Returns the CPU-related settings of a virtual machine.
vcenter_vm_hardware_disk module – Adds a virtual disk to the virtual machine
vcenter_vm_hardware_disk_info module – Returns information about a virtual disk.
vcenter_vm_hardware_ethernet module – Adds a virtual Ethernet adapter to the virtual machine.
vcenter_vm_hardware_ethernet_info module – Returns information about a virtual Ethernet adapter.
vcenter_vm_hardware_floppy module – Adds a virtual floppy drive to the virtual machine.
vcenter_vm_hardware_floppy_info module – Returns information about a virtual floppy drive.
vcenter_vm_hardware_info module – Returns the virtual hardware settings of a virtual machine.
vcenter_vm_hardware_memory module – Updates the memory-related settings of a virtual machine.
vcenter_vm_hardware_memory_info module – Returns the memory-related settings of a virtual machine.
vcenter_vm_hardware_parallel module – Adds a virtual parallel port to the virtual machine.
vcenter_vm_hardware_parallel_info module – Returns information about a virtual parallel port.
vcenter_vm_hardware_serial module – Adds a virtual serial port to the virtual machine.
vcenter_vm_hardware_serial_info module – Returns information about a virtual serial port.
vcenter_vm_info module – Returns information about a virtual machine.
vcenter_vm_libraryitem_info module – Returns the information about the library item associated with the virtual machine.
vcenter_vm_power module – Operate a boot, hard shutdown, hard reset or hard suspend on a guest.
vcenter_vm_power_info module – Returns the power state information of a virtual machine.
vcenter_vm_storage_policy module – Updates the storage policy configuration of a virtual machine and/or its associated virtual hard disks.
vcenter_vm_storage_policy_compliance module – Returns the storage policy Compliance {@link Info} of a virtual machine after explicitly re-computing compliance check.
vcenter_vm_storage_policy_compliance_info module – Returns the cached storage policy compliance information of a virtual machine.
vcenter_vm_storage_policy_info module – Returns Information about Storage Policy associated with a virtual machine's home directory and/or its virtual hard disks.

vcenter_vm_tools module – Update the properties of VMware Tools.
vcenter_vm_tools_info module – Get the properties of VMware Tools.
vcenter_vm_tools_installer module – Connects the VMware Tools CD installer as a CD-ROM for the guest operating system
vcenter_vm_tools_installer_info module – Get information about the VMware Tools installer.
vcenter_vmtemplate_libraryitems module – Creates a library item in content library from a virtual machine
vcenter_vmtemplate_libraryitems_info module – Returns information about a virtual machine template contained in the library item specified by {@param.name templateLibraryItem}

Lookup Plugins
cluster_moid lookup – Look up MoID for vSphere cluster objects using vCenter REST API
datacenter_moid lookup – Look up MoID for vSphere datacenter objects using vCenter REST API
datastore_moid lookup – Look up MoID for vSphere datastore objects using vCenter REST API
folder_moid lookup – Look up MoID for vSphere folder objects using vCenter REST API
host_moid lookup – Look up MoID for vSphere host objects using vCenter REST API
network_moid lookup – Look up MoID for vSphere network objects using vCenter REST API
resource_pool_moid lookup – Look up MoID for vSphere resource pool objects using vCenter REST API
vm_moid lookup – Look up MoID for vSphere vm objects using vCenter REST API

Amazon.Aws
Collection version 7.0.0-dev0
https://ansible-collections.github.io/amazon.aws/branch/main/collections/amazon/aws/index.html
autoscaling_group module – Create or delete AWS AutoScaling Groups (ASGs)
autoscaling_group_info module – Gather information about EC2 Auto Scaling Groups (ASGs) in AWS
aws_az_info module – Gather information about availability zones in AWS
aws_caller_info module – Get information about the user and account being used to make AWS calls
backup_plan module – Manage AWS Backup Plans
backup_plan_info module – Describe AWS Backup Plans
backup_restore_job_info module – List information about backup restore jobs
backup_selection module – Create, delete and modify AWS Backup selection
backup_selection_info module – Describe AWS Backup Selections
backup_tag module – Manage tags on backup plan, backup vault, recovery point
backup_tag_info module – List tags on AWS Backup resources
backup_vault module – Manage AWS Backup Vaults
backup_vault_info module – Describe AWS Backup Vaults
cloudformation module – Create or delete an AWS CloudFormation stack
cloudformation_info module – Obtain information about an AWS CloudFormation stack
cloudtrail module – manage CloudTrail create, delete, update
cloudtrail_info module – Gather information about trails in AWS Cloud Trail.
cloudwatch_metric_alarm module – Create/update or delete AWS CloudWatch ‘metric alarms’
cloudwatch_metric_alarm_info module – Gather information about the alarms for the specified metric
cloudwatchevent_rule module – Manage CloudWatch Event rules and targets
cloudwatchlogs_log_group module – create or delete log_group in CloudWatchLogs
cloudwatchlogs_log_group_info module – Get information about log_group in CloudWatchLogs
cloudwatchlogs_log_group_metric_filter module – Manage CloudWatch log group metric filter
ec2_ami module – Create or destroy an image (AMI) in EC2
ec2_ami_info module – Gather information about ec2 AMIs
ec2_eip module – manages EC2 elastic IP (EIP) addresses.
ec2_eip_info module – List EC2 EIP details
ec2_eni module – Create and optionally attach an Elastic Network Interface (ENI) to an instance
ec2_eni_info module – Gather information about EC2 ENI interfaces in AWS
ec2_instance module – Create & manage EC2 instances
ec2_instance_info module – Gather information about ec2 instances in AWS
ec2_key module – Create or delete an EC2 key pair
ec2_key_info module – Gather information about EC2 key pairs in AWS
ec2_metadata_facts module – Gathers facts (instance metadata) about remote hosts within EC2
ec2_security_group module – Maintain an EC2 security group
ec2_security_group_info module – Gather information about EC2 security groups in AWS
ec2_snapshot module – Creates a snapshot from an existing volume
ec2_snapshot_info module – Gathers information about EC2 volume snapshots in AWS
ec2_spot_instance module – Request, stop, reboot or cancel spot instance
ec2_spot_instance_info module – Gather information about ec2 spot instance requests
ec2_tag module – Create and remove tags on ec2 resources
ec2_tag_info module – List tags on ec2 resources
ec2_vol module – Create and attach a volume, return volume ID and device map
ec2_vol_info module – Gather information about EC2 volumes in AWS
ec2_vpc_dhcp_option module – Manages DHCP Options, and can ensure the DHCP options for the given VPC match what’s requested
ec2_vpc_dhcp_option_info module – Gather information about DHCP options sets in AWS
ec2_vpc_endpoint module – Create and delete AWS VPC endpoints
ec2_vpc_endpoint_info module – Retrieves AWS VPC endpoints details using AWS methods
ec2_vpc_endpoint_service_info module – Retrieves AWS VPC endpoint service details
ec2_vpc_igw module – Manage an AWS VPC Internet gateway
ec2_vpc_igw_info module – Gather information about internet gateways in AWS
ec2_vpc_nat_gateway module – Manage AWS VPC NAT Gateways
ec2_vpc_nat_gateway_info module – Retrieves AWS VPC Managed Nat Gateway details using AWS methods
ec2_vpc_net module – Configure AWS Virtual Private Clouds
ec2_vpc_net_info module – Gather information about ec2 VPCs in AWS
ec2_vpc_route_table module – Manage route tables for AWS Virtual Private Clouds
ec2_vpc_route_table_info module – Gather information about ec2 VPC route tables in AWS
ec2_vpc_subnet module – Manage subnets in AWS virtual private clouds
ec2_vpc_subnet_info module – Gather information about ec2 VPC subnets in AWS
elb_application_lb module – Manage an Application Load Balancer
elb_application_lb_info module – Gather information about Application Load Balancers in AWS
elb_classic_lb module – Creates, updates or destroys an Amazon ELB
iam_group module – Manage AWS IAM groups
iam_instance_profile module – manage IAM instance profiles
iam_instance_profile_info module – gather information on IAM instance profiles
iam_policy module – Manage inline IAM policies for users, groups, and roles
iam_policy_info module – Retrieve inline IAM policies for users, groups, and roles

iam_user module – Manage AWS IAM users
iam_user_info module – Gather IAM user(s) facts in AWS
kms_key module – Perform various KMS key management tasks
kms_key_info module – Gather information about AWS KMS keys
lambda module – Manage AWS Lambda functions
lambda_alias module – Creates, updates or deletes AWS Lambda function aliases
lambda_event module – Creates, updates or deletes AWS Lambda function event mappings
lambda_execute module – Execute an AWS Lambda function
lambda_info module – Gathers AWS Lambda function details
lambda_layer module – Creates an AWS Lambda layer or deletes an AWS Lambda layer version
lambda_layer_info module – List lambda layer or lambda layer versions
lambda_policy module – Creates, updates or deletes AWS Lambda policy statements.
rds_cluster module – rds_cluster module
rds_cluster_info module – Obtain information about one or more RDS clusters
rds_cluster_snapshot module – Manage Amazon RDS snapshots of DB clusters
rds_instance module – Manage RDS instances
rds_instance_info module – obtain information about one or more RDS instances
rds_instance_snapshot module – Manage Amazon RDS instance snapshots
rds_option_group module – Manages the creation, modification, deletion of RDS option groups
rds_option_group_info module – rds_option_group_info module
rds_param_group module – manage RDS parameter groups
rds_snapshot_info module – obtain information about one or more RDS snapshots
rds_subnet_group module – manage RDS database subnet groups
route53 module – add or delete entries in Amazons Route 53 DNS service
route53_health_check module – Manage health-checks in Amazons Route53 DNS service
route53_info module – Retrieves route53 details using AWS methods
route53_zone module – add or delete Route53 zones
s3_bucket module – Manage S3 buckets in AWS, DigitalOcean, Ceph, Walrus, FakeS3 and StorageGRID
s3_object module – Manage objects in S3
s3_object_info module – Gather information about objects in S3

Callback Plugins
aws_resource_actions callback – summarizes all “resource:actions” completed

Inventory Plugins
aws_ec2 inventory – EC2 inventory source
aws_rds inventory – RDS instance inventory source

Lookup Plugins
aws_account_attribute lookup – Look up AWS account attributes
aws_collection_constants lookup – expose various collection related constants
aws_service_ip_ranges lookup – Look up the IP ranges for services provided in AWS such as EC2 and S3.
secretsmanager_secret lookup – Look up secrets stored in AWS Secrets Manager
ssm_parameter lookup – gets the value for a SSM parameter or all parameters under a path

Google.Cloud
Collection version 1.2.0
https://docs.ansible.com/ansible/latest/collections/google/cloud/index.html
gcp_appengine_firewall_rule module – Creates a GCP FirewallRule
gcp_appengine_firewall_rule_info module – Gather info for GCP FirewallRule
gcp_bigquery_dataset module – Creates a GCP Dataset
gcp_bigquery_dataset_info module – Gather info for GCP Dataset
gcp_bigquery_table module – Creates a GCP Table
gcp_bigquery_table_info module – Gather info for GCP Table
gcp_bigtable_instance module – Creates a GCP Instance
gcp_bigtable_instance_info module – Gather info for GCP Instance
gcp_cloudbuild_trigger module – Creates a GCP Trigger
gcp_cloudbuild_trigger_info module – Gather info for GCP Trigger
gcp_cloudfunctions_cloud_function module – Creates a GCP CloudFunction
gcp_cloudfunctions_cloud_function_info module – Gather info for GCP CloudFunction
gcp_cloudscheduler_job module – Creates a GCP Job
gcp_cloudscheduler_job_info module – Gather info for GCP Job
gcp_cloudtasks_queue module – Creates a GCP Queue
gcp_cloudtasks_queue_info module – Gather info for GCP Queue
gcp_compute_address module – Creates a GCP Address
gcp_compute_address_info module – Gather info for GCP Address
gcp_compute_autoscaler module – Creates a GCP Autoscaler
gcp_compute_autoscaler_info module – Gather info for GCP Autoscaler
gcp_compute_backend_bucket module – Creates a GCP BackendBucket
gcp_compute_backend_bucket_info module – Gather info for GCP BackendBucket
gcp_compute_backend_service module – Creates a GCP BackendService
gcp_compute_backend_service_info module – Gather info for GCP BackendService
gcp_compute_disk module – Creates a GCP Disk
gcp_compute_disk_info module – Gather info for GCP Disk
gcp_compute_external_vpn_gateway module – Creates a GCP ExternalVpnGateway
gcp_compute_external_vpn_gateway_info module – Gather info for GCP ExternalVpnGateway
gcp_compute_firewall module – Creates a GCP Firewall
gcp_compute_firewall_info module – Gather info for GCP Firewall
gcp_compute_forwarding_rule module – Creates a GCP ForwardingRule
gcp_compute_forwarding_rule_info module – Gather info for GCP ForwardingRule
gcp_compute_global_address module – Creates a GCP GlobalAddress
gcp_compute_global_address_info module – Gather info for GCP GlobalAddress
gcp_compute_global_forwarding_rule module – Creates a GCP GlobalForwardingRule
gcp_compute_global_forwarding_rule_info module – Gather info for GCP GlobalForwardingRule
gcp_compute_health_check module – Creates a GCP HealthCheck
gcp_compute_health_check_info module – Gather info for GCP HealthCheck
gcp_compute_http_health_check module – Creates a GCP HttpHealthCheck
gcp_compute_http_health_check_info module – Gather info for GCP HttpHealthCheck

gcp_compute_https_health_check module – Creates a GCP HttpsHealthCheck
gcp_compute_https_health_check_info module – Gather info for GCP HttpsHealthCheck
gcp_compute_image module – Creates a GCP Image
gcp_compute_image_info module – Gather info for GCP Image
gcp_compute_instance module – Creates a GCP Instance
gcp_compute_instance_group module – Creates a GCP InstanceGroup
gcp_compute_instance_group_info module – Gather info for GCP InstanceGroup
gcp_compute_instance_group_manager module – Creates a GCP InstanceGroupManager
gcp_compute_instance_group_manager_info module – Gather info for GCP InstanceGroupManager
gcp_compute_instance_info module – Gather info for GCP Instance
gcp_compute_instance_template module – Creates a GCP InstanceTemplate
gcp_compute_instance_template_info module – Gather info for GCP InstanceTemplate
gcp_compute_interconnect_attachment module – Creates a GCP InterconnectAttachment
gcp_compute_interconnect_attachment_info module – Gather info for GCP InterconnectAttachment
gcp_compute_network module – Creates a GCP Network
gcp_compute_network_endpoint_group module – Creates a GCP NetworkEndpointGroup
gcp_compute_network_endpoint_group_info module – Gather info for GCP NetworkEndpointGroup
gcp_compute_network_info module – Gather info for GCP Network
gcp_compute_node_group module – Creates a GCP NodeGroup
gcp_compute_node_group_info module – Gather info for GCP NodeGroup
gcp_compute_node_template module – Creates a GCP NodeTemplate
gcp_compute_node_template_info module – Gather info for GCP NodeTemplate
gcp_compute_region_autoscaler module – Creates a GCP RegionAutoscaler
gcp_compute_region_autoscaler_info module – Gather info for GCP RegionAutoscaler
gcp_compute_region_backend_service module – Creates a GCP RegionBackendService
gcp_compute_region_backend_service_info module – Gather info for GCP RegionBackendService
gcp_compute_region_disk module – Creates a GCP RegionDisk
gcp_compute_region_disk_info module – Gather info for GCP RegionDisk
gcp_compute_region_health_check module – Creates a GCP RegionHealthCheck
gcp_compute_region_health_check_info module – Gather info for GCP RegionHealthCheck
gcp_compute_region_instance_group_manager module – Creates a GCP RegionInstanceGroupManager
gcp_compute_region_instance_group_manager_info module – Gather info for GCP RegionInstanceGroupManager
gcp_compute_region_target_http_proxy module – Creates a GCP RegionTargetHttpProxy
gcp_compute_region_target_http_proxy_info module – Gather info for GCP RegionTargetHttpProxy
gcp_compute_region_target_https_proxy module – Creates a GCP RegionTargetHttpsProxy
gcp_compute_region_target_https_proxy_info module – Gather info for GCP RegionTargetHttpsProxy
gcp_compute_region_url_map module – Creates a GCP RegionUrlMap
gcp_compute_region_url_map_info module – Gather info for GCP RegionUrlMap
gcp_compute_reservation module – Creates a GCP Reservation
gcp_compute_reservation_info module – Gather info for GCP Reservation
gcp_compute_resource_policy module – Creates a GCP ResourcePolicy
gcp_compute_resource_policy_info module – Gather info for GCP ResourcePolicy
gcp_compute_route module – Creates a GCP Route
gcp_compute_route_info module – Gather info for GCP Route
gcp_compute_router module – Creates a GCP Router
gcp_compute_router_info module – Gather info for GCP Router
gcp_compute_snapshot module – Creates a GCP Snapshot
gcp_compute_snapshot_info module – Gather info for GCP Snapshot
gcp_compute_ssl_certificate module – Creates a GCP SslCertificate
gcp_compute_ssl_certificate_info module – Gather info for GCP SslCertificate
gcp_compute_ssl_policy module – Creates a GCP SslPolicy
gcp_compute_ssl_policy_info module – Gather info for GCP SslPolicy
gcp_compute_subnetwork module – Creates a GCP Subnetwork
gcp_compute_subnetwork_info module – Gather info for GCP Subnetwork
gcp_compute_target_http_proxy module – Creates a GCP TargetHttpProxy
gcp_compute_target_http_proxy_info module – Gather info for GCP TargetHttpProxy
gcp_compute_target_https_proxy module – Creates a GCP TargetHttpsProxy
gcp_compute_target_https_proxy_info module – Gather info for GCP TargetHttpsProxy
gcp_compute_target_instance module – Creates a GCP TargetInstance
gcp_compute_target_instance_info module – Gather info for GCP TargetInstance
gcp_compute_target_pool module – Creates a GCP TargetPool
gcp_compute_target_pool_info module – Gather info for GCP TargetPool
gcp_compute_target_ssl_proxy module – Creates a GCP TargetSslProxy
gcp_compute_target_ssl_proxy_info module – Gather info for GCP TargetSslProxy
gcp_compute_target_tcp_proxy module – Creates a GCP TargetTcpProxy
gcp_compute_target_tcp_proxy_info module – Gather info for GCP TargetTcpProxy
gcp_compute_target_vpn_gateway module – Creates a GCP TargetVpnGateway
gcp_compute_target_vpn_gateway_info module – Gather info for GCP TargetVpnGateway
gcp_compute_url_map module – Creates a GCP UrlMap
gcp_compute_url_map_info module – Gather info for GCP UrlMap
gcp_compute_vpn_tunnel module – Creates a GCP VpnTunnel
gcp_compute_vpn_tunnel_info module – Gather info for GCP VpnTunnel
gcp_container_cluster module – Creates a GCP Cluster
gcp_container_cluster_info module – Gather info for GCP Cluster
gcp_container_node_pool module – Creates a GCP NodePool
gcp_container_node_pool_info module – Gather info for GCP NodePool
gcp_dns_managed_zone module – Creates a GCP ManagedZone
gcp_dns_managed_zone_info module – Gather info for GCP ManagedZone
gcp_dns_resource_record_set module – Creates a GCP ResourceRecordSet
gcp_dns_resource_record_set_info module – Gather info for GCP ResourceRecordSet
gcp_filestore_instance module – Creates a GCP Instance
gcp_filestore_instance_info module – Gather info for GCP Instance
gcp_iam_role module – Creates a GCP Role
gcp_iam_role_info module – Gather info for GCP Role
gcp_iam_service_account module – Creates a GCP ServiceAccount
gcp_iam_service_account_info module – Gather info for GCP ServiceAccount
gcp_iam_service_account_key module – Creates a GCP ServiceAccountKey
gcp_kms_crypto_key module – Creates a GCP CryptoKey
gcp_kms_crypto_key_info module – Gather info for GCP CryptoKey
gcp_kms_key_ring module – Creates a GCP KeyRing

gcp_kms_key_ring_info module – Gather info for GCP KeyRing
gcp_logging_metric module – Creates a GCP Metric
gcp_logging_metric_info module – Gather info for GCP Metric
gcp_mlengine_model module – Creates a GCP Model
gcp_mlengine_model_info module – Gather info for GCP Model
gcp_mlengine_version module – Creates a GCP Version
gcp_mlengine_version_info module – Gather info for GCP Version
gcp_pubsub_subscription module – Creates a GCP Subscription
gcp_pubsub_subscription_info module – Gather info for GCP Subscription
gcp_pubsub_topic module – Creates a GCP Topic
gcp_pubsub_topic_info module – Gather info for GCP Topic
gcp_redis_instance module – Creates a GCP Instance
gcp_redis_instance_info module – Gather info for GCP Instance
gcp_resourcemanager_project module – Creates a GCP Project
gcp_resourcemanager_project_info module – Gather info for GCP Project
gcp_runtimeconfig_config module – Creates a GCP Config
gcp_runtimeconfig_config_info module – Gather info for GCP Config
gcp_runtimeconfig_variable module – Creates a GCP Variable
gcp_runtimeconfig_variable_info module – Gather info for GCP Variable
gcp_serviceusage_service module – Creates a GCP Service
gcp_serviceusage_service_info module – Gather info for GCP Service
gcp_sourcerepo_repository module – Creates a GCP Repository
gcp_sourcerepo_repository_info module – Gather info for GCP Repository
gcp_spanner_database module – Creates a GCP Database
gcp_spanner_database_info module – Gather info for GCP Database
gcp_spanner_instance module – Creates a GCP Instance
gcp_spanner_instance_info module – Gather info for GCP Instance
gcp_sql_database module – Creates a GCP Database
gcp_sql_database_info module – Gather info for GCP Database
gcp_sql_instance module – Creates a GCP Instance
gcp_sql_instance_info module – Gather info for GCP Instance
gcp_sql_ssl_cert module – Creates a GCP SslCert
gcp_sql_user module – Creates a GCP User
gcp_sql_user_info module – Gather info for GCP User
gcp_storage_bucket module – Creates a GCP Bucket
gcp_storage_bucket_access_control module – Creates a GCP BucketAccessControl
gcp_storage_default_object_acl module – Creates a GCP DefaultObjectACL
gcp_storage_object module – Creates a GCP Object
gcp_tpu_node module – Creates a GCP Node
gcp_tpu_node_info module – Gather info for GCP Node

Filter Plugins
gcp_kms_decrypt filter –
gcp_kms_encrypt filter –

Inventory Plugins
gcp_compute inventory – Google Cloud Compute Engine inventory source

Azure.Azcollection
Collection version 1.17.0
https://docs.ansible.com/ansible/latest/collections/azure/azcollection/index.html
azure_rm_account_info module – Get Azure Account facts (output of az account show)
azure_rm_adapplication module – Manage Azure Active Directory application
azure_rm_adapplication_info module – Get Azure Active Directory application info
azure_rm_adgroup module – Manage Azure Active Directory group
azure_rm_adgroup_info module – Get Azure Active Directory group info
azure_rm_adpassword module – Manage application password
azure_rm_adpassword_info module – Get application password info
azure_rm_adserviceprincipal module – Manage Azure Active Directory service principal
azure_rm_adserviceprincipal_info module – Get Azure Active Directory service principal info
azure_rm_aduser module – Modify an Azure Active Directory user
azure_rm_aduser_info module – Get Azure Active Directory user info
azure_rm_aks module – Manage a managed Azure Container Service (AKS) instance
azure_rm_aks_info module – Get Azure Kubernetes Service facts
azure_rm_aksagentpool module – Manage node pools in Kubernetes kubernetes cluster
azure_rm_aksagentpool_info module – Show the details for a node pool in the managed Kubernetes cluster
azure_rm_aksagentpoolversion_info module – Gets a list of supported versions for the specified agent pool
azure_rm_aksupgrade_info module – Get the upgrade versions available for a AKS instance
azure_rm_aksversion_info module – Get available kubernetes versions supported by Azure Kubernetes Service
azure_rm_apimanagement module – Manage Azure api instances
azure_rm_apimanagement_info module – Get the infomation of the API Instance
azure_rm_apimanagementservice module – Manage Azure ApiManagementService instance
azure_rm_apimanagementservice_info module – Get ApiManagementService info
azure_rm_appgateway module – Manage Application Gateway instance
azure_rm_appgateway_info module – Retrieve Application Gateway instance facts
azure_rm_applicationsecuritygroup module – Manage Azure Application Security Group
azure_rm_applicationsecuritygroup_info module – Get Azure Application Security Group facts
azure_rm_appserviceplan module – Manage App Service Plan
azure_rm_appserviceplan_info module – Get azure app service plan facts
azure_rm_automationaccount module – Manage Azure Automation account
azure_rm_automationaccount_info module – Get Azure automation account facts
azure_rm_automationrunbook module – Mange automation runbook
azure_rm_automationrunbook_info module – Get Azure automation runbook facts
azure_rm_autoscale module – Manage Azure autoscale setting
azure_rm_autoscale_info module – Get Azure Auto Scale Setting facts
azure_rm_availabilityset module – Manage Azure Availability Set
azure_rm_availabilityset_info module – Get Azure Availability Set facts
azure_rm_azurefirewall module – Manage Azure Firewall instance

azure_rm_azurefirewall_info module – Get AzureFirewall info
azure_rm_backupazurevm module – Back up an Azure Virtual Machine using Azure Backup
azure_rm_backupazurevm_info module – Back up an Azure Virtual Machine using Azure Backup Information
azure_rm_backuppolicy module – Manage Azure Backup Policy
azure_rm_backuppolicy_info module – Get Info on Azure Backup Policy
azure_rm_bastionhost module – Managed bastion host resource
azure_rm_bastionhost_info module – Get Azure bastion host info
azure_rm_batchaccount module – Manages a Batch Account on Azure
azure_rm_batchaccount_info module – Get the Batch Account on Azure facts
azure_rm_cdnendpoint module – Manage a Azure CDN endpoint
azure_rm_cdnendpoint_info module – Get Azure CDN endpoint facts
azure_rm_cdnprofile module – Manage a Azure CDN profile
azure_rm_cdnprofile_info module – Get Azure CDN profile facts
azure_rm_cognitivesearch module – Manage Azure Cognitive Search service
azure_rm_cognitivesearch_info module – Get Azure Cognitive Search service info
azure_rm_containerinstance module – Manage an Azure Container Instance
azure_rm_containerinstance_info module – Get Azure Container Instance facts
azure_rm_containerregistry module – Manage an Azure Container Registry
azure_rm_containerregistry_info module – Get Azure Container Registry facts
azure_rm_containerregistryreplication module – Manage Replication instance.
azure_rm_containerregistryreplication_info module – Get Replication facts.
azure_rm_containerregistrytag module – Import or delete tags in Azure Container Registry
azure_rm_containerregistrytag_info module – Get Azure Container Registry tag facts
azure_rm_containerregistrywebhook module – Manage Webhook instance.
azure_rm_containerregistrywebhook_info module – Get Webhook facts.
azure_rm_cosmosdbaccount module – Manage Azure Database Account instance
azure_rm_cosmosdbaccount_info module – Get Azure Cosmos DB Account facts
azure_rm_datafactory module – Managed data factory
azure_rm_datafactory_info module – Get data factory facts
azure_rm_datalakestore module – Manage Azure data lake store
azure_rm_datalakestore_info module – Get Azure Data Lake Store info
azure_rm_ddosprotectionplan module – Manage DDoS protection plan
azure_rm_ddosprotectionplan_info module – Get Azure DDoS protection plan
azure_rm_deployment module – Create or destroy Azure Resource Manager template deployments
azure_rm_deployment_info module – Get Azure Deployment facts
azure_rm_devtestlab module – Manage Azure DevTest Lab instance
azure_rm_devtestlab_info module – Get Azure DevTest Lab facts
azure_rm_devtestlabarmtemplate_info module – Get Azure DevTest Lab ARM Template facts
azure_rm_devtestlabartifact_info module – Get Azure DevTest Lab Artifact facts
azure_rm_devtestlabartifactsource module – Manage Azure DevTest Labs Artifacts Source instance
azure_rm_devtestlabartifactsource_info module – Get Azure DevTest Lab Artifact Source facts
azure_rm_devtestlabcustomimage module – Manage Azure DevTest Lab Custom Image instance
azure_rm_devtestlabcustomimage_info module – Get Azure DevTest Lab Custom Image facts
azure_rm_devtestlabenvironment module – Manage Azure DevTest Lab Environment instance
azure_rm_devtestlabenvironment_info module – Get Azure Environment facts
azure_rm_devtestlabpolicy module – Manage Azure Policy instance
azure_rm_devtestlabpolicy_info module – Get Azure DTL Policy facts
azure_rm_devtestlabschedule module – Manage Azure DevTest Lab Schedule instance
azure_rm_devtestlabschedule_info module – Get Azure Schedule facts
azure_rm_devtestlabvirtualmachine module – Manage Azure DevTest Lab Virtual Machine instance
azure_rm_devtestlabvirtualmachine_info module – Get Azure DevTest Lab Virtual Machine facts
azure_rm_devtestlabvirtualnetwork module – Manage Azure DevTest Lab Virtual Network instance
azure_rm_devtestlabvirtualnetwork_info module – Get Azure DevTest Lab Virtual Network facts
azure_rm_diskencryptionset module – Create, delete and update Disk encryption set
azure_rm_diskencryptionset_info module – Get disk encryption set facts
azure_rm_dnsrecordset module – Create, delete and update DNS record sets and records
azure_rm_dnsrecordset_info module – Get DNS Record Set facts
azure_rm_dnszone module – Manage Azure DNS zones
azure_rm_dnszone_info module – Get DNS zone facts
azure_rm_eventhub module – Manage Event Hub
azure_rm_eventhub_info module – Get Azure Event Hub
azure_rm_expressroute module – Manage Express Route Circuits
azure_rm_expressroute_info module – Get Azure Express Route
azure_rm_firewallpolicy module – Create, delete or update specified firewall policy.
azure_rm_firewallpolicy_info module – Get firewall policy facts
azure_rm_functionapp module – Manage Azure Function Apps
azure_rm_functionapp_info module – Get Azure Function App facts
azure_rm_gallery module – Manage Azure Shared Image Gallery instance
azure_rm_gallery_info module – Get Azure Shared Image Gallery info
azure_rm_galleryimage module – Manage Azure SIG Image instance
azure_rm_galleryimage_info module – Get Azure SIG Image info
azure_rm_galleryimageversion module – Manage Azure SIG Image Version instance
azure_rm_galleryimageversion_info module – Get Azure SIG Image Version info
azure_rm_hdinsightcluster module – Manage Azure HDInsight Cluster instance
azure_rm_hdinsightcluster_info module – Get Azure HDInsight Cluster facts
azure_rm_hostgroup module – Create, delete and update a dedicated host group
azure_rm_hostgroup_info module – Get host group facts
azure_rm_image module – Manage Azure image
azure_rm_image_info module – Get facts about azure custom images
azure_rm_iotdevice module – Manage Azure IoT hub device
azure_rm_iotdevice_info module – Facts of Azure IoT hub device
azure_rm_iotdevicemodule module – Manage Azure IoT hub device module
azure_rm_iothub module – Manage Azure IoT hub
azure_rm_iothub_info module – Get IoT Hub facts
azure_rm_iothubconsumergroup module – Manage Azure IoT hub
azure_rm_ipgroup module – Create, delete and update IP group
azure_rm_ipgroup_info module – Get IP group facts
azure_rm_keyvault module – Manage Key Vault instance
azure_rm_keyvault_info module – Get Azure Key Vault facts
azure_rm_keyvaultkey module – Use Azure KeyVault keys

azure_rm_keyvaultkey_info module – Get Azure Key Vault key facts
azure_rm_keyvaultsecret module – Use Azure KeyVault Secrets
azure_rm_keyvaultsecret_info module – Get Azure Key Vault secret facts
azure_rm_loadbalancer module – Manage Azure load balancers
azure_rm_loadbalancer_info module – Get load balancer facts
azure_rm_lock module – Manage Azure locks
azure_rm_lock_info module – Manage Azure locks
azure_rm_loganalyticsworkspace module – Manage Azure Log Analytics workspaces
azure_rm_loganalyticsworkspace_info module – Get facts of Azure Log Analytics workspaces
azure_rm_manageddisk module – Manage Azure Manage Disks
azure_rm_manageddisk_info module – Get managed disk facts
azure_rm_managementgroup module – Manage Azure ManagementGroup instance
azure_rm_managementgroup_info module – Get Azure Management Group facts
azure_rm_mariadbconfiguration module – Manage Configuration instance
azure_rm_mariadbconfiguration_info module – Get Azure MariaDB Configuration facts
azure_rm_mariadbdatabase module – Manage MariaDB Database instance
azure_rm_mariadbdatabase_info module – Get Azure MariaDB Database facts
azure_rm_mariadbfirewallrule module – Manage MariaDB firewall rule instance
azure_rm_mariadbfirewallrule_info module – Get Azure MariaDB Firewall Rule facts
azure_rm_mariadbserver module – Manage MariaDB Server instance
azure_rm_mariadbserver_info module – Get Azure MariaDB Server facts
azure_rm_monitordiagnosticsetting module – Create, update, or manage Azure Monitor diagnostic settings.
azure_rm_monitordiagnosticsetting_info module – Get Azure Monitor diagnostic setting facts.
azure_rm_monitorlogprofile module – Manage Azure Monitor log profile
azure_rm_multiplemanageddisks module – Manage Multiple Azure Manage Disks
azure_rm_mysqlconfiguration module – Manage Configuration instance
azure_rm_mysqlconfiguration_info module – Get Azure MySQL Configuration facts
azure_rm_mysqldatabase module – Manage MySQL Database instance
azure_rm_mysqldatabase_info module – Get Azure MySQL Database facts
azure_rm_mysqlfirewallrule module – Manage MySQL firewall rule instance
azure_rm_mysqlfirewallrule_info module – Get Azure MySQL Firewall Rule facts
azure_rm_mysqlserver module – Manage MySQL Server instance
azure_rm_mysqlserver_info module – Get Azure MySQL Server facts
azure_rm_natgateway module – Manage Azure NAT Gateway instance
azure_rm_natgateway_info module – Retrieve NAT Gateway instance facts
azure_rm_networkinterface module – Manage Azure network interfaces
azure_rm_networkinterface_info module – Get network interface facts
azure_rm_notificationhub module – Manage Notification Hub
azure_rm_notificationhub_info module – Get Azure Notification Hub
azure_rm_openshiftmanagedcluster module – Manage Azure Red Hat OpenShift Managed Cluster instance
azure_rm_openshiftmanagedcluster_info module – Get Info onf Azure Red Hat OpenShift Managed Cluster
azure_rm_postgresqlconfiguration module – Manage Azure PostgreSQL Configuration
azure_rm_postgresqlconfiguration_info module – Get Azure PostgreSQL Configuration facts
azure_rm_postgresqldatabase module – Manage PostgreSQL Database instance
azure_rm_postgresqldatabase_info module – Get Azure PostgreSQL Database facts
azure_rm_postgresqlfirewallrule module – Manage PostgreSQL firewall rule instance
azure_rm_postgresqlfirewallrule_info module – Get Azure PostgreSQL Firewall Rule facts
azure_rm_postgresqlserver module – Manage PostgreSQL Server instance
azure_rm_postgresqlserver_info module – Get Azure PostgreSQL Server facts
azure_rm_privatednsrecordset module – Create, delete and update Private DNS record sets and records
azure_rm_privatednsrecordset_info module – Get Private DNS Record Set facts
azure_rm_privatednszone module – Manage Azure private DNS zones
azure_rm_privatednszone_info module – Get private DNS zone facts
azure_rm_privatednszonelink module – Create, delete and update Virtual network link for Private DNS zone
azure_rm_privatednszonelink_info module – Get Virtual Network link facts for private DNS zone
azure_rm_privateendpoint module – Manage Azure private endpoint
azure_rm_privateendpoint_info module – Get private endpoints info
azure_rm_privateendpointconnection module – Managed private endpoint connection
azure_rm_privateendpointconnection_info module – Get private endpoint connection info
azure_rm_privateendpointdnszonegroup module – Create, update, or manage private endpoint DNS zone groups.
azure_rm_privateendpointdnszonegroup_info module – Get private endpoint DNS zone group info.
azure_rm_privatelinkservice module – Managed private link service resource
azure_rm_privatelinkservice_info module – Get private endpoint connection info
azure_rm_proximityplacementgroup module – Create, delete and update proximity placement group
azure_rm_proximityplacementgroup_info module – Get proximity placement group facts
azure_rm_publicipaddress module – Manage Azure Public IP Addresses
azure_rm_publicipaddress_info module – Get public IP facts
azure_rm_recoveryservicesvault module – Create and Delete Azure Recovery Services vault
azure_rm_recoveryservicesvault_info module – Get Azure Recovery Services vault Details
azure_rm_rediscache module – Manage Azure Cache for Redis instance
azure_rm_rediscache_info module – Get Azure Cache for Redis instance facts
azure_rm_rediscachefirewallrule module – Manage Azure Cache for Redis Firewall rules
azure_rm_registrationassignment module – Manage Azure RegistrationAssignment instance
azure_rm_registrationassignment_info module – Get RegistrationAssignment info
azure_rm_registrationdefinition module – Manage Azure RegistrationDefinition instance
azure_rm_registrationdefinition_info module – Get RegistrationDefinition info
azure_rm_resource module – Create any Azure resource
azure_rm_resource_info module – Generic facts of Azure resources
azure_rm_resourcegroup module – Manage Azure resource groups
azure_rm_resourcegroup_info module – Get resource group facts
azure_rm_roleassignment module – Manage Azure Role Assignment
azure_rm_roleassignment_info module – Gets Azure Role Assignment facts
azure_rm_roledefinition module – Manage Azure Role Definition
azure_rm_roledefinition_info module – Get Azure Role Definition facts
azure_rm_route module – Manage Azure route resource
azure_rm_route_info module – Get Route info
azure_rm_routetable module – Manage Azure route table resource
azure_rm_routetable_info module – Get route table facts
azure_rm_securitygroup module – Manage Azure network security groups
azure_rm_securitygroup_info module – Get security group facts

azure_rm_servicebus module – Manage Azure Service Bus
azure_rm_servicebus_info module – Get servicebus facts
azure_rm_servicebusqueue module – Manage Azure Service Bus queue
azure_rm_servicebussaspolicy module – Manage Azure Service Bus SAS policy
azure_rm_servicebustopic module – Manage Azure Service Bus
azure_rm_servicebustopicsubscription module – Manage Azure Service Bus subscription
azure_rm_snapshot module – Manage Azure Snapshot instance
azure_rm_sqldatabase module – Manage SQL Database instance
azure_rm_sqldatabase_info module – Get Azure SQL Database facts
azure_rm_sqlelasticpool module – Manage SQL Elastic Pool instance
azure_rm_sqlelasticpool_info module – Get Azure SQL Elastic Pool facts
azure_rm_sqlfirewallrule module – Manage Firewall Rule instance
azure_rm_sqlfirewallrule_info module – Get Azure SQL Firewall Rule facts
azure_rm_sqlmanagedinstance module – Manage SQL managed instances
azure_rm_sqlmanagedinstance_info module – Get Azure SQL managed instance facts
azure_rm_sqlserver module – Manage SQL Server instance
azure_rm_sqlserver_info module – Get SQL Server facts
azure_rm_storageaccount module – Manage Azure storage accounts
azure_rm_storageaccount_info module – Get storage account facts
azure_rm_storageblob module – Manage blob containers and blob objects
azure_rm_storageshare module – Manage Azure storage file share
azure_rm_storageshare_info module – Get Azure storage file share info
azure_rm_subnet module – Manage Azure subnets
azure_rm_subnet_info module – Get Azure Subnet facts
azure_rm_subscription_info module – Get Azure Subscription facts
azure_rm_trafficmanager module – Manage a Traffic Manager profile.
azure_rm_trafficmanagerendpoint module – Manage Azure Traffic Manager endpoint
azure_rm_trafficmanagerendpoint_info module – Get Azure Traffic Manager endpoint facts
azure_rm_trafficmanagerprofile module – Manage Azure Traffic Manager profile
azure_rm_trafficmanagerprofile_info module – Get Azure Traffic Manager profile facts
azure_rm_virtualhub module – Manage Azure VirtualHub instance
azure_rm_virtualhub_info module – Get VirtualHub info
azure_rm_virtualhubconnection module – Manage Azure VirtualHub instance
azure_rm_virtualhubconnection_info module – Get VirtualHub info
azure_rm_virtualmachine module – Manage Azure virtual machines
azure_rm_virtualmachine_info module – Get virtual machine facts
azure_rm_virtualmachineextension module – Managed Azure Virtual Machine extension
azure_rm_virtualmachineextension_info module – Get Azure Virtual Machine Extension facts
azure_rm_virtualmachineimage_info module – Get virtual machine image facts
azure_rm_virtualmachinescaleset module – Manage Azure virtual machine scale sets
azure_rm_virtualmachinescaleset_info module – Get Virtual Machine Scale Set facts
azure_rm_virtualmachinescalesetextension module – Manage Azure Virtual Machine Scale Set (VMSS) extensions
azure_rm_virtualmachinescalesetextension_info module – Get Azure Virtual Machine Scale Set Extension facts
azure_rm_virtualmachinescalesetinstance module – Get Azure Virtual Machine Scale Set Instance facts
azure_rm_virtualmachinescalesetinstance_info module – Get Azure Virtual Machine Scale Set Instance facts
azure_rm_virtualmachinesize_info module – Get facts for virtual machine sizes
azure_rm_virtualnetwork module – Manage Azure virtual networks
azure_rm_virtualnetwork_info module – Get virtual network facts
azure_rm_virtualnetworkgateway module – Manage Azure virtual network gateways
azure_rm_virtualnetworkpeering module – Manage Azure Virtual Network Peering
azure_rm_virtualnetworkpeering_info module – Get facts of Azure Virtual Network Peering
azure_rm_virtualwan module – Manage Azure VirtualWan instance
azure_rm_virtualwan_info module – Get VirtualWan info
azure_rm_vmbackuppolicy module – Create or Delete Azure VM Backup Policy
azure_rm_vmbackuppolicy_info module – Fetch Backup Policy Details
azure_rm_vmssnetworkinterface_info module – Get information about network interface in virtul machine scale
azure_rm_vpnsite module – Manage Azure VpnSite instance
azure_rm_vpnsite_info module – Get VpnSite info
azure_rm_vpnsitelink_info module – Get VpnSiteLink info
azure_rm_webapp module – Manage Web App instances
azure_rm_webapp_info module – Get Azure web app facts
azure_rm_webappaccessrestriction module – Manage web app network access restrictions
azure_rm_webappaccessrestriction_info module – Retrieve web app network access restriction facts
azure_rm_webappslot module – Manage Azure Web App slot
azure_rm_webappvnetconnection module – Manage web app virtual network connection
azure_rm_webappvnetconnection_info module – Get Azure web app virtual network connection facts

Inventory Plugins
azure_rm inventory – Azure Resource Manager inventory plugin

Lookup Plugins
azure_keyvault_secret lookup – Read secret from Azure Key Vault.

Junipernetworks.Junos
Collection version 5.3.0
https://docs.ansible.com/ansible/latest/collections/junipernetworks/junos/index.html
junos_acl_interfaces module – ACL interfaces resource module
junos_acls module – ACLs resource module
junos_banner module – Manage multiline banners on Juniper JUNOS devices
junos_bgp_address_family module – Manage BGP Address Family attributes of interfaces on Junos devices.
junos_bgp_global module – Manages BGP Global configuration on devices running Juniper JUNOS.
junos_command module – Run arbitrary commands on an Juniper JUNOS device
junos_config module – Manage configuration on devices running Juniper JUNOS
junos_facts module – Collect facts from remote devices running Juniper Junos
junos_hostname module – Manage Hostname server configuration on Junos devices.
junos_interfaces module – Junos Interfaces resource module
junos_l2_interfaces module – L2 interfaces resource module
junos_l3_interfaces module – L3 interfaces resource module

junos_lacp module – Global Link Aggregation Control Protocol (LACP) Junos resource module
junos_lacp_interfaces module – LACP interfaces resource module
junos_lag_interfaces module – Link Aggregation Juniper JUNOS resource module
junos_lldp_global module – LLDP resource module
junos_lldp_interfaces module – LLDP interfaces resource module
junos_logging module – Manage logging on network devices
junos_logging_global module – Manage logging configuration on Junos devices.
junos_netconf module – Configures the Junos Netconf system service
junos_ntp_global module – Manage NTP configuration on Junos devices.
junos_ospf_interfaces module – OSPF Interfaces Resource Module.
junos_ospfv2 module – OSPFv2 resource module
junos_ospfv3 module – OSPFv3 resource module
junos_package module – Installs packages on remote devices running Junos
junos_ping module – Tests reachability using ping from devices running Juniper JUNOS
junos_prefix_lists module – Manage prefix-lists attributes of interfaces on Junos devices.
junos_routing_instances module – Manage routing instances on Junos devices.
junos_routing_options module – Manage routing-options configuration on Junos devices.
junos_rpc module – Runs an arbitrary RPC over NetConf on an Juniper JUNOS device
junos_scp module – Transfer files from or to remote devices running Junos
junos_security_policies module – Create and manage security policies on Juniper JUNOS devices
junos_security_policies_global module – Manage global security policy settings on Juniper JUNOS devices
junos_security_zones module – Manage security zones on Juniper JUNOS devices
junos_snmp_server module – Manage SNMP server configuration on Junos devices.
junos_static_routes module – Static routes resource module
junos_system module – Manage the system attributes on Juniper JUNOS devices
junos_user module – Manage local user accounts on Juniper JUNOS devices
junos_vlans module – VLANs resource module
junos_vrf module – Manage the VRF definitions on Juniper JUNOS devices

Cliconf Plugins
junos cliconf – Use junos cliconf to run command on Juniper Junos OS platform

Netconf Plugins
junos netconf – Use junos netconf plugin to run netconf commands on Juniper JUNOS platform

Cisco.Ios
Collection version 4.6.1
https://docs.ansible.com/ansible/latest/collections/cisco/ios/index.html
ios_acl_interfaces module – Resource module to configure ACL interfaces.
ios_acls module – Resource module to configure ACLs.
ios_banner module – Module to configure multiline banners.
ios_bgp module – Module to configure BGP protocol settings.
ios_bgp_address_family module – Resource module to configure BGP Address family.
ios_bgp_global module – Resource module to configure BGP.
ios_command module – Module to run commands on remote devices.
ios_config module – Module to manage configuration sections.
ios_facts module – Module to collect facts from remote devices.
ios_hostname module – Resource module to configure hostname.
ios_interfaces module – Resource module to configure interfaces.
ios_l2_interfaces module – Resource module to configure L2 interfaces.
ios_l3_interfaces module – Resource module to configure L3 interfaces.
ios_lacp module – Resource module to configure LACP.
ios_lacp_interfaces module – Resource module to configure LACP interfaces.
ios_lag_interfaces module – Resource module to configure LAG interfaces.
ios_linkagg module – Module to configure link aggregation groups.
ios_lldp module – (deprecated, removed after 2024-06-01) Manage LLDP configuration on Cisco IOS network devices.
ios_lldp_global module – Resource module to configure LLDP.
ios_lldp_interfaces module – Resource module to configure LLDP interfaces.
ios_logging module – (deprecated, removed after 2023-06-01) Manage logging on network devices
ios_logging_global module – Resource module to configure logging.
ios_ntp module – (deprecated, removed after 2024-01-01) Manages core NTP configuration.
ios_ntp_global module – Resource module to configure NTP.
ios_ospf_interfaces module – Resource module to configure OSPF interfaces.
ios_ospfv2 module – Resource module to configure OSPFv2.
ios_ospfv3 module – Resource module to configure OSPFv3.
ios_ping module – Tests reachability using ping from IOS switch.
ios_prefix_lists module – Resource module to configure prefix lists.
ios_route_maps module – Resource module to configure route maps.
ios_service module – Resource module to configure service.
ios_snmp_server module – Resource module to configure snmp server.
ios_static_routes module – Resource module to configure static routes.
ios_system module – Module to manage the system attributes.
ios_user module – Module to manage the aggregates of local users.
ios_vlans module – Resource module to configure VLANs.
ios_vrf module – Module to configure VRF definitions.

Cliconf Plugins
ios cliconf – Use ios cliconf to run command on Cisco IOS platform

AWS – GCP – AZURE Equivalents 2023 – adopted from GCP Documentation

https://cloud.google.com/docs/get-started/aws-azure-gcp-service-comparison

Service category	Service type	AWS offering	Azure offering	Google Cloud product	Google Cloud product description
App modernization	CI/CD	AWS CodeBuild, AWS CodeDeploy, AWS CodePipeline	Azure DevOps, GitHub Enterprise	Cloud Build	Build, test, and deploy on Google Cloud serverless CI/CD platform
App modernization	CI/CD	AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy	Azure DevOps	Google Cloud Deploy	Deliver continuously to Google Kubernetes Engine and Anthos.
App modernization	Execution Control	Amazon EventBridge, Amazon Simple Notification Service (SNS)	Azure Service Bus, Azure Storage Queues	Cloud Tasks	Control and observe asynchronous service requests between independent applications using this zonal, execution-control service.
App modernization	Multi-cloud	Amazon EKS Anywhere, Amazon ECS Anywhere, AWS Outposts	Azure Arc	Anthos	Migrate directly from VMs, build, deploy, and optimize apps on GKE, Anthos serverless landing zones and VMs anywhere—simply, flexibly, and securely
App modernization	Multi-cloud	Amazon EKS Anywhere		Anthos Clusters	Extend GKE to work in multiple environments, including attached clusters, AWS, Azure, bare metal, and VMWare.
App modernization	Multi-cloud	AWS Systems Manager	Azure App Configuration	Anthos Config Management	Automate policy and security at scale for your hybrid and multi-cloud Kubernetes deployments.
App modernization	Multi-cloud	AWS Controllers for Kubernetes	Azure Service Operator	Config Connector	Manage Google Cloud resources through Kubernetes.
App modernization	Multi-cloud	AWS Bottlerocket	Azure Container Instances	Container-Optimized OS	Efficiently and securely run Docker containers on Compute Engine VMs.
App modernization	Multi-cloud	AWS Outposts	Azure Stack	Google Distributed Cloud	Extend Google Cloud's infrastructure and services to the edge and your data centers.
App modernization	Multi-cloud	AWS Direct Connect	Azure Express Route	Hybrid Connectivity	Connect your infrastructure to Google Cloud on your terms, from anywhere.
App modernization	Multi-cloud serverless			Cloud Run for Anthos	Flexible serverless development for multicloud environments.
App modernization	Service mesh	AWS App Mesh	Azure Service Fabric	Anthos Service Mesh	Simplify, manage, and secure complex microservices architectures with this fully managed service.
App modernization	Service mesh	Amazon VPC	Azure VPN Gateway	Cloud Router	Dynamically exchange routes between your Virtual Private Cloud (VPC) and on-premises networks by using Border Gateway Protocol (BGP).
App modernization	Service mesh	Istio on Amazon EKS	Istio in Azure Kubernetes Service	Istio on Google Kubernetes Engine	Quickly create GKE clusters with all the components you need to create and run an Istio service mesh in a single step.
AI & ML	Cloud cost optimization	AWS Cost Optimization	Azure Cost Management	Recommender	Optimize your Google Cloud usage with proactive, easily actionable recommendations.
AI & ML	Conversational interface	Amazon Lex	Azure Conversational AI	Dialogflow	Lifelike conversational AI with state-of-the-art virtual agents.
AI & ML	Document understanding	Amazon Textract	Azure Form Recognizer	Document AI	Automate data capture at scale to reduce document processing costs.
AI & ML	Image recognition	Amazon Rekognition Image	Azure Computer Vision	Vision AI	Derive insights from your images in the cloud or at the edge, or use pre-trained Vision API models to detect emotion, understand text, and more.
AI & ML	ML for structured data	Amazon SageMaker	AutoML in Azure ML Studio	Vertex AI AutoML tabular models	Automatically build and deploy state-of-the-art machine learning models on structured data.
AI & ML	ML platform	Amazon SageMaker, Amazon EC2 P3	Azure Data Science Virtual Machines	Deep Learning VM Images	Preconfigured VMs for deep learning applications.
AI & ML	ML platform	Tensorflow on AWS	Azure Databricks	TensorFlow Enterprise	Reliability and performance for AI applications with enterprise-grade support and managed services.
AI & ML	ML platform	Amazon SageMaker	Azure AI Platform	Vertex AI	Train your machine learning models at scale, to host your trained model in the cloud, and to use your model to make predictions about new data.
AI & ML	ML platform	Amazon SageMaker Autopilot	Azure Cognitive Services	Vertex AI AutoML models	Train high-quality custom machine learning models with minimal effort and machine learning expertise.
AI & ML	ML platform	Amazon SageMaker	Azure Machine Learning	Vertex AI custom training	Host your machine learning models and train them with the power and flexibility of TensorFlow, scikit-learn, XGBoost, and custom containers.
AI & ML	ML platform	Amazon SageMaker	Azure AI Platform	Vertex AI custom-trained models	Host your trained models so that you can send them prediction requests with the power and flexibility of TensorFlow, scikit-learn, and XGBoost.
AI & ML	ML platform	Amazon SageMaker	Azure Notebooks	Vertex AI Workbench	Create instances running JupyterLab that come pre-installed with the latest data science and machine learning frameworks in a single click.
AI & ML	Natural language processing	Amazon Comprehend	Azure Text Analytics	Natural Language AI	Derive insights from unstructured text using Google machine learning.
AI & ML	Personalization	Amazon Personalize	Azure Personalizer	Recommendations AI	Deliver highly personalized product recommendations at scale.
AI & ML	Speech recognition	Amazon Transcribe	Azure Speech to Text	Speech-to-Text	Accurately convert speech into text using an API powered by Google's AI technologies.
AI & ML	Speech synthesis	Amazon Polly	Azure Text to Speech	Text-to-Speech	Convert text into natural-sounding speech using an API powered by Google's AI technologies.
AI & ML	Translation	Amazon Translate	Azure Translator	Translation AI	Dynamically translate between languages using Google machine learning.
AI & ML	Video intelligence	Amazon Rekognition Video	Azure Video Indexer	Video Intelligence API	Quickly categorize video content using thousands of predefined labels and creating additional custom labels to suit your specific needs.
Backup & DR	SaaS	AWS Resilience Hub	Azure Backup and Disaster Recovery	Actifio	Protect your data and business with this backup and disaster recovery offering that supports Google Cloud workloads as well as hybrid workloads like VMware, SAP HANA, Oracle, or SQL Server.
Compute	Core compute	Amazon Elastic Compute Cloud (EC2) P3	GPU Optimized VMs	Cloud GPUs	Train and run machine learning models faster than before.
Compute	Core compute	AWS UltraClusters	Azure Virtual Machines	Cloud TPU	Train and run machine learning models faster than ever before.
Compute	Core compute	Amazon Elastic Compute Cloud (EC2)	Azure Virtual Machines	Compute Engine	Accelerate your digital transformation with high-performance VMs.
Compute	Core compute	AWS EC2 Autoscaling	Azure Autoscale, Azure Virtual Machine Scale Sets	Compute Engine Autoscaler	Automatically add or delete VM instances from a managed instance group (MIG) based on increases or decreases in load.
Compute	Core compute	Amazon EC2 Instance Connect		OS Login	Manage SSH access to your instances using IAM without having to create and manage individual SSH keys.
Compute	Core compute	Amazon Elastic Block Store (EBS)	Azure Managed Disks	Persistent Disk	Reliable, high-performance block storage for VM instances.
Compute	Core compute	AWS EC2 Instance Connect	Azure Bastion	SSH from the browser	Connect to a Compute Engine virtual machine (VM) instance using SSH with the Google Cloud console in your web browser.
Compute	Core Compute	AWS Systems Manager		VM Manager	Manage operating systems for large virtual machine (VM) fleets running Windows and Linux on Compute Engine.
Compute	Dedicated VMs	Amazon EC2 Dedicated Host	Azure Dedicated Host	Sole-tenant nodes	Host your VMs on hardware dedicated only to your project.
Compute	Infrastructure modernization	SAP on AWS	SAP on Azure	SAP on Google Cloud	Run SAP on Google Cloud.
Compute	PaaS	AWS Lambda, AWS Fargate, AWS App Runner	Azure App Service	App Engine	Build highly scalable applications on a fully managed serverless platform.
Compute	VMware connectivity	VMware Cloud on AWS	Azure VMware Solution	VMware Engine	Migrate and run your VMware workloads on Google Cloud.
Containers	CaaS	Amazon Elastic Kubernetes Service (EKS), Amazon Elastic Container Service (ECS)	Azure Kubernetes Service (AKS)	Google Kubernetes Engine	Secured and managed Kubernetes service with four-way autoscaling and multi-cluster support.
Containers	Container registry	Amazon Elastic Container Registry (ECR)	Azure Container Registry	Artifact Registry	Store, manage, and secure your container images.
Containers	Container Security			Binary Authorization	Require images to be signed by trusted authorities during the development process and then enforce signature validation when deploying.
Containers	Gaming	Amazon GameLift	Azure for Gaming	Game Servers	Deliver seamless multiplayer gaming experiences with simpler multicluster management.
Data analytics	Business intelligence	Amazon QuickSight	Microsoft Power BI	Looker	Explore, share, and visualize your company's data so that you can make better business decisions.
Data analytics	Data discovery/ metadata mgmt	AWS Glue Data Catalog	Azure Purview, Azure Data Explorer	Data Catalog	Discover, understand, and manage data at scale with powerful search and seamless integration to BigQuery, Pub/Sub, Cloud Storage, secured via IAM and Cloud Data Loss Prevention.
Data analytics	Data integration / ETL	Amazon AppFlow, Amazon Data Pipeline, AWS Glue	Azure Data Factory	Cloud Data Fusion	Activate fully managed, cloud-native data integration at scale.

Data analytics	Data processing	Amazon Elastic MapReduce (EMR), AWS Batch, AWS Glue	Azure Data Lake Analytics, HDInsight	Dataproc	Deploy open-source data and analytics processing services (Apache Hadoop, Apache Spark, etc.) with improved efficiency and security.
Data analytics	Data warehouse	Amazon Athena, Amazon Redshift	Azure Synapse Analytics	BigQuery	Serverless, highly scalable, and cost-effective multi-cloud data warehouse designed for business agility.
Data analytics	Data wrangling	AWS Glue Data Brew	Azure Data Factory	Dataprep by Trifecta	An intelligent cloud data service to visually explore, clean, and prepare data for analysis and machine learning.
Data analytics	Messaging	AWS Kinesis, Amazon MQ	Azure Service Bus Messaging	Pub/Sub	Messaging and ingestion for event-driven systems and streaming analytics.
Data analytics	Messaging	Amazon Simple Notification Service, Amazon Simple Queueing Service	Azure Service Bus Messaging	Pub/Sub Lite	Send and receive messages between independent applications using this zonal, real-time messaging service.
Data analytics	Query service	Amazon Redshift Spectrum	Azure Synapse Analytics	BigQuery	Analyze petabytes of data at scale using ANSI SQL and gain 26%–34% lower three-year total cost of ownership (TCO) than competing cloud data warehouses.
Data analytics	Stream data ingest	Amazon Kinesis	Azure Event Hubs	Pub/Sub	Create scalable messaging and ingestion for event-driven systems and streaming analytics.
Data analytics	Stream data processing	Amazon Kinesis Data Firehose	Azure Stream Analytics	Dataflow	Unifiy stream and batch data processing that's serverless, fast, and cost-effective.
Data analytics	Workflow orchestration	Amazon Data Pipeline, AWS Glue, Managed Workflows for Apache Airflow	Azure Data Factory	Cloud Composer	Author, schedule, and monitor pipelines that span across hybrid and multi-cloud environments using this fully managed workflow orchestration service built on Apache Airflow.
Database	Document data storage	Amazon DocumentDB, Amazon DynamoDB	Azure Cosmos DB	Firestore	Easily develop rich applications using a fully managed, scalable, and serverless document database.
Database	In-memory data store	Amazon ElastiCache	Azure Cache	Memorystore	Reduce latency with scalable, secure, and highly available in-memory service for Redis and Memcached.
Database	NoSQL: Indexed	Amazon DynamoDB	Azure Cosmos DB	Datastore	A highly scalable NoSQL database for your web and mobile applications.
Database	NoSQL: Key-value	Amazon DynamoDB	Azure Cosmos DB	Cloud Bigtable	Run large analytical and operational workloads using this fully managed, scalable NoSQL database service.
Database	RDBMS	Amazon Aurora	Azure Cosmos DB for PostgreSQL, Azure SQL Database	AlloyDB for PostgreSQL	Run transactional workloads 4x faster than standard PostgreSQL, and analytical queries up to 100x faster.
Database	RDBMS	Amazon Aurora	Azure SQL Database	Cloud Spanner	Manage relational data with massive scale, strong consistency worldwide, and up to 99.999% availability.
Database	RDBMS	Amazon Relational Database Service (RDS), Amazon Aurora	Azure Database for MySQL and Azure Database for PostgreSQL	Cloud SQL	Manage relational data for MySQL, PostgreSQL, and SQL Server for workloads under 64 TB.
Database	Relational	Amazon RDS for Oracle	Azure Oracle Database Enterprise Edition	Bare Metal Solution	Lift and shift Oracle workloads to Google Cloud.
Developer tools	Client libraries	AWS SDKs	Azure SDKs	Cloud SDK	Tools and libraries for interacting with Google Cloud products and services.
Developer tools	Cloud development IDE plugin	AWS Toolkit for IntelliJ	Azure Toolkit for IntelliJ	Cloud Code for IntelliJ	Write, debug, and deploy your cloud-based applications for IntelliJ, VS Code, or any browser.
Developer tools	Cloud development IDE plugin	AWS Toolkit for Visual Studio Code	Azure Tools for Visual Studio Code	Cloud Code for VS Code	Write, debug, and deploy your cloud-based applications for IntelliJ, VS Code, or any browser.
Developer tools	Cloud-based IDE	AWS CloudShell	Azure Cloud Shell	Cloud Shell	Manage your infrastructure and develop your applications from any browser.
Developer tools	Command-line interface (CLI)	AWS CLI	Azure CLI	Cloud SDK	Tools and libraries for interacting with Google Cloud products and services.
Developer tools	Error handling			Error Reporting	Real-time exception monitoring and alerting for your applications.
Developer tools	Git Repositories	AWS Code Commit	Azure Repos	Cloud Source Repositories	Access fully featured, private Git repositories hosted on Google Cloud.
Developer tools	Job scheduling	Amazon EventBridge	Azure Scheduler	Cloud Scheduler	Fully managed cron job service.
Developer tools	No-code	AppSheet, Amazon Honeycode	Microsoft Power Platform	AppSheet	Enable anyone to build business applications and automated workflows, without coding.
Developer tools	Parallel task execution	Amazon Simple Queue Service (SQS), Amazon Simple Notification Service (SNS)	Azure Service Bus, Azure Storage Queues	Cloud Tasks	Control and observe asynchronous service requests between independent applications using this zonal, execution-control service.
Developer tools	PowerShell	AWS Tools for PowerShell	Azure Tools for PowerShell	Cloud Tools for PowerShell	Full cloud control from Windows PowerShell.
Enterprise	Abuse prevention	AWS WAF CAPTCHA, AWS Fraud	Microsoft Dynamics Fraud	reCAPTCHA Enterprise	Help protect your website from fraudulent activity, spam, and abuse without creating friction.
Enterprise	Marketplace	AWS Marketplace	Azure Marketplace	Marketplace	Scale procurement for your enterprise via online discovery, purchasing, and fulfillment of enterprise-grade cloud solutions.
Enterprise	ML workflows	Tensorflow on AWS	Azure DataBricks	Tensorflow Enterprise	Scale resources across CPUs, GPUs, and record-setting Cloud TPUs.
Enterprise	Solutions catalog	AWS Service Catalog	Azure Custom Images, Azure API Management	Private Catalog	Control internal enterprise solutions and make them easily discoverable.
Government services	Regulated services	AWS GovCloud	Azure Government	Assured Workloads	Run more secure and compliant workloads on Google Cloud.
Internet of things (IoT)	IoT platform	AWS IoT Core	Azure IoT Hub	Cloud IoT	Easily and securely connect, manage, and ingest data from globally dispersed devices with this fully managed service.
Management tools	API management	Amazon API Gateway	Azure API Management	API Gateway	Develop, deploy, secure, and manage APIs with a fully managed gateway.
Management tools	API management	Amazon API Gateway	Azure API Management	Apigee API Management	Design, secure, analyze, and scale APIs anywhere with visibility and control.
Management tools	Cost management	AWS Cost Explorer, AWS Budgets	Azure Cost Management	Cost Management	Tools for monitoring, controlling, and optimizing your Google Cloud costs.
Management tools	Deployment	AWS CloudFormation, AWS Serverless Application Model (SAM), AWS Cloud Development Kit (CDK)	Azure Deployment Manager	Cloud Deployment Manager	Create and manage cloud resources with simple templates.
Management tools	Monetization	Amazon Publisher Services, Mobile Ads	Azure API Management	Apigee API Management	Easy-to-use and flexible way to monetize your APIs so that you can generate revenue whenever your APIs are used.
Media	AI	Amazon Rekognition Video	Azure Video Analyzer for Media	Video AI	Enable powerful content discovery and engaging video experiences.
Media	Encoding and streaming	AWS MediaLive	Azure Media Services	Livestream API	Encode and transform live video content for use across a variety of user devices.
Media	Encoding and streaming	AWS Media Convert	Azure Media Services	Transcoder API	Convert video files and package them for optimized delivery to web, mobile, and connected TVs.
Media	Monetization	AWS MediaTailor	Azure Media Services	Video Stitcher API	Dynamically insert content and ads for targeted personalization of video-on-demand (VOD) and live content.
Migration	Container migration	AWS App2Container	Azure Migrate	Migrate to Containers	Intelligently extract, migrate, and modernize applications to run natively on containers in GKE and Anthos clusters.
Migration	Server migration	AWS Server Migration Service	Azure Migrate	Migrate to Virtual Machines	Migrate VM instances to Google Cloud from AWS, Azure, or VMWare VSphere.
Migration	SQL database migration	AWS Database Migration Service	Azure Database Migration Service	Database Migration Service	Migrate databases to Cloud SQL from on-premises, Compute Engine, and other clouds.
Migration	Storage migration	AWS Storage Gateway, AWS DataSync	Azure Migrate	Storage Transfer Service	Complete large-scale online data transfers from online and on-premises sources to Cloud Storage.
Migration	Storage migration	AWS Snowcone, AWS Snowball, AWS Snowmobile	Azure Data Box	Transfer Appliance	Securely migrate large volumes of data (from hundreds of terabytes up to one petabyte) to Google Cloud without disrupting business operations.
Networking	CDN	Amazon CloudFront	Azure Content Delivery Network, Azure Front Door	Cloud CDN	Serve web and video content globally, efficiently, and reliably.
Networking	CDN	Amazon CloudFront	Azure Content Delivery Network	Media CDN	Deliver exceptional media experiences through Google's planet-scale cache network.
Networking	Domains and DNS	Amazon Route 53	Azure DNS	Cloud DNS	Publish your zones and records in DNS without the burden of managing your own DNS servers and software.
Networking	Domains and DNS	Amazon Route 53		Cloud Domains	Register and configure a domain in Google Cloud.
Networking	Firewall	AWS WAF, AWS Shield	Azure Web Application Firewall (WAF), Azure Front Door	Google Cloud Armor	Help protect your applications and websites against denial of service and web attacks.
Networking	Firewall	AWS Network Firewall, AWS Security Groups, AWS Access Control List	Azure Firewall	Cloud Firewall	Protect your network with firewalls that are fully embedded in the cloud networking fabric, highly scalable, and granular.
Networking	Load balancing	Elastic Load Balancing	Azure Load Balancer	Cloud Load Balancing	Efficiently distribute network traffic across Compute Engine VMs.
Networking	Network connectivity	AWS Direct Connect	Azure ExpressRoute	Cloud Interconnect	Extend your on-premises network to Google's network through a highly available, low-latency connection. You can use Dedicated Interconnect to connect directly to Google or use Partner Interconnect to connect to Google through a supported service provider.
Networking	Network connectivity	AWS Virtual Private Network (VPN)	Azure Virtual Private Network (VPN)	Cloud VPN	Connect your peer network to your Virtual Private Cloud (VPC) network through an IPsec VPN connection.

Networking	Network connectivity	Amazon Cloud WAN, AWS Transit Gateway	Azure Virtual WAN	Network Connectivity Center	Reimagine how you deploy, manage, and scale your networks on Google Cloud and beyond.
Networking	Network connectivity	AWS PrivateLink	Azure Private Link	Private Service Connect	Create a private and secure connection from your VPCs to Google, third parties, or your own services.
Networking	Network monitoring	AWS Network Manager, Amazon CloudWatch	Azure Network Watcher	Network Intelligence Center	Centralize your network monitoring functions to verify network configurations, optimize network performance, increase network security, and reduce troubleshooting time.
Networking	Premium networking	Amazon Global Accelerator		Network Service Tiers	Optimize your network for performance or cost.
Networking	Service mesh	AWS App Mesh	Open Service Mesh	Traffic Director	Easily deploy global load balancing across clusters and VM instances in multiple regions, offload health checking from service proxies, and configure sophisticated traffic control policies.
Networking	Services discovery (DNS)	AWS Cloud Map	Hashicorp Consul Service on Azure	Service Directory	Publish, discover, and connect services from a single directory.
Networking	Virtual networks	NAT Gateways for Amazon VPC	Azure Virtual Network NAT	Cloud NAT	Send and receive packets using Google Cloud private GKE clusters or Compute Engine VM instances with no external IP address.
Networking	Virtual networks	Amazon Virtual Private Cloud (VPC)	Azure Virtual Network	Virtual Private Cloud	Provide managed networking functionality for your cloud-based services running on Compute Engine VM instances, Google Kubernetes Engine, App Engine flexible environment instances, and other Google Cloud products built on Compute Engine VMs.
Operations	Audit logging	AWS CloudTrail	Azure Audit Logs	Cloud Audit Logs	Log all user activity on Google Cloud.
Operations	Debugging	AWS X-Ray	Azure Monitor Application Insights Snapshot Debugger	Cloud Debugger	Investigate your code's behavior in production.
Operations	Logging	Amazon CloudWatch Logs	Azure Monitor Logs	Cloud Logging	Manage logging and analysis in real time at scale.
Operations	Monitoring	Amazon CloudWatch	Azure Monitor	Cloud Monitoring	Monitor the performance, availability, and health of your applications and infrastructure.
Operations	Performance tracing	AWS X-Ray	Azure Monitor Application Insights Distributed Tracing	Cloud Trace	Find performance bottlenecks in production.
Operations	Profiling	Amazon CodeGuru Profiler	Azure Monitor Application Insights Profiler	Cloud Profiler	Understand resource consumption in your code and see the ways the code is actually called.
Security & identity	Certificate management	AWS Certificate Manager	Azure Active Directory Certificate Authority	Certificate Authority Service	Simplify the deployment and management of private certificate authorities without managing infrastructure.
Security & identity	CIAM	Amazon Cognito	Azure Active Directory B2C	Identity Platform	Add Google-grade identity and access management to your apps.
Security & identity	Cloud provider access mgmt	AWS CloudTrail		Access Transparency and Access Approval	Help expand visibility and control over your cloud provider with admin access logs and approval controls.
Security & identity	Container security	Amazon Elastic Container Registry (ECR)	Azure Container Registry	Artifact Registry	Deploy only trusted containers on GKE.
Security & identity	Container security	Amazon ECR Image Scanning	Azure Defender for container registries	Container Analysis	Perform vulnerability scans on container images in Artifact Registry and Container Registry, and monitor vulnerability information to keep it up to date.
Security & identity	Data loss prevention (DLP)	Amazon Macie	Azure Information Protection	Cloud Data Loss Prevention	Discover, classify, and help protect your most sensitive cloud data.
Security & identity	Encryption	AWS Nitro Enclaves	Azure Confidential Computing	Confidential Computing	Encrypt data in-use with Confidential Computing and Confidential GKE Nodes.
Security & identity	Exfiltration prevention	Amazon VPC, AWS IAM, AWS PrivateLink		VPC Service Controls	Isolate resources of multi-tenant Google Cloud services to help mitigate data exfiltration risks.
Security & identity	Hardware security module (HSM)	AWS CloudHSM	Azure Dedicated HSM	Cloud HSM	Host encryption keys and perform cryptographic operations in a cluster of FIPS 140-2 Level 3 certified hardware security modules (HSMs).
Security & identity	IAM	AWS IAM Identity Center	Azure Active Directory	Cloud Identity	A unified identity, access, app, and endpoint management (IAM/EMM) platform.
Security & identity	IAM	Amazon Identity and Access Management	Azure Identity Management	Identity and Access Management	Provide fine-grained access control and visibility for centrally managing resources.
Security & identity	IAM	AWS Systems Manager	Azure Bastion, Azure AD Application Proxy	Identity-Aware Proxy (IAP)	Use identity and context to guard access to your applications and VMs.
Security & identity	IAM	AWS Managed Microsoft AD	Azure Active Directory Domain Services	Managed Service for Microsoft Active Directory	Use a highly available, hardened service running actual Microsoft Active Directory (AD).
Security & identity	Resource access management	AWS Organizations policies	Azure Policy	Organization Policy Service	Configure restrictions on how resources can be used.
Security & identity	Resource monitoring	AWS Config	Azure Security Control	Cloud Asset Inventory	View, monitor, and analyze all your Google Cloud and Anthos assets across projects and services using this metadata inventory service.
Security & identity	Resource monitoring	AWS Resource Access Manager, AWS Organizations, AWS Control Tower	Azure Resource Manager	Resource Manager	Hierarchically manage resources by project, folder, and organization.
Security & identity	SIEM	AWS Security Hub, Amazon CloudWatch	Azure Sentinel	Chronicle	Normalizes, indexes, correlates, and analyzes security and network data to provide instant analysis and context on risky activity.
Security & identity	Secret management	AWS Secrets Manager, AWS Systems Manager Parameter Store	Azure Key Vault	Secret Manager	Store API keys, passwords, certificates, and other sensitive data.
Security & identity	Security administration	AWS Key Management Service (KMS)	Azure Key Vault	Cloud Key Management Service	Manage encryption keys on Google Cloud.
Security & identity	Security and risk management	Amazon Guard Duty, AWS Security Hub, AWS Audit Manager	Azure Security Center, Azure Defender	Security Command Center	Security and risk management platform for Google Cloud.
Security & identity	Zero trust		Azure AD Conditional Access	BeyondCorp Enterprise	Enable secure access to critical applications and services, with integrated threat and data protection.
Serverless	Build	AWS Simple Storage Service (S3)	Azure Blob Storage	Cloud Storage for Firebase	Store and serve user-generated content from Firebase apps, such as photos or videos, including bandwidth-friendly transactions and automated ML, synced automatically in real time.
Serverless	Build	Amazon Cognito	Azure Active Directory (AD)	Firebase Auth	Sign in users to your Firebase app, either by using Firebase UI as a complete drop-in authentication solution, or by using the Firebase Authentication SDK to manually integrate one or several sign-in methods into your app.
Serverless	Build	AWS Amplify Hosting	GitHub Pages, Static Web Apps	Firebase Hosting	Provides fast and secure hosting for your Firebase web app, static and dynamic content, and microservices, including a generous free tier.
Serverless	Build	Amazon DynamoDB, AWS AppSync	Azure Cosmos DB	Firebase Realtime Database	Store and sync data from your Firebase application with our NoSQL cloud database. Data is synced across all clients in real time, and remains available when your application goes offline.
Serverless	Containers w/o infrastructure	AWS App Runner, AWS Fargate, AWS Lambda	Azure Container Apps, Azure Container Instances	Cloud Run	Develop and deploy highly scalable containerized applications on a fully managed serverless platform.
Serverless	Engage	Amazon Pinpoint	Azure Playfab	Firebase A/B Testing	Deploy A/B experiments to test how a change to your application's UI, features, or engagement campaigns affects key metrics (like revenue) before you implement the change widely.
Serverless	Engage	Amazon Device Messaging (ADM), Amazon Simple Notification Service (SNS)	Azure Notification Hubs	Firebase Cloud Messaging	Send and receive notifications across platforms with this reliable and battery-efficient connection between your server and devices, including iOS, Android, and the web.
Serverless	Engage			Firebase Dynamic Links	Provide users with deep-link smart URLs that bypass the application installation process, allowing you to send first-time or returning users to any location within your iOS or Android app.
Serverless	Engage	Amazon Device Messaging (ADM), Amazon Simple Notification Service (SNS)	Azure Notification Hubs	Firebase In-App Messaging	Engage active users of your Firebase application by sending them targeted, contextual messages to complete key actions, such as beating a game level, buying an item, or subscribing to content.
Serverless	Engage		Azure App Configuration	Firebase Remote Config	Control and optimize your app on the fly.
Serverless	Engage			Google Analytics for Firebase	Make informed decisions regarding application marketing and performance optimizations by understanding user behavior using the Firebase SDK and integration with the Google ecosystem.
Serverless	Event handling	AWS EventBridge	Azure Event Grid	Eventarc	Asynchronously deliver events from Google services, SaaS, and your own apps using loosely coupled services that react to state changes.

Serverless	FaaS	AWS Lambda	Azure Functions Serverless Compute	Cloud Functions	Run your code with zero server management with this scalable, pay-as-you-go functions-as-a-service (FaaS) offering.
Serverless	Mobile FaaS	AWS Lambda	Azure Functions Serverless Compute	Firebase Cloud Functions	Run your mobile backend code without managing servers.
Serverless	Monetization	Amazon Publisher Services, Mobile Ads	Azure API Management	AdMob and Firebase	Google AdMob is an easy way to monetize mobile apps with targeted, in-app advertising.
Serverless	Release & monitor		Azure App Center	Firebase App Distribution	Distribute your Firebase apps to trusted testers quickly and easily.
Serverless	Release & monitor		Azure App Center	Firebase Crashlytics	Get real-time, actionable insight into Firebase application issues with this native crash reporting solution for iOS, Android, and Unity, including streaming data export.
Serverless	Release & monitor			Firebase Performance Monitoring	Gain insight into your app's performance issues.
Serverless	Release & monitor	AWS Device Farm	Azure App Center	Firebase Test Lab	Test your Firebase application on devices hosted in a Google data center.
Serverless	Workflow orchestration	AWS Step Functions	Azure Logic Apps	Workflows	Orchestrate and automate Google Cloud and HTTP-based API services with serverless workflows.
Storage	Block storage	Amazon Elastic Block Store (EBS)	Azure Disk Storage	Persistent Disk	Store data from VM instances running in Compute Engine or GKE, Google Cloud's state-of-the-art block storage offering.
Storage	File storage	Amazon Elastic File System (EFS)	Azure Disk Storage, Azure Files	Filestore	Provide fully managed NFS file servers on Google Cloud for applications running on Compute Engine VMs (VMs) instances or GKE clusters.
Storage	Infrequently accessed object store	Amazon S3 Glacier	Azure Archive Storage	Cloud Storage Archive	Store infrequently accessed data using Google Cloud's ultra low-cost, highly durable, highly available archival storage.
Storage	Object storage	AWS Simple Storage Service (S3)	Azure Blob Storage	Cloud Storage	Store any amount of data and retrieve it as often as you'd like, using Google Cloud's object storage offering.

Network Topics (Net+ CCNA CCNP Etc)

01. OSI and DARPA Network Models/ Network basics I Transport protocols
- Network protocols
- Data-Link protocols
- Ping and Traceroute(and their relatives)
- Round trip ping trip explained PC1-〉router-〉PC2 and back

02. Subnetting and binary overview- Addressing crash course
- IPv4 Fast subnetting technique, VLSM, binary practice
- IPv4 quick reference sheet and worksheet (for dry erase marker)
- IPv6 overview and SLAAC/DHCPv6, Cisco routing in IPv6
- IPv6 subnetting large ISP blocks down /32 to /60 (intro only)

03. Cisco-based switching topics
- Cisco switchports: access and trunks, port security IVR/SVI VLANs, VTP, RoaS
- Spanning Tree, EtherChannel (PAgP and LACP)

04. Cisco Misc. topics
Load Balancing with VRRP, HSRP, GLBP
Cisco-based DHCP, NAT (non-ASA), Access control lists

05. Routing on Cisco Devices
- OSPF and LSAs, Stub areas, NSSA, etc.
- EIGRP, k-values and metric, RD/FD, SR/FS in topology IPv4 Interior routing- static, RIP, OSPF, EIGRP examples IPv6 version (show changing routing to another type)

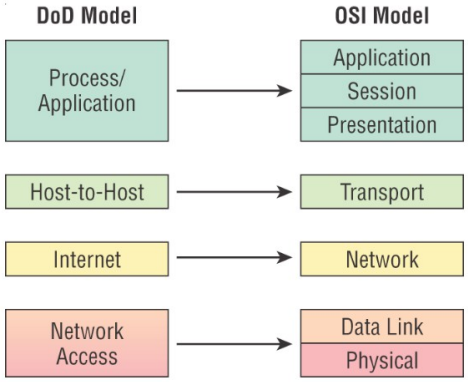
06. BGP

07. Short subjects and other stuff
- BGP route leaks and BGP hijacking intro (more of a 1 pg "what is...")
- Setting up GRE with IPSEC for a Typical VPN
- DMVPN - Dynamic Multipoint VPN (multipoint GRE, IPSec, NHRP)
- TCP Ports - Interactions and Scanning (nmap concepts, port responses)
- SSH Tunnels; port forwarding, jump hosts, etc (Linux)
- Socat to Netcat Commands Task Quick Reference
- Cables and connectors, line speeds (sortof Network+ junk)

The OSI and DARPA TCP/IP Networking Models

DEC/IBM (SNA) before 1980s, into 90s when TCP/IP prevailed and overtook the OSI model
The TCP/IP model is not a top-down comprehensive design reference - the purpose is illustrating the logical groups and scopes of functions needed. In general, direct or strict comparisons of the OSI and TCP/IP models should be avoided, because the layering in TCP/IP is not a principal design criterion. It just makes it easier to understand the interaction of the overlying technologies.

OSI Networking Model (DARPA's TCP/IP stack remains, but OSI won the terminology war)

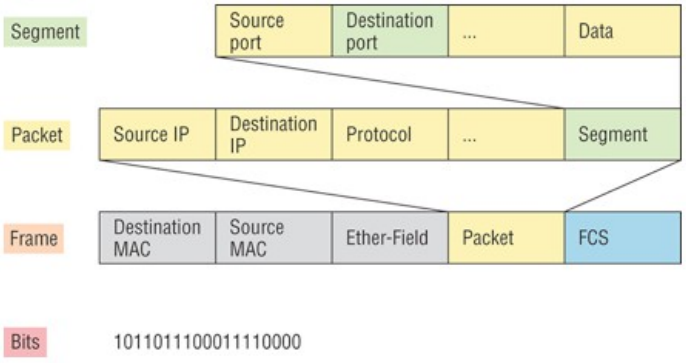


The DoD's Application layer is split into 3 layers in the OSI.
Transport is often still called Host-to-Host in the DoD model.
and it calls the network layer the Internet layer
Finally, the Data Link and Physical layers are unified in the DoD model, and called the Network Access layer, aka Network Interface, or Link layer.

The OSI has remained- as the network stack is always referred to by it's 7 layers. Additionally, layer 2 refers to data link, and layer 4 for transport. The DoD model retains layers 5, 6, and 7 as one application layer.

The TCP/IP and OSI stack are *reference models* of a *layered architecture.*

Same-Layer Interactions *are between two computers at the same layer (like HTTP to HTTP)*
Adjacent-Layer *- A layer passes info to neighboring layer (up or down) on the same computer*
A layer's data is referred as a Protocol Data Unit. A layer 2 frame is a L2PDU, a network IP packet is a L3PDU, and a UDP or TCP segment is a L4PDU.
An example of outgoing data, HTTP sends to TCP to put into a segment, it then sends to IP to pack up in a packet, and data-link frames it to go out the wire as bits. Likewise, when frame is pulled out of the incoming bits, it's frame is unpacked, the IP packet passed up the stack to be opened it up, revealing a segment to give to the adjacent transport layer.



Layer 7 - Application Layer - Telnet, HTTP, FTP, SMTP, POP3, VoIP, SNMP. Provides an interface between the communications software and any applications that need to communicate outside the computer on which the application resides. It also defines processes for user authentication

Layer 6 - Presentation Layer - ASCII, EBCDIC; de/compression, de/encryption. "Defines and negotiates data formats"; "how standard data should be formatted." Includes JPEG, TIFF, GIF, PICT, MPEG, MIDI, etc.

Layer 5 - Session Layer - PPTP, L2TP, NetBIOS, NFS, PAP, RPC, SOCKS. From a client to a server; three different modes: simplex, half-duplex, and full-duplex. How to start, control, and end conversations (sessions). This includes the control and management of multiple bidirectional messages so that the application can be notified if only some of a series of messages are completed. Sockets associate traffic with the transport layer's ports

Layer 4 - The Transport Layer - TCP, UDP, and SCTP - Determines how to handle data sent/ delivered over the network layer. Depending on the protocol, may provide for retransmission, i.e., error recovery, and may use flow control to prevent unnecessary congestion by attempting to send data at a rate that the network can accommodate. Multiplexing of incoming data for different flows to applications on the same host is also performed (ports and sessions). Reordering of the incoming data stream when packets arrive out of order is included.

The Network Layer (Layer 3) defines end-to-end delivery of packets and defines logical addressing to accomplish this. It also defines how routing works and how routes are learned; and how to fragment a packet into smaller packets to accommodate media with smaller maximum transmission unit sizes. Examples include. IP, IPX, and ICMP. Both IP and IPX define logical addressing, routing, the learning of routing information, and end-to-end delivery rules.

The Data-Link Layer (Layer 2) is concerned with getting data across one particular link or medium. The data link protocols define delivery across an individual link. These protocols are necessarily concerned with the type of media in use. Examples include. IEEE 802.3/802.2, HDLC, Frame Relay, PPP, FDDI, ATM. The lower level works with standards governing the physical transmission medium (Layer 1), including use of connector pins, electrical properties like voltage, etc.

The Physical Layer (Layer 1)
This is the medium: copper wire, fiber, or over the airwaves. This will be in an appendix of sorts to talk protocols sooner here.

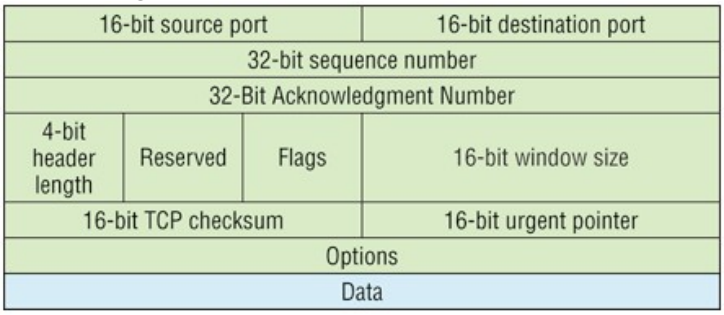
Layer 4 - Transport Layer (aka Host-to-Host) - end-to-end data transport services between hosts. TCP is a protocol with sequencing, acknowledgements- communications between the participating hosts. UDP is connectionless and there is no verification either side gets the other host's data.

TCP implements *reliable networking* - requiring acknowledgments, sequencing, and flow control will all be used. The 3-way handshake virtual circuit setup; removes a lot of programming work, but for real-time video and VoIP, UDP is often better because using it results in less overhead, but is less reliable)
Connection establishment and termination: SYN-->ACK-->SYN-ACK-->SYN and later, FIN-->FINACK?-->FIN!

The types of flow control are buffering, windowing, and congestion avoidance.
Sliding Window - TCP allows the receiving device to dictate the amount of data the sender can send before receiving an acknowledgment, the mechanism to grant future windows is typically just a number that grows (slides) upward slowly after each acknowledgment.

"Positive acknowledgment with retransmission" - The sender documents each segment measured in bytes, then sends and waits for acknowledgment before sending the next segment. The transmitting machine starts a timer and will retransmit if it expires before it gets an acknowledgment back from the receiving end.

The TCP Segment's Format and Header Fields



Source port: port number of the application on the host sending the data
Destination port: This is the port number of the application requested on the destination host.
Sequence number: Used to put the data back in the correct order (or retransmit missing or damaged data) during sequencing process.
Acknowledgment number: The value is the TCP octet that is expected next.
Header length aka Offset: Specifies the size of the TCP header in 32-bit words (a "word" is 4 bytes). The minimum is 5 words and the maximum is 15 words (so minimum is 20 bytes and maximum of 60 bytes, allowing for up to 40 bytes of options in the header)
Reserved: Always set to zero.
Code bits/flags: Control bits for functions to manage a session. (SYN, ACK, FIN, URG, PSH, etc)
Window: The window size the sender is willing to accept, in octets.
Checksum: This checksum results in combining data such as addresses involved, segment size and types with a formula that can be checked on the receiving end (it is recalculated in a misleadingly- labeled 'pseudoheader' to do it) - it isn't a CRC which checks all data, but it is sufficient.
Urgent: A valid field only if the Urgent pointer in the code bits is set. If so, this value indicates the offset from the current sequence number, in octets, where the segment of non-urgent data begins.
Options: May be 0, meaning that no options have to be present, or a multiple of 32 bits. However, if any options are used that do not cause the option field to total a multiple of 32 bits, padding of 0s must be used to make sure the data begins on a 32-bit boundary (aka "words").
Data: Handed down to the TCP PDU by the upper-layer headers

TCP - Transport Control Protocol		Code:	%011000
Source Port:	5973		Ack is valid, Push Request
Destination Port:	23	Window:	61320
Sequence Number:	1456389907	Checksum:	0x61a6
Ack Number:	1242056456	Urgent Pointer:	0
Offset:	5	No TCP Options	
Reserved:	%000000	TCP Data Area:	vL.5.+.5.+.5.+.5 76 4c

The TCP handshake and termination - quick overview:
The Initial Sequence and Response Numbers (ISN and IRN) are numbers exchanged in TCP segments during computer network communication between a client and a server. These are central in a SYN flood defense known as SYN cookies.

Here is a sample session:
1. The client sends a SYN with an ISN of 1664882716.
2. Server replies with a SYNACK with an IRN of 829007135 and an ACK value of 1664822717. The ACK reports the next the server expects from the client in this sequence (1664822717)
3. Client sends an ACK back 829007136 to increment the server's IRN in the SYNACK, which also reports it expects from the server in this sequence (829007136) It sends this with a sequence number of 1664882717, just like the server expects.

1. Client:	SYN	seq 1664882716	
2. Server:	SYNACK	seq 829007135	ack 1664882717
3. Client:	ACK	seq 1664882717	ack 829007136
4. Server:	ACK	seq 829007136	ack 1664882718
5. Client:	ACK	seq 1664882718	ack 829007137
Then later, server terminates the connection:			
1. Client:	ACK	seq 1664882733	ack 829008199
2. Server:	FIN-ACK	seq 829008199	ack 1664882734
3. Client:	ACK!	seq 1664882734	ack 829008200
4. Client:	FIN-ACK?	seq 1664882734	ack 829008200 (yes, two different responses)
5. Server:	ACK	seq 1664882735	

User Datagram Protocol (**UDP**)
There are times that it's wise for developers to opt for UDP rather than TCP, one of them being when reliability is already taken care of at the upper layers. If the segments arrive out of order, which is commonplace in IP networks, they'll simply be passed up to the next layer in whatever order they were received, without sequencing or other features TCP provides. A UDP header is only 8 bytes (compared to TCP's 20 bytes) - has 4 fields, each of which are 2 bytes. Source and checksum are optional in IPv4 (only source is optional in IPv6)

UDP - User Datagram Protocol

Source Port:	1085
Destination Port:	5136
Length:	41
Checksum:	0x7a3c
UDP Data Area:	..Z.....00 01 5a 96 00 01 00 00 00 00 00 11 0000 00

Port numbers for communicating with upper layers
Ordered data transfer and data segmentation. Ports with numbers 0-1023 are called well-known ports; ports with numbers 1024-49151 are called egistered ports, and ports with numbers 49152-65535 are called dynamic, private or ephemeral ports.*The source port number is arbitrary- usually ephemeral. Destination port is specific to process/application (workstation sends SSH connection out it's port using an ephemeral port #, to port 22 at the listening SSH process on the other end)* . The virtual circuit is defined by the source and destination port number plus the source and destination IP address and called a socket.

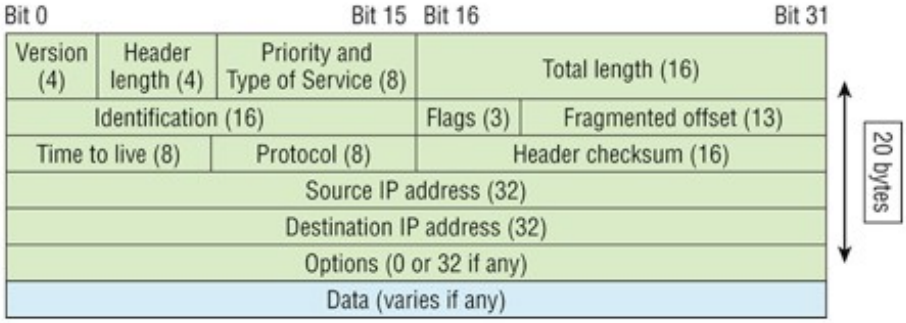
20-21 FTP	110 POP3	465 SMTP over SSL	587 SMTP
22 SSH/SCP	123 NTP	500 ISAKMP	636 LDAP over SSL
23 Telnet	135 Microsoft RPC	512 rexec	646 LDP (MPLS)
25 SMTP	137-139 NetBIOS	513 rlogin	860 iSCSI
49 TACACS	143 IMAP4	514 syslog	902 VMware Server
53 DNS (uses both)	161-162 SNMP	515 LPD/LPR	989-990 FTP over SSL
67-68 DHCP/BOOTP	389 LDAP	520 RIP	993 IMAP4 over SSL
69 TFTP	443 HTTP over SSL	521 RIPng (IPv6)	995 POP3 over SSL
80 HTTP	445 Microsoft DS	546-547 DHCPv6	1025 Microsoft RPC
88 Kerberos	464 Kerberos		

IANA assigns a port number for both TCP and UDP even if the service uses only one. Lists like the one above are ok, but often not very accurate (especially with VoIP). In Wireshark, it has a builtin tool for that.
Authoritative list here:
https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Layer 3 - Network/ Internet Layer - IPv4, IPv6, IPX, IPSec, ICMP.

Has three main features: logical addressing, routing (forwarding), and path determination. Routing defines how devices (typically routers) forward packets to their final destination. Logical addressing defines how each device can have an address that can be used by the routing process. Path determination refers to the work done by routing protocols to learn all possible routes and determine the best way to move data. The network layer handles two types: data packets and route update packets. *Ultimately, Layer 3 doesn't care about where a particular host is located- only about where networks are located and the best way to reach them.*

IPv4 (below) has more on it in the following pages, in the form of a subnetting chart and worksheet on subnetting.



Header length: HLEN in 32-bit words.
Priority/TOS/Precedence: how to handle the datagram. The first 3 bits are the priority bits, called the differentiated services bits.
Total length: including header and data.
Identification: Unique IP-packet value used to differentiate fragmented packets from different datagrams.
Fragmentation Flags: Specifies whether fragmentation should occur.
Fragment offset: Provides fragmentation and reassembly if the packet is too large to put in a frame. It accommodates different MTUs on the Internet.
Time To Live: TTL set when packet is generated. If it doesn't get to where it's supposed to go before the TTL expires, it's dropped.
Protocol: Port of ecapsulated payload protocol; for example, TCP is port 6 or UDP is port 17
Header checksum: CRC on header only.
Source IP address: 32-bit IP address of sending station.
Destination IP address: 32-bit IP address of the station this packet is destined for.
Options: Used for network testing, debugging, security, and more.
Data: The upper-layer data.

IP Header - Internet Protocol Datagram

Version:	4
Header Length:	5
Precedence:	0
Type of Service:	%000
Unused:	%00
Total Length:	187
Identifier:	22486

Fragmentation Flags:	%010 Do Not Fragment
Fragment Offset:	0
Time To Live:	60
IP Type:	0x06 TCP
Header Checksum:	0xd031
Source IP Address:	10.7.1.30
Dest. IP Address:	10.7.1.10
No Internet Datagram Options	

Protocols found in the protocol field of an IP header
ICMP 1, TCP 6, UDP 17, IPv6 41, GRE 47, EIGRP 88, OSPF 89, L2TP 115
For more: http://www.iana.org/assignments/protocol-numbers

- IP packets in WiFi have a noticeably larger size, which is mainly to hold not only the source and destination IP addresses of the workstation, but also the IP addresses of the access points on either end. We may get into that more in the section on WiFi.

- IPSec is it's own big topic which includes it's mechanism and tunneling uses. This also will have it's own dedicated area when we talk about VPN.

- IPv6 is saved toward the end of the routing and switching material, since it has a lot to talk about those things and the differences in IPv6 v.s. IPv4 (not much but it keeps things simpler to explain everything in IPv4 terms first instead of complicating it for people that might be new to all this).

ICMP
As it's name implies, this is a simple general messaging protocol, mostly about traffic stuff.

IANA maintains lists of ICMPv4 and ICMPv6 message types:
http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml
http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml

From a network scanning perspective, the following types of ICMP messages are useful:
Type 8 (echo request) aka ping packets. Perform ping sweeping and identify accessible hosts. Type 0 is the reply.
Type 13 (timestamp request) get system time from host; in decimal, number of milliseconds since midnight GMT.

Common ICMP type 3 message (destination unreachable) codes
Firewalls and routers generate type 3 ICMP responses, providing insight into network configuration.
0 - Network unreachable
1 - Host unreachable
2 - Protocol unreachable
3 - Port unreachable
6 - Destination network unknown
7 - Destination host unknown
9 - Communication administratively prohibited (network)
10 - Communication administratively prohibited (host)
13 - Communication administratively prohibited (general)
Cisco exams used to like mentioning "buffer full/source quench (deprecated Type 3 msg 4)- router's buffer for incoming datagrams is full.

There is more on ICMP in the section about the tools PING and traceroute, since those also discuss TCP.

ICMP error messages include a copy of the IPv4 header, plus at least the first 8 bytes of data from the error-triggering frame The length of ICMP error messages should not exceed 576 bytes. It is also intended to hold the type field indicating the upper layers contained within.

The ICMP data field can be exploited, as in the "Ping of death", large or fragmented ICMP packets are used for denial-of-service attacks. ICMP data can also be used to create covert tunnels for communication. There are more specific uses- encapsulation of IP packets into ICMP packets, leads to TCP session inclusion which opens the door for creative applications - there is also a telnet-like shell that can be opened.

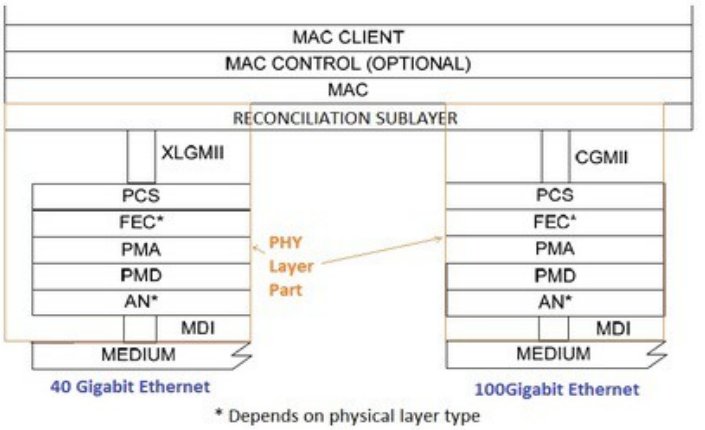
Layer 2 - Data Link Layer
ARP, MAC (IEEE 802.3 Ethernet, DSL, ISDN, FDDI), PPP, L2TP, HDLC
Defines the rules that determine when a device can send data over a particular medium. Data link protocols also define the format of a header and trailer that allows devices attached to the medium to successfully send and receive data.

This layer transfers data between adjacent network nodes in a wide area network (WAN) or between nodes on the same local area network (LAN) segment. Frames do not cross the boundaries of a local network (these frames are encapsulated/replaced by new frames native to the WAN environment). Internetwork routing and global addressing are higher-layer functions, allowing data-link protocols to focus on local delivery, addressing, and media arbitration (between parties contending for access to a medium, without concern for their ultimate destination). When devices attempt to use a medium simultaneously, frame collisions can occur. Data-link protocols specify how devices detect and recover from such collisions, and often provide mechanisms to reduce or prevent them.

Logical link control sublayer
The uppermost sublayer, LLC, multiplexes protocols running atop the data link layer, and optionally provides flow control, acknowledgment, and error notification. The LLC provides addressing and control of the logical links between local devices: which mechanisms are to be used for addressing stations over the transmission medium and for controlling the data exchanged between the originator and recipient machines. It basically provides services to the network layer and hides the rest of the data link layer. Is outlined by IEEE 802.2 - https://en.wikipedia.org/wiki/IEEE_802.2

Media access control sublayer
The sublayer defines methods to control access to the physical layer. Since many networks use a shared medium (such as a single network cable, or a series of cables that are electrically connected into a single virtual medium) it is necessary to have rules for managing the medium to avoid conflicts. Examples include ethernet's CSMA/CD method of media access control, while Token Ring uses token passing. It also determines where one frame of data ends and the next one starts- frame synchronization. There are four means of frame synchronization: time based, character counting, byte stuffing and bit stuffing.

MAC Services: physical addressing, channel-access control (CSMA/CD and CSMA/CA), LAN switching (packet switching), including MAC filtering, Spanning Tree Protocol and Shortest Path Bridging (SPB), data packet queuing or scheduling, Store-and-forward switching or cut-through switching, Quality of Service (QoS) control, encoding schemes. It is outlined by IEEE 802.3 - See https://en.wikipedia.org/wiki/IEEE_802.3



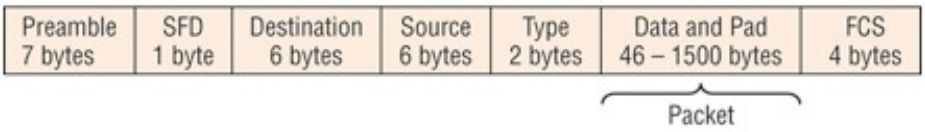
IEEE 802.3ba architecture

Network engineers have it easy
The MAC sublayer of Layer 2 almost has it's own huge protocol stack inside it, dealing with the gritty details of electrical signal properties of pins on connectors, light modulation for fiber optics, things we never have to think about with the physical medium.

Thankfully, most network engineers don't have to deal with this. This diagram is tame compared to some I have seen. This is the MAC layer substack for gigabit ethernet. We only have to think about the very top one layer, "MAC client".

Most networking classes never mention this- it is just "layer two 2nd half, then layer 1, da wire". I guess they think it will make their student's head explode or something.

Layer 2 Protocols - Ethernet Frames at the Data Link Layer

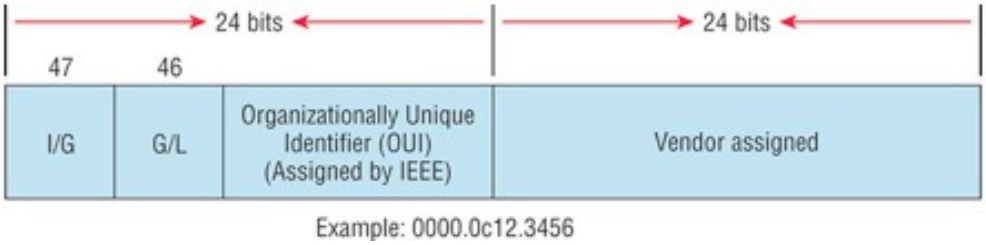


Preamble - 7 bytes - pattern of alternating 1 and 0 bits, allowing network devices to easily synchronize their receiver clocks. Since least significant bits are transmitted first, the one breaking the alternating pattern tends to be last.
Start Frame Delimiter (SFD): 1 byte - The SFD octet is designed to break the bit pattern of the preamble and signal the start of the actual frame. Arguably, the SFD could more easily be called part of the preamble, and often you will just see 8 bytes: preamble. Wireshark won't see this (L2 strips it off)
Destination Address (DA) - 6 bytes - The 48-bit value of the intended recipient. Can also be BCast or MCast.
Source Address (SA) - 6 bytes - Sender. BC and MC address formats are illegal within the SA field.
802.1Q tag (optional) - 4 bytes - Ethernet II only
Length or Type - 2 bytes - Old 802.3 uses a Length field, but Ethernet_II frame uses a Type field to ID the layer 3 PDU. 0x86dd for IPv6 data; 0x0800 for IPv4; 0x8100 for VLAN-tagged (802.1Q); 0x0806 for ARP; 0x8147/48 MPLS; 0x88CC LLDP; 0x8163/64 PPPoE; 0x8906 FCoE; 0x8100 Jumbo;
Data - 46-1500 bytes - This is a packet sent down to the Data Link layer from the Network layer. The size can vary from 46 to 1,500 bytes. (44 if 802.1Q tag is not present, and greater than 1500 if jumbo). Padding to meet the minimum 46 bytes is added if necessary.
Frame Check Sequence (FCS/CRC) - Algorithm is run when each frame is built based on the data in the frame. If CRCs don't match, the frame is discarded, assuming errors have occurred.

The original IEEE 802.3 defined the min/max Ethernet frame size as 64/1518 bytes (max later increased to 1522 bytes for VLAN tagging. MTU in an Ethernet LAN 1500 bytes by default.
The minimum size of an Ethernet frame that carries an ICMP packet is 74 bytes. (ping packet with no options will generate a 74 byte packet with a 60 byte IP Header, 8 byte ICMP header
Minimum 64 bytes (header + data + FCS) -- 64-(14+4)=46 bytes for data, padded if needed

Size of Ethernet frame - 24 Bytes - [this doesn't include preamble 8 bytes]
Size of IPv4 Header (without any options) - 20 bytes
Size of TCP Header (without any options) - 20 Bytes
So total size of empty TCP datagram - 24 + 20 + 20 = 64 bytes
Size of UDP header - 8 bytes
So total size of empty UDP datagram - 24 + 20 + 8 = 52 bytes
Ethernet HW usually filters preamble and FCS on incoming packets (so applications won't see them- Wireshark)

Data Link and Ethernet - Hex to Dec and MAC Addresses



Example: 0000.0c12.3456

48-bit (6-byte) MAC address written in a hexadecimal format (instead of binary for readability).
The organizationally unique identifier (OUI) is assigned by the IEEE to an organization. It's composed of 24 bits, or 3 bytes, and it in turn assigns a globally administered address

Individual/Group (I/G) bit. When it has a value of 0, we can assume that the address is the MAC address of a device and that it may well appear in the source portion of the MAC header. When it's a 1, we can assume that the address represents either a broadcast or multicast address in Ethernet."

Global/local bit, sometimes called the G/L bit or U/L bit, where U means universal. When set to 0, this bit represents a globally administered address, as assigned by the IEEE, but when it's a 1, it represents a locally governed and administered address. This becomes clearer when discussing IPv6 and MAC addresses.

Manufacturer-assigned code commonly starts with 24 0s for the first card made and continues to 16,777,216 (24 ones in binary). It is often incorporated into the serial number as well.
Hex to Binary conversion: A "nibble" is 4 bits and a byte is 8 bits (or an octet)
If we have a 1 placed in each spot of our nibble, we would then add up 8 + 4 + 2 + 1 to give us a maximum value of 15. Another example for our nibble values would be 1001, meaning that the 8 bit and the 1 bit are turned on, which equals a decimal value of 9. If we have a nibble binary value of 0110, then our decimal value would be 6, because the 4 and 2 bits are turned on. You take these two nibbles as binary and run them together as a byte: you get the real value 10010110, which is 128+16+2+4 = 150

Example: 0x6A- each hex character is one nibble and that two hex characters joined together make a byte. To figure out the binary value, put the hex characters into two nibbles and then join them together into a byte. 6 = 0110; A, which is 10 in hex = 1010; so the complete binary byte would be 01101010, and 64+32+8+2= 106 is the decimal value.

A binary number: 11001100. What's it in hex? Split it: 1100 = 12 and 1100 = 12, so therefore, it's representation is CC in hex, but the decimal conversion would be 128 + 64 + 8 + 4 = 204, the real value.
What about 10110101? The hex answer would be 0xB5, since 1011 converts to B and 0101 converts to 5 in hex value. The decimal equivalent is 128 + 32 + 16 + 4 + 1 = 181

Hex	**Bin**	**Dec**	5	0101	5	B	1011	11
0	0000	0	6	0110	6	C	1100	12
1	0001	1	7	0111	7	D	1101	13
2	0010	2	8	1000	8	E	1110	14
3	0011	3	9	1001	9	F	1111	15
4	0100	4	A	1010	10			

Layer 2 Protocols - Address Resolution Protocol (ARP)
ARP was defined by RFC 826 in 1982, and is used for mapping a IPv4 address to a physical MAC address. In IPv6 the functionality of ARP is provided by the Neighbor Discovery Protocol (NDP). When IP has a datagram to send, it must inform a Network Access protocol, such as Ethernet, of the destination's hardware address on the local network. If IP doesn't find the destination host's hardware address in the ARP cache, it uses ARP to find this information.

A continuing discussion about ARP is where Layer 2 and 3 begin and end, since it seems to almost straddle both. It is a good reminder that the OSI model is just that- a conceptual model for design and development of protocols. This is a good example (there aren't many) of where the real world is a bit less tangible and technological requirements move things beyond the roadmap given to us. In the end it is best to call it L2 and be done with it- it is most likely that the LLC invokes service calls to the MAC sublayer in order to report to the CAM table and layer 3 services

ARP - Address Resolution Protocol

Hardware	1 (Ethernet 10Mb)
Protocol:	0x0800 (IP)
Hardware Address Length:	6
Protocol Address Length:	4
Operation:	1 ARP Request
Sender Hardware Address:	00:A0:24:48:60:A5
Sender Internet Address:	172.16.10.3
Target Hardware Address:	00:00:00:00:00:00 (ignored)
Target Internet Address:	172.16.10.10

ARP related terms, (or, terms with ARP in the name)

Inverse ARP is primarily used in Frame Relay (DLCI) and ATM networks, which need the corresponding Layer 3 addresses before those virtual circuits can be used

An ARP probe is an ARP request constructed with an all-zero sender IP address. Before beginning to use an IPv4 address a host must test to see if the address is already in use, by broadcasting ARP probe packets

A *gratuitous ARP* announcement updates any cached entries in the ARP tables of other hosts that receive the packet. Many operating systems perform gratuitous ARP during startup. It helps in case a network card was recently changed and other hosts need to update ARP caches. Gratuitous ARP is also needed in teaming network cards, and can be used to defend link-local IP addresses in zeroconf.

ARP spoofing - ARP has no authentication, so replies can come from systems other than the one with the required address. ARP spoofing is where a system impersonates another system's address with the aim of intercepting data bound for that system, resulting in man-in-the-middle and DoS attacks. An ARP proxy is a legitimate system which answers the ARP request on behalf of another system for which it will forward traffic, such as for a dialup internet service.

ARP mediation is WAN resolution of Layer 2 addresses through a Virtual Private Wire Service (VPWS) when different resolution protocols are used circuits- e.g., Ethernet on one end and Frame Relay on the other. In IPv4, each Provider Edge (PE) device discovers the IP address of the locally attached Customer Edge (CE) device and distributes that IP address to the corresponding remote PE device. Then each PE device responds to local ARP requests using the IP address of the remote CE device and the hardware address of the local PE device.

In IPv6, ARP is replaced with NDP (Network Discovery Protocol) - each device discovers the IP address of both local and remote CE devices and then intercepts local Neighbor Discovery (ND) and Inverse Neighbor Discovery (IND) packets and forwards them to the remote PE device.

Layer 2 - Switching basic decision-making, other issues
IEEE 802.1D/w Spanning Tree puts each port in forward or blocking state. Ports in a blocking/ discarding state won't process any frames except STP messages - the switch physically receives the frame on blocked port, but ignores it.

That being said, for all ports in a forwarding state:
If destination address is:
- same as source? Ignore (filter).
- known? Forward to correct port
- unknown? LEARN: map the source MAC address to it's port number. Flood out all ports except entry

Address learning:
- When flooded out, source MAC is added to table, but the destination will NOT be learned.
For each received frame, examine the source MAC, note the interface.
- If not in table, add- *set inactivity timer to 0.*
- If it is in table, *reset the inactivity timer for the entry to 0.*
On new frame, if MAC-to-port is different than previously recorded, it will be updated.
- MAC table instability (info repeatedly updated erroneously from loops) is prevented by STP

Modes of Layer 2 forwarding:
Store-and-forward - receive all bits in the frame and check the FCS before forwarding.
Fragment-free - receiving the first 64 bytes first to weed out collision-damaged frames.
Cut-through - checks dest MAC, forwards frame ASAP -reduces latency but no FCS check (later versions check for QoS and/or ACLs)
If asked, the three switch Layer 2 functions: address learning, forward/filter, loop avoidance (STP)
Forward vs Filter simply refers to: if MAC table says for Port 1, not port 2, forward to port 1, filter from port 2.
Is looked at by some people as two decisions:
1. Forwarding decision: to send it to the right port (associated with that MAC address)
2. Filtering decision: to NOT send it out the other ports.

IP Routing basic decisionmaking:
Four-step process of how routers route (forward) packets
Layer 2 first- check FCS. If errors occurred, discard the frame.
Trash the old layer 2 header and trailer, leaving the IP packet, which is sent to Layer 3.
Compare to the routing table, and find the outgoing interface of the router (next-hop IP).
Encapsulate into new layer 2 header/trailer for the outgoing interface and forward the frame. (how does it know the destination MAC? The ARP table)

Goals of IP routing protocols
- dynamically learn and fill the routing table with a route to each subnet on the network.
- choose and place the best route in the routing table.
- notice when table's routes are no longer valid, remove them. Get new one from neighbor.
- work quickly - for fast convergence time and routing updates
- prevent routing loops.

Layer 2: Broadcast and Collision Domains - FF:FF:FF:FF:FF:FF
1. Each router port is a separate broadcast domain (also a separate collision domain)
2. Each port on a switch (or bridge) is a separate collision domain.
3. Everything connected to a hub is in one collision domain (switches break up collision domains).

Collision domains: sending and receiving data frames at same time is not supported (half-duplex).
If devices send at the same time the frames collide, a jamming signal (backoff) is sent, devices set timers to try again, which slows everything down. Rely on CSMA/CD to try to mediate collisions.

Routers move traffic between IP networks/subnets. Routers also separate broadcast domains.
If a router interconnects two networks, each network is its own broadcast domain
If a router has two ports in the same subnet, those are two different broadcast domains.
Because, routers do not forward layer 2 broadcast traffic

- At a home office: adding collision domains means increased bandwidth. With a hub (1 collision domain), only one PC can send at a time, for a theoretical maximum capacity of 100 Mbps for the entire LAN. Replace with a switch, and you get 100 Mbps per link, for a total of 1000 Mbps (1Gbps) and the ability to use full-duplex on each link, effectively doubling the capacity to 2000 Mbps (2Gbps)

Ping (Packet Internet Groper)
The Ping program uses the alphabet in the data portion of the packet as a payload, typically around 100 bytes by default, unless, of course, you are pinging from a Windows device, which thinks the alphabet stops at the letter W

Controlling the Source IP Address with Extended ping
R1# ping
Protocol [ip]:
Target IP address: 172.16.2.101
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 172.16.1.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 172.16.2.101, timeout is 2 seconds: Packet sent with a source address of 172.16.1.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/2/4 ms

Typically you'll see three values at the end separated with the slash "/," the minimum, average, and maximum RTT
Cisco ping Return Codes:
! - Each exclamation point indicates receipt of an ICMP echo reply. The ping completed successfully.
. - Each period indicates one timed out - blocked by access list or firewall, connectivity problem, or "unreachable" error
U - An ICMP unreachable message was received. - A router along the path did not have a route to the destination address.
C - An ICMP source quench message was received. - A device along the path- may be receiving to much traffic.
& - An ICMP time exceeded message was received. - A routing loop may have occurred.

Demystifying the lost first ping
When pinging a directly-attached host (end-station) from a router, it's quite common to lose the first reply, as shown in the following example (the same symptom might occur when pinging a remote host that has been inactive).
Actually, it's not the reply that was lost, the request was never sent out. Whenever a router has to send a packet to the next-hop (or directly attached destination) that has no entry in the ARP table, the ARP request is sent out, but the original packet is unconditionally dropped.

Debugging ARP requests while using the ping command:

```
Router2#show arp
Protocol  Address      Age (min)   Hardware Addr   Type   Interface
Internet  10.0.0.6     -           0016.c876.8b38  ARPA   FastEthernet0/0
Internet  10.0.0.5     0           0016.c7fe.f150  ARPA   FastEthernet0/0
Router2#debug arp
ARP packet debugging is on
Router2#ping 10.0.0.10
Sending 5, 100-byte ICMP Echos to 10.0.0.10, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/4 ms
08:26:21: IP ARP: creating incomplete entry for IP address: 10.0.0.10 interface FastEthernet0/0
08:26:21: IP ARP: sent req src 10.0.0.6 0016.c876.8b38, dst 10.0.0.10 0000.0000.0000 FastEthernet0/0
08:26:21: IP ARP: rcvd rep src 10.0.0.10 000c.29a7.8ade, dst 10.0.0.6 FastEthernet0/0
```

ICMPv6 includes a different set of message types that are particularly useful when performing local network testing. Type 128 (echo request) messages, which you can use to remotely sweep IPv6 networks.

Firewalk first calculates the distance to the gateway, and then sends packets destined for the target with a TTL that is one hop beyond. Based on the responses, the tool is able to map the filtering policy.

- If an ICMP type 11 code 0 (TTL exceeded in transit) message is received, the packet passed through the filter and a response was later generated.
- If the packet is dropped without comment, it was probably done at the firewall.
- If an ICMP type 3 code 13 (communication administratively prohibited) message is received, a simple filter such as a router ACL is being used.

If the packet is dropped without comment, this doesn't necessarily mean that traffic to the target host and port is filtered. Some firewalls know that the packet is due to expire and will send the expired message whether the policy allows the packet or not.

Traceroute
Cisco traceroute sends 3 UDP datagrams for each hop- first, three with a TTL field value set to 1 to cause the datagram to "timeout" as soon as it hits the first router, which responds with ICMP "time exceeded". Next, three more with the TTL value set to 2 for the second router, and so forth, incrementing the TTL for each hop. Since these datagrams try to access an invalid port (default 33434) at the destination host, the host will respond with ICMP "port unreachable" which tells the program to finish (Windows destinations send an ICMP echo reply back)
Standard traceroute defaults to using UDP on Linux, ICMP echo on Windows, but the principle is the same.

On Cisco IOS, running ping or traceroute without an IP address will allow the option to use extended versions of the programs. This will query for source and destination IPs in order to run those programs between specified gateways to check routing, troubleshoot routing loops, or packet loss; to determine the type of connectivity problem, to narrow down where the problem occurs.

Cisco IOS Extended Traceroute example:
Router A#traceroute
Protocol [ip]:
Target IP address: 192.168.40.2
!--- The address to which the path is traced.
Source address: 172.16.23.2
Numeric display [n]:
Timeout in seconds [3]:
Probe count [3]:
Minimum Time to Live [1]:
Maximum Time to Live [30]:
Port Number [33434]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Type escape sequence to abort.
Tracing the route to 192.168.40.2
1 172.31.20.2 16 msec 16 msec 16 msec
2 172.20.10.2 28 msec 28 msec 32 msec
3 192.168.40.2 32 msec 28 msec *

Cisco IOS traceroute returns these codes, here's their possible cause.
nn msec - This gives, for each node, the round-trip time (in milliseconds) for the specified number of probes
* - The probe timed out. - A device along the path either did not receive the probe or did not reply with an ICMP "packet life exceeded" message.
A - Administratively prohibited. - a firewall or router, may be blocking the probe and possibly other or all traffic; check access lists.
Q - Source quench. - A device along the path may be receiving to much traffic; check input queues.
H - An ICMP unreachable message has been received. A routing loop may have occurred.

TCP Traceroute
Both traceroute and tcptraceroute work on the same basic principle. Employing a tool for performing TCP traceroute is becoming a more popular option since it is less likely to be filtered. All will generate ICMP error messages when the TTL expires. Some firewalls block the "time exceeded" ICMP packets. SYN packets are used in TCP traceroutes since firewalls will often block other TCP packets that aren't part of an established connection.

intrace waits for an existing TCP connection. When it sees a connection it will send short-lived packets which appear as being part of the observed connection. intrace can do that because it has seen the packets, and so knows the IP addresses, ports and sequence numbers. The short-lived packets are adjusted so that they will not disrupt the TCP connection (i.e. they are simple "ACK" packets with no data by themselves, so the destination OS will simply ignore them).

Classic traceroute rely on ICMP type 11 (Time exceeded) responses from each hop along the route. If ICMP type 11 responses are being blocked by your firewall, traceroute will not work. These packets are inbound, not outbound.

TCP SYN packets will cause either a RST packet or a SYN ACK packet in response when they reach their destination. There is more about those sort of TCP interactions and what they mean in the section on scanning

It is possible to get ICMP type 3 code 4 responses back instead of ICMP type 11 responses if you send a large packet with the "Do not fragment" flag set, however this is likely only to allow you to find the hop with the smallest MTU. You will normally only get this sort of response from one hop along the route. Not all of them.

Windows has the pathping utility, and there is also MTR (https://github.com/White-Tiger/WinMTR). Older tools include vtrace and Cheops which you may need to use archive.org's WayBackMachine to find.

Simple routing and switching round trip *HostA (172.16.10.2) pings HostB (172.16.20.5) through one router*
Note: this example ignores the issue mentioned in the ping section "what happens to the first ping?"

1- ICMP's ping process creates it's echo request payload, which is simply the alphabet in the data field. It's handed to IP to form a packet. At a minimum, this packet contains an IP source address, an IP destination address, and a Protocol field with 0x01

3- IP checks to see if the destination IP address is a device on the local LAN or on a remote network. Since the destination device is on a remote network, the packet needs to be sent to the default gateway, so it finds it in the routing table. *Remember that hardware addresses are always local, so...*

4- It finds the default GW is 172.16.10.1, so the ARP cache is checked for the MAC if it has already been resolved. If the ARP cache doesn't have it, an ARP broadcast is sent out on local network for 172.16.10.1, router responds, and the cache updated with the info and an activity timer for it.

It turns out that the ARP table had it, and says it's the router's interface with the MAC address on Ethernet0, so it's activity timer is reset to 0, and the packet is *then* ready to be handed to a different part of the Data Link layer for framing.

5- With destination gateway and source MAC addresses, ether-type (0x0800 for IP) and the packet payload, the LAN driver is used to provide media access via the type of LAN (Ethernet), a frame is then generated, adding the FCS field (the result of the CRC).

6- Frame is given to the wire bit-by-bit. *THE FRAME HAS FINALLY ENTERED THE NETWORK*

7- All devices on the collision domain get the bits, rebuild the frame, and run a CRC to compare the FCS field for integrity. If it fails the integrity check it's discarded. *(note: this takes account if we are in a collision domain)*

8- If it passes, the destination MAC is checked to see if it matches it's own address. Eth0 on the router gets it, checks the ether-type field and gives the enclosed packet to IP (the rest is discarded).

9- IP on the router checks the packet header for errors (it doesn't run a complete CRC like the Data Link layer does on the frame). Since the packet's destination IP doesn't match any of the addresses configured on the router's interfaces, the router will look in its routing table. *If there is no entry for the network 172.16.20.0 the packet will be discarded immediately and an ICMP message will be sent back to the originating device with a destination network unreachable message.*
In this case it was found- the packet is routed to the exit interface (Eth1) for that subnet. Since the routing table shows "directly connected," no routing protocols are needed.

10- As the IP layer hands the packet to Layer 2, the MAC of HostB's IP address is either found in the ARP cache, or an ARP request is sent out for it (just as before in steps 3-6). We can safely say that the IP packet has been switched to a buffer before it's framing. It's framed, and sent out, just like in steps 5 and 6. *THE FRAME HAS LEFT THE ROUTER*

11- HostB receives the frame and immediately runs a CRC. If the result matches the information in the FCS field, the hardware destination address will be then checked next. If the host finds a match, the Ether-Type field is finally checked to determine the Network layer protocol that the packet belongs with (IPv4), and it is passed up the stack.

12- IP gets the packet, runs a CRC on the header, checks the destination address and sees it's own address, so it checks Protocol field to find out which Layer 4 protocol gets the payload segment. DESTINATION REACHED!

13- The payload is handed to ICMP, which understands that this is an echo request. ICMP responds to this by immediately discarding the packet and generating a new payload as an echo reply.

14- A packet is then created including the source and destination addresses, Protocol field, and payload. The destination device is now HostA.

15- Steps 3-6 are repeated on HostB: determining it goes to the gateway's IP address, getting the gateway's MAC, framing it, and sending it to wire for the router.

16- The router's Eth1 interface receives the bits and builds a frame. The CRC is run, and the FCS field is checked to make sure the answers match. It's handed to IP and steps 7-10 are repeated with destination swapped to HostA

17- To route the packet, the device knows to get to network 172.16.10.0 it should exit Eth0, and the MAC for 172.16.10.2 is already cached from the originating trip to HostB, so it builds a frame and sends it out on the wire. (LEAVES THE ROUTER)

18- The destination host receives the frame, runs a CRC, checks the destination MAC, then looks at the Ether-Type field, saying to hand it to off at the Network layer, it checks the Protocol field which says ICMP, and ICMP determines the packet to be an ICMP echo reply. ICMP acknowledges that it has received the reply by sending an exclamation point (!) to the user interface. ICMP then attempts to send four more echo requests to the destination host.

Fast Subnetting Examples - Subnet and Binary Practice

Remember to always place largest subnets first. How many hosts and how many subnets?

Class C Subnetting - Subnetting for Hosts

You are given 192.168.1.0 to start with, and are being asked for:

- one subnet that contains at least 50 usable IP addresses.

- two subnets that contains at least 14 usable IP addresses each.

Start all this with the biggest subnet. Consider our bit increments:

We need 50, so draw the line where 50 is:

128 64 | 32 16 8 4 2 1

|

32+16+8+4+2+1=63 This fits so we use it. *That 64 marks where our first network bit is.*

Our block ends at 192.168.1.63 (our broadcast address) and our first network is 192.168.1.0

What's the subnet mask? Add the unused bits: 128+64=192. The subnet mask is 255.255.255.192

CIDR/ Prefix length? 8 8 8 2 Count all the network bits- 26, 8+8+8+(128 and 64= 2 bits)

So, this subnet is 192.168.1.0/26 Part 1 is done, move on to part 2:

Second subnet starts at 192.168.1.64

We need 14, so draw the line:

128 64 32 16 | 8 4 2 1

|

8+4+2+1=15 - This fits so we use it. *That 16 marks where our first network bit is.*

64 is this new network number, and with a block of 15, it's broadcast is 192.168.1.79

What's the subnet mask? Add unused bits: 128+64+32+16=240 The subnet mask is 255.255.255.240

CIDR/ Prefix length? 8 8 8 4 Count all the network bits- 28 (128, 64, 32, 16= 4 bits)

So, our second subnet is 192.168.1.64/28 Part 2 is done, and we are almost finished

---

Third subnet starts at 192.168.1.80

We need 14 again:

128 64 32 16 | 8 4 2 1

|

8+4+2+1=15 - Just like the last step

Our block ends at 192.168.1.95 (our broadcast address)

What's the subnet mask? Add unused bits: 128+64+32+16=240 The subnet mask is 255.255.255.240

CIDR/ Prefix length? 8 8 8 4 Count all the network bits- 28 (128, 64, 32, 16= 4 bits)

So, our third subnet is 192.168.1.80/28, and we are done!

Next block available to continue begins at 192.168.1.96 so you have 96-255 to use in the future (block of 159 numbers for the 192.168.1.0 network we own)

You may be wondering... The line for subnet 1 was drawn between 64 and 32. 128 and 64 were included in counting up network bits when we determine subnet mask and the CIDR, but 64 is the block size, which is the 7th bit, not the 6th. Why? The reason for that is that is the 7th bit is now designated to chop up into networks, so it's a network bit. It has been appropriated (i.e. "borrowed) from the host bits. The block size is designated by the number to the left of where we draw the line.

Class B Subnetting - Subnetting for Hosts

You are given 170.70.0.0 to start with, and are being asked for:

- two subnets that contain 1000 usable IP addresses.
- one subnet that contains 500 usable IP addresses.
- one subnet that contains 100 usable IP addresses.

128 64 32 16 | 8 4 2 1

So what we were handed and are starting off with 170.70.0.0, 255.255.0.0
To make the first subnet of 1000 hosts. We need to add the 3rd octet in the diagram like this:

32768 16384 8192 4096 2048 1024 | 512 256 • 128 64 32 16 8 4 2 1

1023 - This fits so let's use it. *That 1024 marks where our first network bit is.*
This uses 10 host bits (8 bits for the 4th octet + 2 used in the 3rd= 10)
32 bits-10 bits= 22 bits for the network 8 8 6 0 - 170.70.0.0/22
The line was drawn in the 3rd octet going up to the 1023 mark.
/22 has a block size of 4, a subnet of 252 (256-4)
The subnet mask is = 255.255.252.0
Broadcast is 170.70.3.255 - the next network is 170.70.4.0

The second subnet starts at 170.70.4.0.
We need 1000 hosts again- what we just did, so this is easy. Just update the network number with what we already figured out. We know this is a /22 subnet with increments of 4 as the block size.
Subnet mask is 255.255.252.0
Broadcast is 170.70.7.255 - the next network is 170.70.8.0

Third subnet starts at 170.70.8.0
To make the subnet of 500 hosts:

32768 16384 8192 4096 2048 1024 512 | 256 • 128 64 32 16 8 4 2 1

511 - This fits so we use it. This equals 9 bits, leaving 23 remaining (32 bits-9 bits)
CIDR notation? 8 8 7 0 Network bits used- 23 (512 leaves 7 bits in 3rd octet)
170.70.8.0/23 with a block size of 2
Subnet mask is 255.255.254.0
Broadcast is 170.70.9.255 - the next network is 170.70.10.0

Fourth subnet: 170.70.10.0
To make the subnet of 100 hosts:

32768 16384 8192 4096 2048 1024 512 256 • 128 | 64 32 16 8 4 2 1

127 - This fits so we use it. This equals 7 bits, leaving 25 remaining (32 bits-7 bits)
CIDR notation? 8 8 8 1 Count the bits used- 25 (128 leaves 1 bits in 4rd octet)
170.70.10.0/25 - We didn't even have to touch the third octet for this one.
128 is the 25nd bit, thus subnet mask is 255.255.255.128
Broadcast is 170.70.10.127

Next network will start at 170.70.10.128. *170.70.10.128 through 170.70.244.128 are still unused!*

At the beginning, we determined how many bits, which is the first step. Here it was easy- we found where we could fit 50 bits on the bit counting line, but notice that in a way we counted from the right side. When provisioning networks instead of hosts, we have to figure out how many bits will do the job, and then count from the LEFT, adding to the original network numbers, borrowing from host bits.

Class B Subnetting - Subnetting for Networks

We get a class B: 140.78.0.0, 255.255.0.0; asked for 29 networks. Add 2 for NetID and BC is 31

First question! How many bits? 1+2+4+8+16 = 31 = uses 5 bits

Count 5 bits (from the LEFT for networks) and borrow from the host bits to extend network bits:

32768	16384	8192	4096	2048	1024	512	256	•	128	64	32	16	8	4	2	1

Our new starting subnet mask is going to be 255.255.248.0 (the 5 bits)

This is our first network - 140.78.8.0

140.78.	0	0	0	0	1	0	0	0	•	0	0	0	0	0	0	0	0

This is our first network's broadcast- 140.78.15.255

140.78.	0	0	0	0	1	1	1	1	•	1	1	1	1	1	1	1	1

Increment 1 binary in the network portion (the left). This is our second network - 140.78.16.0

140.78.	0	0	0	1	0	0	0	0	•	0	0	0	0	0	0	0	0

This is our second network's broadcast- 140.78.23.255

140.78.	0	0	0	1	0	1	1	1	•	1	1	1	1	1	1	1	1

Increment 1 binary in the network portion. This is our third network- 140.78.24.0

140.78.	0	0	0	1	1	0	0	0	•	0	0	0	0	0	0	0	0

This is our third network's broadcast- 140.78.31.255

140.78.	0	0	0	1	1	1	1	1	•	1	1	1	1	1	1	1	1

Increment 1 binary in the network portion. This is our fourth network - 140.78.32.0

140.78.	0	0	1	0	0	0	0	0	•	0	0	0	0	0	0	0	0

This is our fourth network's broadcast- 140.78.39.255

140.78.	0	0	1	0	0	1	1	1	•	1	1	1	1	1	1	1	1

...And so forth. We begin to see the pattern- multiples of 8, our dividing line gave us the block size

Net ID	**1st address**	**Last address**	**Broadcast**
140.78.8.0	140.78.8.1	140.78.15.254	140.78.15.255
140.78.16.0	140.78.16.1	140.78.16.254	140.78.23.255
140.78.24.0	140.78.24.1	140.78.31.254	140.78.31.255
140.78.32.0	140.78.32.1	140.78.39.254	140.78.39.255

What's the last network you can give out? 140.78.240.0:

140.78.	1	1	1	1	0	0	0	0	•	0	0	0	0	0	0	0	0

It's broadcast is 140.78.247.255

140.78.	1	1	1	1	0	1	1	1	•	1	1	1	1	1	1	1	1

Class B Subnetting - Subnetting for Networks - Example 2

We are given a Class B: 150.9.0.0, BC: 255.255.0.0
We are asked for 10 networks- always add 2 = We really need 12; for networks count from the LEFT

How many bits? 12, so 1100=12 = 4 bits. Count 4 from the left - New magic number is 16
Our new starting subnet mask is going to be 255.255.240.0 (the 5 bits)

32768	16384	8192	4096	2048	1024	512	256 •	128	64	32	16	8	4	2	1

This is our first network - 150.9.16 .0

150.9.	0	0	0	1	0	0	0	0	•	0	0	0	0	0	0	0	0

This is our first network's broadcast- 150.9.31.255

150.9.	0	0	0	1	1	1	1	1	•	1	1	1	1	1	1	1	1

The last example walked through discovering our networks and we saw the pattern for block of 8 bits repeating.
Below it is the same thing using the new info, but the block is 16.

Net ID	**1st address**	**Last address**	**Broadcast**
150.9.0.0	150.9.0.1	150.9.15.254	150.9.15.255
150.9.16.0	150.9.16.1	150.9.31.254	150.9.31.255
150.9.32.0	150.9.32.1	150.9.47.254	150.9.47.255

150.9.48.0, 150.9.64.0, 150.9.80.0, 150.9.96.0, 150.9.112.0, 150.9.128.0, 150.9.144.0, 150.9.160.0...

What if...? Subnet Masks and borrowing from the fourth octet

What if we instead were working with 10 bits instead of 4? How do we deal with the subnet mask?
Remember we are given a Class B: 150.9.0.0

32768	16384	8192	4096	2048	1024	512	256 •	128	64	32	16	8	4	2	1

This is our first network - 150.9.0.0- 150.9.0.63

150.9.	0	0	0	1	0	0	0	0	•	0	0	0	0	0	0	0	0

Net ID	**1st address**	**Last address**	**Broadcast**
150.9.0.0	150.9.0.1	150.9.0.62	150.9.0.63
150.9.64.0	150.9.0.64	150.9.0.126	150.9.1.127

Use 64 (the 7th bit), 255.255.255.192 - since we are borrowing from the 4th octet it might look like a class C, but we can ultimately see by the first octet of the address (150) that this is Class B, but it's the net/host boundary of the address dictates the subnet mask.

Class B Subnetting - Subnetting for Hosts - Review

We are given a Class B: 160.12.0.0, BC: 255.255.0.0 We are asked for 4080 hosts so add 2= 4082 Bits
So we need 12 bits- for hosts count from the RIGHT

32768	16384	8192	4096	2048	1024	512	256 •	128	64	32	16	8	4	2	1

Subnet mask is 240 (128+64+32+16) "Magic number" is 16
160.12.0.0... 160.12.15.255
160.12.16.0… 160.12.31.255
160.12.32.0, 160.12.48.0, 160.12.64.0, etc

Class C Subnetting - Subnetting for Networks - Review

We are given 201.9.6.0 told to make 25 subnets- add 2 and it's 27.
27 = 5 bits - count from the left
201.9.6. 0 0 0 0 1

128	64	32	16	8	4	2	1

CIDR is /29 and 255.255.255.248 mask
Network IDs are 201.9.6.8, 201.9.6.16, 201.9.6.24, 201.9.6.32, 201.9.6.40 ... etc.

Variable Length Subnet Mask - VLSM #1
Rule 1: Always make networks from highest to lowest in size
We have a starting address of 172.16.0.0
We need 4 subnets, sized with 8000, 1000, 400, and 100 hosts respectively.

First net of 8000 hosts- how many bits? 13 Bits start with the originating network ID (172.16.0.0) *Count from the right:*

32768 16384 8192 | 4096 2048 1024 512 256 • 128 64 32 16 8 4 2 1

172.16.	0	0	0	1	0	0	0	0	•	0	0	0	0	0	0	0	0

We get 172.16.0.0/19 and know the next network ID is 172.16.32.0.
So, the first VLSM block is 172.16.0.0/19, mask is 255.255.224.0, and broadcast of 172.16.31.255

Second network: 1000 hosts. Line is now between 1024 and 512 (10 bits).
Depending how you do yours, you can leave the 32 marker from the first network turned on, but move our dividing line to the new spot. So we start with 172.16.32.0

172.16.	0	0	1	0	0	0	0	0	•	0	0	0	0	0	0	0	0

32768 16384 8192 4096 2048 1024 | 512 256 • 128 64 32 16 8 4 2 1

This VLSM block is 172.16.32.0/22, subnet mask is 255.255.252.0, with a broadcast of 172.16.35.255

Third network: 400 hosts - starts at 172.16.36.0

172.16.	0	0	1	0	0	1	0	0	•	0	0	0	0	0	0	0	0

Do the same thing we just did with previous markers.
Line is between 256 and 512

32768 16384 8192 4096 2048 1024 512 | 256 • 128 64 32 16 8 4 2 1

This is a /23 subnet (8+8+7). We know from 32 + 4 + 2=38 that the next network ID is 172.16.38.0
This VLSM block is 172.16.36.0/23, subnet mask is 255.255.254.0, with a broadcast of 172.16.37.255

Fourth network: 100 hosts - starts at 172.16.38.0

172.16.	0	0	1	0	0	1	1	0	•	0	0	0	0	0	0	0	0

The line is between 128 and 64 we are now borrowing host bits from the 4th octet

128 | 64 32 16 8 4 2 1

This is a /25 subnet (8+8+9). 128 is the next network ID
This VLSM block is 172.16.38.0/25, subnet mask is 255.255.128.0, with a broadcast of 172.16.127.255

Hosts	Net ID	Last address	Broadcast
8000	172.16.0.0	172.16.31.254	172.16.31.255
1000	172.16.32.0	172.16.35.254	172.16.35.255
400	172.16.36.0	172.16.37.254	172.16.37.255
100	172.16.38.0	172.16.38.126	172.16.38.127

Next network would begin at 172.16.38.128 for future space (room for about 216 /24 networks or in ip addressed formatted binary 00000000.00000000.11010010.10000000)

Variable Length Subnet Mask - VLSM #2
140.58.0.0 is our starting IP. We need networks that are big enough to cover these host blocks:
Blocks for 7000, 2500, 1800; and we need 2 blocks of 900, a 500 block, and 2 tiny two-host networks.

First network: 7000 hosts- how many bits? 13 Bits:

32768 16384 8192 | 4096 2048 1024 512 256 • 128 64 32 16 8 4 2 1

So we have this- start with the originating network ID:

140.58.	0	0	0	1	0	0	0	0	•	0	0	0	0	0	0	0	0

This is a /19 subnet (8+8+3).
This VLSM block is 140.58.0.0/19, subnet mask is 255.255.254.0, with a broadcast of 140.58.31.255

So we have 140.58.32.0 as a starting network ID:

140.58.	0	0	1	0	0	0	0	0	•	0	0	0	0	0	0	0	0

Second network: 2500 hosts. Line is now between 2048 and 4096 (12 bits)

32768 16384 8192 4096 | 2048 1024 512 256 • 128 64 32 16 8 4 2 1

This is a /20 subnet (8+8+4). 32 + 16 (48) is the next network ID
This VLSM block is 140.58.32.0/20, subnet mask is 255.255.240.0, with a broadcast of 140.58.47.255

So we have 140.58.48.0 as a starting network ID:

140.58.	0	0	1	1	0	0	0	0	•	0	0	0	0	0	0	0	0

Third network: 1800 hosts. Line is now between 1024 and 2048(11 bits)

32768 16384 8192 4096 2048 | 1024 512 256 • 128 64 32 16 8 4 2 1

This is a /21 subnet (8+8+5). 32 + 16 + 8 (56) is the next network ID
This VLSM block is 140.58.48.0/21, subnet mask is 255.255.248.0, with a broadcast of 140.58.55.255

So we have 140.58.56.0 as a starting network ID:

140.58.	0	0	1	1	1	0	0	0	•	0	0	0	0	0	0	0	0

Fourth network: 900 hosts. Line is now between 512 and 1024 (10 bits)

32768 16384 8192 4096 2048 1024 | 512 256 • 128 64 32 16 8 4 2 1

This is a /22 subnet (8+8+6). 32 + 16 + 8 + 4 (60) is the next network ID
This VLSM block is 140.58.56.0/22, subnet mask is 255.255.252.0, with a broadcast of 140.58.59.255

So we have 140.58.60.0 as a starting network ID:

140.58.	0	0	1	1	1	1	0	0	•	0	0	0	0	0	0	0	0

Fifth network: also 900 hosts. Line is between 512 and 1024 (10 bits) again (see 4th network just above for diagram)

Ok- we need to draw our line, but it's already at the spot we need to put it. The interval hasn't changed so add a 4 to the network. The number line is just a workspace to help us keep track of things, so it's not a big deal.
This is a /22 subnet (8+8+6). 32 + 16 + 8 + 4 **+ 4** (64) is the next network ID
This block is 140.58.60.0/22, mask is 255.255.252.0, broadcast of 140.58.63.255; next network number is 140.58.64.0

If you are doing this on paper/ by hand it can get messy and you may have 172.16.00111100.00000000 down from drawing the number line over and over. Clean up the 64 in the 3rd octet like this since it is 64: 172.16.01000000.00000000

Continuing, we have 140.58.60.0 as a starting network ID:

140.58.	0	1	0	0	0	0	0	•	0	0	0	0	0	0	0	0

Sixth network: 500 hosts. Line is now between 256 and 512 (9 bits)

32768 16384 8192 4096 2048 1024 512 | 256 • 128 64 32 16 8 4 2 1

This is a /23 subnet (8+8+7). 64 + 2 (66) is the next network ID
This VLSM block is 140.58.64.0/23, subnet mask is 255.255.254.0, with a broadcast of 140.58.65.255

So we have 140.58.66.0 as a starting network ID:

140.58.	0	1	0	0	0	0	1	0	•	0	0	0	0	0	0	0	0

Seventh and eighth networks: 2 networks of 2 hosts each. We are zooming in to the bottom of the 4th octet. Hypothetically, you could ask if it's a point-to-point not needing reserved network ID and BC addresses, or just throw those in anyway just to save the trouble of having the conversation. I am opting for the latter (2 IP addresses, 4 total, big deal)

128 64 32 16 8 4 2 | 1

These are /30 subnets (8+8+8+6). Now that we are in the 4th octet, our network IDs are 140.58.66.0 (with .1 and .2 hosts, and .3 BC), and 140.58.66.4 (with .5 and .6 hosts, and .7 broadcast)

Hosts	**Net ID**	**Last address**	**Broadcast**
7000	140.58.0.0	140.58.31.254	140.58.31.255
2500	140.58.32.0	140.58.47.254	140.58.47.255
1800	140.58.48.0	140.58.55.254	140.58.55.255
900	140.58.56.0	140.58.59.254	140.58.59.255
900	140.58.60.0	140.58.63.254	140.58.63.255
500	140.58.64.0	140.58.64.254	140.58.64.255
2	140.56.66.0	140.58.66.2	140.58.66.3
2	140.58.66.4	140.58.66.6	140.58.66.7

Next network would begin at 140.58.66.8

The /30 is considered the smallest network you can make to provision a tiny network for isolated connection coupling. In practice, you will hear about /31s working fine for serial connections which need only 2 addresses provisioned for them, but this is NOT recognized by Cisco as working. A /31 may work fine in a production network, but is guaranteed to be the wrong answer to Cisc exams.

140.58.0.0, our starting class B. For network 1, we needed 7000 provisioned and we got 8190 so there is plenty of room for growth. Network 2 the same thing: we needed 2500 and got a 4094 ceiling for future growth.

All networks have significant overhead, and the only snag might be is if we needed to add routers to the networks with 2 hosts, in which case it is easy to just make some more tiny networks, since all of these networks to handle the enterprise of thousands of hosts barely made a dent in our original Class B address-space.

The previous pages (subnetting for hosts, subnetting for networks, VLSM 1 & 2) all were made out of the information presented in the classroom videos available here: ***https://www.youtube.com/user/TheLITCNTT*** *I transcribed them since they were so fast and simple, and didn't require more than making a number line and stuff from memory (for the most part). I am not affiliated with that school, just found the videos of the whiteboard presentation.*

For templating/ reuse:

128		64		32		16		8		4		2		1

32768		16384		8192		4096		2048		1024		512		256	•	128		64		32		16		8		4		2		1

Subnet Shortcuts
Using binary AND: 1 and 1 = 1, all others = 0 (see below)

What is the network number of the address 192.168.100.115 with a subnet mask of 255.255.255.240?
192.168.100.115 = 11000000.10101000.01100100.01110011
255.255.255.240 = 11111111.11111111.11111111.11110000
ANDed result = 11000000.10101000.01100100.01110000 = 192.168.100.112
192.168.100.115 belongs to the 192.168.100.112 network when a mask of 255.255.255.240 is used.

It get's simpler: octets of the addresses with a common 255 or 0 can be ignored
What's the broadcast for the IP address 192.168.100.164 if it has a subnet mask of 255.255.255.248?
192.168.100.164 = 192.168.100.10100100
255.255.255.248 = ...11111000
ANDed result = 192.168.100.10100000 = 192.168.100.160 (here's our subnet number)

Separate the network bits from the host bits:
255.255.255.248 = /29 = the first 29 bits are network bits - so the last three are host bits
----.10100 **000**. Change all host bits to 1, so it's -----.10100111 = 192.168.100.167
The broadcast of 192.168.100.164 is 192.168.100.167 when the subnet mask is 255.255.255.248.

To what network does 131.186.227.43 belong, if its subnet mask is 255.255.240.0?
Based on the two shortcut rules, the answer should be 131.186.???.0
So now you only need to convert one octet to binary for the ANDing process:
227 = 11100011
240 = 11110000
11100000 = 224 Therefore, the answer is 131.186.224.0

What subnet does this belong to?
What subnet does 192.168.12.78/29 belong to?
Our mask is a /29. The next boundary is 32. So 32-29=3.
Now 2^3 = 8 which gives us our block size i.e. 2 to the power of 3 equals 8.

We have borrowed from the last octet as the 29th bit is in the last octet. We start from zero and count up in our block size.
Therefore it follows that the subnets are:
192.168.12.0
192.168.12.8
192.168.12.16
...
192.168.12.72
192.168.12.80
Our address is 192.168.12.78 so it must sit on the 192.168.12.72 subnet.

What subnet does 172.16.116.4/19 sit on?
Our mask is /19 and our next boundary is 24. Therefore 24-19=5. The block size is 2^5 = 32.
We have borrowed into the third octet as bit 19 is in the third octet so we count up our block size in that octet. The subnets are:
172.16.0.0
172.16.32.0
172.16.64.0
172.16.96.0
172.16.128.0
Our address is 172.16.116.4 so it must sit on the 172.16.96.0 subnet.

What subnet does 10.34.67.234/12 sit on?
Our mask is 12. Our next boundary is 16. Therefore 16-12=4. 2^4=16 which gives us our block size.
We have borrowed from the second octet as bit 12 sits in the second octet so we count up the block size in that octet. The subnets are:
10.0.0.0
10.16.0.0
10.32.0.0
10.48.0.0
.....etc
Our address is 10.34.67.234 which must sit on the 10.32.0.0 subnet.

What is the *valid* host range of the Nth subnet of abc.xxx.yyy.zzz/xx?

What is the valid host range of the 4th subnet of 192.168.10.0/28?
The block size is 16 since 32-28=4 and 2^4 = 16.
We need to count up in the block size in the last octet as bit 28 is in the last octet.

192.168.10.0
192.168.10.16
192.168.10.32
192.168.10.48
192.168.10.64
The 4th subnet is 192.168.10.48 and the host range must be 192.168.10.49 to 192.168.10.62, remembering that the subnet and broadcast address cannot be used.

What is the valid host range of the 1st subnet of 172.16.0.0/17?
The block size is 128 since 24-17=7 and 2^7 = 128.
We are borrowing in the 3rd octet as bit 17 is in the 3rd octet. Our subnets are:

172.16.0.0
172.16.128.0

The first subnet is 172.16.0.0 and the valid host range is 172.16.0.1 to 172.16.127.254. You must remember not to include the subnet address (172.16.0.0) and the broadcast address (172.16.127.255).

What is the valid host range of the 7th subnet of address 10.0.0.0/14?
The block size is 4, from 16-14=2 then 2^2 = 4.
We are borrowing in the 2nd octet, so count in the block size from zero 7 times to get the 7th subnet.

The seventh subnet is 10.24.0.0. Our valid host range must be 10.24.0.1 to 10.27.255.254 again remembering not to include our subnet (10.24.0.0) and the broadcast address (10.27.255.255).

10.0.0.0
10.4.0.0.0
10.8.0.0.0
...
10.20.0.0.0
10.24.0.0.0

Dotted decimal mask? What's the block size?
A mask of 255.255.192.0 - you would simply count up in 256-192 = 64 in the third octet.
Or 255.224.0.0 - block size is 256-224=32 in the second octet.

Question: You are designing a subnet mask for the 10.0.0.0 network. You want 3800 subnets with up to 3800 hosts on each subnet. What subnet mask should you use?
Answer: 255.255.240.0

Question: How many subnets and hosts per subnet can you get from the network 172.29.0.0 255.255.254.0?
Answer: 128 subnets and 510 hosts

Just memorize how many items you get for each number of bits (subtract two for hosts)

000000000001 = 2 (1)
000000000011 = 4 (2)
000000000111 = 8 (3)
000000001111 = 16 (4)
000000011111 = 32 (5)
000000111111 = 64 (6)
000001111111 = 128 (7)
000011111111 = 256 (8)
000111111111 = 512 (9)
001111111111 = 1024 (10)
011111111111 = 2048 (11)
111111111111 = 4096 (12)

If you can't remember what one is, just remember it's twice the number of the one before it (or halve the number after it if you're counting down). There's no math, tricks, etc. involved.

So for the first question, you'll need 12 bits for both because the value is between 2048 and 4096 (you obviously have to go with the higher value).

You can look at the other question the same way. /23 on a class B will give you 7 subnet bits and 9 host bits. 7 bits is 128 and 9 bits is 512 (minus 2 gives you 510).

Different answer:
Bits do you need to borrow to accommodate 3800 subnets?
12 bits as 2 ^ 12 = 4096.
Your network address is a /8 by default as it is a Class A address so 8 + 12 = /20 mask.

The second question is straightforward.
Your address is Class B so it has a default mask of /16.
255.255.254.0 is the same as /23, so 23-16=7.
2^7 = 128 subnets
(2 ^ (32 - 23))-2 = 2^9-2 = 510 hosts.

Interesting:
Question: *What is the last valid host on the subnetwork 172.25.248.0/21?*
Answer: 172.25.255.254

Subnet mask: inside an octet, mask is your current block size added to the next-highest neighboring block size (except 128 has a mask of 128)
128+64=192, 192+32=224, 224+16=240, etc:

128	192	224	240	248	252	254	255
128	64	32	16	8	4	2	1

Subnetting Chart Tools for Quick Reference

Constructed by Tristan Mendoza

CIDR is the number of network bits used. 255.255.255.192 = 11111111.11111111.11111111.11000000 = /26 network bits, 32-26 = 6 host bits
Many teachers and some exams require you know the long way to do things- longhand calculation of usable hosts per subnet= 2^{H}-2 (H= host bits)
Usable subnets=2^{N} with N for network. If you know how to draw the number line of binary with bits up high enough, it's clear every bit after 4096 is just as before going to double the previous bit's number. It's easier to draw what you need like done in this sheet.
Block Sizes: Subtract subnet mask from 256 to get the bit position or "magic number." Mask of 192 (256-192=64) gives the network numbers of 0, 64, 128, and 192. So 255.255.255.192 has a /26 CIDR. Since 32-26= 6 host bits, we get 2^{6}-2=62 usable hosts per subnet.
Remembering mask numbers: Add bit value to it's bigger neighbor's mask like this: 128+64=192, 192+32=224, 224+16=240, etc.

Normal Bit Value/ Position	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
2-exponent representation	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
Subnet Mask for Bit Position	128	192	224	240	248	252	254	255
Class C CIDR	/25	/26	/27	/28	/29	/30	/31	/32
Subnets	2	4	8	16	32	64	128	n/a
Hosts per Subnet	126	62	30	14	6	2	2-no NetID/BC	n/a
			A /24 is all host bits in the 4th octet, thus 1 class C (2 /25's, 4 /26's, or 8 /27's, etc.)					
Class B CIDR (3rd Octet)	/17	/18	/19	/20	/21	/22	/23	/24
Subnets	128 /24's	64 /24's	32 /24's	16 /24's	8 /24's	4 /24's	2 /24's	1 Class C
Hosts (unsubnetted)	32766	16382	8190	4094	2046	1022	510	254
Class B CIDR (4th Octet)	/25	/26	/27	/28	/29	/30		
Subnets	512	1024	2048	4096	8192	16384		
Hosts (unsubnetted)	126	62	30	14	6	2		
			Like /24, a /16 is a whole class B, holding all host bits in the 3rd and 4th octets					
Class A CIDR (2nd Octet)	/9	/10	/11	/12	/13	/14	/15	/16
Subnets	128 /16's	64 /16's	32 /16's	16 /16's	8 /16's	4 /16's	2 /16's	1 Class B
Hosts per Subnet (subtract 2)	8,388,606	4,194,302	2,097,150	1,048,574	524,286	262,142	131,070	65,534
Class A CIDR (3rd Octet)	/17	/18	/19	/20	/21	/22	/23	/24
Subnets	512	1024	2048	4096	8192	16,384	32,768	65,536
Hosts per Subnet (subtract 2)	32766	16382	8190	4094	2046	1022	510	254

Default Classful System

	Leading Bits		Starts At	Net Bits	Bits Left	# of Nets	Hosts per Net	Default Mask	Native CIDRs
Class A	0xxx, 1-126	-	0.0.0.1	8	24	128	16,777,216	255.0.0.0	/9 to /16
Class B	10xx, 128-191	-	128.0.0.0	16	16	16,384	65,534	255.255.0.0	/17 to /24
Class C	110x, 192-223	-	192.0.0.0	24	8	2,097,152	254	255.255.255.0	/25 to /32
Class D	1110, 224-239	Multicast	224.0.0.0	-	-	-	-	-	
Class E	1111, 240-254	Reserved	240.0.0.0	-	-	-	-	-	

Reserved and Private Addresses

Class A	*10.0.0.0 - 10.255.255.255*	*10.0.0.0/8*
Class B	*172.16.0.0 - 172.31.255.255*	*172.16.0.0/12*
Class C	*192.168.0.0 - 255.255*	*192.168.0.0/16*
Loopback	*127.x.x.x*	*127.0.0.0/8*
APIPA	*169.254.x.x*	*169.254.0.0/16*
Carrier NAT	*100.64.0.0 – 100.127.255.255*	*100.64.0.0/14*
Stress-testing	*198.18.0.0 – 199.19.255.255*	*198.18.0.0/15*

The main reason the chart above lists CIDR info for unusual-looking address space is to assist with imagining the "subnetting" of large chunks of private address space as is mostly the case in 10.0.0.0/8, but also 172.16.0.0/12 blocks. In those cases, for ease of reference mimicking the traditional routed addressing while adding a more human-recognizable patterns might help, (i.e., 10.1.x.x for eastern US, 10.2.x.x for central U.S., 10.3.x.x for western U.S., and so forth, as appropriate). An entire 10.0.0.0/8 has plenty of space for creativity.

Using the worksheet on the right hand side

Class B Subnetting - Subnetting for Hosts
We are given a Class B: 160.12.0.0, BC: 255.255.0.0. We are asked for 4080 hosts so add 2= 4082 Bits: where is the line drawn? 4096 ^ 2048 1024 512 256 128 64 32 16 8 4 2 1
So we need 12 bits- for hosts count from the RIGHT
160.12. 0 0 0 1
128 64 32 16 || 8 4 2 1 . 128 64 32 16 8 4 2 1
Subnet mask is 240 (128+64+32+16) "Magic number" is 16
160.12.16.0… 160.12.31.255
160.12.32.0, 160.12.48.0, 160.12.64.0, etc

Class C Subnetting - Subnetting for Networks
We are given 201.9.6.0 told to make 25 subnets- add 2 and it's 27.
27 = 5 bits - count from the left
201.9.6. 0 0 0 0 1
128 64 32 16 8 || 4 2 1
CIDR is /29 and 255.255.255.248 mask
Network IDs are 201.9.6.8, 201.9.6.16, 201.9.6.24, 201.9.6.32, 201.9.6.40 ... etc.

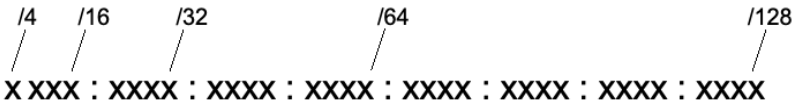
Class C Subnetting - Subnetting for Hosts
We have 195.12.8.0 and need 40 hosts +2 = 42 = 6 bits to hold
This time count FROM THE RIGHT to place the divider:
195.12.8.
128 64 || 32 16 8 4 2 1
Subnet mask is 255.255.255.192 (128+64), CIDR is /26
Network IDs 195.12.8.64, 195.12.8.128. That's it since 192 is subnet for these subnets (and it's only a class C

/9	128	128	24	2^{23}	8,388,608
/10	192	64	23	2^{22}	4,194,304
/11	224	32	22	2^{21}	2,097,152
/12	240	16	21	2^{20}	1,048,576
/13	248	8	20	2^{19}	524,288
/14	252	4	19	2^{18}	262,144
/15	254	2	18	2^{17}	131,072
/16	255	1	17	2^{16}	65,536
			.		
/17	128	128	16	2^{15}	32,768
/18	192	64	15	2^{14}	16,384
/19	224	32	14	2^{13}	8,192
/20	240	16	13	2^{12}	4,096
/21	248	8	12	2^{11}	2,048
/22	252	4	11	2^{10}	1,024
/23	254	2	10	2^{9}	512
/24	255	1	9	2^{8}	256
			.		
/25	128	128	8	2^{7}	128
/26	192	64	7	2^{6}	64
/27	224	32	6	2^{5}	32
/28	240	16	5	2^{4}	16
/29	248	8	4	2^{3}	8
/30	252	4	3	2^{2}	4
/31	254	2	2	2^{1}	2
/32	255	1	1	2^{0}	1
← CIDR (network bits)	← subnet mask	← Binary bit by octet	← Binary bit placement	← Power of 2	← # of hosts by position

IPv6 Overview

- IPv6 core protocol definition is RFC 2460
- RIPng, EIGRPv6, OSPFv3, ICMPv6, traceroute6, ping6, MP-BGP-4 (mulitprotocol BGP v4)
- ARP is replaced with Neighbor discovery, Neighbor solicitation/ duplicated address detection, multiprotocol router discovery and router advertisement.

IPv6 Numbering Summary:
Each character is 4 bits, in hexadecimal, so every block of 4 (a quartet) of those is 16 bits. Typically, IPv6 addresses refer to a certain length or place using the backslash notation. This diagram shows the structure of an IPv6 address to demonstrate bit placement/ordering 0-128 bits.



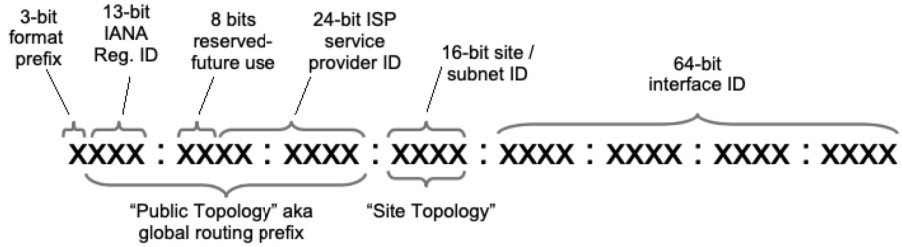
So, a prefix ending at the /48th bit, we can say something like 2001:1234:5678::/48 (how many bits are used)

You'll see IPv6 addresses shortened often. Here's how that works:
Start with expanded address: 2001:0DB8:0000:0000:0000:FF00:0042:8329
- Remove any leading zeros - All leading zeros within each group of 4 hexadecimal digits can go. So, 0DB8 becomes DB8, 0000 becomes 0, and 0042 becomes 42 - 2001:DB8:0:0:0:FF00:42:8329
- Replace consecutive groups of zeros with double colons (::). You can only do this once per address, so do it to the largest grouping or the leftmost if there are a few the same size. Now we have 2001:DB8::FF00:42:8329
- This is more a style thing instead of shortening: make all letters lowercase. It helps with both ease of reading, but also with case-sensitive searching or filtering. We finally get: 2001:db8::ff00:42:8329
- Expanding is easy: inverse the rules and pad out so there are 8 quartets of 4 bits each.

Address Types

If asked what the 3 types of IPv6 addresses there typically the expected answer is unicast, multicast and anycast. It's most efficient to start of with unicast, of which there are also 3 types, one routable on the internet and the other two that have a local scope. It is also the one that has the most components so it makes the others easier.

Global Unicast (GUA)



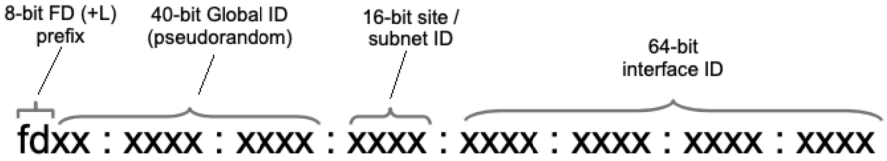
So, using the global unicast address as a starting point, take 2001:0db8:56a4:0001:0000:0000:0000:0002 - or - 2001:db8:56a4:1::2 In this example, the generalized prefix would be 2001:0db8:56a4::/48, the subnet would be 0001, and interface ID (simplified) 2

Global unicast addresses are routable and what you need to use to do anything internet. The other two can't leave your local nets. The also can have a variety of prefixes (the other 2 have the same ones and are easy to spot). As of Feb 2023, IANA lists 2001:0000::/23 – 2c00:0000::/12 as being currently registered/available global unicast prefix blocks ffrom regional registries (RIRs). For more info, see https://www.iana.org/assignments/ipv6-unicast-address-assignments.

If your company was only given one /64, you will likely have a subnet ID assigned by the ISP along with the rest of the prefix illustrated above. The interface ID represents a single interface on any given device. Typically you shouldn't need to worry about altering anything in the prefix your ISP gave you including the subnet ID. You may have been given a larger address space to work with like /32 or /48. There is a section on subnetting those after talking about addresses and such.

2001:1234:3333::/48 -Company B
2001:1234:3333:0001::/64 -Company B's subnet 1
2001:1234:3333:0002::/64 -Company B's subnet 2
2001:1234:3333:2::1 -A host in Company B's subnet 2

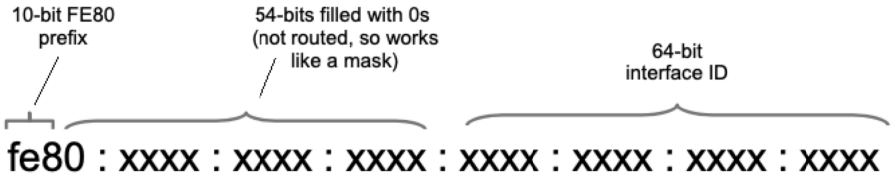
Unique Local Addresses (ULA) Unicast



Unique Locals are like the reserved IPv4 addresses for private networks (192.168.0.0/16, 172.16.0.0/12, and 10.0.0.0/8). They aren't routable over the internet (we'll talk about NAT later) but they can move around the local network. They replace an old one named “site local”. These don't use the IANA/ISP's prefix like global unicast, and instead is a “random” replacement (still often called a global ID... out of habit?). This can be autogenerated by the device or done manually, which will be discussed soon.

fc00::/8 is for unique local and assigned from say a DHCP or other provider server
fd00::/8 is only for a locally assigned (IE fd00:: + 40 random bits) without a provider of some kind to generate/hand out addresses.

Link Local (Unicast), fe80::/10



Link locals are auto-generated and only useful inside one subnet- can't be routed. They mainly help with the functions of neighbor discovery and next hop configuration. A great way to check out all 3 of these unicast types is to run ifconfig on your workstation.

Multicast address ff00::/8
Identifies a group of nodes or interfaces with traffic forwarded to all the nodes in the group. Addresses are all assigned out of the FF00::/8 block; also have a scope associated: link local being just like the scope of unicast LL, 'organization' with same scope as unique local, and global.

Anycast address
Identifies a group of nodes or interfaces, with traffic forwarded to the nearest node in the group. An anycast address is essentially a unicast address assigned to multiple devices with a host ID = 0000:0000:0000:0000. (Anycast addresses are not widely used today.)

Key IPv6 Multicast Addresses

ff01::1	All nodes in the interface-local	
ff02::1	All nodes in the link-local, RA/RS	
ff0X::1	All nodes address, identify the group of all IPv6 nodes	
ff0X::2	All routers	1 (interface-local), 2 (link-local), 5 (site-local)
ff01::2	All routers in the interface-local	
ff02::2	All routers in the link-local	
ff05::2	All routers in the site-local	
ff02::5	OSPFv3	2 (link-local)
ff02::6	OSPFv3 Designated Routers	2 (link-local)
ff02::9	RIP Routers	2 (link-local)
ff02::a	EIGRPv6 Routers	2 (link-local)
ff02::d	All PIM Routers	2 (link-local)
ff02::1a	All RPL Routers	2 (link-local)
ff0X::fb	mDNSv6	All scopes
ff0X::101	All NTP servers	All scopes
ff02::1:1	Link Name	2 (link-local)
ff02::1:2	All-dhcp-agents	2 (link-local)
ff02::1:3	Link-local Multicast Name Resolution	2 (link-local)
ff05::1:3	All-dhcp-servers	5 (site-local)
ff02::1:ff00:0/104	Solicited-node multicast address	2 (link-local)
ff02::2:ff00:0/104	Node Information Queries	2 (link-local)

Host Addressing: SLAAC and EUI-64 (RFC 2373)
Stateless Address Auto-Configuration (SLAAC) enables hosts to generate a unique **routable** address on their own. For this the router sends out a Router Advertisement (RA) periodically, and a host can send a Router Solicitation(RS) in order to trigger an RA. The RA has the prefix and length to use, default gateway. Before using the address it makes, the host uses Duplicate Address Detection (DAD) to make sure it's unique. Since things like DNS servers aren't in RAs, DHCPv6 still has to be used. (DAD takes the place of ARP in IPv6)

For the interface ID, the host either uses a EUI-64 (Extended Unique Identifier) made from the interface's MAC address (most likely), or can optionally generate it randomly. The EUI-64 creation works like this:
1. Split the 6-byte (12-hex digit) MAC address in two down the middle.
2. Insert FFFE in between the two halves.
3. Invert the seventh bit from the beginning.

Examples:
Interface MAC address is aa12:bcbc:1234
10101010 represents the first 8 bits of the MAC address (aa) - when inverting the 7th bit it becomes 10101000.
The answer is A8 and the EUI-64 address: 2001:0db8:0:1:a812:bcff:febc:1234 EUI-64

MAC address 0b34:ba12:1234
0b in binary is 00001011, the first 8 bits of the MAC address, which then becomes 00001001
The answer is 09, and the IPv6 EUI-64 address: 2001:0db8:0:1:0934:baff:fe12:1234 EUI-64

Generating EUI-64 is usually done for us:
```
ipv6 address 2001:db8:1111:1::/64 eui-64

interface GigabitEthernet0/0
  ipv6 address 2001:db8:1111:1::/64 eui-64
interface serial0/0/0
  ipv6 address 2001:db8:1111:2::/64 eui-64
show ipv6 interface brief
    GigabitEthernet0/0 [up/up]
        fe80::1ff:fe01:101                  <---link local, employing eui64
        2001:db8:1111:1:0:1ff:fe01:101    <---EUI64 global unicast address
    Serial0/0 [up/up]
        fe80::1ff:fe01:101
        2001:db8:1111:2:0:1ff:fe01:101
```

Cisco router, setting up SLAAC
```
Router>enable
Router#configure terminal
Router(config)#ipv6 unicast-routing
Router(config)#interface interface
Router(config-if)#ipv6 address ipv6-address/prefix-length
Router(config-if)#no shutdown
```

Problems with EUI-64
It has been pointed out that a user cannot connect anonymously to any network if someone knows the EUI-64 interface identifier of that device, which could be exploited such as websites and apps associating different IPv6 addresses to a particular device or user (whether malicious, monitoring, or benign). It is recommended that if the host OS or router does not support autoconfig with Random Interface Identifiers, that static IPv6 address should be used. (MS Windows generates by default a random interface ID for SLAAC)

Best answer to how to make clients go random instead of using EUI-64:
https://superuser.com/questions/243669/how-to-avoid-exposing-my-mac-address-when-using-ipv6

Stateless and Stateful DHCPv6 in Cisco IOS
DHCPv6 Stateless mode
- provides network info not in an RA, (no IPv6 address since already provided by SLAAC).
- DNS domain name and server(s), other DHCP options.

```
ipv6 unicast-routing
ipv6 dhcp pool IPV6_DHCPPOOL
address prefix 2001:db8:5:10::/64
domain-name cisco.com
dns-server 2001:db8:6:6::1
interface Vlan20
description IPv6-DHCP-Stateless
ip address 192.168.20.1 255.255.255.0
ipv6 nd other-config-flag
ipv6 dhcp server IPV6_DHCPPOOL
ipv6 address 2001:DB8:0:20::1/64
```

DHCPv6 Stateful aka managed mode assigns unique addresses instead of the client generating one

```
ipv6 unicast-routing
ipv6 dhcp pool IPV6_DHCPPOOL
address prefix 2001:db8:5:10::/64
domain-name cisco.com
dns-server 2001:db8:6:6::1
interface Vlan20
description IPv6-DHCP-Stateful
ip address 192.168.20.1 255.255.255.0
ipv6 address 2001:DB8:0:20::1/64
ipv6 nd prefix 2001:DB8:0:20::/64 no-advertise
ipv6 nd managed-config-flag
ipv6 nd other-config-flag
ipv6 dhcp server IPV6_DHCPPOOL
```

This interface configuration is for a Cisco IOS IPv6 router implementing stateful DHCPv6 on an external DHCP server:

```
ipv6 unicast-routing
domain-name cisco.com
dns-server 2001:db8:6:6::1
interface Vlan20
description IPv6-DHCP-Stateful
ip address 192.168.20.1 255.255.255.0
ipv6 address 2001:DB8:0:20::1/64
ipv6 nd prefix 2001:DB8:0:20::/64 no-advertise
ipv6 nd managed-config-flag
ipv6 nd other-config-flag
ipv6 dhcp_relay destination 2001:DB8:0:20::2
```

Cisco IOS Routing in IPv6

Static routing in IPv6

```
ipv6 unicast-routing
interface serial0/0/0
  ipv6 address 2001:5432:1111:4::1/64
interface serial0/0/1
  ipv6 address 2001:db8:1111:5::1/64
interface gigabitethernet0/0
  ipv6 address 2001:db8:1111:1::1/64
```

Static IPv6 with Next-Hop Address

```
!First command is on R1, listing R2's global unicast
R1(config)#ipv6 route 2001:db8:1111:2::/64 2001:db8:1111:4::2
!This command is on R2, listed R1's global unicast
R2(config)#ipv6 route 2001:db8:1111:1::/64 2001:db8:1111:4::1
!Verify routes with show ipv6 route static
```

Default Route

```
B1(config)#ipv6 route ::0 S0/0/1
```

```
show ipv6 route
show ipv6 route static
show ipv6 route local
```

OSPFv3 Routing - area goes on interfaces

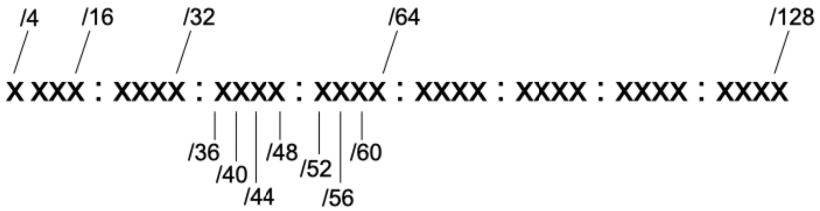
```
ipv6 unicast-routing
interface serial0/0/0
  no ip address
  ipv6 address 2001:abcd:1234:4::1/64
interface s0/0/1
  no ip address
  ipv6 address 2001: abcd 1234:5::1/64
interface GigabitEthernet0/0
  no ip address
  ipv6 address 2001:abcd:1234:5::1/64
ipv6 router ospf 1
    router-id 1.1.1.1
interface s0/0/0
    ipv6 ospf 1 area 0
int gi0/0
    ipv6 ospf 1 area 0

ipv6 unicast-routing
ipv6 router ospf 2
  router-id 2.2.2.2
int s0/0/1
  ipv6 address 2001: abcd:1234:4::2
  ipv6 ospf 2 area 0
int gi0/0
  ipv6 address 2001:abcd:1234:2::2
  ipv6 ospf 2 area 0

show ipv6 ospf
show ipv6 protocols
show ipv6 ospf interface
show ipv6 ospf interface brief
show ipv6 ospf neighbor
show ipv6 ospf database
show ipv6 route ospf
```

Subnetting IPv6

In IPv4, subnetting is done to get more addresses for hosts,optimize network distribution to help with that.
In IPv6 we have tons of addresses, so it becomes more of an issue of things like network organization.
Bulk blocks of IPv6 space purchased from RIRs are bought in lengths of /32 - /48 that can be subnetted into multiple /64 subnets. Subnets sized over /64 (like /84 or /96) are not advised since IPv6 features like addressing autoconfiguration rely on being used in a /64 subnet.

Prefix	Appearance	# of hosts	Number of /64s
/32	xxxx:xxxx::	2^{96}	4294967296
/36	xxxx:xxxx:x::	2^{92}	268435456
/40	xxxx:xxxx:xx::	2^{88}	16777216
/44	xxxx:xxxx:xxx::	2^{84}	1048576
/48	xxxx:xxxx:xxxx::	2^{80}	65536
/52	xxxx:xxxx:xxxx:x::	2^{76}	4096
/56	xxxx:xxxx:xxxx:xx::	2^{72}	256
/60	xxxx:xxxx:xxxx:xxx::	2^{68}	16
/64	xxxx:xxxx:xxxx:xxxx::	2^{64}	1

A /64 holds 18,446,744,073,709,551,616 addresses



Switchports for Trunking and Access; related display commands

switchport mode {access | trunk}
switchport access vlan vlan-number -- defines the VLAN interface resides in.
switchport trunk encapsulation [dot1q | isl] -- almost always use dot1q - you may have to do this first.
switchport mode dynamic auto | dynamic desirable
    dynamic auto - becomes a trunk if the neighboring interface is set to trunk or desirable- not if auto!
    dynamic desireable - becomes a trunk if neighboring interface is set to ANY trunk mode *(default!)*
switchport nonegotiate - prevents generating DTP frames, or converting dynamically to anything
switchport trunk allowed vlan 4,6,12,15 --this should eliminate vlans not specifically listed
switchport trunk allowed vlan [remove 4-8 | all | none]
no switchport trunk native vlan
no switchport trunk vlan 4
switchport trunk pruning vlan {add | except | none | remove} vlan-list Specify VLANs eligible for pruning

The proper way to remove VLANs from a switch
If you delete a VLAN from a switch with "no vlan 30", it won't make it disappear from the interface configs that it was previously added to- it will remain in the swtchport configurations! (and cause confusion) First, run on those interfaces "switchport trunk allowed vlan remove 30' or "switchport trunk allowed vlan 10,20,40,50' Then run "no vlan 30" on the switch to remove it.

show interfaces [type,#] switchport - settings, status, trunking, access/voice/native VLAN
show interfaces [type,#] trunk - lists info on trunks (or the specific trunk) and the VLANs
show vlan brief, show vlan - each VLAN and all assigned interfaces, but no trunks!
show vlan [vlan] - access and trunk ports in the VLAN.
show vtp status - VTP mode, configuration and status info

Expected Trunking Operational Mode Based on the Configured Administrative Modes w/ DTP

Administrative Mode	***Access***	***Dynamic Auto***	***Trunk***	***Dynamic Desirable***
access	Access	Access	Access	Access
dynamic auto	Access	Access	Trunk	Trunk
trunk	Access	Trunk	Trunk	Trunk
dynamic desirable	Access	Trunk	Trunk	Trunk

Port Security
As soon as you enable port-security, it defaults to violation shutdown and a maximum of 1 MAC address.
switchport port-security
switchport port-security violation restrict
switchport port-security mac-address aa.bb.cc.dd.ee.ff [sticky, often used with maximum]
switchport port-security maximum value
    the max number of MAC addresses that can be assigned - default is one.
    A max value can also be set if it's a switch connected - recieving frames for multiple MACs
switchport port-security violation {protect | restrict | shutdown}
        protect: unauthorized frames would just be dropped
        restrict: authorized frames would be dropped and violations count toggled
        shutdown: disable the interface (err-disabled - this is the default action)
show port-security interface

To open an interface shut down with port-security, you first issue "shutdown", then "no shutdown"

Inter-VLAN Routing (IVR) and Switched Virtual Interfaces (SVIs)
VLANs can't bridge without a router or Inter-VLAN Routing (IVR). Implementing trunking and Inter-VLAN routing on a layer 3 switch uses Switched Virtual Interfaces (SVIs)

If each router interface is plugged into an access link, each of the routers' interfaces would be the default gateway address for each host in each respective VLAN. IVR in the "router on a stick" (ROAS) implementation puts VLANs into a trunk to a layer 3 device. which performs the routing on logical interfaces and send back out the trunk to the switch on the proper VLAN. This can be done without the external router with a layer 3 switch, which seems a bit more efficient, and relying on the one external router creates a potential bottleneck, as well as a single point of failure.

Basic VLAN Management

Best practice dictates to always have separate voice VLANs, data VLANs, management VLAN, native VLAN, black hole VLAN, and the control plane VLAN (VLAN1). First, we need to discus what each of these are.

"Default" VLAN and the Control Plane

The default VLAN is simply the VLAN that all the ports on a switch are members of when a switch is reset to factory defaults. All switch ports are members of the default VLAN after the initial boot of the switch. Default really means exactly that and nothing more- factory defaults.

The default VLAN for Cisco switches is VLAN 1 and you cannot rename or delete it. However, since it defaults to being both Management and Native VLAN, best practices recommend moving those roles to other separate independent VLANs. After doing so, applying shutdown will prevent all data traffic from VLAN 1, so only control protocols are permitted on VLAN 1 (DTP, VTP ,STP BPDU's, PAgP, LACP, CDP, etc) Even if you prune VLAN 1 from trunks, these protocols always use VLAN 1 for controls communication.
Consider VLAN 1 to serve as a conduit only for Layer 2 control plane traffic, supporting no other traffic.

Management VLAN

A management VLAN is defined to access the management capabilities of a switch via HTTP, Telnet, SSH, but also includes SNMP and Syslog. It should be obvious why it's segregation is important security-wise

Native VLAN - Untagged Frames

Native VLAN is an 802.1q trunk concept for how to transport untagged traffic, and also serves as a common identifier on opposing ends of a trunk link. Traffic from access ports unassigned to a VLAN will be untagged, as well as some legacy LAN traffic. It should be separate and distinct from all other VLANs defined in the switched LAN

By default the native VLAN is always untagged, however there is a command available on some switches where you can tell the switch to tag all VLANs including the native

- Can be manually set on either end of the trunk, using the **switchport trunk native vlan** vlan-id
- If native VLANs differ on either end, it will accidentally cause frames to leave one VLAN and enter another.
  So the number of Native VLAN's can be equal to the number of trunk ports if you have a REALLY messed up network!
- Native has to do with the trunk itself, not the switch. We can only configure one native vlan per port. However, a switch with multiple ports, could have different vlans specified as the native vlan for that port. In other words, trunkport 2 could have a native vlan of 20, and trunkport 3 could have a native vlan of 30 (if you choose to be really complicated in your design). The switches on either end of the trunk just need to agree so strange things don't happen. [This can get really weird: imagine mismatched native vlans on each end of a trunk. In that case, the native vlan on one side becomes part of the same broadcast domain of a differently labeled vlan of the other end]

Consider: Switch A is connected to Switch B with a trunk. SA tags its native VLAN while SB doesn't, although it indeed uses that same VLAN number. Frames on the that vlan would flow properly from SA to SB, but not from SB to SA. While SB does not expect its traffic destined for its native vlan to be tagged, it will not reject it. Since SA is configured to expect traffic received for its native vlan to be tagged, it will discard the incoming untagged traffic. This is not quite a native vlan mismatch, but traffic will only flow in one direction.

The definition of a native vlan in 802.1q is indeed an untagged vlan. There are some exploits that can take advantage of this by stacking two sets of tags. If a user builds a frame that has an outer dot1q tag for a known native vlan and an inner tag of a vlan he wishes to attack, the first tag will be removed when the frame traverses the first trunk. The next trunk that is encountered will put the frame on the vlan that is the attack destination. As a result, there is a new command introduced to tag all frames on a trunk. This global command is "vlan tag dot1q native"

interface fa0/1
 switchport mode access
 switchport access vlan 10
 switchport voice vlan 20
 switchport trunk native vlan 30
Here PVID is 10, as untagged frames will get into VLAN 10.

interface fa0/2
 switchport mode trunk
 switchport access vlan 10
 switchport voice vlan 20
 switchport trunk native vlan 30
Here PVID is 30, as untagged frames will get into VLAN 30

Consider having a dedicated native vlan-id used on all trunks and never used on access ports to defeat double VLAN hopping attacks. On trunks, using a native vlan that is then not used can provide some L2 security advantages this makes all traffic on the trunk ports tagged. Another advantage is there is no concern about native VLAN mismatch if untagged frames are not allowed.

```
interface GigabitEthernet1/0/23
switchport mode trunk
switchport trunk encapsulation dot1q
switchport trunk native vlan 800
switchport trunk allowed vlan 252
```

Another way is to create (for example) VLAN 100 as the native vlan. then shut it down and make all trunk ports native vlan 100.

Finally, **vlan dot1q tag native** is a global command to tag native VLAN traffic, and admit only 802.1Q tagged frames on 802.1Q trunks, dropping any untagged traffic, including untagged traffic in the native VLAN.

Black Hole VLAN - Suspended Ports
A black hole (AKA parking or holding) VLAN is defined to assign all unused ports to it so that any device traffic connecting is not allowed on trunk links, thus preventing communicating beyond the switch. It is an extra way of ordering a port inoperable. The **state suspend** command in VLAN configuration mode will cause all received frames to be dropped. Syntax - state {active | suspend}

```
S1(config)#vlan 10
S1(config-vlan)#name Data
S1(config-vlan)#vlan 20
S1(config-vlan)#name Voice
S1(config-vlan)# vlan 30
S1(config-vlan)# name Native
S1(config-vlan)#vlan 99
S1(config-vlan)#name Management
S1(config-vlan)#vlan 5
S1(config-vlan)#name Suspended
S1(config-vlan)#state suspend
S1(config-vlan)#exit
S1(config)#interface vlan 99
S1(config-if)#ip add 192.168.128.10 255.255.255.0
S1(config-if)#no shut
S1(config-if)#exit
S1(config)#ip default-gateway  192.168.128.1
S1(config)#interface range  fa0/5-24
S1(config-if-range)#switchport mode access
S1(config-if-range)#switchport access vlan 5
S1(config-if-range)#shutdown
S1(config)#interface  fa0/1
S1(config-if)#switchport mode trunk
S1(config-if)#switchport trunk native vlan 30
S1(config-if)#no shutdown
S1(config)#interface range  fa0/2-4
S1(config-if-range)#switchport mode access
S1(config-if-range)#switchport access vlan 10
S1(config-if-range)#switchport voice vlan 20
S1(config-if-range)#no shutdown
```

VLAN Trunking Protocol (VTP)

Designed as a method to manage VLANs across a large numbers of switches: addition, deletion, and renaming from a central point of control, and all switches participating can use any related VLANs. Add a VLAN on the server and it gets set up on others. Not used as much in modern networks, Cisco says best practice is having switches in "off" or "transparent" modes; you may never see it but may find it and want to disable or otherwise manage it.

A switch can belong to just one management domain, and there is no communication between domains.
VTP advertisements contain info about the domain itself (including revision number), it's VLANs and their info.

VTP Modes: vtp mode {server | client | transparent | off}

Server mode (is default)

- full control over management of it's domain. Each domain should have at least one server
- advertises updates to other switches in the domain, receives info to synchronize domain members
- The first server defined in a network defines the domain that will be used by future VTP servers and clients.
- Multiple VTP servers can coexist in a domain, and is recommended for redundancy.
- There is no election for primary or secondary server
- If one server is configured with a new VLAN or VTP parameter, other servers synchronize just as any client

Client mode:

- listen to VTP advertisements from other switches and modify configurations accordingly.
- forward/relay VTP messages out trunk links to neighboring switches in the domain

Transparent mode:

- do not participate in VTP, does not advertise or synchronize its VLAN database with received advertisements.
- can create and delete VLANs that are local only to itself without changes being advertised
- Works only as a relaying member.
- [VTP v1, transparent mode does not relay VTP info unless its domain and VTP version numbers match]
- [VTP v2 and 3, transparent mode does forward VTP advertisements regardless of the VTP domain name]

Off mode simply disables all VTP activity on the switch.

Revision Numbers - watch out when adding hardware!
VTP uses a revision number to track the most recent information.
Starts of at 0 and is incremented by the VTP server with each change in domain info it will advertise
An advertisement with a greater revision number than before says it has new and updated information.
That advertisement is stored and overwrites any previously stored VLAN information.

Important!

VTP VLAN data is saved in vlan.dat file in flash memory is retained even when the switch power is off.
It ensures a switch can recover last known VTP/VLAN configuration from its VTP database after it reboots. Even a device previously configured in client mode will send a summary advertisement using info from it's vlan.dat after powering up and discovering it has a higher revision number.
- Care must be taken to not plug in a VTP enabled switch containing a higher revision number, as it will obliterate existing VTP database information throughout the domain it matches to!

- Always force revision number 0 before being attaching *EVEN if previously configured as a only a VTP client*
- On a new device, set to VTP transparent and then later change back to server- or change the VTP domain to a bogus name
- For critical portions of your network, consider using VTP transparent or off mode to prevent synchronization problems
- Eliminate the chance for duplicate, overlapping VLANs in a large network with transparent mode. For example, two administrators might configure VLANs on switches in their respective areas but use the same VLAN identification or VLAN number. They could overlap if both administrators advertised them using VTP servers

Domains use unsecure advertisements (default), but a password can be required to participants (secure mode)

Cisco Catalyst switches - default is VTP mode server - Don't forget to disable!
It turns out that by default Cisco switches operate in VTP server mode for the management domain NULL (a blank string), no password. If it hears a VTP summary advertisement it automatically learns the VTP domain name, VLANs, revision number. This makes it easy to bring up a new switch in an existing VTP domain.
Be sure to remember that the new switch stays in VTP server mode until you change it..

show vtp status

```
Switch# show vtp status
VTP Version capable             : 1 to 3
VTP version running             : 1
VTP Domain Name                 :
VTP Pruning Mode                : Disabled
VTP Traps Generation            : Disabled
Device ID                       : aca0.164f.3f80
Configuration last modified by 0.0.0.0 at 0-0-00 00:00:00
Local updater ID is 0.0.0.0 (no valid interface found)
Feature VLAN:
--------------
VTP Operating Mode                : Server
Maximum VLANs supported locally   : 1005
Number of existing VLANs          : 5
Configuration Revision            : 0
MD5 digest                        : 0x57 0xCD 0x40 0x65 0x63 0x59 0x47 0xBD
                                    0x56 0x9D 0x4A 0x3E 0xA5 0x69 0x35 0xBC
```

Advertisement types

Summary advertisements

- VTP domain servers send every 300 seconds and every time a VLAN database change occurs
- VTP version, domain name, revision number, MD5 hash, and number of subset advertisements to follow.
- When config changes, one or more subset advertisements with more details are sent afterwards

Subset advertisements

- list specific changes: add/ delete/ suspend/ activate a VLAN, changing of VLAN name, number, MTU.
- Even change in VLAN type (such as Ethernet or Token Ring), or security association identifier (SAID- 802.10)
- each VLAN gets it's own individual sequential subset advertisement per change, as needed.

VTP advertisements are multicast 01-00-0C-CC-CCCC and an SNAP type value of 0×2003.

Cisco switches default to v1. Versions are not fully backward compatible with each other
Versions 1 and 2 support VLAN numbers 1 to 1005. Only VTP v3 supports extended VLAN range 1-4094.

VTP v3 Features

Extended VLAN range VLANs 1 through 4094 can be advertised throughout a VTPv3 domain
Enhanced authentication - the password can be hidden (only a hash of the password is saved in the running configuration) or secret (the password is saved in the running configuration).
Database propagation - databases other than VTP can be advertised
By default, all VTPv3 switches operate as secondary servers and can send updates throughout the domain.
A primary server is only needed to take control of a domain.
Per-port VTP - VTPv3 can be enabled on a per-trunk port basis, rather than a switch as a whole

```
Switch(config)# vtp version 3
Switch(config)# vtp domain MyCompany
Switch(config)# vtp mode server
Switch(config)# vtp password bigsecret
```

VLAN Pruning

VTP pruning makes more efficient use of trunk bandwidth by reducing unnecessary flooded traffic.
Broadcast, multicast, and unknown unicast frames on a VLAN are forwarded over a trunk link only if the switch on the receiving end of the trunk has ports in that VLAN.
When a switch has an active port associated with a VLAN, the switch advertises that to its neighbor switches.
The neighbors then to decide whether flooded traffic from a VLAN should be allowed on certain trunk links.

Even when VTP pruning has determined that a VLAN is not needed on a trunk, an instance of the Spanning Tree will run for every VLAN that is **allowed** on the trunk link. To reduce the number of STP instances, you should manually “prune” unneeded VLANs from the trunk and allow only the needed ones. Use the **switchport trunk allowed vlan** command to identify the VLANs that should be added or removed from a trunk.

VTP pruning is disabled by default. To enable pruning, use **vtp pruning**
On a VTP server, it says pruning needs to be enabled for the entire domain (all will also enable pruning.
When pruning is enabled, all general-purpose VLANs become eligible for pruning on all trunk links, if needed. However, you can modify the default list of pruning eligibility with the following interface-configuration command:

switchport trunk pruning vlan {{{add | except | remove} vlan-list} | none}

vlan-list	List of eligible VLAN numbers (2-1001), separate by commas or dashes, no spaces.
add	Will add to the already configured list
except	All VLANs are eligible except for the VLAN numbers
remove	VLAN numbers to remove from the already configured list
none	No VLAN will be eligible for pruning.

Obviously, VTP pruning has no effect on switches in the VTP transparent mode.
Those switches must be configured manually to “prune” VLANs from trunk links (same command)
VLAN 1 is never eligible for pruning, same with 1002-1005 (reserved for Token Ring and FDDI VLANs)

Troubleshooting VTP - Not updating information from a VTP server?
- Is the switch in VTP transparent mode
- If a VTP client, there might not be an VTP server. In this case, just make it a VTP server itself.
- Is the link to the VTP server a trunk? VTP advertisements are sent only over trunks.
- Is the VTP domain name is configured to match the one on the VTP server.
- Check if the VTP version is compatible matches the VTP domain.
- Does the VTP domain use a password? If the server doesn't, make sure the password is disabled or cleared.

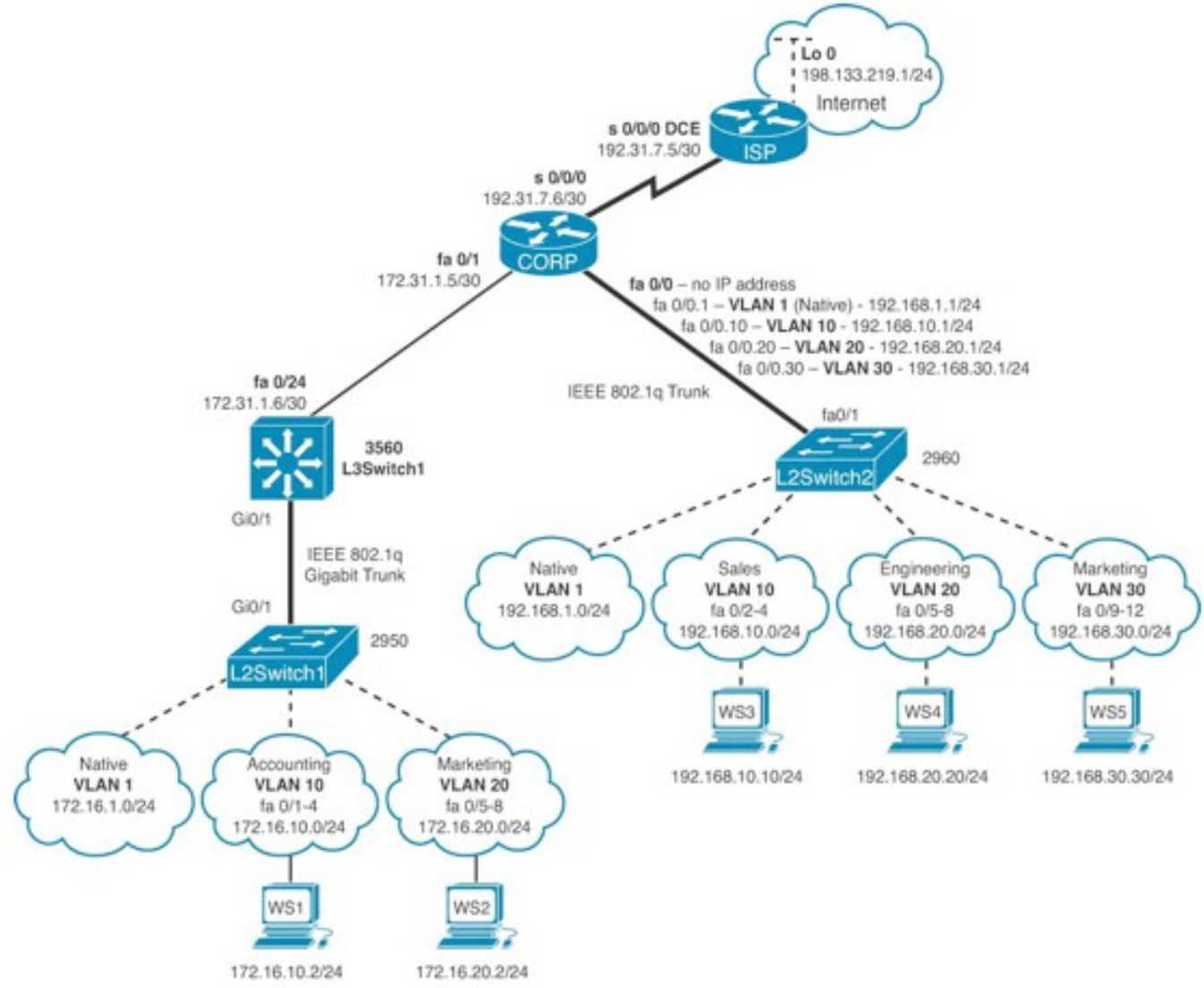
VTP Troubleshooting Commands

show vtp status	Current VTP parameters, incl. the last VTP server
show vlan brief	Displays defined VLANs
show interface type member/module/number switchport	Displays trunk status, including pruning eligibility
show interface type member/module/number pruning	Displays VTP pruning state

VTP Configuration Commands

vtp domain domain-name	Define the VTP domain.
vtp mode {server | client | transparent | off}	Set the VTP mode.
vtp password password [hidden | secret]	Define an optional VTP password.
vtp version {1 | 2 | 3}	Configure VTP version.
vtp pruning	Enable VTP pruning.
switchport trunk pruning vlan {add | except | none | remove} vlan-list	Specify VLANs eligible for pruning

Configuration Example- Inter-VLAN Communication - pp113, Empson



ISP Router
Router>enable.
Router>#configure terminal
Router(config)#hostname ISP

ISP(config)#interface loopback 0
ISP(config-if)#description simulated address for remote website
ISP(config-if)#ip address 198.133.219.1 255.255.255.0
ISP(config-if)#interface serial 0/0/0
ISP(config-if)#description WAN link to the Corporate Router
ISP(config-if)#ip address 192.31.7.5 255.255.255.252
ISP(config-if)#clock rate 56000 *--DCE cable so don't forget the clock rate*
ISP(config-if)#no shutdown

ISP(config-if)#router eigrp 10 *--- initialize EIGRP for AS*
ISP(config-router)#network 198.133.219.0
ISP(config-router)#network 192.31.7.0
ISP(config-router)#no auto-summary
ISP#copy running-config startup-config

CORP Router
Router>enable.
Router>#configure terminal
Router(config)#hostname CORP

```
CORP(config)#interface serial 0/0/0
CORP(config-if)#description link to ISP Router
CORP(config-if)#ip address 192.31.7.6 255.255.255.252
CORP(config-if)#no shutdown
CORP(config)#interface fastethernet 0/1
CORP(config-if)#description link to 3560 Switch
CORP(config-if)#ip address 172.31.1.5 255.255.255.252
CORP(config-if)#no shutdown
CORP(config)#interface fastethernet 0/0
CORP(config-if)#description link to L2Switch2
CORP(config-if)#duplex full  ---full-duplex to ensure trunking will take effect with L2Switch2.
CORP(config-if)#no shutdown
CORP(config-if)#interface fastethernet 0/0.1  ---Creates a virtual subinterface
CORP(config-subif)#description Management VLAN 1 – Native VLAN
CORP(config-subif)#encapsulation dot1q 1 native
CORP(config-subif)#ip address 192.168.1.1 255.255.255.0
CORP(config-subif)#interface fastethernet 0/0.10
CORP(config-subif)#description Sales VLAN 10
CORP(config-subif)#encapsulation dot1q 10
CORP(config-subif)#ip address 192.168.10.1 255.255.255.0
CORP(config-subif)#interface fastethernet 0/0.20
CORP(config-subif)#description Engineering VLAN 20
CORP(config-subif)#encapsulation dot1q 20
CORP(config-subif)#ip address 192.168.20.1 255.255.255.0
CORP(config-subif)#interface fastethernet 0/0.30
CORP(config-subif)#description Marketing VLAN 30
CORP(config-subif)#encapsulation dot1q 30
CORP(config-subif)#ip add 192.168.30.1 255.255.255.0
CORP(config-subif)#exit
CORP(config-if)#exit
CORP(config)#router eigrp 10
CORP(config-router)#network 192.168.1.0   ---Advertise our networks
CORP(config-router)#network 192.168.10.0
CORP(config-router)#network 192.168.20.0
CORP(config-router)#network 192.168.30.0
CORP(config-router)#network 172.31.0.0
CORP(config-router)#network 192.31.7.0
CORP(config-router)#no auto-summary
```

L2Switch2 (Catalyst 2960)

```
Switch(config)#hostname L2Switch2
L2Switch2(config)#vlan 10
L2Switch2(config-vlan)#name Sales  ---Assign names to the VLANs
L2Switch2(config)#vlan 20
L2Switch2(config-vlan)#name Engineering
L2Switch2(config-vlan)#vlan 30
L2Switch2(config-vlan)#name Marketing
L2Switch2(config)#interface range fastethernet 0/2 - 4   ---Assign access ports to the VLANs
L2Switch2(config-if-range)#switchport mode access
L2Switch2(config-if-range)#switchport access vlan 10
L2Switch2(config-if-range)#interface range fastethernet 0/5 - 8
L2Switch2(config-if-range)#switchport mode access
L2Switch2(config-if-range)#switchport access vlan 20
L2Switch2(config-if-range)#interface range fastethernet 0/9 - 12
L2Switch2(config-if-range)#switchport mode access
L2Switch2(config-if-range)#switchport access vlan 30
L2Switch2(config-if-range)#exit
L2Switch2(config)#interface fastethernet 0/1
L2Switch2(config-if)#description Trunk Link to CORP Router
L2Switch2(config-if)#switchport mode trunk
L2Switch2(config-if)#exit
L2Switch2(config)#interface vlan 1  ---Creates virtual interface for VLAN 1
L2Switch2(config-if)#ip address 192.168.1.2 255.255.255.0
L2Switch2(config-if)#no shutdown
L2Switch2(config-if)#exit
L2Switch2(config)#ip default-gateway 192.168.1.1  ---Assigns the default gateway address.
L2Switch2(config)#exit
L2Switch2#copy running-config startup-config
```

L3Switch1 (Catalyst 3560)

```
Switch(config)#hostname L3Switch1
L3Switch1(config)#vtp mode server  ---Changes the switch to VTP server mode.
L3Switch1(config)#vtp domain testdomain  ---Configures the VTP domain name to testdomain.
L3Switch1(config)#vlan 10
L3Switch1(config-vlan)#name Accounting
L3Switch1(config-vlan)#exit
L3Switch1(config)#vlan 20
L3Switch1(config-vlan)#name Marketing
L3Switch1(config-vlan)#exit
L3Switch1(config)#interface gigabitethernet 0/1
L3Switch1(config-if)#description Gigabit Trunk to L2Switch1
L3Switch1(config-if)#switchport trunk encapsulation dot1q
L3Switch1(config-if)#switchport mode trunk
L3Switch1(config-if)#exit
L3Switch1(config)#ip routing  ---Enables IP routing on this device.
L3Switch1(config)#interface vlan 1  ---Creates a virtual interface for VLAN 1
L3Switch1(config-if)#ip address 172.16.1.1 255.255.255.0
L3Switch1(config-if)#no shutdown
L3Switch1(config-if)#interface vlan 10
L3Switch1(config-if)#ip address 172.16.10.1 255.255.255.0
L3Switch1(config-if)#no shutdown
L3Switch1(config-if)#interface vlan 20
L3Switch1(config-if)#ip address 172.16.20.1 255.255.255.0
L3Switch1(config-if)#no shutdown
L3Switch1(config-if)#exit
L3Switch1(config)#interface fastethernet 0/24
L3Switch1(config-if)#description Link to CORP
L3Switch1(config-if)#no switchport  ---Creates a Layer 3 port on the switch.
L3Switch1(config-if)#ip address 172.31.1.6 255.255.255.252
L3Switch1(config-if)#exit
L3Switch1(config)#router eigrp 10
L3Switch1(config-router)#network 172.16.0.0
L3Switch1(config-router)#network 172.31.0.0
L3Switch1(config-router)#no auto-summary
L3Switch1(config-router)#exit
L3Switch1(config)#exit
L3Switch1#copy running-config startup-config
```

L2Switch1 (Catalyst 2960)

```
Switch(config)#hostname L2Switch1
L2Switch1(config)#vtp domain testdomain  ---VTP domain name to testdomain.
L2Switch1(config)#vtp mode client  ---Changes the switch to VTP client mode.
L2Switch1(config)#interface range fastethernet 0/1 - 4
L2Switch1(config-if-range)#switchport mode access
L2Switch1(config-if-range)#switchport access vlan 10
L2Switch1(config-if-range)#interface range fastethernet 0/5 - 8
L2Switch1(config-if-range)#switchport mode access
L2Switch1(config-if-range)#switchport access vlan 20
L2Switch1(config-if-range)#exit
L2Switch1(config)#interface gigabitethernet 0/1
L2Switch1(config-if)#switchport mode trunk
L2Switch1(config-if)#exit
L2Switch1(config)#interface vlan 1
L2Switch1(config-if)#ip address 172.16.1.2 255.255.255.0  --VLAN1 on diagram is 172.16.1.0/24 - typo?
L2Switch1(config-if)#no shutdown
L2Switch1(config-if)#exit
L2Switch1(config)#ip default-gateway 172.16.1.1
L2Switch1(config)#exit .
L2Switch1#copy running-config startup-config
```

Spanning Tree (802.1D)

Spanning Tree is a managed system where each port is in a forwarding or blocking state. Blocking state means the port doesn't process any frames except STP messages- the switch physically receives the frame on blocked port, but ignores it. If topology changes (a switch goes down), STP convergence updates port states accordingly. This system has a "root bridge" and uses bridge IDs for each switch containing a priority # and MAC address. Distance to the root switch/bridge are measured as a "root cost" from each switch and port, and governs the forwarding and blocking of ports, and ultimately, the paths Layer 2 stuff can take around the network.

The Root Bridge/ Root Switch and the System's Basics

- The root switch is always the designated switch on all directly connected segments
- All ports on the root switch are designated ports, in a forwarding state
- Nonroot switches have a root port (with lowest cost to the root switch); if directly connected, it's the root port.
- Each nonroot switch has at least one designated port (DP) to the other non-root switches.
- Similarly, a designated switch is chosen by lowest "root cost" among other paths to root, other non-root switches connect to the root through their neighboring designated switch.
- Throughout this system of non-root switches, up/up ints become either DPs or blocked
- The designated port's job is to forward STP messages (BDPUs) back and forth from the root

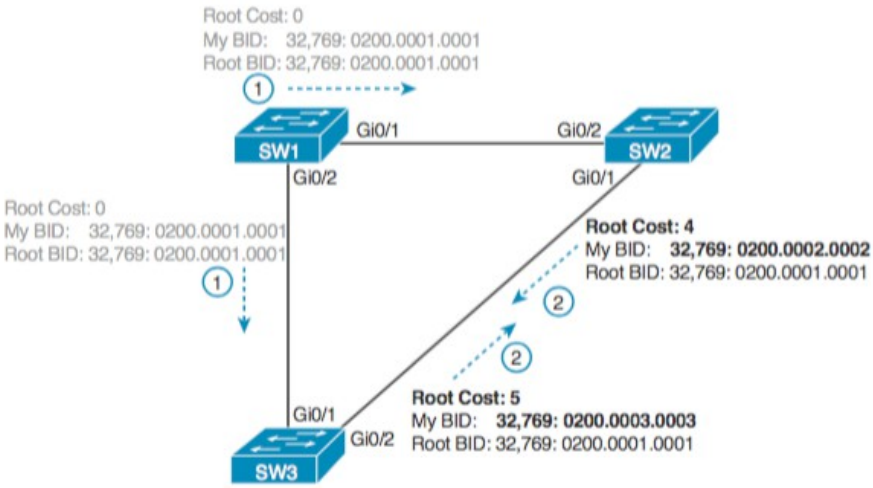
Electing the Root Switch [more explanation in Bridge ID (BID) section]

Root switch has the lowest value priority field in their BID

If SW1 Priority = 4096 and SW2 Priority = 8192, then SW1 is Root

If a priority tie occurs, switch with lowest MAC address field in BID is Root

If SW1 Pri= 4096 & MAC= 0200.xxxx.xxxx and SW2 Pri= 4096 & MAC= 0911.xxxx.xxxx, SW1 is Root



Bridge Protocol Data Units (BPDU) exchange info among switches

Hello messages are a BPDU containing:

- Root Bridge ID, Sender's Bridge ID and root cost
- Timer values on the root switch [hello, MaxAge, and forward delay timers]

Configuration BPDUs originate from Root Bridge. These are the majority of BPDUs on a healthy network

Topology Change Notification (TCN) BPDUs sent to the Root Bridge to alert that active topology has changed. (notification and acknowledgement subtypes)

Timers

Hello - 2 seconds default - Time period between recieving hellos created by the root

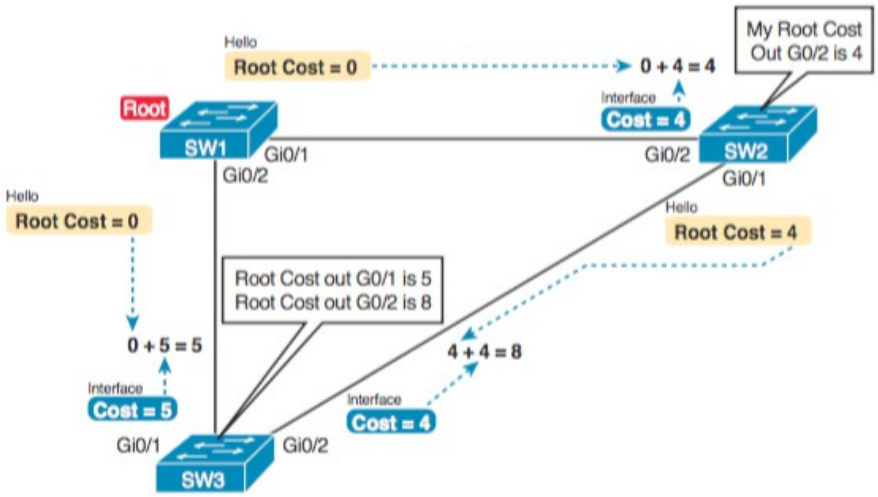
MaxAge- 10x hello timer (20 sec default) -How long switch waits after no hello BDPUs, before changing states

Forward delay- 15 secs - time interface goes from blocking to forwarding in each of the listening, learning states

Electing Designated Ports - the port with the lowest-cost hello on a LAN segment.

(Hello msgs have a cost of 0 when they leave the root switch) Cost by line type, 2nd # is "long path cost":

10 Mbps	100	2000000
100 Mbps	19	200000
1 Gbps	4	20000
10 Gbps	2	2000
100 Gbps		200



Both SW2 and SW3 list their respective cost to the root switch (cost 4 on SW2 and cost 5 on SW3). SW2 lists the lower cost, so SW2's Gi0/1 port is the designated port on that LAN segment. ***The order is lowest root cost, then lowest BID, then lowest neighbor port priority, then finally, lowest neighbor internal port number***

50-second convergence delay: STP has the interface in both *learning and listening states* for a time equal to the forward delay timer, which defaults to 15 seconds each. A convergence event that causes an interface to change from blocking to forwarding requires 30 sec to transition those PLUS the MaxAge to first move from blocking

Temporary states when going from blocking to forwarding:
- Listening- doesn't forward- does MAC table cleanup
- Learning- also doesn't forward- learn the MAC address of received frames

State	Forwards?	Learns MACs?	Transitory or Stable State?
Blocking	No	No	Stable
Listening	No	No	Transitory
Learning	No	Yes	Transitory
Forwarding	Yes	Yes	Stable
Disabled	No	No	Stable

Rapid STP (IEEE 802.1w - RSTP): (is backward-compatible w/ non-RSTP switches)
- Calls blocking "discarding", no listening state. Only has learning, forwarding or discarding states
- Hello time = 2 secs, Max Age = 6 secs (3 x hello, 3 missed BDPUs)
- improves convergence from about 50 seconds to about 10 seconds
- adds two port types allowed to immediately enter the forwarding state rather than passively wait for the network to converge. In addition to forwarding, root, designated, and disabled:
    - Alternate port - best alternate path to the root bridge if there is a failure on the designated port
    - Backup port. Applies to scenarios with a hub. Antiquated.

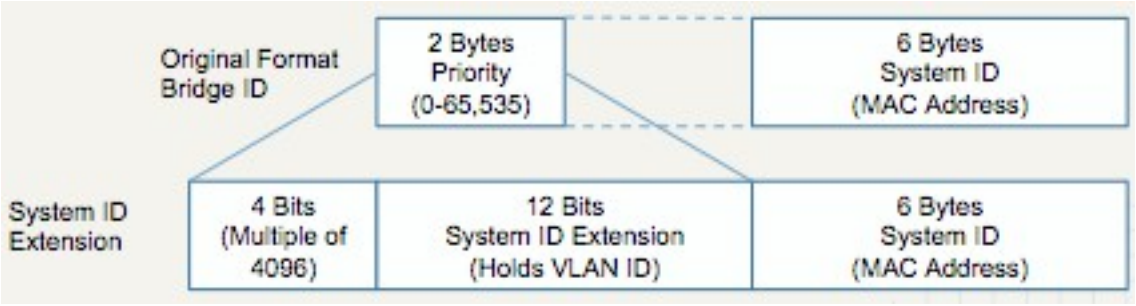
Check Before Connecting New HW
When plugged into the network, switches send out hello BPDUs listing their own BID as the root BID, then, if it hears a hello that lists a better (lower) BID, that switch stops advertising itself as root and starts forwarding the superior hello. This is how an unauthorized or improperly vetted device can screw up a STP topology- if one shows up with a lower BID, convergence spreads it as a new root switch and everything goes haywire.

PortFast
- Allows immediately change from blocking to forwarding. No listening or learning states
- For endpoint/edge devices (PCs) - NOT bridges/switches
- *Don't put on interfaces receiving BPDU (not another switch) - w/o BDPU Guard can create loops*
- If voice VLAN set, PortFast turns on automatically- will stay on even if voice VLAN is unset

BPDU Guard
- BDPU Guard disables a port if any BPDUs are received on it and prevents PortFast problems
- STP opens up the following security exposures:
    Attacker adds switch with low STP priority and becomes the Root Switch
    Attacker could plug into multiple ports/switches, become root, & forward (or TCPDump) LAN traffic
    Users can harm the LAN when they connect a non-STP switch (can be counted in elections, etc.)

The Bridge ID - { [Priority(4096-61440) + VLAN ID(1-4094)] + the MAC address}
STP Bridge ID (BID) is a unique 8-byte value- basically a priority field slapped onto a MAC address
- a 2-byte priority field The last 12 bits holds a VLAN ID 0-4096 (it's just basically a 16-bit number)
- a 6-byte system ID - the 48-bit MAC address

The 2-byte Priority field is separated into 2 parts
4-bit priority field; the part that can be changed with the "priority" directive
12-bit *system ID extension* - basically, just space for the VLAN ID)



When the bits are written out, those leading 4 bits reside starting at the 4096 bit (the 13th bit), so each number starting there would naturally be a multiple of 4096. Since VLAN IDs can range from 1 to 4094, it all makes perfect sense and looks a lot less complicated like this:

[Priority 4096-61440 + VLAN 1-4094] + tack on the MAC address.
Means, [0000 + 000000000000] + your MAC address
0000 - [111111111111] <---- 4095 in binary (enough bits reserved to plop a "priotrity" onto)
We configure the priority field below in square brackets:
[1000] - 000000000000 <---- Default base priority of 32768 in binary
[0110] - 000000000000 <---- Secondary root bridge designation number in binary
[????] - 000000000000 <-----you can set a multiple here for a priority (from 4096 x ? up to 61440)

The Easy Way
- spanning-tree vlan vlan-id **root primary** [auto-set this vlan's priority low enough to become root now]
Cisco switches use a default base priority of 32,768
If current root base priority > 24,576, make base priority 24,576.
If 24,576 or lower, sets base priority to highest multiple of 4096 that results in becoming root.
For our numbers below, 28,672 is 0110 000000000000

- spanning-tree vlan vlan-id **root secondary** [again, this auto-sets it for us]
A priority value worse than the primary switch but better than all the other switches.
Sets the switch's base priority to 28,672 *regardless* of the current root's priority value.
For our numbers below, 28,672 is 0111 000000000000

The Hard Way
It is much easier to just designate using root primary and secondary to avoid doing it yourself. If you insist on doing "spanning-tree vlan vlan-id priority x":

- spanning-tree vlan vlan-id **priority** x
A switch configured with VLANs 1 through 4, with a default base priority
of 32,768, has a default STP priority of 32,769 in VLAN 1, 32,770 in VLAN 2, 32,771 in VLAN 3, and so on. So, you can view the 16-bit priority as a base priority (as configured on the **spanning-tree vlan** vlan-id **priority** x command) plus the VLAN ID."
0001:000000000001 = 4097
0010:000000000001 = 8193
0100 = 16384, 1000 = 32768

Vlan 1 would be 32769, Vlan 2 32770, Vlan 10 32778, Vlan 100 32868
1111= 61,440 <--- if there are more than one bit turned on, things get weird. One trick is to pretend the VLAN-ID part is all 1's so it flattens out to multiples of 4096 only, when trying to calculate stuff)
Final word: If binary makes you uncomfortable, forget it. Use "root primary" and "root secondary"

Determining the Root Switch and the Root Port on Nonroot Switches

Use **show spanning-tree**, **show spanning-tree root** to rule out any switches that have an RP, because root switches do not have an RP. **show spanning-tree** identifies the local switch as root directly: "This switch is the root". In **show spanning-tree root**, the RP column is empty if the local switch is the root.

Using **show spanning-tree vlan** x on a few switches, and recording the root switch, RP, and DP ports can quickly show you most STP facts. Chase the RPs. If starting with SW1, and SW1's G0/1 is an RP, try the switch on the other end of SW1's G0/1 port.

STP Tiebreakers When Choosing the Root Port

The three tiebreakers are, in the order: lowest neighbor bridge ID, lowest neighbor port priority, finally, lowest neighbor internal port number. Only root paths that tie are considered when thinking about tiebreakers.

Below, SW3 is not root and that its two paths to reach the root tie with their root costs of 8. The first tiebreaker is the lowest neighbor's BID. SW1's BID value is lower than SW2's, so SW3 chooses its G0/1 interface as its RP.

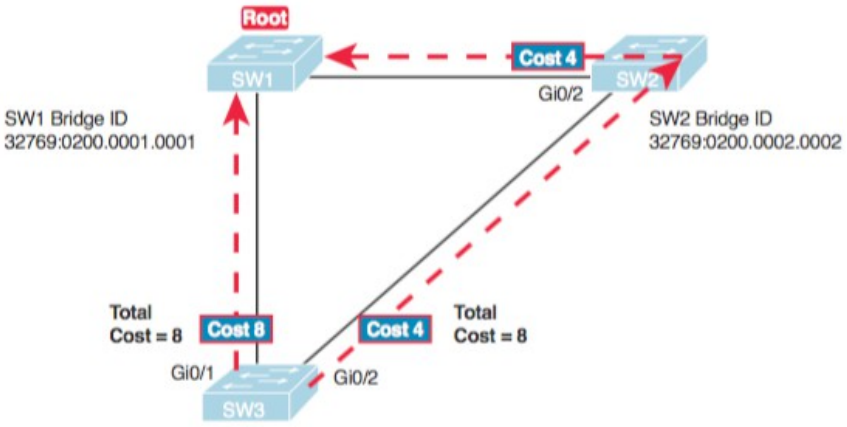


Figure 2-8 *SW3's Root Cost Calculation Ends in a Tie*

The last two RP tiebreakers come into play only when two switches connect to each other with multiple links, as shown in above. In that case, a switch receives hellos on more than one port from the same neighboring switch, so the BIDs tie.

So, SW2 becomes root, and SW1 needs to choose its RP. SW1's root cost over each path will tie at 19. SW2 sends hellos over each link to SW1, so SW1 cannot break the tie based on SW1's neighbor BID because both list SW2's BID. So, SW1 has to turn to the other two tiebreakers.

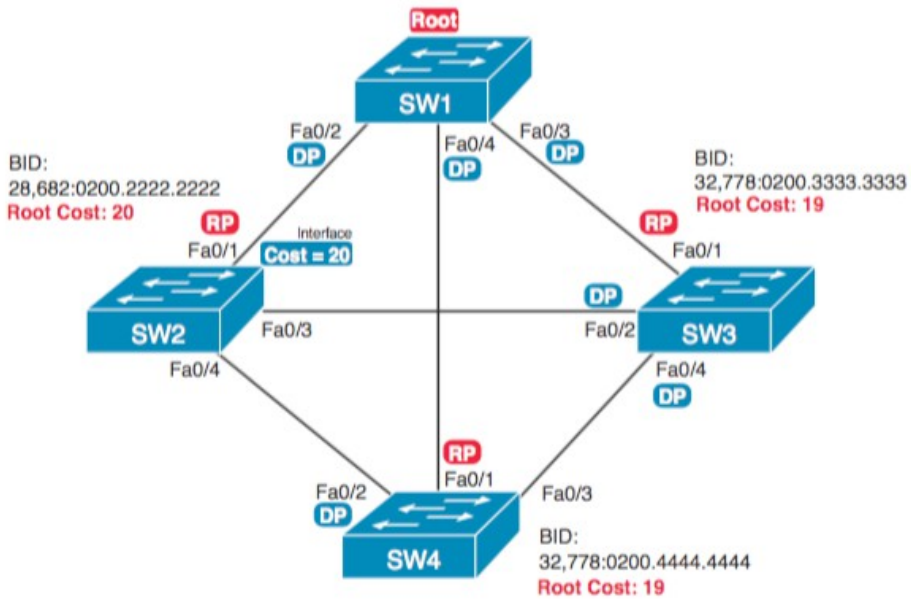
Figure 2-9 *Topology Required for the Last Two Tiebreakers for Root Port*

The next tiebreaker is configurable: the neighboring switch's port priority on each neighboring switch interface. Cisco switch ports default to a setting of 128, with a range of values from 0 through 255, with lower being better (as usual). Here, SW2's F0/16 was manually set with **spanning-tree vlan 10 port-priority 112**. SW1 learns that the neighbor has a port priority of 112 on the top link and 128 on the bottom, so SW1 uses its top (F0/14) interface as the root port.

If the port priority ties, which it often does due to the default values, STP relies on an internal port numbering on the neighbor. Cisco switches assign an internal integer to identify each interface on the switch. The nonroot looks for the neighbor's lowest internal port number (as listed in the hello messages) and chooses its RP based on the lower number.

With Fa0/1 having the lowest number, then Fa0/2, then Fa0/3, and so on- SW2's Fa0/16 would have a lower internal port number than Fa0/17; SW1 would learn those numbers in the hello; and SW1 would use its Fa0/14 port as its RP. (In real life, most engineers would put these two links into an EtherChannel)

This figure notes the root, RPs, and DPs and each switch's least cost to reach the root over its respective RP.



Remember the order of the criteria:
Lowest root cost, lowest BID, lowest neighbor port priority, lowest neighbor internal port number

Focus on the segments that connect the nonroot switches for a moment.
SW2–SW4 segment: SW4 wins because of its root cost of 19, compared to SW2's root cost of 20.
SW2-SW3 segment: SW3 wins because of its root cost of 19, compared to SW2's root cost of 20.
SW3-SW4 segment: SW3 and SW4 tie on root cost, both with 19. SW3 wins due to its better (lower) BID value.
SW2 loses and does not become DP on the links to SW3 and SW4 even though SW2 has the better (lower) BID value. The first DP criteria is the lowest root cost, and SW2's root cost happens to be higher than SW3's and SW4's.

---

show spanning-tree vlan Bridge and RootID info, timers; port roles for local ports, switch role for device.
show spanning-tree vlan 10 root - lists root's BID for each VLAN, local switch's root cost and root port
show spanning-tree vlan 10 bridge - lists the local switch's BID, MAC, timers
debug spanning-tree events

spanning-tree portfast - **spanning-tree bpduguard enable** (to be run on individual interfaces)
Default is no portfast and no BPDU Guard. Change the opposite to make all on the new default:
spanning-tree portfast default - **spanning-tree portfast bpduguard default** (Global config)
spanning-tree portfast disable - **spanning-tree bpduguard disable** (Run per interface config)

UplinkFast: pg 197 - UplinkFast: Access Layer Uplinks
BackboneFast pg 198 - BackboneFast: Redundant Backbone Paths

Commands for Spanning Tree - PortFast - BDPUGuard - UplinkFast - BackboneFast

spanning-tree vlan-id	Enable STP.
spanning-tree mode { **pvst | rapid-pvst | mst** }	Set the STP mode.
spanning-tree vlan vlan-number **root primary**	Changes this switch to the root.
spanning-tree vlan vlan-number **root secondar y**	Sets this switch's STP base
spanning-tree [**vlan** vlan-id] {**priority** priority}	Changes the bridge priority of this switch for this VLAN.
spanning-tree [**vlan** vlan-number] **cost** cost	Changes the STP cost .
spanning-tree [**vlan** vlan-number] **port-priority** priority	Changes STP port priority in VLAN (0 to 240, count by 16).
spanning-tree vlan vlan-id **priority** bridge-priority	Set bridge priority.
spanning-tree vlan vlan-id **root** {**primary** | **secondary**} [**diameter** diameter]	Set root bridge (macro).
spanning-tree [**vlan** vlan-id] **hello-time** seconds **spanning-tree** [**vlan** vlan-id] **forward-time** seconds **spanning-tree** [**vlan** vlan-id] **max-age** seconds	Set STP timers.
show spanning-tree	STP info all VLANs, ports summarized
show spanning-tree detail	STP info all VLANs, ports detailed.
show spanning-tree interface interface-id	Lists STP info for the specified port
show spanning-tree vlan vlan-id	Lists STP info for the specified VLAN
show spanning-tree [vlan vlan-id] **summary**	Show switch ports currently in each of the STP states.
show spanning-tree [**vlan** vlan-id] **root**	Show a VLAN's root bridge ID, root port, and root path cost.
show spanning-tree [**vlan** vlan-id] **bridge**	Show this switch's bridge ID and STP timers
show spanning-tree interface type number **portfast**	1-line status message about PortFast
channel-group channel-group-number **mode** {**auto** | **desirable** | **active** | **passive** | **on**}	Enables EtherChannel.
show etherchannel [chan-grp-#] {**brief** | **detail** | **port** | **port-channel** | **summary**}	Info on state of EtherChannels
debug spanning-tree events	Provide info messages about changes in the STP topology
spanning-tree portfast (disable)	Enables/disables PortFast.
spanning-tree bpduguard enable | disable	Enables/disables BPDU Guard
spanning-tree portfast default	Changes switch default for PortFast to enabled.
spanning-tree portfast bpduguard default	Changes switch default for BPDU Guard to enabled.
spanning-tree uplinkfast [**max-update-rate** pkts-per-sec]	Set UplinkFast on a switch.
spanning-tree backbonefast	Set BackboneFast on a switch.
show spanning-tree uplinkfast	Show the STP UplinkFast status.
show spanning-tree backbonefast	Show the STP BackboneFast status.
show spanning-tree summary	Display global BPDU Guard, BPDU filter, and Loop Guard states.
show spanning-tree interface type mod/num [**detail**]	Look for detailed reasons for inconsistencies.
show spanning-tree inconsistentports	List ports labeled in an inconsistent state.
show udld [type mod/num]	Display the UDLD status on one or all ports.
udld reset	Reenable ports that UDLD aggressive mode errdisabled.

Task	**Global Command Syntax**	**Interface Command Syntax**
Enable Root Guard.	--	**spanning- tree guard root**
Enable BPDU Guard.	**spanning-tree portfast bpduguard default**	**spanning- tree bpduguard enable**
Enable Loop Guard.	**spanning-tree loopguard default**	**spanning- tree guard loop**
Enable UDLD.	**udld {enable | aggressive | message time}**	**udld {enable | aggressive | disable}**
Enable BPDU filtering.	**spanning-tree bpdufilter default**	**spanning- tree bpdufilter enable**

Switch(config-if)**# spanning-tree portfast** Define an edge port
Switch(config-if)**# spanning-tree link-type point-to-point** Override a port type
Switch(config)**# spanning-tree mode mst** Enable MST on a switch
Switch(config)**# spanning-tree mst configuration** Enter MST configuration mode
Switch(config-mst)**# name** name Name the MST region.
Switch(config-mst)**# revision** version Set the configuration revision number

spanning-tree mst instance-id **root {primary | secondary**} [**diameter** diameter] Set root bridge (macro).
spanning-tree mst instance-id **priority** bridge- priority Set bridge priority.
spanning-tree mst instance-id **cost** cost Set port cost.
spanning-tree mst instance-id **port-priority** port-priority Set port priority.

Set STP timers.
spanning-tree mst hello-time seconds
spanning-tree mst forward-time seconds
spanning-tree mst max-age seconds

STP and RTSP Port Path Cost values

Data rate	**STP 802.1D-1998**	**RSTP 802.1W-2004**
Base Metric	**1 Gigabit per sec**	**2 Terabit per sec**
Calculation	**1,000,000/(*N* Kb/s)**	**20,000,000,000/ (*N* Kb/s)**
	(1998 is nonlinear)	
	(1990 was linear)	
<=100 Kb/s		200,000,000
1 Mbit/s		20,000,000
4 Mbit/s	250	5,000,000
10 Mbit/s	100	2,000,000
16 Mbit/s	62 (was 63)	1,250,000
45 Mbit/s	39 (was 22)	444,445
100 Mbit/s	19 (was 10)	200,000
155 Mbit/s	14 (was 6)	129,032
622 Mbit/s	6 (was 2)	32,154
1 Gbit/s	4 (was 1)	20,000
2 Gbit/s	3	10,000
10 Gbit/s	2	2,000
100 Gbit/s		200
1 Tbit/s		20
10 Tbit/s		2

802.1W-2004: Limiting the range of the Path Cost parameter to 1–200 000 000 ensures that the accumulated Path Cost cannot exceed 32 bits over a concatenation of 20 hops
The grey cells indicate values presented by Cisco in the Switch 300-115 Official Cert Guide and are not in the IEEE 802.1W-2004.
Cisco also says in the CIEE 5.0 Routing and Switching book the short format is Cisco's, and not IEEE.
Indeed, in IEEE 802.1W-2004, it is the long format which remains (ed.: I am thankful since a nonlinear format is nonsensical)

Switching: EtherChannel

Cisco's Port Aggregation Protocol (PAgP) and Link Aggregation Control Protocol (LACP - 802.3ad)
- Combines multiple parallel segments of equal speed into an EtherChannel (single interface)
- Multiple segments of equal speed up to 8 links. If one fails, the rest stay up.
- Prevents STP convergence when only a single failure occurs; with at least one up, no STP convergence
- Aggregated links provide some load balancing on switches (and better use of bandwidth)

Create the port-channel interface, exit, then enter interface config mode for ports and add them with a mode:

```
Cat2950(config)# interface port-channel 1   <exit>
Cat2950(config)# interface fa0/2
Cat2950(config-if)# channel-group 1 mode on
```

PAgP: **channel-group 1 mode {desirable | auto}**
LACP: **channel-group 1 mode {active | passive}**
Manually run with **channel-group** 1 **mode on** (has no encapsulation; set "on" on both ends)
- One side must be either desirable or active to get the other end to begin negotiations (like trunks)
- Number must remain the same on the current device. Another device could call it 4 or 5

Three terms as synonyms: EtherChannel, PortChannel, and Channel-group.
The **channel-group** configuration command
show etherchannel shows status of a “PortChannel” (instead of EtherChannel or Channel-group)
show spanning-tree also says Port-channel

Misconfiguration of channel-group commands:
- All members of a switch's specific etherchannel need to be using the same channel-group number.
- The neighboring switches can refer to that etherchannel as any number they want.
- If using the "on" keyword, it has to be on the other end too (and remember no encaps'ing).
- If using the "desirable" keyword, it's PAgP and the other must use either "desirable" or "auto".
- If using "active" keyword, it's LACP; the other must use either "active" or "passive".
- No "auto" or "passive" keyword on both switches! Then both wait on the other to begin negotiations.
- Don't mix LACP and PAgP commands on one end or the other- the protocols need to match.

show etherchannel [chan-grp-#] {**brief** | **detail** | **port** | **port-channel** | **summary**}

```
SW1# show etherchannel 1 summary
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator
        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port
Number of channel-groups in use: 1
Number of aggregators:           1
Group  Port-channel  Protocol    Ports
1      Po1(SU)           -        Fa0/14(P)   Fa0/15(P)
```

The **show etherchannel summary** command. The D code letter means that the channel itself is down, with S meaning a Layer 2 EtherChannel. The bottom shows Portchannel (Po1) as L2 EtherChannel in a down state (SD)
"Down" only refers to the EtherChannel state - the two physical interfaces are still connected (**sh int status**)

Configuration Checks for EtherChannel
Things need to match or be compatible. Watch for common sense stuff (no shutdown, speed, duplex)
- Operational access or trunking state (all must be access, or all must be trunks)
- On an access port, check the access VLAN;
- Trunk port? Check allowed VLAN list - **switchport trunk allowed** - also check the native VLAN
- Check STP interface settings (ie, cost)

For example, if you change the STP port cost on one of the interfaces but not the other you get this:
ERR_Disable: channel-misconfig (STP) error detected on Po1, putting Fa0/14 in err-disable state
ERR_Disable: channel-misconfig (STP) error detected on Po1, putting Fa0/15 in err-disable state
ERR_Disable: channel-misconfig (STP) error detected on Po1, putting Po1 in err-disable state
Make them match and it should bring everything back up again

EtherChannel Stuff from CCNP

Load balancing uses a hashing operation on either MAC or IP addresses and can be based solely on source or destination addresses, or both. To configure frame distribution for all EtherChannel switch links:

Switch(config)**# port-channel load-balance** *method*

The method is set with a global configuration command. It's set globally for the switch, not on a per-port basis.

Table 10-3 Types of EtherChannel Load-Balancing Methods

Method Value	Hash Input
src-ip	Source IP
dst-ip	Destination IP
src-dst-ip	Source and destination IP
src-mac	Source MAC
dst-mac	Destination MAC
src-dst-mac	Source and destination MAC
src-port	Source port number
dst-port	Destination port number
src-dst-port	Source and destination port number

- All hash operations are performed on bits, except src-dst combo methods use XOR
- Port number only specified in CCNP Switch book as available in models 4500, 6500

Switch**# show etherchannel load-balance**
EtherChannel Load-Balancing Configuration:
src-mac

EtherChannel Load-Balancing Addresses Used Per-Protocol:
Non-IP: Source MAC address
IPv4: Source MAC address
IPv6: Source MAC address

To verify efficiency of load-balancing method use the **show etherchannel port-channel** command

EtherChannel Troubleshooting Commands

Current EtherChannel status of each member port	**show etherchannel summary** **show etherchannel port**
Time stamps of EtherChannel changes	**show etherchannel port-channel**
Detailed status about each EtherChannel component	**show etherchannel detail**
Load-balancing hashing algorithm	**show etherchannel load-balance**
Load-balancing port index used by hashing algorithm	**show etherchannel port-channel**
EtherChannel neighbors on each port	**show** {**pagp** | **lacp**} **neighbor**
LACP system ID	**show lacp sys-id**

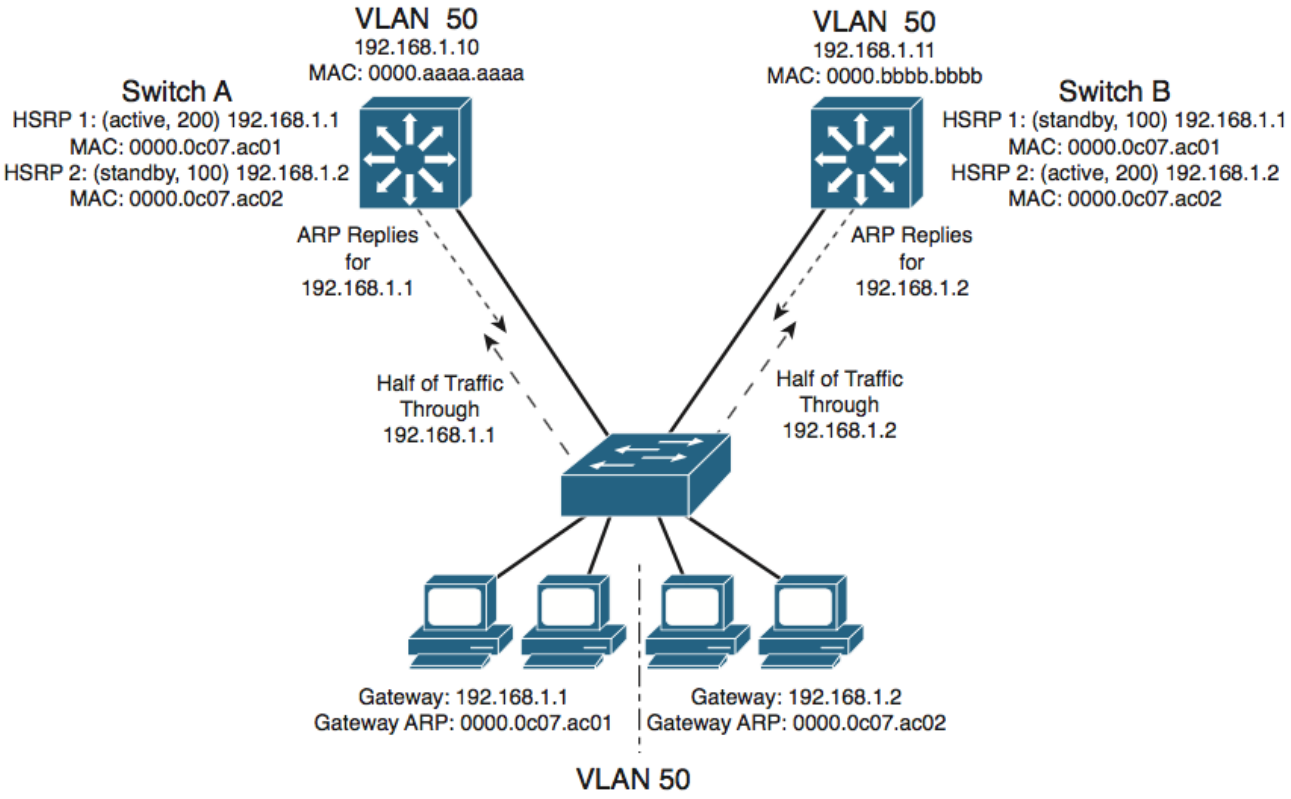
EtherChannel Configuration Commands

Select load-balancing method for switch.	**port-channel load-balance** method
Use a PAgP mode on an interface.	**channel-protocol pagp** **channel-group** number **mode** {**on |** {{**auto** | **desirable**} [**non-silent**]}
Assign the LACP system priority.	**lacp system-priority** priority
Use an LACP mode on an interface.	**channel-protocol lacp** **channel-group** number **mode** {**on | passive | active**} **lacp port-priority** priority
Configure EtherChannel Guard	[**no**] **spanning-tree etherchannel guard misconfig**

From SVI section
Keep in mind that a Layer 3 port assigns a network address to one specific physical interface. If several interfaces are bundled as an EtherChannel, the EtherChannel can also become a Layer 3 port. In that case, the network address is assigned to the port-channel interface—not to the individual physical links within the channel.

HSRP and VRRP Load Balancing Example
[two HSRP groups, one VLAN]

Load balancing traffic across two uplinks to two HSRP routers with a single HSRP group is not possible.
We must use two HSRP groups where each group assigns an active router to one of the switches
- Make each switch function as the standby router for its partner's HSRP group.
- (Each router is active for one group and standby for the other group)
- Switch A is active router for Group 1 (192.168.1.1), and standby for Group 2 (192.168.1.2).
- Switch B is configured similarly, but with its roles reversed.
- Configure half of the PCs with the Group 1 virtual router address and the other half with the Group 2 address.
- Each half of the hosts uses one switch as its default gateway over one uplink

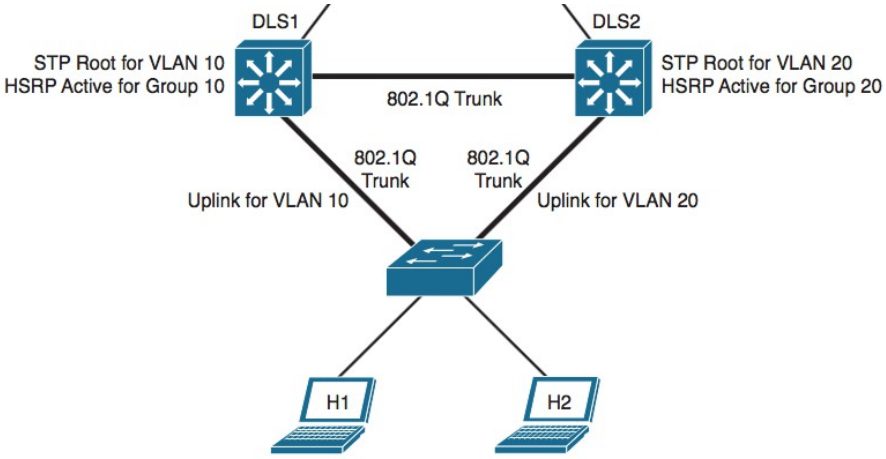


HSRP Version	**VRRP Version (minus authentication)**
Switch-A(config)# **interface vlan 50** Switch-A(config-if)# **ip address 192.168.1.10 255.255.255.0** Switch-A(config-if)# **standby 1 priority 200** Switch-A(config-if)# **standby 1 preempt** Switch-A(config-if)# **standby 1 ip 192.168.1.1** Switch-A(config-if)# **standby 2 priority 100** Switch-A(config-if)# **standby 2 ip 192.168.1.2**	Switch-A(config)# **interface vlan 50** Switch-A(config-if)# **ip address 192.168.1.10 255.255.255.0** Switch-A(config-if)# **vrrp 1 priority 200** Switch-A(config-if)# **vrrp 1 ip 192.168.1.1** Switch-A(config-if)# **vrrp 2 priority 100** Switch-A(config-if)# **no vrrp 2 preempt** Switch-A(config-if)# **vrrp 2 ip 192.168.1.2**
Switch-B(config)# **interface vlan 50** Switch-B(config-if)# **ip address 192.168.1.11 255.255.255.0** Switch-B(config-if)# **standby 1 priority 100** Switch-B(config-if)# **standby 1 ip 192.168.1.1** Switch-B(config-if)# **standby 2 priority 200** Switch-B(config-if)# **standby 2 preempt** Switch-B(config-if)# **standby 2 ip 192.168.1.2**	Switch-B(config)# **interface vlan 50** Switch-B(config-if)# **ip address 192.168.1.11 255.255.255.0** Switch-B(config-if)# **vrrp 1 priority 100** Switch-B(config-if)# **no vrrp 1 preempt** Switch-B(config-if)# **vrrp 1 ip 192.168.1.1** Switch-B(config-if)# **vrrp 2 priority 200** Switch-B(config-if)# **vrrp 2 ip 192.168.1.2**

HSRP Load Balancing Example 2

[two HSRP groups, two VLANs, Spanning Tree roots (primary and secondary]

- configure DLS1 as STP root, HSRP active for VLAN 10 - also backup root and HSRP standby for VLAN 20
- configure DLS2 as STP root, HSRP active for VLAN 20 - also backup root and HSRP standby for VLAN 10.
- The configuration for DLS2 below is basically the opposite of what DLS1 has.



STP forwarding to the correct VLANs ensured by setting spanning-tree primary root for those VLANs.

DLS1(config)**#spanning-tree vlan 10 root primary**	Configures STP root primary for VLAN 10.
DLS1(config)**#spanning-tree vlan 20 root secondary**	Configures STP root secondary for VLAN 20.
DLS1(config)**#interface vlan10**	Moves to interface configuration mode.
DLS1(config-if)**#ip address 10.1.10.2 255.255.255.0**	Assigns IP address and netmask.
DLS1(config-if)**#standby 10 ip 10.1.10.1**	Activates HSRP group 10, virtual IP of 10.1.10.1
DLS1(config-if)**#standby 10 priority 110**	Assigns a priority value of 110 to standby group 10. This will be the active forwarder for VLAN 10.
DLS1(config-if)**#standby 10 preempt**	Switch will take control of VLAN 10 forwarding if local priority is higher than active switch VLAN 10 priority.
DLS1(config-if)**#interface vlan20**	Moves to interface configuration mode.
DLS1(config-if)**#ip address 10.1.20.2 255.255.255.0**	Assigns IP address and netmask.
DLS1(config-if)**#standby 20 ip 10.1.20.1**	Activate HSRP group 20, virtual IP address 10.1.20.1.
DLS1(config-if)**#standby 20 priority 90**	Assigns a priority value of 90 to standby group 20. This switch will be the standby device for VLAN 20.
DLS1(config-if)**#standby 20 preempt**	This switch will take control of VLAN 20 forwarding if local priority is higher than active switch for VLAN 20

DLS2(config)**#spanning-tree vlan 20 root primary**	Configures STP root primary for VLAN 20.
DLS2(config)**#spanning-tree vlan 10 root secondary**	Configures STP root secondary for VLAN 10.
DLS2(config)**#interface vlan20**	Moves to interface configuration mode.
DLS2(config-if)**#ip address 10.1.20.3 255.255.255.0**	Assigns IP address and netmask.
DLS2(config-if)**#standby 20 ip 10.1.20.1**	Activates HSRP group 10, virtual IP of 10.1.20.1
DLS2(config-if)**#standby 20 priority 110**	Assigns a priority value of 110 to standby group 20. This will be the active forwarder for VLAN 10.
DLS2(config-if)**#standby 20 preempt**	Switch will take control of VLAN 20 forwarding if local priority is higher than active switch VLAN 20 priority.
DLS2(config-if)**#interface vlan10**	Moves to interface configuration mode.
DLS2(config-if)**#ip address 10.1.10.3 255.255.255.0**	Assigns IP address and netmask.
DLS2(config-if)**#standby 10 ip 10.1.10.1**	Activate HSRP group 10, virtual IP address 10.1.10.1.
DLS2(config-if)**#standby 10 priority 90**	Assigns a priority value of 90 to standby group 10. This switch will be the standby device for VLAN 10.
DLS2(config-if)**#standby 10 preempt**	This switch will take control of VLAN 20 forwarding if local priority is higher than active switch for VLAN 10

HSRP Load Balancing Example 1 (cont.)
(output of "show standby" command)

```
Switch-A# show standby vlan 50 brief
                     P indicates configured to preempt.
                     |
Interface   Grp Prio P State    Active addr     Standby addr    Group addr
Vl50        1   200  P Active   local           192.168.1.11    192.168.1.1
Vl50        2   100    Standby  192.168.1.11    local           192.168.1.2
Switch-A#
Switch-A# show standby vlan 50
Vlan50 - Group 1

  Local state is Active, priority 200, may preempt
  Hellotime 3 sec, holdtime 10 sec
  Next hello sent in 2.248
  Virtual IP address is 192.168.1.1 configured
  Active router is local
  Standby router is 192.168.1.11 expires in 9.860
  Virtual mac address is 0000.0c07.ac01
  Authentication text "MyKey"
  2 state changes, last state change 00:11:58
  IP redundancy name is "hsrp-Vl50-1" (default)
Vlan50 - Group 2
  Local state is Standby, priority 100
  Hellotime 3 sec, holdtime 10 sec
  Next hello sent in 1.302
  Virtual IP address is 192.168.1.2 configured
  Active router is 192.168.1.11, priority 200 expires in 7.812
  Standby router is local
  Authentication text "MyKey"
  4 state changes, last state change 00:10:04
  IP redundancy name is "hsrp-Vl50-2" (default)
```

Switch B is the same, just flipped.

VRRP Load Balancing example - output of "show" commands

Table 18-3 *Verifying VRRP Status for multiple VRRP Groups*

Switch A

```
Switch-A# show vrrp
Vlan50 - Group 1
  State is Master
  Virtual IP address is 192.168.1.1
  Virtual MAC address is 0000.5e00.0101
  Advertisement interval is 1.000 sec
  Preemption is enabled
    min delay is 0.000 sec
  Priority is 200
  Authentication is enabled
  Master Router is 192.168.1.10 (local),
    priority is 200
  Master Advertisement interval is 1.000
    sec
  Master Down interval is 3.218 sec

Vlan50 - Group 2
  State is Backup
  Virtual IP address is 192.168.1.2
  Virtual MAC address is 0000.5e00.0102
  Advertisement interval is 1.000 sec
  Preemption is disabled
  Priority is 100
  Authentication is enabled
  Master Router is 192.168.1.11, priority
    is 200
  Master Advertisement interval is 1.000
    sec
  Master Down interval is 3.609 sec
(expires in 2.977 sec)
Switch-A#
```

Switch B

```
Switch-B# show vrrp
Vlan50 - Group 1
State is Backup
Virtual IP address is 192.168.1.1
Virtual MAC address is 0000.5e00.0101
Advertisement interval is 1.000 sec
Preemption is disabled
Priority is 100
Authentication is enabled
Master Router is 192.168.1.10, priority
is 200
Master Advertisement interval is 1.000
sec
Master Down interval is 3.609 sec
(expires in 2.833 sec)

Vlan50 - Group 2
State is Master
Virtual IP address is 192.168.1.2
Virtual MAC address is 0000.5e00.0102
Advertisement interval is 1.000 sec
Preemption is enabled
min delay is 0.000 sec
Priority is 200
Authentication is enabled
Master Router is 192.168.1.11 (local),
priority is 200
Master Advertisement interval is 1.000
sec
Master Down interval is 3.218 sec
Switch-B#
```

```
Switch-A# show vrrp brief
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Vlan50             1   200 3218        Y  Master  192.168.1.10    192.168.1.1
Vlan50             2   100 3609           Backup  192.168.1.11    192.168.1.2
Switch-A#

Switch-B# show vrrp brief
Interface          Grp Pri Time  Own Pre State   Master addr     Group addr
Vlan50              1   100 3609           Backup  192.168.1.10    192.168.1.1
Vlan50              2   200 3218        Y  Master  192.168.1.11    192.168.1.2
Switch-B#
```

IEEE Standard: VRRP - Virtual Router Redundancy Protocol (RFC 2338)

Almost the same as HSRP. Here, "master" takes the place of "active" in HSRP terms, and the rest are termed "backup." The VRRP preempt option is enabled by default (the switch that is the IP address owner will preempt, regardless of this explicit setting). Default preempt delay is 0 sec (easily extended)
- Group number can be from 1- 255; Priority range 1-254. The default is 100.
- The virtual router MAC is of the form 0000.5e00.01xx, where xx is a two-digit hex group number.
- Sends advertisements to multicast 224.0.0.18 (VRRP), using IP protocol 112
- Uses a timer for advertisements by the virtual switch master.
- All switches in a group must use the same timer values, or the group will not communicate.
- Default interval value is 1 sec, range is 1-255 sec. If you use the msec argument, you change the timer globally to measure in milliseconds. The range then is 50 to 999ms.

Switch(config)**#interface vlan10**	(all of these are interface config mode)
ip address 172.16.100.5 255.255.255.0	Assigns IP address and netmask to VLAN interface.
vrrp 10 ip 172.16.100.1	Enables VRRP group 10 virtual IP of 172.16.100.1 (using a real interface IP here will make the router with that address become the master)
vrrp 10 description Sales Group	Assigns a text description to the group.
vrrp 10 priority 110	Sets the priority level for this VLAN.
vrrp 10 preempt	This switch will take over as virtual switch master for group 10 if it has a higher priority than the current one
vrrp 10 preempt delay minimum 60	This switch will preempt, only after a delay of 60 sec
vrrp 10 timers advertise 15	Configures the interval between successful advertisements by the virtual switch master.
vrrp 10 timers learn	Configures a virtual switch backup, to learn the advertisement interval from the virtual switch master.
(**no**) **vrrp 10 shutdown**	Disables/ re-enables VRRP on the interface, in a way that the configuration is still retained.
no vrrp 10 shutdown	Reenables the VRRP group using the previous configuration.
vrrp 10 authentication text ottawa	Plain-text auth for group 10 using the key **ottawa**.
vrrp 10 authentication md5 key-string winnipeg	MD5 auth for group 10 using the key **winnipeg**.

~~Interface~~ Object Tracking
VRRP does not have a native interface tracking mechanism, but even better, has the ability to track objects. This allows the VRRP master to lose its status if a tracked object (interface, IP SLA, and so on) fails.

track 10 interface fastethernet0/0 line-protocol	Creates a tracked object, where the status of the uplink interface is tracked
interface fastethernet0/1	
vrrp 1 track 10 decrement 30	Track previously created object and decrease the VRRP priority by 30 should the uplink interface fail

Verifying VRRP (same for IPv6 and IPv4)

Switch**#show vrrp**	Displays VRRP information
Switch**#show vrrp brief**	Displays a brief status of all VRRP groups
Switch**#show vrrp 10**	Displays detailed information about VRRP group 10
Switch**#show vrrp interface vlan10**	Displays info about VRRP enabled on int Vlan10
Switch**#show vrrp interface vlan10 brief**	Displays a brief summary about VRRP on int Vlan10

Debugging VRRP

Switch**#debug vrrp all**	Displays all VRRP messages
Switch**#debug vrrp error**	Displays all VRRP error messages
Switch**#debug vrrp events**	Displays all VRRP event messages
Switch**#debug vrrp packets**	Displays messages about packets sent and received
Switch**#debug vrrp state**	Displays messages about state transitions

VRRP is only partially supported on some Cisco hardware. Verify VRRP capabilities by platform datasheets and appropriate Cisco IOS command and configuration guides.

Cisco Proprietary: HSRP - Hot Standby Redundancy Protocol

interface vlan10	Make VLAN an interface - activate switch virtual interface (SVI)
ip address 172.16.0.10 255.255.255.0	Assigns IP address and netmask.
standby 1 ip 172.16.0.1 [**secondary**]	Activates HSRP group 1 and creates virtual IP address of 172.16.0.1 If interface has secondary IP addresses, you can add **secondary** so HSRP has a redundant secondary gateway address.
standby 1 priority 120	Assigns a priority value of 120 to standby group 1.

- Group number can be from 0 to 255. The default is 0.
- Some Catalyst switches limit to 16 unique group numbers- just make the group number the same (that is, 1) for every VLAN interface. HSRP groups are locally significant only on an interface: HSRP Group 1 on interface VLAN 10 is unique and independent from HSRP Group 1 on interface VLAN 11.
- The actual interface address and the virtual (standby) address must be configured to be in the same subnet.
- Priority value 1-255, default is 100. A higher priority will result in that switch being elected the active switch.
- If priorities of all switches in the group are equal, switch with highest IP address becomes active switch.
- HSRP hello messages - multicast 224.0.0.2 (“all routers”) - UDP port 1985.
- Only the standby router (with the second-highest priority) monitors hello messages from the active router.
- Decreasing hello time allows a router failure to be detected more quickly, yet increases traffic on the interface.

Router interface states before becoming active:

Disabled **=>** Init**=>** Listen **=>** Speak **=>** Standby **=>** Active

- HSRP defines a virtual MAC address in the form 0000.0c07.acxx, where xx represents the group number as a two-digit hex value. (Group 1 appears as 0000.0c07.ac01, Group 16 appears as 0000.0c07.ac1, etc.)

show standby	Displays HSRP information
show standby brief	Displays single-line output summary of each standby group
show standby vlan 1	Displays HSRP information on the VLAN 1 group

Defaults for HSRP

HSRP version	Version 1 (v1 and v2 have different packet structure)
HSRP groups	None configured.
Standby group number	0
Standby MAC address	System assigned as 0000.0c07.ac*XX*, where *XX* is the HSRP group number. For HSRPv2, the MAC address will be 0000.0C9F.FXXX.
Standby priority	100
Standby delay	0 (no delay)
Standby track interface priority	10
Standby hello time	3 seconds
Standby holdtime	10 seconds

Preempt - switch will take control of the active switch if local priority is higher than priority of active switch

interface vlan10	
standby 1 preempt	Designate this switch to preempt
standby 1 preempt delay minimum 180 reload 140	Set to preempt 180 sec since that switch was last restarted or 140 sec since switch was last reloaded.
standby delay minimum 30 reload 60	Delay for HSRP group initialization 30 sec when interface comes up and 60 sec after switch reloads.
no standby 1 preempt delay	Disables the preemption delay
no standby 1 preempt	Disable the preempt option completely

If a router is not already active, it cannot become active again until the current active router fails- even if its priority is higher than that of the active router. When routers are just being powered up or added to a network, the first router to bring up its interface becomes the HSRP active router, even if it has the lowest priority of all. This is where setting up preempt comes to the rescue.

- Use **reload** to force router to wait after it has been reloaded or restarted before preempt (you should consider and allow time for routing protocols (e.g.) that need time to converge). Use of **minimum** only refers to time after interface is ready for HSRP

Timers - Hello timer 1-254, default is 3; Hold timer 1-255, default is 10. The default unit of time is seconds.

interface vlan10	
standby 1 timers 5 15	Sets the hello to 5 sec and hold to 15 sec
	Hold normally set to be ≧ 3X hello
standby 1 timers msec 200 msec 600	Sets hello to 200 milliseconds, hold to 600 msec.

If the **msec** argument is used, the timers can be an integer from 15 to 999

Track - assigns a value that the priority will be decreased if the tracked interface goes down

Switch(config)**#interface vlan10**	
Switch(config-if)**#standby 1 track f0/0 25**	HSRP will track the availability of interface FastEthernet0/0. If it goes down, the priority of the switch in group 1 will be decremented by 25.

Default value of the **track** is 10. In the example, assuming default priority of 100, the new priority will be 75. Using track facilitates in election eligibility when failures occur and replacement candidates must be considered.

Authentication

Switch(config)**#key chain HSRP**	Creates authentication key chain called **HSRP**.
Switch(config-keychain)**#key 1**	Adds a first key to the key chain.
Switch(config-keychain-key)**#key-string australia**	Configures a key string of **australia**.
Switch(config)**#interface vlan10**	
Switch(config-if)**#standby 1 authentication text canada**	Configures **canada** as plain-text authentication string used by group 1.
Switch(config-if)**#standby 2 authentication md5 key-string england**	Configures **england** as MD5 key string for group 2.
Switch(config-if)**#standby 3 authentication md5 key-chain HSRP**	Configures MD5 using key chain **HSRP**. Queries key chain for current live key and key ID.

HSRPv2 for IPv6

HSRPv2 must be enabled on an interface before HSRP for IPv6 can be configured.
When configuring the IPv6 virtual address, if an IPv6 global address is used, it must include an IPv6 prefix length. If a link-local address is used, it does not have a prefix

standby version 2	Enables HSRPv2 on an interface
standby 1 ipv6 autoconfig	Use a virtual link-local address that will be generated automatically from the link-local prefix and a modified EUI-64 format interface identifier, where the EUI-64 interface identifier is created from the relevant HSRP virtual MAC address
standby 1 ipv6 FE80::1:1	Use an explicitly configured link-local address to be used as the virtual IPv6 address for group 1
standby 1 ipv6 2001::0DB8:2/64	Use a global IPv6 address as the virtual address for group 1

All other relevant HSRP commands (preempt, priority, authentication, tracking, and so on) are identical in HSRPv1 and HSRPv2.

Debugging HSRP

debug standby	All HSRP debugging information, including state changes and transmission/ reception of HSRP packets
debug standby terse	All HSRP errors, events, and packets, except for hellos and advertisements
debug standby errors	Displays HSRP error messages
debug standby events	Displays HSRP event messages
debug standby events terse	Displays all HSRP events except for hellos and advertisements
debug standby events track	Displays all HSRP tracking events
debug standby packets	Displays HSRP packet messages

Cisco Proprietary: GLBP - Gateway Load Balancing Protocol

Introduced in Cisco IOS 12.2(14)S for routers, but is not consistently supported across all switching platforms.
- Group members elect one gateway to be the **active virtual gateway (AVG)**.
- Members are **active virtual forwarders (AVF)**, backup for AVG in the event it becomes unavailable.
- The AVG assigns a different virtual MAC address to each AVF (which becomes their identifier).
- Each AVF assumes responsibility for forwarding packets sent to the virtual MAC address assigned
- If an AVF fails, one of others assumes responsibility for the virtual MAC address.
- Precedence is highest priority value, or the highest IP address in the group
- Automatic selection and simultaneous use of multiple available gateways; automatic failover
- No configuring multiple groups or multiple default gateway configurations like HSRP/VRRP
- Maximum of four routers in each forwarding group. Technically the AVG is also an AVF (it is just "special")
- Up to 1024 virtual routers as GLBP groups on each router's physical interface (number range 0-1023)
- Up to four AVFs per group at a time, secondaries can be backups if one fails

- Priority is 1- 255, default 100. A higher number is preferred.
- Preemption is disabled by default for AVG; AVF preempt has default delay of 30 seconds.
- Hello timer default 3 sec; range 1-60 sec. If msec argument is used, the timer will switch to msec, range of 50- 60,000 msec.
- Hold timer default 10 sec; range 19-180 sec. If msec argument is used, the timer will switch to msec, range of 18,020-180,000 msec.
- Hello messages - multicast 224.0.0.102 - UDP 3222.
- Virtual MAC have the form 0007.b4xx.xxyy. The 16-bit value denoted by xx.xx represents six 0 bits followed by a 10-bit GLBP group number. The 8-bit yy value is the virtual forwarder number.

Three different types of load balancing

Host-dependent uses MAC address of a host to determine which AVF MAC address to use.
- The option if using stateful NAT (needs each host to be returned to the same virtual MAC address each time it sends an ARP request for the virtual IP address)
- Not recommended for where there are fewer than 20 end hosts.

Weighted - place a weight on each device when calculating the amount of load sharing.
- If router A has twice the forwarding capacity of router B, weighting value should be configured accordingly
- Use the **glbp x weighting y** where **x** is the GLBP group number, and **y** is the weighting value (1-254)

Round-robin (default) each AVF MAC is used sequentially in ARP replies for virtual IP address.
- If no load balancing is used, GLBP will operate like HSRP, where the AVG will only respond to ARP requests with its own AVF MAC address, and all traffic will be directed to the AVG. Use **no glbp load-balancing**.

It is recommended that unless you are extremely familiar with your network design and mechanisms of GLBP that you don't change the timers. Reset timers back to default **no glbp x timers**, where x is group number.

Router(config)**#interface fastethernet0/0**	(all of these are interface config mode)
ip address 172.16.100.5 255.255.255.0	Assigns IP address and netmask.
glbp 10 ip 172.16.100.1	Put group 10 on this int w/ virtual IP of 172.16.100.1.
glbp 10 preempt	Take over as AVG for group 10 if this router has higher priority than current AVG
glbp 10 preempt delay minimum 60	Wait before preempt AVG- delay of 60 seconds
glbp 10 forwarder preempt	Take over as AVF for group 10 if this router has higher priority than current AVF.
glbp 10 forwarder preempt delay minimum 60	Wait before preempt AVF- delay of 60 seconds
glbp 10 priority 150	Sets the priority level of the router.
glbp 10 timers 5 15	Configures the hello timer 5 sec and the hold 15 sec
glbp 10 timers msec 20200 msec 60600	Hello timer 20,200 msec hold timer to 60,600 millisec
glbp 10 authentication text edmonton	Configures GLBP for plain text authentication
glbp 10 authentication md5 key-chain vancouver	Configures GLBP for MD5 authentication
glbp 10 load- balancing host-dependent	Load balance using the host-dependent method.
glbp 10 load- balancing weighted	Load balance using the weighted method.
glbp 10 weighting 80	Set maximum weighting value for this interface
glbp 10 load balancing round robin	Load balance using the round-robin method.

Interface Tracking

Router(config)**#track 2 interface fa0/1 line-protocol**	Config FastEthernet0/1 interface to be tracked. The **line-protocol** tracks whether the interface is up
Router(config-track)**#exit**	Notice prompt "(config-track)" Use ? for more options
Router(config)**#interface fastethernet0/0**	
Router(config-if)**#glbp 10 weighting 110 lower 20 upper 50**	Initial weighting value; upper/lower thresholds, for GW
Router(config-if)**#glbp 10 weighting track 2 decrement 50**	Track the object and decrement the weight by 50 when the Fast Ethernet 0/1 interface fails

Verifying GLBP

Router**#show glbp**	Displays GLBP information
Router**#show glbp brief**	Displays a brief status of all GLBP groups
Router**#show glbp 10**	Displays information about GLBP group 10
Router**#show glbp vlan10**	Displays GLBP information on interface Vlan10
Router**#show glbp vlan20 10**	Displays GLBP group 10 info on interface Vlan20

Debugging GLBP

Router**#debug condition glbp**	Displays GLBP condition messages
Router**#debug glbp errors**	Displays all GLBP error messages
Router**#debug glbp events**	Displays all GLBP event messages
Router**#debug glbp packets**	Displays messages about packets sent and received
Router**#debug glbp terse**	Displays a limited range of debugging messages

A redirect timer is used to determine when the AVG will stop using a stale virtual MAC address in ARP replies. The AVF corresponding to the old address continues to act as a stand-in gateway for any clients that try to use it.

When the timeout timer expires, the old MAC address and the virtual forwarder using it are flushed from all the GLBP peers. The AVG assumes that the previously failed AVF will not return to service, so the resources assigned to it must be reclaimed.
At this point, clients still using the old MAC address in their ARP caches must refresh the entry to obtain the new virtual MAC address.

The redirect timer defaults to 600 seconds (10 minutes) and can range from 0 to 3600 seconds (1 hour). The timeout timer defaults to 14,400 seconds (4 hours) and can range from 700 to 64,800 seconds (18 hours). You can adjust these timers with the following interface configuration command:
Switch(config-if)# **glbp** *group* **timers redirect** *redirect timeout*

Switch(config)# **track** *object-number* **interface** *type member/module/number* {**line- protocol** | **ip routing**}
The object-number is an arbitrary index (1 to 500) that is used for weight adjustment. The condition that triggers an adjustment can be **line-protocol** (the interface line protocol is up) or **ip routing**. (IP routing is enabled, the interface has an IP address, and the interface is up.)

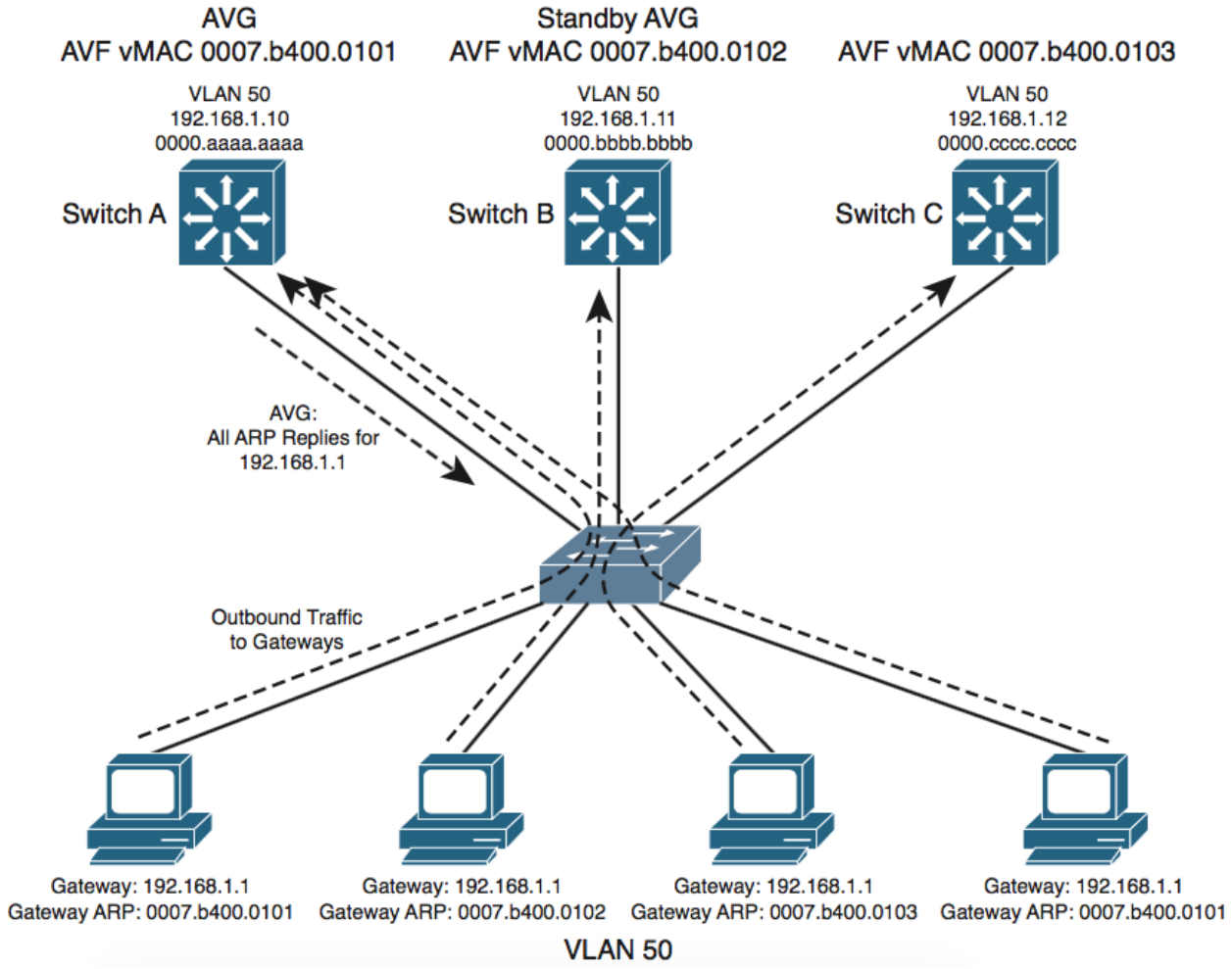
GLBP can also be used with IPv6. Rather than specifying an IPv6 address, use the following command to autoconfigure the address:
Switch(config-if)# **glbp** *group* **ipv6 autoconfigure**

Typical 3 Switch GLBP Setup Example

Three multilayer switches participating in a common GLBP group. with switch A elected AVG, to coordinate the GLBP process, answer all ARP requests for the virtual router 192.168.1.1.

It has identified itself, and other two switches as AVFs for the group, using default round-robin load balancing, so each of the client PCs sends an ARP request to look for the virtual router address (192.168.1.1) in turn, from left to right. Each time the AVG replies, the next sequential virtual MAC is sent back to a client. After the fourth PC sends a request, all three virtual MAC addresses (and AVF routers) have been used, so the AVG cycles back to the first virtual MAC address.

One GLBP group has been configured, clients know of only one gateway IP address: 192.168.1.1, all uplinks are being used, and all routers are proportionately forwarding traffic.



Switch-A(config)# **interface vlan 50**
Switch-A(config-if)# **ip address 192.168.1.10 255.255.255.0**
Switch-A(config-if)# **glbp 1 priority 200**
Switch-A(config-if)# **glbp 1 preempt**
Switch-A(config-if)# **glbp 1 ip 192.168.1.1**

Switch-B(config)# **interface vlan 50**
Switch-B(config-if)# **ip address 192.168.1.11 255.255.255.0**
Switch-B(config-if)# **glbp 1 priority 150**
Switch-B(config-if)# **glbp 1 preempt**
Switch-B(config-if)# **glbp 1 ip 192.168.1.1**

Switch-C(config)# **interface vlan 50**
Switch-C(config-if)# **ip address 192.168.1.12 255.255.255.0**
Switch-C(config-if)# **glbp 1 priority 100**
Switch-C(config-if)# **glbp 1 ip 192.168.1.1**

You can verify GLBP operation with the **show glbp** [**brief**] command. With the **brief** keyword, the GLBP roles are summarized showing the interface, GLBP group number (Grp), virtual forwarder number (Fwd), GLBP priority (Pri), state, and addresses.

```
Switch-A# show glbp brief
Interface   Grp  Fwd Pri State    Address         Active router   Standby router
Vl50        1    -   200 Active   192.168.1.1     local           192.168.1.11
Vl50        1    1   7   Active   0007.b400.0101  local           -
Vl50        1    2   7   Listen   0007.b400.0102  192.168.1.11    -
Vl50        1    3   7   Listen   0007.b400.0103  192.168.1.12    -
Switch-A#

Switch-B# show glbp brief
Interface   Grp  Fwd Pri State    Address         Active router   Standby router
Vl50        1    -   150 Standby  192.168.1.1     192.168.1.10    local
Vl50        1    1   7   Listen   0007.b400.0101  192.168.1.10    -
Vl50        1    2   7   Active   0007.b400.0102  local           -
Vl50        1    3   7   Listen   0007.b400.0103  192.168.1.12    -
Switch-B#

Switch-C# show glbp brief
Interface   Grp  Fwd Pri State    Address         Active router   Standby router
Vl50        1    -   100 Listen   192.168.1.1     192.168.1.10    192.168.1.11
Vl50        1    1   7   Listen   0007.b400.0101  192.168.1.10    -
Vl50        1    2   7   Listen   0007.b400.0102  192.168.1.11    -
Vl50        1    3   7   Active   0007.b400.0103  local          -
```

Notice that Switch A is shown to be the AVG because it has a dash in the Fwd column and is in the Active state. It also is acting as AVF for virtual forwarder number 1.

Because the GLBP group has three routers, there are three virtual forwarders and virtual MAC addresses.

Switch A is in the Listen state for forwarders number 2 and 3, waiting to be given an active role in case one of those AVFs fails.

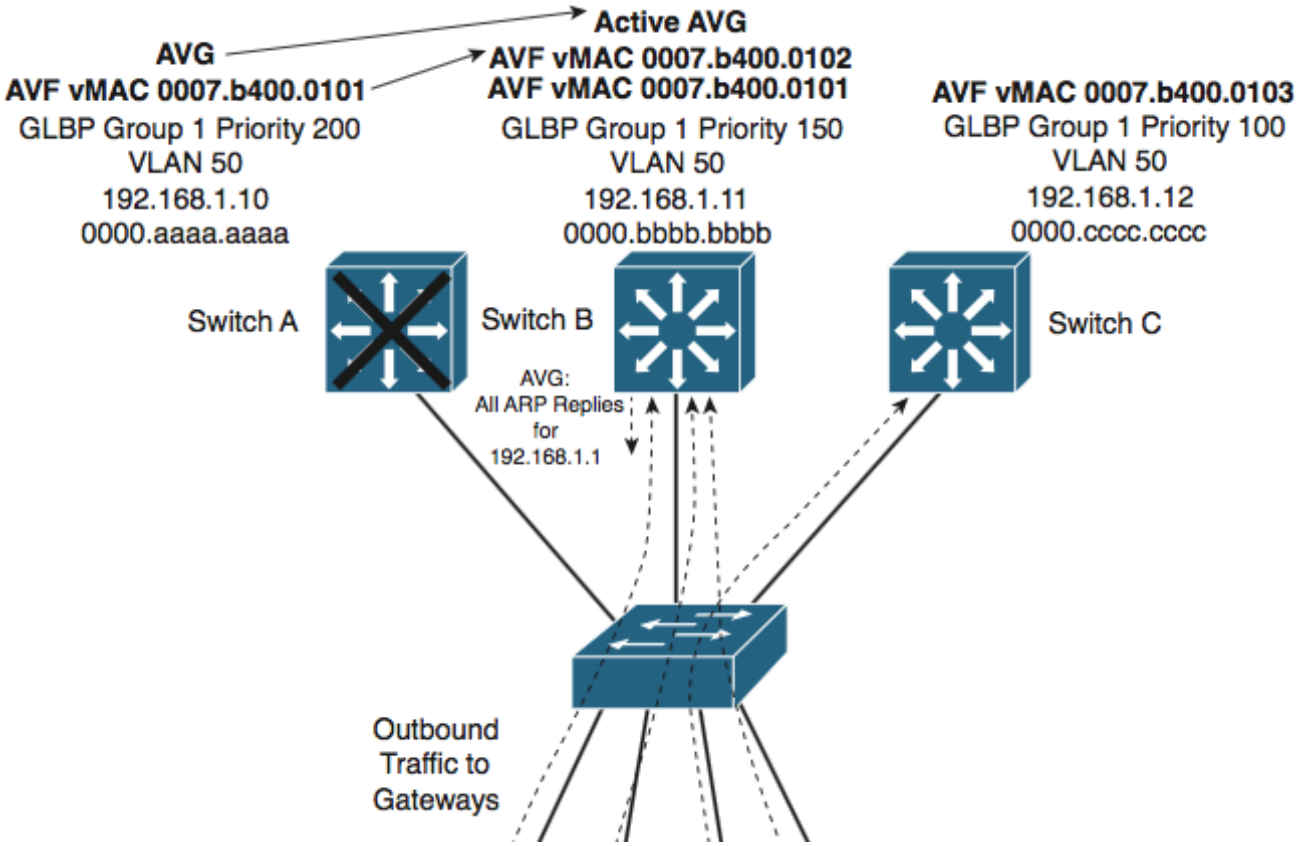
Switch B is shown to have the Standby role, waiting to take over in case the AVG fails. It is the AVF for virtual forwarder number 2.

Finally, Switch C has the lowest GLBP priority, so it stays in the Listen state, waiting for the active or standby AVG to fail. It is also the AVF for virtual forwarder number 3.

Taking off the **brief** keyword and just using **show glbp** displays more detailed info about config and status

The AVG here shows the virtual forwarder roles it has assigned to each of the routers in the GLBP group.

```
Switch-A# show glbp
Vlan50 - Group 1
  State is Active
    7 state changes, last state change 03:28:05
  Virtual IP address is 192.168.1.1
  Hello time 3 sec, hold time 10 sec
    Next hello sent in 1.672 secs
  Redirect time 600 sec, forwarder time-out 14400 sec
  Preemption enabled, min delay 0 sec
  Active is local
  Standby is 192.168.1.11, priority 150 (expires in 9.632 sec)
  Priority 200 (configured)
  Weighting 100 (default 100), thresholds: lower 1, upper 100
  Load balancing: round-robin
  There are 3 forwarders (1 active)
  Forwarder 1
    State is Active
      3 state changes, last state change 03:27:37
    MAC address is 0007.b400.0101 (default)
    Owner ID is 00d0.0229.b80a
    Redirection enabled
    Preemption enabled, min delay 30 sec
    Active is local, weighting 100
  Forwarder 2
    State is Listen
    MAC address is 0007.b400.0102 (learnt)
    Owner ID is 0007.b372.dc4a
    Redirection enabled, 598.308 sec remaining (maximum 600 sec)
    Time to live: 14398.308 sec (maximum 14400 sec)
    Preemption enabled, min delay 30 sec
    Active is 192.168.1.11 (primary), weighting 100 (expires in 8.308 sec)
  Forwarder 3
    State is Listen
    MAC address is 0007.b400.0103 (learnt)
    Owner ID is 00d0.ff8a.2c0a
    Redirection enabled, 599.892 sec remaining (maximum 600 sec)
    Time to live: 14399.892 sec (maximum 14400 sec)
    Preemption enabled, min delay 30 sec
    Active is 192.168.1.12 (primary), weighting 100 (expires in 9.892 sec)
```



Redundancy is also inherent in the GLBP group: Switch A is the AVG, but the next-highest priority router can take over if the AVG fails.

All routers have been given an AVF role for a unique virtual MAC address in the group.

If one AVF fails, some clients remember the last-known virtual MAC address that was handed out. Therefore, another of the routers also takes over the AVF role for the failed router, causing the virtual MAC address to remain alive at all times.

Above is illustrated how these redundancy features react when the current active AVG fails. Before its failure, Switch A was the AVG because of its higher GLBP priority.

After it failed, Switch B became the AVG, answering ARP requests with the appropriate virtual MAC address for gateway 192.168.1.1.

Switch A also had been acting as an AVF, participating in the gateway load balancing.

Switch B also picks up this responsibility, usingits virtual MAC address 0007.b400.0102 along with the one Switch A had been using, 0007.b400.0101.

Therefore, any hosts that know the gateway by any of its virtual MAC addresses still can reach a live gateway or AVF.

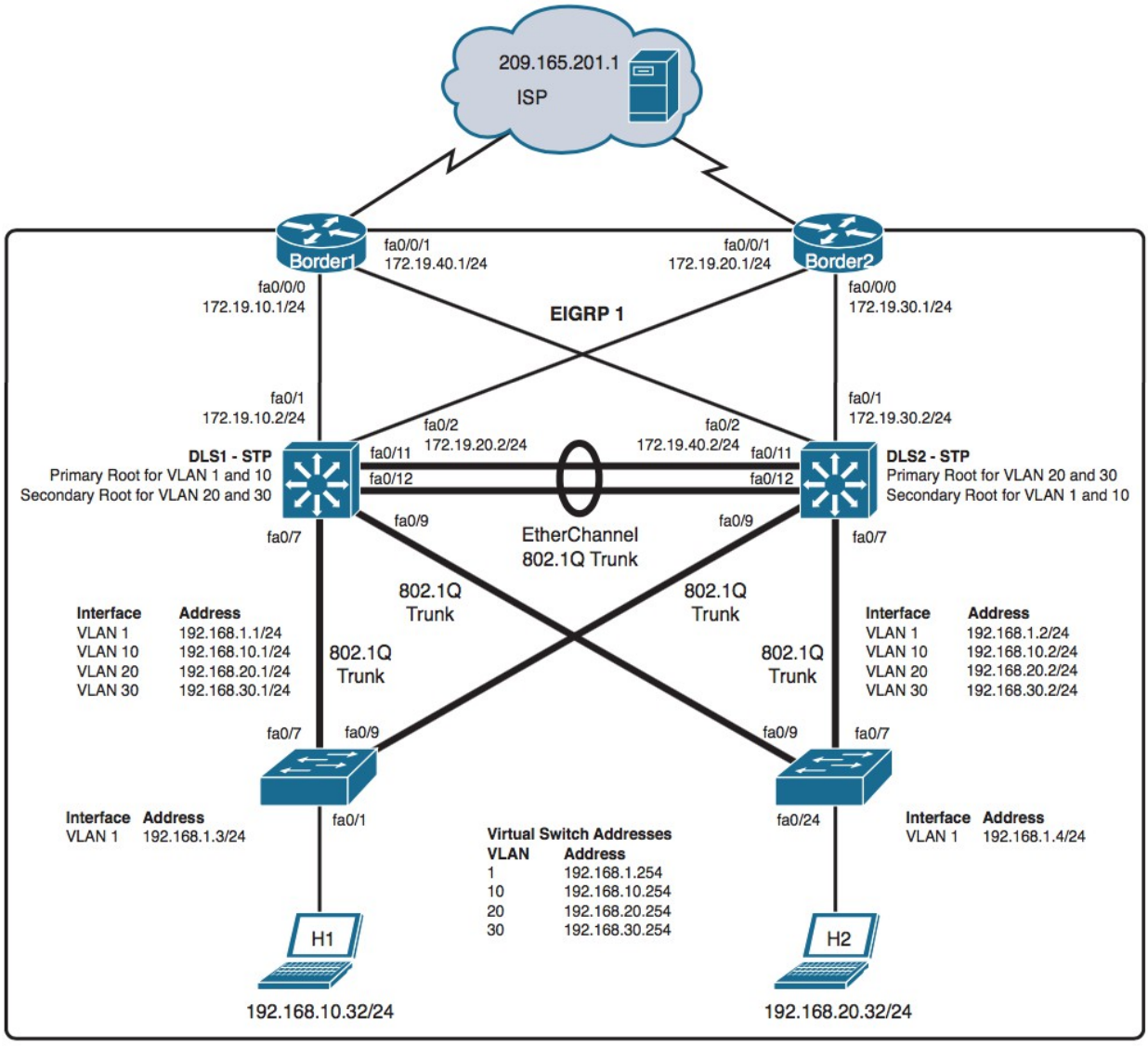


Figure 13-2 Network Topology for HSRP Configuration Example

DHCP Setup on Cisco Devices (IPv4)

Switch(config)#service dhcp -- makes sure it's enabled - "no service dhcp" turns it off
Switch(config)#ip dhcp pool Sales_Wireless --create the pool
Switch(dhcp-config)#network 192.168.10.0 255.255.255.0 -- netID and mask for pool
Switch(dhcp-config)#default-router 192.168.10.1 --default gateway
Switch(dhcp-config)#dns-server 4.4.4.4
Switch(dhcp-config)#lease 3 12 15 --days, hours, minutes - default 24hours
Switch(dhcp-config)#exit
Switch(config)#ip dhcp excluded-address 192.168.10.1 192.168.10.10

DHCP Relay
If you need to provide addresses from a DHCP server to hosts that aren't on the same LAN as the DHCP server, you can configure your router interface to relay DHCP client requests.
Router(config)#interface fa0/0
Router(config-if)#ip helper-address 10.10.10.254

The router and fa0/0 are on 192.168.10.1. 10.10.10.254 is obviously on a different subnet, but is the DHCP server. ip helper-address forwards more than just DHCP client requests.

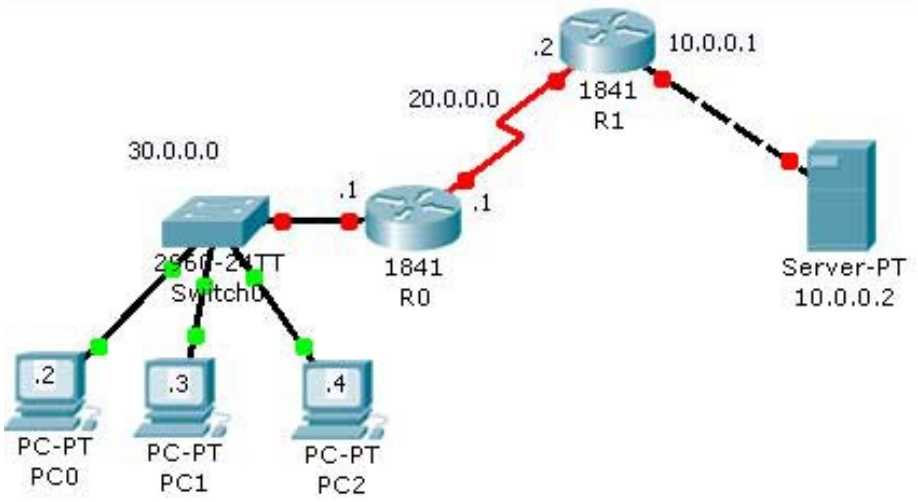
Verifying DHCP
show ip dhcp binding - Lists state information about each IP address currently leased
show ip dhcp pool [poolname] - Lists scope of addresses, plus stats for leased addresses
show ip dhcp server statistics - Lists DHCP server statistics
show ip dhcp conflict - show duplicate addresses

More Pools Example
Corp#config t
Corp(config)#ip dhcp excluded-address 192.168.10.1
Corp(config)#ip dhcp excluded-address 192.168.20.1
Corp(config)#ip dhcp pool SF_LAN
Corp(dhcp-config)#network 192.168.10.0 255.255.255.0
Corp(dhcp-config)#default-router 192.168.10.1
Corp(dhcp-config)#dns-server 4.4.4.4
Corp(dhcp-config)#exit
Corp(config)#ip dhcp pool LA_LAN
Corp(dhcp-config)#network 192.168.20.0 255.255.255.0
Corp(dhcp-config)#default-router 192.168.20.1
Corp(dhcp-config)#dns-server 4.4.4.4

Add these to the two routers so they will forward DHCP:
LA(config)#int f0/0
LA(config-if)#ip helper-address 172.16.10.5

SF(config)#int f0/0
SF(config-if)#ip helper-address 172.16.10.1

Network Address Translation : Static and Dynamic NAT and PAT



IPv4 Static NAT

R1(config-if)#ip address 10.0.0.1 255.0.0.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#interface serial 0/0/0
R1(config-if)#ip address 20.0.0.2 255.0.0.0
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#ip route 30.0.0.0 255.0.0.0 20.0.0.1
R1(config)#ip nat inside source static 10.0.0.2 50.0.0.1
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip nat inside
R1(config-if)#exit
R1(config)#interface serial 0/0/0
R1(config-if)#ip nat outside
R1(config-if)#exit

R0(config)#interface fastethernet 0/0
R0(config-if)#ip address 30.0.0.1 255.0.0.0
R0(config-if)#no shutdown
R0(config-if)#exit
R0(config)#interface serial 0/0/0
R0(config-if)#ip address 20.0.0.1 255.0.0.0
R0(config-if)#clock rate 64000
R0(config-if)#bandwidth 64
R0(config-if)#no shutdown
R0(config-if)#exit
R0(config)#ip route 50.0.0.0 255.0.0.0 20.0.0.2

There is not direct route for 10.0.0.2. So PC from network of 30.0.0.0 will never know about it. They will access 50.0.0.1 as the web server IP. Ping 50.0.0.1 and it works. Ping 10.0.0.2 and you will get destination host unreachable error.

IP4 Dynamic NAT
This is using 192.168.0.0 network as internal. We have five public ip address 50.0.0.1 to 50.0.0.5 to use. Router1(1841 Router0) is going to be NAT device. Starting off by configuring Router1(1841 Router0):

R1(config)#interface fastethernet 0/0
R1(config-if)#ip address 192.168.0.1 255.0.0.0
R1(config-if)#no shutdown
R1(config)#interface serial 0/0/0
R1(config-if)#ip address 30.0.0.1 255.0.0.0
R1(config-if)#clock rate 64000
R1(config-if)#bandwidth 64
R1(config-if)#no shutdown
R1(config-if)#exit
R1(config)#ip route 0.0.0.0 0.0.0.0 serial 0/0/0
R1(config)#access-list 1 permit 192.168.0.0 0.0.0.255
R1(config)#ip nat pool test 50.0.0.1 50.0.0.5 netmask 255.0.0.0
R1(config)#ip nat inside source list 1 pool test
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip nat inside
R1(config-if)#exit
R1(config)#interface serial 0/0/0
R1(config-if)#ip nat outside
R1(config-if)#exit

R1#debug ip nat
IP NAT debugging is on
NAT: s=192.168.0.7->50.0.0.1, d=20.0.0.2[1]
NAT*: s=20.0.0.2, d=50.0.0.1->192.168.0.7[1]
R1#no debug ip nat
IP NAT debugging is off

As you can see in output 192.168.0.5 is translated with 50.0.0.1 before leaving the router
Webpage loads as well.

IP4 PAT
If you have few global IP address and hundred of inside local address to translate. In such a situation you need to use PAT. This time we are using only one global IP address 50.0.0.1

R1(config)#access-list 1 permit 192.168.0.0 0.0.0.255
R1(config)#ip nat pool test 50.0.0.1 50.0.0.1 netmask 255.0.0.0
R1(config)#ip nat inside source list 1 pool test overload
R1(config)#interface fastEthernet 0/0
R1(config-if)#ip nat inside
R1(config-if)#exit
R1(config)#interface serial 0/0/0
R1(config-if)#ip nat outside
R1(config-if)#exit

To verify PAT go on R1 and run show ip nat translations
R1#show ip nat translations

Pro	Inside global	Inside local	Outside local	Outside global
icmp	50.0.0.1:1	192.168.0.7:1	20.0.0.2:1	20.0.0.2:1
icmp	50.0.0.1:2	192.168.0.7:2	20.0.0.2:2	20.0.0.2:2

Access Control Lists on Cisco Devices

Set ACL in global config, then apply to interfaces.
Put the right ACL on the right port! Outbound or inbound?
Put the ACL on the right device!
- standard ACLs should be close to destination, and extended ACLs close to source
- Prevent unecessary network traffic that will just be denied by the subnet on the other side.
- Use as-needed granularity on the client end prevent a mass denial of traffic.

Corp(config)#access-list {1-99 | 1300-1999} - for standard
Corp(config)#access-list {100-199 | 2000-2699} - for IP extended access list

None of these source addresses should be ever be allowed to enter a production network:
Deny any source addresses from your internal networks.
Deny any local host (127.0.0.0/8) and reserved private addresses (RFC 1918).
Deny any addresses in the IP multicast address range (224.0.0.0/4).

Put most specific (single IPs) first. Subnet or range of subnets lower on list
Put permits first then place deny entries lower on the lists to filter properly
e.g.: Place 10.1.1.1 allow first, and have a later entry for deny 10.1.1.x for the rest of that subnet.

After adding restrictions (deny statements) append an explicit "permit any" at end of rules to counteract implicit deny. See the set ACLs listed in the running-config
You need to disable an ACL before making changes: no ip access-list 3

Corp(config)#access-list 10 [deny | permit | remark] [any | subnet | host] [wildcard (for subnets)]
Wildcard masks: for a block of addresses. Each block size must start at 0 or a multiple of the block size.

Corp(config)#access-list 10 deny 172.16.0.0 0.0.255.255
Match the first two octets and that the last two octets can be any value
Corp(config)#access-list 10 deny 172.16.16.0 0.0.3.255
This configuration tells the router to start at network 172.16.16.0 and use a block size of 4
The range would then be 172.16.16.0 through 172.16.19.255
Corp(config)#access-list 10 deny 172.16.16.0 0.0.7.255
172.16.16.0 going up a block size of 8 to 172.16.23.255
Corp(config)#access-list 10 deny 172.16.32.0 0.0.15.255
172.16.32.0 and goes up a block size of 16 to 172.16.47.255
Corp(config)#access-list 10 deny 172.16.64.0 0.0.63.255
172.16.64.0 and goes up a block size of 64 to 172.16.127.255
Corp(config)#access-list 10 deny 192.168.160.0 0.0.31.255
192.168.160.0 and goes up a block size of 32 to 192.168.191.255

Router(config)#access-list 1 deny 172.16.128.0 0.0.31.255 --- /19 is a 32 block means 144 is in 128-160.0
Router(config)#access-list 1 deny 172.16.48.0 0.0.15.255 --- /20 is a 16 block means 50 is in 48-60.0
Router(config)#access-list 1 deny 172.16.192.0 0.0.63.255 ---/18 is a 64 block means 198 is in 192-254.0
Router(config)#access-list 1 deny 172.16.88.0 0.0.7.255 ---/21 is an 8 block means 92 is in 88-106.0
Router(config)#access-list 1 permit any
Router(config)#interface serial 0
Router(config-if)#ip access-group 1 out
It turns out the rules could have been done this with one line:
Router(config)#access-list 1 deny 172.16.0.0 0.0.255.255
The value of 0.0.0.0/255.255.255.255 can be specified as any.

Need: a standard access list that permits only the host or hosts you want to be able to telnet into the routers, then apply the access list to the VTY line with the access-class in command.
Lab_A(config)#access-list 50 permit host 172.16.10.3
Lab_A(config)#line vty 0 4
Lab_A(config-line)#access-class 50 in <--- use "-class" when configuring lines instead of "-group"

Users on the Sales LAN (172.16.40.0/24 - Fa0/0) should not have access to the Finance LAN (172.16.50.0/24 Fa0/1), but they should be able to access the Internet and the marketing department files. The Marketing LAN needs to access the Finance LAN for application services.
Lab_A(config)#access-list 10 deny 172.16.40.0 0.0.0.255
Lab_A(config)#access-list 10 permit any
Place it close to the source

If you place it as an incoming access list on Fa0/0, you might as well shut down the interface because all of the Sales LAN devices will be denied access to all networks attached to the router. The best place to apply this access list is on the Fa0/1 interface as an outbound list:
Lab_A(config)#int fa0/1
Lab_A(config-if)#ip access-group 10 out

Extended ACLs

access-list 165 [deny | permit | remark] [protocol] {sourceIP maskWC} {destIP maskWC} [logical op] [port | tos]
For a single host, use "host" before the IP address and no wildcard mask is required

interface Serial 0/0
Access-group 100 in
access-list 100 remark Begin -- Allow BGP IN and OUT
access-list 100 permit tcp host 1.1.1.1 host 2.2.2.2 eq bgp
access-list 100 permit udp any host 2.2.2.2 gt 33000
access-list 100 remark End
access-list 100 deny ip any any log

interface BRI 0/0
Access-group 100 in
Access-group 100 out

access-list 100 permit ip any any log
debug IP packet detail XXX (access list number)

access-list 101 permit tcp any any eq telnet
debug ip packet detail 101
[IP packet debugging is on (detailed) for access list 101]

As always, once our access list is created, we must apply it to an interface with the same command used for the IP standard list:
Corp(config-if)#ip access-group 110 in
Or this:
Corp(config-if)#ip access-group 110 out

Extended ACLs #2
Lab_A#config t
Lab_A(config)#access-list 110 deny tcp any host 172.16.50.5 eq 21
Lab_A(config)#access-list 110 deny tcp any host 172.16.50.5 eq 23
Lab_A(config)#access-list 110 permit ip any any
Lab_A(config)#int fa0/1
Lab_A(config-if)#ip access-group 110 out

Router(config)#access-list 110 deny tcp any 172.16.48.0 0.0.15.255 eq 23
Router(config)#access-list 110 deny tcp any 172.16.192.0 0.0.63.255 eq 23
Router(config)#access-list 110 permit ip any any
Router(config)#interface Ethernet 1
Router(config-if)#ip access-group 110 out
Router(config-if)#interface Ethernet 2
Router(config-if)#ip access-group 110 out

Allow HTTP access to the Finance server from source Host B only. All other traffic will be permitted. We need to be able to configure this in only three test statements:
Lab_A(config)#access-list 110 permit tcp host 192.168.177.2 host 172.22.89.26 eq 80
Lab_A(config)#access-list 110 deny tcp any host 172.22.89.26 eq 80
Lab_A(config)#access-list 110 permit ip any any
Lab_A(config)#interface fastethernet 0/1
Lab_A(config-if)#ip access-group 110 out
webserver:
access-list 253 permit tcp 127.16.5.0 0.0.0.255 eq www any <--permit from 172.16.5.* of www to anyone

Named Lists - these start with "ip access-"

```
Lab_A(config)#ip access-list standard BlockSales
Lab_A(config-std-nacl)#deny 172.16.40.0 0.0.0.255
Lab_A(config-std-nacl)#permit any
Lab_A(config-std-nacl)#exit
Lab_A(config)#int fa0/1
Lab_A(config-if)#ip access-group BlockSales out
```

Adding #comments#

```
R2(config)#access-list 110 remark Permit Bob from Sales Only To Finance
R2(config)#access-list 110 permit ip host 172.16.40.1 172.16.50.0 0.0.0.255
R2(config)#access-list 110 deny ip 172.16.40.0 0.0.0.255 172.16.50.0 0.0.0.255
R2(config)#ip access-list extended No_Telnet
R2(config-ext-nacl)#remark Deny all of Sales from Telnetting to Marketing
R2(config-ext-nacl)#deny tcp 172.16.40.0 0.0.0.255 172.16.60.0 0.0.0.255 eq 23
R2(config-ext-nacl)#permit ip any any
```

Adding a line to the sequence list:

```
Lab_A(config)#do show access-list
Extended IP access list 110
    10 deny tcp any host 172.16.30.5 eq ftp
    20 deny tcp any host 172.16.30.5 eq telnet
    30 permit ip any any
    40 permit tcp host 192.168.177.2 host 172.22.89.26 eq www
    50 deny tcp any host 172.22.89.26 eq www
Lab_A (config)#ip access-list extended 110
Lab_A (config-ext-nacl)#21 deny udp any host 172.16.30.5 eq 69
```

show access-list - displays all access lists and parameters. Shows stats on usage. Doesn't show interfaces.
show access-list 110 - Reveals parameters for 110. Doesn't show interfaces.
show ip access-list - Shows only the IP access lists configured on the router.
show ip interface - Displays which interfaces have access lists set on them.
show running-config - Shows the access lists and the specific interfaces that have ACLs applied on them

Remember: You can’t have two lists on the same interface, in the same direction- one will overwrite the other. configuration.

This extended ACL is used to permit traffic on the 10.1.1.x network (inside) and to receive ping responses from the outside while it prevents unsolicited pings from people outside, permitting all other traffic.

```
interface Ethernet0/1
ip address 172.16.1.2 255.255.255.0
ip access-group 101 in
access-list 101 deny icmp any 10.1.1.0 0.0.0.255 echo
access-list 101 permit ip any 10.1.1.0 0.0.0.255
```

Note: Some applications such as network management require pings for a keepalive function. If this is the case, you might want to limit blocking inbound pings or be more granular in permitted/denied IPs.

OSPF - Link State Routing - Dijkstra's algorithm

Central1(config)#int loopback 0
Central1(config-if)#ip address 172.31.1.1 255.255.255.255
Central1(config)#router ospf 132
Central1(config-router)#network 10.10.10.1 0.0.0.0 area 0
Central1(config-router)#network 172.16.10.0 0.0.0.3 area 1 <--this mask is 255 minus mask, per octet (here 252)
Central1(config-router)#network 172.16.10.4 0.0.0.3 area 2
Central1(config-router)#passive-interface f0/0 <--- says no neighbors should be discovered on the interface

Have a default route for your networks (so they know the way out), propagate to other routers.
Central1(config)#ip route 0.0.0.0 0.0.0.0 Fa0/0
Central1(config)#router ospf 1
Central1(config-router)#default-information originate

Loopback and Router-IDs in OSPF
If you don't configure a loopback interface on a router, the highest active IP address on a router will become that router's RID during bootup if it isn't already specified. A loopback interface will not override the router-id command, and we don't have to reboot the router to make it take effect as the RID.

1. Highest IP on an active interface by default.
2. Highest logical interface IP (loopback) overrides a physical interface.
3. The router-id overrides the interface and loopback interface.

Reload or use "clear ip ospf process" command, to force changes to take effect

IPv6 OSPF Routing - Routing info is also attached to the interface.
Central1(config)#int f0/0
Central1(config-if)#ipv6 ospf 1 area 0
The same IPv4 OSPF RID is still there- change the RID under the OSPF process ID in the global config:
Central1(config)#ipv6 router ospf 1
Central1(config-rtr)#router-id 1.1.1.1
Central1(config-rtr)#do clear ip ospf process

Designated Router Elections
- By default, all OSPF routers are assigned a DR priority of 1.
- Elections are first won based upon a router's priority level, then highest RID (often the highest IP address)
- Backup designated router (BDR) is the runner-up in the DR election, a standby for the DR
  - receives all routing updates but does not distribute them; steps in if the DR goes down

Suppose we wanted R1 to become the DR and R2 to become the BDR:
R1(config)# **interface f0/0**
R1(config-if)# **ipv6 ospf priority 100**
R2(config)# interface f0/0
R2(config-if)# **ipv6 ospf priority 90**

Prevent routers from ever becoming a DR or BDR by configuring a priority of zero
R2(config)# interface f0/0
R2(config-if)# **ipv6 ospf priority 0**

Choosing the best route - cost=reference_bandwidth /interface_bandwidth

Interface	Default Bandwidth	Formula (Kbps)	OSPF Cost
Serial	1544 Kbps	100,000/1544	64
Ethernet	10,000 Kbps	100,000/10,000	10
Fast Ethernet	100,000 Kbps	100,000/100,000	1

The default ref bandwidth assumes the fastest link in the network runs at 100 Mbps.
Use **auto-cost reference-bandwidth 10000 to** accommodate links up to 10 Gbps in speed.

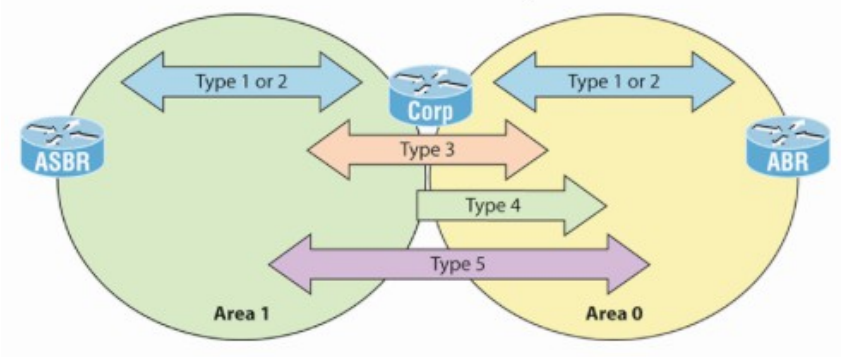
Here are the rules for how a router sets its OSPF interface costs:
- Set the cost explicitly, with **ip ospf cost x interface**, to a value between 1-65,535
- To affect all routes on the device, use **distance 80**. This also affects the AD of non-OSPF routes.
- Change interface bandwidth with the **bandwidth** *speed* command, with speed in Kbps
- Change the reference bandwidth, using subcommand **auto-cost reference-bandwidth** *ref-bw,* in Mbps

Instead of copying routes into the routing table from the LSDB, a router must do the SPF math, choose the best route, and add the route with a subnet number/mask, an outgoing interface, and a next-hop router IP address.

Left: R1–R7–R8 10 + 180 + 10 = 200
Middle: R1–R5–R6–R8 20 + 30 + 40 + 10 = 100
Right: R1–R2–R3–R4–R8 30 + 60 + 20 + 5 + 10 = 125

De facto load balancing occurs when metrics tie for multiple routes to the same subnet, the router can put multiple routes in the routing table (the default is four max- use **maximum-paths** subcommand to change).

Link-State Advertisements - Basic LSAs
Type 1 LSA - Router LSA - The common routers describe themselves amongst themselves
Type 2 - Network LSAs - The DR describes itself and it's network to it's area routers
Type 3 LSA - Summary LSAs - Area Border Routers brag to other areas about thier contents
Type 4 LSAs - Area Border Routers give directions to the ASBR
Type 5 LSAs - Autonomous System LSAs - AS external link advertisements



Timers have to match for neighborship
- Flood changed LSAs to neighbors
and re-flood unchanged LSAs when their lifetime expires (default 30 min)
- Maintain neighborship with Hello msgs, listening for Hellos before the dead/hold interval expires.
    - the dead/hold timer is 4x the hello interval
    - 10-sec hello timer and 40-sec hold timer on broadcast and point-to-point links
    - 30-sec hello timer and 120-sec hold timer for all other network types
If a router goes down, the LSDB is populated so all of the routers can recalculate their Shortest Paths

Becoming neighbors. When a point-to-point link comes up, the routers on the ends go through interim neighbor states (INIT and 2-Way) exchanging their basic LSAs for discovery, compatibility checks (routerIDs, timers). When both reach 2-Way state, it means they both acknowledge each other as neighbors with matching settings and are prepared to exchange LSDB's. ExStart/Exchange states are when they share DB descriptors, Loading means sending each other Link-State Updates, and Full means their LSDB's match with the same LSA info. These LSA structures comprise the topology. Down means no Hellos are received yet; and Attempt is when NBMA needs manual config.

LSAs in a Multi-Area Design
Migrating from a single-area design to a multi-area design has a couple of effects on LSAs:
- Each area has a smaller number of router and network LSAs.
- ABRs have a copy of the LSDB for each area they connect to and submit a router LSA in each area's LSDB.
- Each area has a need for summary (Type 3) LSAs to describe subnets in other areas.

OSPF: General Verification and Troubleshooting Commands
sh int [**brief**] **x0/0** - up/down, IP address, encapsulation, keepalive, and loopback status
show ip ospf interface (**brief**) - address, area, process, router ID, cost, state, priority, timers, DR and BDR
sh ip ospf - router ID, area information, SPF statistics, and LSA timer information
show ip route ospf - shows the routing table, and displays any injected routes
show ip ospf neighbor (**detail**)- neighbor table, related interfaces, hold time, queue, seq number, DR and BDR
show ip ospf database - view topology table - output per LSA, organized by type, neighbor RIDs, etc
show ip ospf topology (or topology-info)- displays topology table
show ip ospf traffic - shows the # and type of traffic received
show ip protocols - shows the routing process ID and which protocols are enabled, K values
show key-chain - display authentication info
show ip ospf border-routers
show ip ospf data --lots more on links

- Type 1, or Router LSA, is created and advertised by every router.
- Each internal router has one Router LSA, while each ABR has multiple Router LSAs, one for each area.
- Router LSAs are flooded throughout its (intended) area by sending a copy to all connected neighbors.

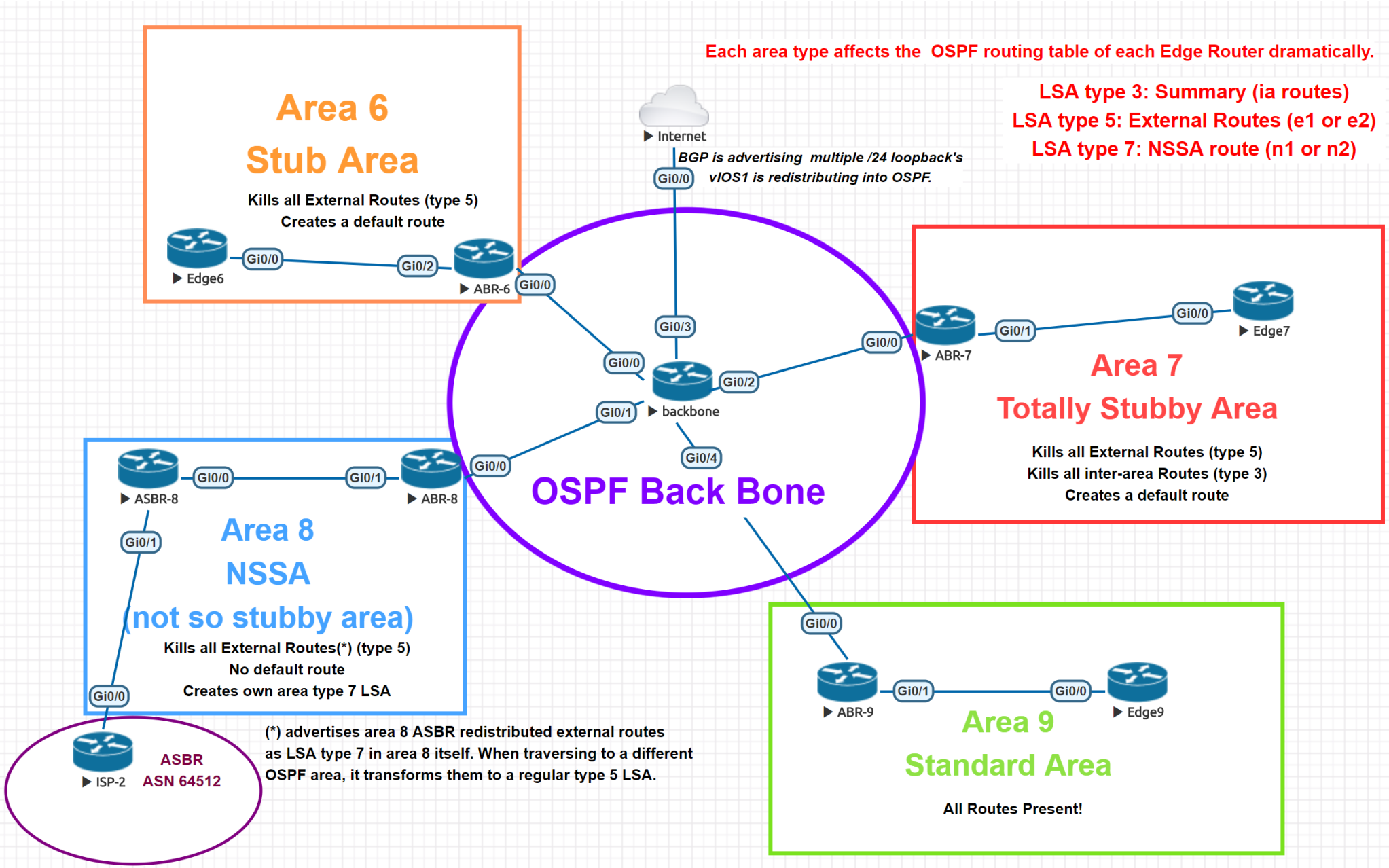
Each one includes: all neighbors directly connected, the router Interface address the LSA was sent

Basic LSA Layout (This can vary but is a basic look)

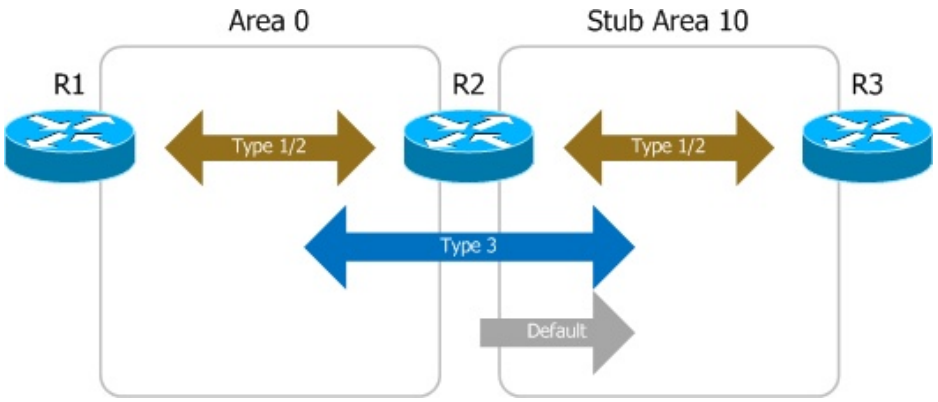
<table>
<tr><td colspan="6">20-byte LSA Header</td></tr>
<tr><td>0 [4 bits]</td><td>V = on if virtual link</td><td>E = on if ASBR</td><td>B = on if ABR</td><td>0 [9 bits]</td><td>Number of Links: count of all router links [16 bits]</td></tr>
<tr><td colspan="6">Link state identifier ID [32 bits]</td></tr>
<tr><td colspan="6">Link Data [32 bits]</td></tr>
<tr><td colspan="2">Type [8 bits]</td><td colspan="2">Number TOS = 0</td><td colspan="2">Metric</td></tr>
<tr><td colspan="6">Starting from Link ID again...</td></tr>
</table>

An overview of Link State Advertisement Types

LSA Type	Description	Routing Table Code
1	Router LSA. Advertises intra-area routes. Generated by each OSPF router to ID itself to peers. Flooded only within the area. Represents stub networks as well	O
2	Network LSA. Generated by a DR, advertises routers on a multiaccess link.. Flooded only within the area. Represents the subnet and the router interfaces connected to the subnet. One per transit network.	O
3	Network Summary LSA. Advertises interarea routes. Generated by an ABR, flooded to adjacent areas. Defines the links (subnets) in the origin area, and cost, but no topology data. "I am the ABR and these are my area's subnets"	O IA
4	ASBR Summary LSA. Advertises the route to reach an ASBR. Generated and advertised by ABR. Flooded to adjacent areas.	O IA
5	AS External LSA. Advertises external routes (in another routing domain) Generated by an ASBR and flooded to adjacent areas. E1–metric increases at each router as it is passed through the network. E2–metric does not increase (this is the default).	O
6	Multicast LSA. Used in multicast OSPF (MOSPF) operations, not supported by Cisco IOS.	
7	Not-so-stubby area (NSSA) LSA. Advertises routes in another routing domain. Generated by an ASBR, advertised within a NSSA (instead of a type-5 LSA) **Totally NSSA bends this rule- that's why it accepts Type 4 LSA.** N1–metric increases as it is passed through the network. N2–metric does not increase (default).	O
8	External Attributes LSA, aka Link LSA. For OSPF and BGP interaction. Not implemented in Cisco. **Another definition at Cisco says this is an IPv6 tool:** "IPv6 local-link flooding scope (never flooded beyond), provide the link-local address to all other routers attached to the link, also gives a list of IPv6 prefixes to associate with the link. Allows the router to assert a collection of Options bits to associate with the network LSA that will be originated for the link"	
9	Intra-Area-Prefix LSAs - A router can originate multiple intra-area-prefix LSAs for each router or transit network, each with a unique link-state ID. The link-state ID for each intra-area-prefix LSA describes its association to either the router LSA or the network LSA and contains prefixes for stub and transit networks.	
8,10,11	Opaque LSAs. Used for specific applications, these are generic LSAs to allow for easy future extension of OSPF; for example, type 10 has been adapted for MPLS traffic, BGP, IPv6 stuff. Others more vendor-specific- are becoming more defined.	


Each area type affects the OSPF routing table of each Edge Router dramatically.
LSA type 3: Summary (ia routes)
LSA type 5: External Routes (e1 or e2)
LSA type 7: NSSA route (n1 or n2)
Area 6
Stub Area
Kills all External Routes (type 5)
Creates a default route
Edge6
Gi0/0
Gi0/2
ABR-6
Gi0/0
Internet
BGP is advertising multiple /24 loopback's
vIOS1 is redistributing into OSPF.
Gi0/0
Gi0/3
Gi0/0
Gi0/2
Gi0/1
backbone
Gi0/4
OSPF Back Bone
Gi0/0
ABR-7
Gi0/1
Gi0/0
Edge7
Area 7
Totally Stubby Area
Kills all External Routes (type 5)
Kills all inter-area Routes (type 3)
Creates a default route
ASBR-8
Gi0/0
Gi0/1
ABR-8
Gi0/1
Area 8
NSSA
(not so stubby area)
Kills all External Routes(*) (type 5)
No default route
Creates own area type 7 LSA
Gi0/0
ISP-2
ASBR
ASN 64512
(*) advertises area 8 ASBR redistributed external routes
as LSA type 7 in area 8 itself. When traversing to a different
OSPF area, it transforms them to a regular type 5 LSA.
Gi0/0
ABR-9
Gi0/1
Gi0/0
Edge9
Area 9
Standard Area
All Routes Present!

Stub Areas



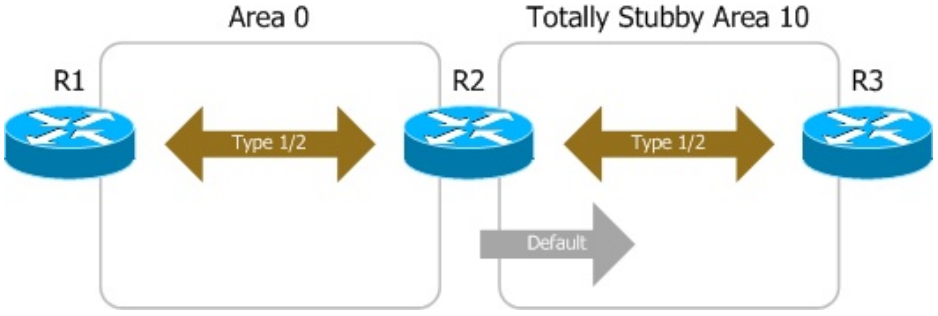
In this next example, R2 and R3 share a common stub area. Instead of propagating external routes (type 5 LSAs) into the area, the ABR injects a type 3 LSA containing a default route into the stub area. This ensures that routers in the stub area will be able to route traffic to external destinations without having to maintain all of the individual external routes. Because external routes are not received by the stub area, ABRs also do not forward type 4 LSAs from other areas into the stub.

For an area to become a stub, all routers belonging to it must be configured to operate as such. Stub routers and non-stub routers will not form adjacencies.

```
Router(config-router)# area 10 stub
```

This idea of substituting a single default route for many specific routes can be applied to internal routes as well, which is the case of *totally stubby areas*.

Totally Stubby Areas



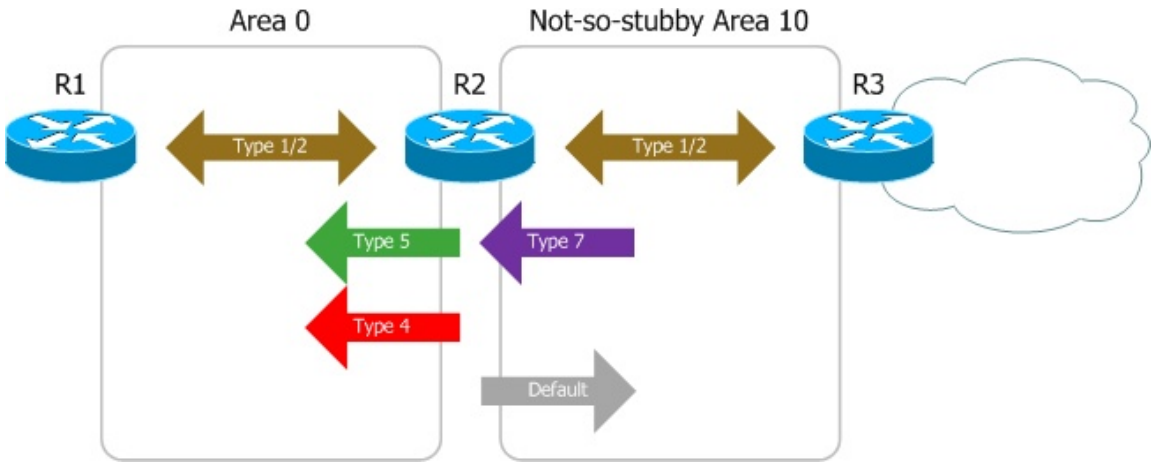
Like stub areas, totally stubby areas do not receive type 4 or 5 LSAs from their ABRs. However, they also do not receive type 3 LSAs; all routing out of the area relies on the single default route injected by the ABR.

A stub area is extended to a totally stubby area by configuring all of its ABRs with the `no-summary` parameter:

```
Router(config-router)# area 10 stub no-summary
```

Stub and totally stubby areas can certainly be convenient to reduce the resource utilization of routers in portions of the network not requiring full routing knowledge. However, neither type can contain an ASBR, as type 4 and 5 LSAs are not permitted inside the area. To solve this problem, and in what is arguably the worst naming decision ever made, Cisco introduced the concept of a *not-so-stubby area (NSSA)*.

Not-so-stubby Areas



An NSSA makes use of type 7 LSAs, which are essentially type 5 LSAs in disguise. This allows an ASBR to advertise external links to an ABR, which converts the type 7 LSAs into type 5 before flooding them to the rest of the OSPF domain.

An NSSA can function as either a stub or totally stubby area. To designate a normal (stub) NSSA, all routers in the area must be so configured:

```
Router(config-router)# area 10 nssa
```

Type 3 LSAs will pass into and out of the area. Unlike a normal stub area, the ABR will *not* inject a default route into an NSSA unless explicitly configured to do so. As traffic cannot be routed to external destinations without a default route, you'll probably want to include one by appending `default-information-originate` (thanks to Adam for pointing this out).

```
Router(config-router)# area 10 nssa default-information-originate
```

To expand an NSSA to function as a totally stubby area, eliminating type 3 LSAs, all of its ABRs must be configured with the `no-summary` parameter:

```
Router(config-router)# area 10 nssa no-summary
```

The ABR of a totally stubby NSSA (or not-so-totally-stubby area, if you prefer) injects a default route without any further configuration.

Summary

- **Standard areas** can contain LSAs of type 1, 2, 3, 4, and 5, and may contain an ASBR. The backbone is considered a standard area.
- **Stub areas** can contain type 1, 2, and 3 LSAs. A default route is substituted for external routes.
- **Totally stubby areas** can only contain type 1 and 2 LSAs, and a single type 3 LSA. The type 3 LSA describes a default route, substituted for all external and inter-area routes.
- **Not-so-stubby areas** implement stub or totally stubby functionality yet contain an ASBR. Type 7 LSAs generated by the ASBR are converted to type 5 by ABRs to be flooded to the rest of the OSPF domain.

EIGRP - Enhanced Interior Gateway Routing Protocol

- A "balanced hybrid" routing protocol, instead of link-state/ distance vector. Sometimes "adv distance vector"
- Sends routing table once, and uses partial updates for changes
- Updates to multicast 224.0.0.10; and the unicast IP of the neighbor if specific.
- Reliable Transport Protocol (RTP) to send updates - resend any EIGRP messages that are not received
- For route poisoning (communicating that a route has "failed" to either warn or stop traffic over the route); EIGRP uses 2^{32}-1 as "infinity" (just over 4 billion), some new IOS versions raise that value to 2^{56}-1 (over 10,000 trillion)

EIGRP default timers are 5 and 15 seconds for Hello and Hold Intervals (Hold is default 3x Hello interval) Not required, but they should be the same for stability. (if Hold time expires it's a sign the other router is down)

For Neighborship: same configured autonomous system number (ASN), the source IP in the neighbor's Hello must be from the same subnet as the interface, matching K values, and pass authentication process if needed.

3-steps when a router first joins a network, 3 tables to populate

- Neighbor discovery: Hello messages discover potential neighboring EIGRP routers to add to neighbor table.
- Topology exchange: Exchange full topology update, only partial updates later as needed on changes
- Routing table: Analyzes topology tables, choose lowest-metric route to each subnet.

Reported Distance (RD) and Feasible Distance (FD), Routing and Topology Tables

- Reported/advertised distance (RD) is how far a neighbor says it is from a remote network.
- Feasible distance (FD) is calculated by taking that and adding how far away that neighbor is.
- The route with the lowest FD is the route that you'll find in the routing table - it's considered the best path.

D 10.0.0.0/8 [90/2195456] via 172.16.10.2, 00:27:06, Serial0/0
D (for DUAL) says this an EIGRP injected route. Get to the 10.0.0.0 network via its neighbor 172.16.10.2.
The first number (90) is the administrative distance (AD) of the routing protocol, the second number is the feasible distance (FD)- the entire cost for this router to get to network 10.0.0.0/8 through that neighbor, 2195456

- The neighbor router has a reported (advertised) distance (RD) between it and network 10.0.0.0
- Feasible Distance (FD), or total cost to that network adds RD to the calculated distance to that router.

Successor Routes and Feasible Successors

Each router keeps information about adjacent neighbors in the neighbor table and the topology tables hold all destinations advertised by the neighbors, with associated distances and other metrics. That best path is called a successor route, and it is determined by which path has the lowest Feasible Distance.

Also elected is a backup "feasible successor" route, which is kept in the topology table if they are needed. If a nonsuccessor route's RD is less than the FD of the successor, the route is a feasible successor route.
Router E has 3 routers/ paths to choose to get to Router A/ Subnet 1:

Router E Topology Table:		**Router E Routing Table**
Next Hop FD RD		Subnet 1 **Metric 14,000**, Through D
Router B - 19,000	15,000	
Router C - 17,500	**13,000 <-- RD makes it FS**	
Router D - **14,000**	10,000 **<-- FD makes it SR**	

Rule is, the lowest FD will determine the Successor Route, and Router D wins with a FD of 14000. You might think that the next smaller FD would be the runner up, right? Absolutely not. In the case of a Feasible Successor, we look at the Successor Route 's FD and find a Reported Distance that is smaller than *that*. Router C wins out as Feasible Successor by the rule because it's RD 13,000 < FD of the current Successor at 14,000.

Verifying EIGRP - The Configuration and the Neighbor, Topology and Routing Tables

show ip eigrp interfaces
show ip eigrp interfaces detail
show ip eigrp interfaces *type number*
show ip protocols
show ip eigrp neighbors
show ip eigrp neighbors *type number*
show ip eigrp topology
show ip eigrp topology *subnet/prefix*
show ip route
show ip route eigrp
show ip route *subnet mask*
show ip route | section *subnet*

EIGRP Metric Calculations

$$\text{Metric} = \left(\left(\frac{10^7}{\text{least-bandwidth}}\right) + \text{cumulative-delay}\right) * 256$$

In this formula, the term least-bandwidth represents the lowest-bandwidth link in the route, using a unit of Kbps. For instance, if the slowest link in a route is a 10Mbps Ethernet link, the first part of the formula is 10^7 / 10^4, which equals 1000. You use 10^4 in the formula because 10Mbps is equal to 10,000 Kbps (10^4 Kbps).
Cumulative-delay is the sum of delay values for all outgoing interfaces in the route, *in tens of microseconds* (µsec) (most **show** commands, including **show ip eigrp topology** and **show interfaces**, list delay in µseconds)
You can set both bandwidth and delay for each link with **bandwidth** and **delay** interface commands.

A Simple Example
R1 learns about subnet 10.1.3.0/24 from R2's update listing min. BW of 100,000 Kbps, and delay of 100 µsec.
R1's S0/1 has an interface bandwidth set to 1544 Kbps and a delay of 20,000 µsec
- 1.544 Mbps < 100,000 Kbps, or 100 Mbps. R1 needs to use this slower bandwidth in the metric calculation, (data's being squeezed down a smaller pipe)
- For interface delay, the router always adds its interface delay to the delay listed in the EIGRP Update.
- The update lists 100 µseconds (10 tens of µseconds.

Add R1's S0/1 is 2000 tens of = 2010 tens of µsec.
Metric = [(10^7 / 1544) + (10 + 2000)] * 256 = 2,172,416
- Later this is revisited, for Reported Distance that R1 was given. It turns out we have the info already:

Metric = [(10^7 / 100,000) + (10)] * 256 = 28,160 - So FD for R1 calc'd at 2,172,416; and the RD from R2: 28160
- On R1, it listed in the sh ip route like this with the AD/FD:
D 10.1.3.0/24 [90/2172416] via ???.??.??.?, 23:58:00, S0/1 [no IP address for R2's Serial port (ooops)]
sh ip eigrp topology might have something like this, which shows the FD/RD
P 10.1.3.0/24, 1 successors, FD is 2172416 via ?.?.?.? (2172416/28160), Serial0/1

The Entire Equation
Metric = {K1 × Bandwidth + [(K2 × Bandwidth) / (256 – Load)] + K3 × Delay} × [K5 / (Reliability + K4)]
K1 Bandwidth (Be) = lowest bandwidth of the links along the path - [10000000 / Kbps] × 256]
K2 Load (utilization on path)
K3 Delay (Dc) = sum of all the delays of the links along the path - [in 10s of µsec] × 256
K4 Reliability (r)
K5 MTU

- K values are only multiples of the metric calculation. Default values are: K1=1, K2=0, K3=1, K4=0, K5=0.
- K1 and K3 are static. K2, K4, and K5 are variable, and we just get overhead calculating them and reporting them if they are on. MTU is exchanged between EIGRP neighbors but not used. By default only K1 and K3 are enabled and we don't use K2 or K4. (only bandwidth and delay are used in the formula)

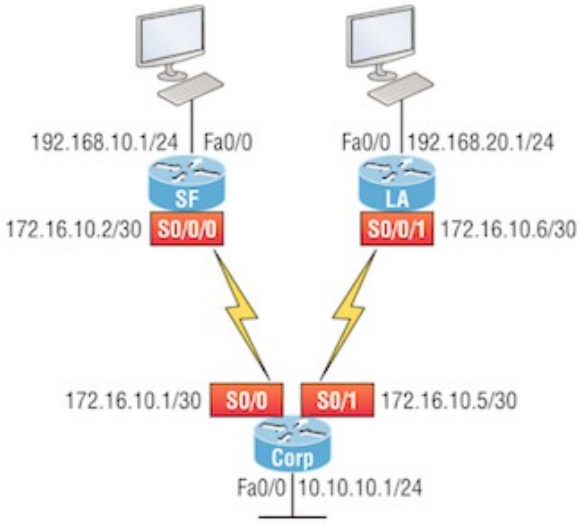
Serial Links and a Special Problem
Serial links default to a bandwidth of 1544 and a delay of 20,000 µseconds. A slow 64 Kbps serial link's **bandwidth** command is used to reflect the correct speed (instead of the default 1544

<u>EIGRP Configuration Commands</u>
router eigrp as-number
network ip-address [wildcard-mask] (address should match an int, and often wildcards are unavoidable*)
eigrp router-id value
ip hello-interval eigrp asn time
[for interfaces: **ip hold-time eigrp** asn time]
[for interfaces: **bandwidth** value
[for interfaces: **delay** value
maximum-paths number
variance multiplier to tweak k-values
auto-summary

Failed Routes, DUAL Query and Reply to Update Routing Table
When a route fails, and there's no feasible successor, EIGRP uses Diffusing Update Algorithm (DUAL) to choose a replacement. DUAL sends queries looking for a loop-free route, and when it's found, DUAL adds it to the routing table. Messages just confirm that a route exists, and would not create a loop. In most networks, convergence can still occur in less than 10 seconds.

Basic IGP Routing in IPv4

The three types of routing are static (in which routes are manually configured at the CLI), dynamic (in which the routers share routing information via a routing protocol), and default routing (in which a special route is configured for all traffic without a more specific destination network found in the table)



Basic layout to setup
Central1 - in diagram is CORP
Serial 0/0: 172.16.10.1/30
Serial 0/1: 172.16.10.5/30
Fa0/0: 10.10.10.1/24

DFW - in diagram is SF
S0/0/0: 172.16.10.2/30
Fa0/0: 192.168.10.1/24

HOU - in diagram is LA
S0/0/0: 172.16.10.6/30
Fa0/0: 192.168.20.1/24

Basic Local IPv4 Routing (Direct Connections)

Every interface below should have "no shutdown" added. It has been removed here for brevity

```
Router(config)#hostname Central1
Central1(config)#int f0/0
Central1(config-if)#desc Connection to LAN
Central1(config-if)#ip address 10.10.10.1 255.255.255.0
Central1(config-if)#int s0/0
Central1(config-if)#desc WAN connection to DFW
Central1(config-if)#ip address 172.16.10.1 255.255.255.252
Central1(config-if)#int s0/1
Central1(config-if)#desc WAN connection to HOU
Central1(config-if)#ip address 172.16.10.5 255.255.255.252
```

Central1>sh controllers s0/0

Interface Serial0/0
Hardware is PowerQUICC MPC860
DTE V.35 TX and RX clocks detected.

Remember to set clock rates on the DCEs!

```
Router(config)#hostname DFW
DFW(config)#int s0/0/0
DFW(config-if)#desc WAN Connection to Central1
DFW(config-if)#ip address 172.16.10.2 255.255.255.252
DFW(config-if)#clock rate 1000000   - DFW's DCE to Central's DTE -we need a clock rate!
DFW(config-if)#int f0/0
DFW(config-if)#desc DFW LAN
DFW(config-if)#ip address 192.168.10.1 255.255.255.0

Router(config)#hostname HOU
HOU(config)#int s0/0/1
HOU(config-if)#ip address 172.16.10.6 255.255.255.252
HOU(config-if)#clock rate 1000000        - HOU's DCE to Central1's DTE -we need a clock rate
HOU(config-if)#description WAN To Central1
HOU(config-if)#int f0/0
HOU(config-if)#ip address 192.168.20.1 255.255.255.0
HOU(config-if)#description HOU LAN
```

Static Routing: Manually Adding to Routing Table

```
Router(config)#ip route 172.16.3.0 255.255.255.0 192.168.2.4
```

172.16.3.0 is the remote network, and 255.255.255.0 is it's mask of the remote network.
192.168.2.4 is the next hop that packets will be sent to. Can also be the interface out to it.

Below, the 150 is where the administrative distance goes if you want to override the default. This is set here because in the next example, we will add routes with RIP, and using 150 (rather than the default of 1) means we won't have to remove static routes first- RIP (120) will just override them.

Administrative Distances (the smaller # wins out):

Connected	0	OSPF	110
Static route	1	RIP	120
EIGRP summary	5	iBGP	200
eBGP	20	Unknown	255
EIGRP	90		

Central1(config)#ip route 192.168.10.0 255.255.255.0 172.16.10.2 150 --use the dest IP out
Central1(config)#ip route 192.168.20.0 255.255.255.0 s0/1 150 -- use an interface out

DFW(config)#ip route 10.10.10.0 255.255.255.0 172.16.10.1 150
DFW(config)#ip route 172.16.10.4 255.255.255.252 172.16.10.1 150
DFW(config)#ip route 192.168.20.0 255.255.255.0 172.16.10.1 150

HOU#config t
HOU(config)#ip route 10.10.10.0 255.255.255.0 172.16.10.5 150
HOU(config)#ip route 172.16.10.0 255.255.255.252 172.16.10.5 150
HOU(config)#ip route 192.168.10.0 255.255.255.0 172.16.10.5 150

A stub indicates that the networks in this design have only one way out to reach all other networks; use only a default route. Here's an alt config instead of typing in the static routes. to make this a "stubby network" (first we turn off the routes we just configured):
HOU(config)#no ip route 10.10.10.0 255.255.255.0 172.16.10.5 150
HOU(config)#no ip route 172.16.10.0 255.255.255.252 172.16.10.5 150
HOU(config)#no ip route 192.168.10.0 255.255.255.0 172.16.10.5 150
HOU(config)#ip route 0.0.0.0 0.0.0.0 172.16.10.5

Convert from Static IPv4 to Dynamic IPv4 RIPv2
As a shortcut, you can verify directly connected networks to know what to configure RIP with:
Central1#sh ip int brief

Interface	IP-Address	OK?	Method	Status	Protocol
FastEthernet0/0	10.10.10.1	YES	manual	up	up
Serial0/0	172.16.10.1	YES	manual	up	up
FastEthernet0/1	unassigned	YES	unset	admin. down	down
Serial0/1	172.16.10.5	YES	manual	up	up

> *When we declare our routes in RIP, we need to refer to them in a classful way instead of by the specific subnet. This is illustrated below. RIP then finds the subnets and fills in the routing table (RIP is NOT a classful routing protocol btw).*

Central1(config)#router rip
Central1(config-router)#network 10.0.0.0
Central1(config-router)#network 172.16.0.0
Central1(config-router)#version 2
Central1(config-router)#no auto-summary

Add the default static route, and enter RIP config to set default-info originate to propagate it.
Central1(config)#ip route 0.0.0.0 0.0.0.0 Fa0/0
Central1(config)#router rip
Central1(config-router)#default-information originate

DFW(config)#router rip
DFW(config-router)#network 192.168.10.0
DFW(config-router)#network 172.16.0.0
DFW(config-router)#version 2
DFW(config-router)#no auto-summary *---disabling auto-summary makes RIP look at our subnets*
DFW(config-router)#do show ip route

```
C    192.168.10.0/24 is directly connected, FastEthernet0/0
L    192.168.10.1/32 is directly connected, FastEthernet0/0
     172.16.0.0/30 is subnetted, 3 subnets
R       172.16.10.4 [120/1] via 172.16.10.1, 00:00:08, Serial0/0/0
C       172.16.10.0 is directly connected, Serial0/0/0
L       172.16.10.2/32 is directly connected, Serial0/0
S    192.168.20.0/24 [150/0] via 172.16.10.1
     10.0.0.0/24 is subnetted, 1 subnets
R       10.10.10.0 [120/1] via 172.16.10.1, 00:00:08, Serial0/0/0
```

- Note above the administrative distances set for RIP (R) and our previous static routes (S)

HOU(config)#no ip route 0.0.0.0 0.0.0.0 *--- get rid of that temporary default route we set for HOU*
HOU(config)#router rip
HOU(config-router)#network 192.168.20.0
HOU(config-router)#network 172.16.0.0
HOU(config-router)#no auto
HOU(config-router)#vers 2

Switch from IPv4 RIP to Dynamic IPv4 OSPF
Central1(config)#no router rip
Central1(config)#router ospf 132
Central1(config-router)#network 10.10.10.1 0.0.0.255 area 0
Central1(config-router)#network 172.16.10.1 0.0.0.3 area 0
Central1(config-router)#network 172.16.10.5 0.0.0.3 area 0

DFW(config)#no router rip
DFW(config)#router ospf 300
DFW(config-router)#network 192.168.10.1 0.0.0.255 area 0
DFW(config-router)#network 172.16.10.0 0.0.0.7 area 0
[may need "area 1 range" for this, since it's a summary of two listed on Central1- see below]

HOU(config)#router ospf 100
HOU(config-router)#network 192.168.20.0 0.0.0.255 area 0
HOU(config-router)#network 172.16.10.0 0.0.0.7 area 0

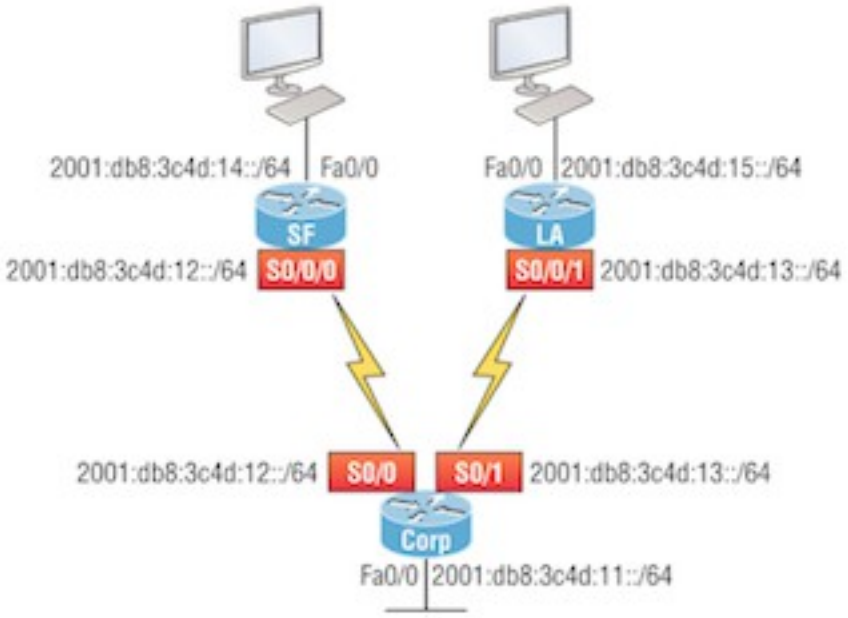
Adding a non-OSPF network? Use passive-interface:
HOU(config)#router ospf 100
HOU(config-router)#passive-interface fastEthernet 0/1

Add a SanAnt router:
Router(config)#hostname SanAnt
SanAnt(config)#int f0/0
SanAnt(config-if)#ip address 10.10.10.2 255.255.255.0
SanAnt(config-if)#no shut
SanAnt(config-if)#router ospf 2
SanAnt(config-router)#network 10.10.10.0 0.0.255.255 area 0

OSPF Route summarization:
Router(config)#router ospf 100
Interarea: area 1 range 192.168.5.8 0.0.0.63
External: summary-address: 192.168.64.0 0.0.224.0

RouterID - So what if you didn't put in a loopback first and need to force a router to become DR?
Corp(config-router)#router-id 223.255.255.254
Corp(config-router)#do clear ip ospf process
[Just set a routerID which will beat any other RID or loopback set on other routers, then reset OSPF]

Basic IGP Routing in IPv6



Adding IPv6 (including local default routing)

Add IPv6 to the Central1, DFW, and HOU routers by using a simple subnet scheme of 11, 12, 13, 14, and 15

```
Central1#config t
Central1(config)#ipv6 unicast-routing
Central1(config)#int f0/0
Central1(config-if)#ipv6 address 2001:db8:3c4d:11::/64 eui-64
Central1(config-if)#int s0/0
Central1(config-if)#ipv6 address 2001:db8:3c4d:12::/64 eui-64
Central1(config-if)#int s0/1
Central1(config-if)#ipv6 address 2001:db8:3c4d:13::/64 eui-64

DFW(config)#ipv6 unicast-routing
DFW(config)#int s0/0/0
DFW(config-if)#ipv6 address 2001:db8:3c4d:12::/64 eui-64
DFW(config-if)#int fa0/0
DFW(config-if)#ipv6 address 2001:db8:3c4d:14::/64 eui-64

HOU(config)#ipv6 unicast-routing
HOU(config)#int s0/0/1
HOU(config-if)#ipv6 address 2001:db8:3c4d:13::/64 eui-64
HOU(config-if)#int f0/0
HOU(config-if)#ipv6 address 2001:db8:3c4d:15::/64 eui-64
```

Static IPv6 Routing

First static route line uses the next-hop address, and the exit interface on the second entry

(On the DFW router, use show ipv6 int brief, and then copy the interface address used for the next hop)

```
Central1(config)#ipv6 route 2001:db8:3c4d:14::/64  2001:DB8:3C4D:12:21A:2FFF:FEE7:4398 150
Central1(config)#ipv6 route 2001:DB8:3C4D:15::/64 s0/1 150
Central1(config)#do sho ipv6 route static
        S   2001:DB8:3C4D:14::/64 [150/0]
            via 2001:DB8:3C4D:12:21A:2FFF:FEE7:4398
```

For DFW and HOU routers put a single entry in each router to get to remote subnet 11 (Central1):

```
DFW(config)#ipv6 route 2001:db8:3c4d:11::/64 s0/0/0 150
HOU(config)#ipv6 route ::/0 s0/0/1  -- a default route
```

IPv6 RIPng (a different pair of routers for this one)

```
Austin(config)#ipv6 unicast-routing
Austin(config)#interface fastethernet 0/0
Austin(config-if)#ipv6 enable
Austin(config-if)#ipv6 address 2001:db8:c18:2::/64 eui-64
Austin(config-if)#ipv6 rip RIPNG1 enable
Austin(config-if)#interface fastethernet 0/1
Austin(config-if)#ipv6 enable
Austin(config-if)#ipv6 address 2001:db8:c18:1::/64 eui-64
Austin(config-if)#ipv6 rip RIPNG1 enable
Austin(config-if)#no shutdown

Houston(config)#ipv6 unicast-routing
Houston(config)#interface fastethernet 0/0
Houston(config-if)#ipv6 enable
Houston(config-if)#ipv6 address 2001:db8:c18:2::/64 eui-64
Houston(config-if)#ipv6 rip RIPNG1 enable
Houston(config-if)#interface fastethernet 0/1
Houston(config-if)#ipv6 enable
Houston(config-if)#ipv6 address 2001:db8:c18:3::/64 eui-64
Houston(config-if)#ipv6 rip RIPNG1 enable
```

Switch to IPv6 OSPF Routing

```
Central1(config)#int f0/0
Central1(config-if)#ipv6 ospf 1 area 0
Central1(config-if)#int s0/0
Central1(config-if)#ipv6 ospf 1 area 0
Central1(config-if)#int s0/1
Central1(config-if)#ipv6 ospf 1 area 0

DFW(config)#int f0/0
DFW(config-if)#ipv6 ospf 1 area 0
DFW(config-if)#int s0/0/0
DFW(config-if)#ipv6 ospf 1 area 0
[Same for others]

SanAnt(config)#int f0/0
SanAnt(config-if)#ipv6 address autoconfig default
SanAnt(config-if)#ipv6 ospf 1 area 0
```

The same IPv4 OSPF RID is still there- change the RID under the OSPF process ID in the global config:

```
Central1(config)#ipv6 router ospf 1
Central1(config-rtr)#router-id 1.1.1.1
Central1(config-rtr)#do clear ip ospf process
```

Switching from IPv6 OSPF to IPv6 EIGRP

```
        (you have to turn off OSPF on the interfaces, then add)
Central1(config)#int f0/0
Central1(config-if)#no ipv6 ospf 1 area 0
Central1(config-if)# ipv6 eigrp 1
Central1(config-if)#exit
        [Do for all interfaces]
DFW(config)#int f0/0
DFW(config-if)#no ipv6 ospf 1 area 0
DFW(config-if)# ipv6 eigrp 1
DFW(config-if)#exit
DFW(config-if)#int s0/0/0
 [Same for others]
```

Dynamic IGP Routing Protocols - Quick Overview

RIP for IPv4
Central1(config)#router rip
Central1(config-router)#network 10.0.0.0
Central1(config-router)#network 172.16.0.0
Central1(config-router)#version 2
Central1(config-router)#no auto-summary
Add the default as a static route, and enter RIP config to set default-info originate to propagate it.
Central1(config)#ip route 0.0.0.0 0.0.0.0 Fa0/0
Central1(config)#router rip
Central1(config-router)#default-information originate

RIPng for IPv6 - add to interfaces
Austin(config)#ipv6 unicast-routing
Austin(config)#interface fastethernet 0/0
Austin(config-if)#ipv6 enable
Austin(config-if)#ipv6 address 2001:db8:c18:2::/64 eui-64
Austin(config-if)#ipv6 rip RIPNG1 enable
Austin(config-if)#interface fastethernet 0/1
Austin(config-if)#ipv6 enable
Austin(config-if)#ipv6 address 2001:db8:c18:1::/64 eui-64
Austin(config-if)#ipv6 rip RIPNG1 enable
Austin(config-if)#no shutdown

OSPF for IPv4
int loopback 0
ip address 172.31.1.1 255.255.255.255
router ospf 300
Test(config-router)#network 192.168.10.64 0.0.0.15 area 0
Test(config-router)#network 192.168.10.80 0.0.0.15 area 0

OSPFv3 for IPv6 - add to interfaces
Corp(config)#ipv6 router ospf 1
Corp(config)#ipv6 router-id 1.1.1.1
Corp(config)#int f0/0
Corp(config-if)#ipv6 ospf 1 area 0

EIGRP for IPv4
Corp#config t
Corp(config)#router eigrp 20
Corp(config-router)#network 10.10.11.0 0.0.0.255
Corp(config-router)#network 172.16.10.0 0.0.0.3
Corp(config-router)#network 172.16.10.4 0.0.0.3
Corp(config-router)#no auto-summary

SF(config)#router eigrp 20
SF(config-router)#network 172.16.0.0
SF(config-router)#network 10.0.0.0
SF(config-router)#no auto-summary

EIGRPv6 for IPv6 - add to interfaces
Corp(config)#ipv6 unicast-routing
Corp(config)#ipv6 router eigrp 10
Corp(config-rtr)#no shut
Corp(config-rtr)#router-id 1.1.1.1
Corp(config-rtr)#int s0/0/0
Corp(config-if)#ipv6 eigrp 10
Corp(config-if)#int s0/0/1
Corp(config-if)#ipv6 eigrp 10

Bigger Picture: Routing Table Manager; Timers

Administrative Distance- Lowest score wins
How the device's Routing Table Manager (RTM) chooses from multiple active routing protocols available to get to the same subnet advertised on one of them. Got a router with a few turned on? Lower AD is chosen.

To understand this, a directly connected device has an AD of 0 (as it should)

Static route	1	IS-IS	115
EIGRP summary	5	RIP	120
external BGP	20	external EIGRP	170
Internal EIGRP	90	internal BGP	200
OSPF	110	Unknown*	255

Timers - How to choose a good protocol in your designs:
These are the defaults. Refer to documentation to how to change them, like "ip hello-interval eigrp" or "ip hold-time eigrp"

Protocol	Type	Hello/Keepalive	Hold/Dead	Notes
Interior Routing Protocols				
RIP	DV	Update 30 sec	90 sec	Timeout 180 sec; Flush 120 sec
OSPF	LS	BC+PtP 10 sec	BC+PtP 40 sec (4x hello)	For NBMA 30 sec hello/ 120 sec dead (4x)
EIGRP	AdDV	BC+PtP 5 sec	BC+PtP 15 sec (3x hello)	For NBMA 60 sec hello/ 180 sec dead (3x)
Exterior Routing Protocols				
IS-IS	LS	10 sec Juniper 9 sec?	30 sec (3x hello) Juniper 27 sec?	Also has L1 and L2 timers. PtP vs NBMA. Differ in Juniper. See documentation.
(IS-IS has several timers and issues that are outside the scope of this table)				
BGP	PathV	60 sec keepalive	180 sec (3x keepalive)	
(BGP also has iBGP and eBGP with timers outside the scope of this table)				

OSPF Path Cost in Routing - Don't get confused with Spanning Tree Cost in Switching!
Choosing the best route - cost=reference_bandwidth /interface_bandwidth

Interface	Default Bandwidth	Formula (Kbps)	OSPF Cost
Serial	1544 Kbps	100,000/1544	64
Ethernet	10,000 Kbps	100,000/10,000	10
Fast Ethernet	100,000 Kbps	100,000/100,000	1

The default ref bandwidth assumes the fastest link in the network runs at 100 Mbps.
Use **auto-cost reference-bandwidth 10000 to** accommodate links up to 10 Gbps in speed.

Here are the rules for how a router sets its OSPF interface costs:
- Set the cost explicitly, with **ip ospf cost x interface**, to a value between 1-65,535
- To affect all routes on the device, use **distance 80**. This also affects the AD of non-OSPF routes.
- Change interface bandwidth with the **bandwidth** *speed* command, with speed in Kbps
- Change the reference bandwidth, using subcommand **auto-cost reference-bandwidth** *ref-bw,* in Mbps

STP Path Cost:

10 Mbps	100	2000000
100 Mbps	19	200000
1 Gbps	4	20000
10 Gbps	2	2000
100 Gbps		200

Distance vector routing:
Split Horizon is just not advertising the route back from where it was learned from. Poison reverse labels "where it came from" an unreachable link with "infinite metric" (16 for RIP)
Split-horizon routing with poison reverse is a variant of split-horizon route advertising in which a router actively advertises routes as **unreachable** over the interface over which they were learned by setting the route metric to infinite (16 for RIP). The effect of such an announcement is to immediately remove most looping routes before they can propagate through the network.

The main disadvantage of poison reverse is that it can significantly increase the size of routing announcements in certain fairly common network topologies, but it allows for the improvement of the overall efficiency of the network in case of faults. Split horizon states that if a neighboring router sends a route to a router, the receiving router will not propagate this route back to the advertising router on the same interface.

With route poisoning, when a router detects that one of its connected routes has failed, the router will poison the route by assigning an infinite metric to it and advertising it to neighbors. When a router advertises a poisoned route to its neighbors, its neighbors break the rule of split horizon and send back to the originator the same poisoned route, called a poison reverse. In order to give the router enough time to propagate the poisoned route and to ensure that no routing loops occur while propagation occurs, the routers implement a hold-down mechanism.

BGP4 Overview

- Built for reliability, scalability, and control - not speed.
- Is a **path-vector protocol**. Routing between AS's is called *interdomain routing*
- Uses TCP port 179. BGP peers exchange incremental, triggered route updates and periodic keepalives.
- The administrative distance for eBGP routes is 20, for iBGP routes is 200.
- Routers run only one instance/process of BGP at a time; are called *BGP speakers*; neighbors are *BGP peers*
- Networks not in the local router's routing table will not be sent out in updates
- Routes learned by the BGP process are propagated by default but are often filtered by a routing policy.
- When an update about a network leaves an AS, it's ASN is prepended to the list of ASs that have handled that update. Once it receives an update, it examines the AS list. If it finds its own ASN in that list, the update is discarded (loop prevention/built in split-horizon)
- no auto-summary and no synchronization are defaults in BGP

Because BGP uses TCP, it has reliability, error recovery, and flow control.
Before a BGP speaker can peer with neighbor, neighbor must be statically defined; TCP session-capable, IP reachable. You also get the buffering and other TCP benefits (resent segments if timer reaches 0, acknowledgement may be delayed up to a 1 second to determine if any data should be sent, etc.) The underlying TCP session can be shown with 'show tcp brief'.

Autonomous Systems and ASNs
AS is a group of networks under a common administration. IANA ASN range from 0 to 65,535, with 64,512 - 65,534 ASNs to be private (private means can connect to only one other ASN or multiple ASNs that can't cause loop.

ASN range	0	1 - 64,495	64,496 - 64,511	64,512 - 65,534	65,535
Purpose	Reserved	Public ASN	Documentation	Private ASN	Reserved

Public ASNs are assigned and registered by RIPE NCC; private ASNs can be removed in eBGP using the 'neighbor ebgp-neighbor-address remove-private-as' command.

Path Attributes: AS_PATH and routing
- Path attributes (PAs) are factors that allow BGP to select a route over another. By default, no BGP PAs have been set, and BGP uses AS_PATH PA when choosing the best route among many routes.
- When a router advertises a route with AS_PATH, it will list ASNs the path will go through, to help find the shortest path and prevent routing loops.
- Looping is prevented by ignoring route updates that contain the current AS's ASN.

Internal and External BGP
BGP peering used: iBGP if same AS, eBGP if different AS. If the ASN configured in a device's **router bgp** and **neighbor** statements match, BGP initiates an internal session (iBGP); otherwise an external session (eBGP).
When advertising to an iBGP peer, no ASN is added- not needed. To an eBGP peer, it's ASN is added.
There are 2 types of routing:
- Hot-potato routing: traffic exits the AS via the closest exit point.
- Cold-potato routing: traffic exits the AS via the path closest to the destination.

BGP Databases
BGP uses three databases:
- BGP DB, or RIB (Routing Information Base): Networks known by BGP, with their paths PAs **- show ip bgp**
- Routing table: Paths to each network used by the router, and the next hop for each **- show ip route**
- Neighbor database: All configured BGP neighbors. **show ip bgp summary**

BGP States and Message Types
The BGP process has different states, managed by a Finite State Machine (FSM). The messages don't have specific names like LSA's do in OSPF, just typical hello, update, etc.
1. Router tries to establish TCP connection with IP address configured in 'neighbor' command at port 179
2. After 3-way handshake, first BGP message sent (Open), with parameters to establish neighborship.
3. If a match, neighborship formed (Established), update messages sent (list of PAs and prefixes)

State	Meaning
Idle	Administratively down or waiting- needs TCP session initiated by peer. Listening.
Connect	TCP handshake. If successful, router sends Open message and changes to OpenSent. If not, router resets the ConnectRetry timer and goes to Active state.If timer reaches 0 while the router is in Connect state, timer is reset and another attempt is made, remaining in Connect.
Active	TCP connection ok, but no BGP messages sent to peer yet. Can also mean Connect wasn't successful so it's trying again. If ConnectRetry timer expires, it kicks back to Connect state. If peering stops here, can also mean TCP session attempts were made from an unexpected IP address, which are rejected. If that's the case, it stays in Active and resets ConnectRetry. (attack vector?)
OpenSent/	TCP connection exists, an Open message has been sent to peer. Transitions to OpenReceive to wait for

	initial keepalive from peer to move into OpenConfirm state.
OpenReceive	If TCP disconnect received, router terminates session, resets ConnectRetry timer, and goes back to Active.
OpenConfirm	Open messages finished, initial keepalive received. Sends Keepalives to peer to see configs match, and listens. On receipt of keepalive, moves to Established, else moves to Idle
Established	Achieved after receiving more Keepalives from peer. All neighbor parameters match, the neighbor relationship works, and the peers can now exchange Update messages.

To check state, use **sh ip bgp summary**. The State/PfxRcd column shows a numeric value for a state of Established- if not in an Established state, the value in this heading lists the text name of the current BGP state

All BGP messages share the same header, composed of:
- 16 byte marker field: set to all 1s to detect a loss of synchronization.
- 2 byte length field: indicate total length of BGP message, range from 19 to 4096
- There is a PA field in Update (AS Path Attribute)
- 1 byte type field: type code, indicate different BGP messages - the number in () below0

Message (type#)	Purpose
Open (1)	First message after TCP connection has established to get neighborship started. BGP version, ASN, hold timer, BGP identifier, optional authentication
Keepalive (4)	Maintain peering. Contains only header. Keepalives exchanged every 60 sec by default. By default, hold timer is 3x keepalive interval. Updates can also reset keepalive interval
Update (2)	Used to exchange PAs and the associated prefix/length (NLRI) that uses those attributes. Each contains a single set of PAs and all related NLRI. Also contain withdrawn routes.
Notification (3)	Signals error with code, subcode, and data. Often signals to resets connection

All prefixes, except filtered by **neighbor <ipaddr> route-map <name> in**, are in the routing table for calculation.

eBGP neighborship
To configure BGP, you need at least 2 commands:
config)#router bgp <asn>
config-router)#neighbor <ipaddr> remote-as <remote-asn>

For BGP neighbor relationship to form,
- TCP connection between them must first be established
- Both routers need complimentary **neighbor <ipaddr> remote-as <asn>** with matching ASN and the IP address of the interface for which TCP packets will exit (if the interface is not explicitly defined)
- BGP RID must not be the same
- MD5 authentication must pass (if configured)

BGP RID is established by
1) **bgp router-id <rid>** - OR -
2) highest IP address of any up/up loopback interfaces at BGP initialization
3) highest IP address of any up/up normal interface at BGP initialization

To configure authentication, use **neighbor <ipaddr> password <password>** on both routers. The <ipaddr> refers to the IP address of the other router, while password must be identical for neighborship to succeed.

To verify neighborship **show ip bgp neighbors**, and **show tcp brief** to display the TCP connection
Use **neighbor <ipaddr> shutdown** to shut down that connection, move to Idle (keeps neighbor config).

The 'neighbor' command can be configured with the **peer-group** parameter, so one set of commands can be applied to all neighbors in the peer-group (either external or internal, not both)
Simplify configuration and reduce updates.
neighbor <name> peer-group
neighbor <ipaddr> peer-group <name>
(neighbor must have 'neighbor remote-as' set).

In most IGPs, the network command starts the routing process on an interface.
In BGP, the command tells the router to originate an advertisement for that network.
- network does not have to be connected - it just has to be in the routing table.
- can even be a network in a different AS (not usually recommended, but next-hop-self sort of does it)

BGP assumes you are using the default classful subnet mask.
- need to advertise a subnet? Use the optional keyword **mask** and specify the subnet mask to use
- routing table must contain an exact match (prefix and subnet mask)

Network declarations need to be SPECIFIC

If you misconfigure a **network** command, like **network 192.168.1.1 mask 255.255.255.0**, BGP will look for exactly 192.168.1.1/24 in the routing table. With no match, BGP won't announce it to neighbors.

If you issue the command **network 192.168.0.0 mask 255.255.0.0** to advertise a CIDR block, BGP will look for ONLY 192.168.0.0/16 in the routing table. It may find 192.168.1.0/24 or 192.168.1.1/32; however, it may never find 192.168.0.0/16. In this case, you can configure a static route towards a null interface so BGP can find an exact match in the routing table: **ip route 192.168.0.0 255.255.0.0 null0**

BGP and Loopback Addresses

- Recall that a loopback interface provides a more robust configuration - it never goes down, so it adds more stability than physical the ones, and BGP will still operate if the link to the closest interface fails.

If the physically connected port is down:

- The router can still send the packet without being interrupted because loopback is always up.
- If another interface tries to reach the neighbor it will be blocked when source address doesn't match.

When multiple links exist between 2 BGP routers, and you would like to establish BGP neighborship between them, there are 2 options:

- configure a connection for each physical interface; consumes bandwidth and memory.
- Configure connections using virtual (loopback) interfaces, which requires less bandwidth and memory, and ensure the interface is always up/up.

eBGP assumes that neighbors are directly connected and peering with the IP of that specific interface. If not, you must use **neighbor *ip-address* ebgp-multihop *number-of-hops*** command. If you are peering with loopback interface IP addresses, you are going to have to use it.

iBGP assumes that internal neighbors might not be directly connected, so this command is not needed. With loopback IP addresses in iBGP, you must change the source to match the loopback with **neighbor** *ip-address* **update-source** *interface*

Router(config-router)**#neighbor 172.16.1.2 update-source loopback0**

- Without neighbor **update-source**, iBGP will use the closest IP interface to the peer. In the case of a point-to- point eBGP session, this command is not needed because there is only one path for BGP to use.

Using loopback and update-source for connections whether iBGP or eBGP is considered a best practice. It ensures an alternate route to the specific router is available if the physical interface goes down

BGP Path Attributes - Manipulating Path Selection

Routes learned via BGP have properties referred to as *BGP attributes*, and an understanding of how they influence route selection is required for the design of robust networks.

PA	Description	Route Direction
NEXT_HOP	Next-hop IP address used to reach a prefix.	N/A
Weight[1]	Range 0 - 2^16 -1, set when receiving updates. Not advertised to any BGP peers. Cisco proprietary	Outbound
LOCAL_PREF	Range 0 - 2^32 -1, set and spread throughout an AS to influence the choice of best route for all routers in that AS	Outbound
AS_PATH (length)	The number of ASNs in the AS_Path PA.	Outbound, Inbound
ORIGIN	Value implying the route was injected into BGP; I (IGP), E (EGP), or ? (incomplete information).	Outbound
Multi-Exit Discrim (MED)	Set and advertised by routers in an AS, impacting the BGP decision of routers in the other AS. Smaller is better.	Inbound

Four categories of attributes exist as follows:
- Well-known mandatory: Must be recognized by all BGP routers, present in all BGP updates, and passed on to other BGP routers. For example, **AS path, origin, and next hop.**
- Well-known discretionary: Must be recognized by all BGP routers and passed on to other BGP routers but need not be present in an update, for example, **local preference.**
- Optional transitive: Might or might not be recognized by a BGP router but is passed on to other BGP routers. If not recognized, it is marked as partial, for example, **aggregator, community.**
- Optional nontransitive: Might or might not be recognized by a BGP router and is not passed on to other routers, for example, **Multi-Exit Discriminator (MED), originator ID.**

BGP Path Selection Criteria
1. Highest weight (most local).
2. Highest LOCAL_PREF (inside an AS this is global).
3. Choose routes that this router originated (next-hop = 0.0.0.0)
4. Shortest AS_PATH
5. Lowest origin type/code - (i)iBGP is lowest, (e)eBGP is next, and ? is last
6. Lowest MED (if the same AS advertises the possible routes)
7. eBGP wins over an iBGP route.
8. Nearest IGP neighbor (lowest IGP metric)
9. Oldest route (for eBGP to minimize the effects of route flapping)
10. Lowest router ID.
11. Lowest IP address.

BGP only chooses one route as the best route (thus, no load balancing). Step 1, 2, and 4 are typically used to influence outbound routes, while MED for outbound routes. 9, 10 and 11 aren't used except in a tie
Mnemonic: N WLLA OMNI

Step	Mnemonic Letter	Short Phrase	Which Is Better?
0	N	Next hop: reachable?	If no route to reach Next_Hop, router cannot use this route.
1	W	Weight	Bigger weight.
2	L	LOCAL_PREF	Larger preference.
3	L	Locally injected routes	Locally injected ‘network’ is better than iBGP/eBGP learned.
4	A	AS_PATH length	Smaller paths win
5	O	ORIGIN	Prefer I over E. Prefer E over ?
6	M	MED	Smaller Multi-Exit Discriminator (MED)
7	N	Neighbor Type	Prefer eBGP over iBGP.
8	I	IGP metric to Next_Hop	Smaller metric wins. If no IGP is used, consider tied.
9	A	Age equals seniority	Oldest eBGP route
10	O	think not OSPF	Route with lowest BGP RID
11	S	think not OSPF	Route with lowest neighbor IP address

BGP - Basic Configuration

router bgp 100.	Starts BGP routing process 100
neighbor 192.31.7.1 remote-as 200	Add a peer for sessions. The ASN tells if eBGP or iBGP
network 192.135.250.0	What locally learned networks to advertise.
network 128.107.0.0 mask 255.255.255.0	Specify an individual subnet.
(**no) neighbor 24.1.1.2 shutdown**"	Disable/-enable an active session
timers bgp 90 240	Timers. Keepalive (default 60sec) Holdtime (default 180sec)
neighbor 172.16.1.2 update-source loopback0	Use this specific interface
(**no**) **synchronizatio**n	iBGP route must be added to table by other IGP before used

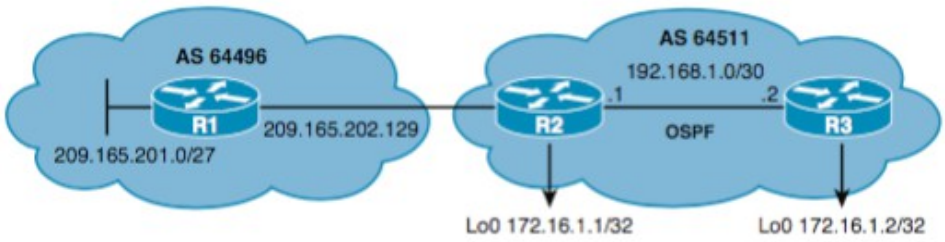
Sychronization is off by default in later IOS. Using iBGP meshes is preferred. Redistribution from BGP into an IGP when using BGP for MPLS is reasonable and commonly done. See SWITCH book pg 617 on avoiding loops

iBGP Next-Hop Behavior

- The eBGP next-hop attribute is the IP address that is used to reach the advertising router; an edge router belonging to the autonomous system "next door", in most cases, the IP address of the connection between peers.
- For iBGP, the eBGP next-hop address is not changed, carried into the local autonomous system as-is.

The next-hop-self command is a way to get around this (illustrated below)

- R2 tells R3 that it instead will be the next-hop address when trying to reach networks outside its autonomous system.

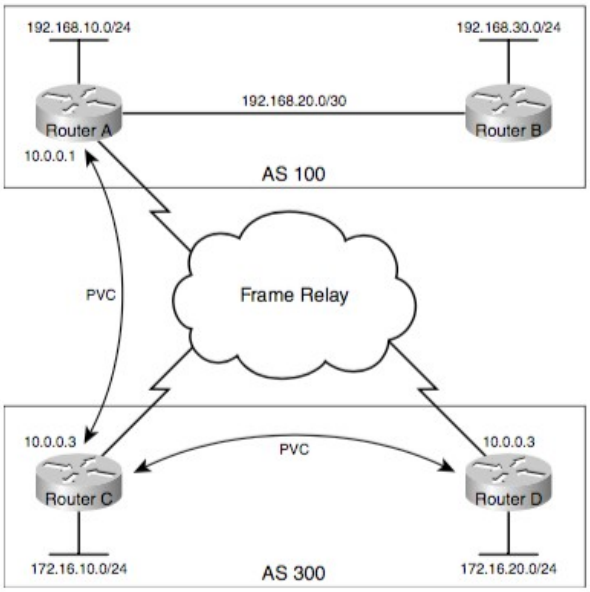


Setup on R2:

router bgp 64511	Starts the BGP routing process.
neighbor 209.165.202.129 remote-as 64496	Identifies R1 as an eBGP neighbor
neighbor 172.16.1.2 remote-as 64511	Identifies R3 as an iBGP neighbor
neighbor 172.16.1.2 update-source loopback0	Loopback0 IP is source for all BGP TCP packets to R3
neighbor 172.16.1.2 next-hop-self	Advertises itself as next hop for networks from other AS

R3 will then use R2 as the next hop instead of using the eBGP next-hop of 209.165.202.129.

Here is another example in a slightly different setup where it is almost mandatory:



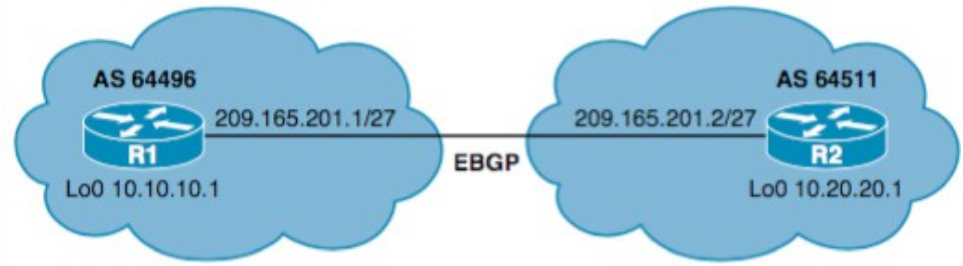
RouterC(config)**#router bgp 300**
RouterC(config-router)**#neighbor 10.0.0.1 remote-as 100**
RouterC(config-router)**#neighbor 10.0.0.1 next-hop-self**
- Forces all updates to 10.0.0.1 to advertise this router as next hop

- Router C advertises 172.16.20.0 to Router A with next hop of 10.0.0.3 as if the common media were Ethernet.
- Routing will fail - Router A has no direct PVC to Router D and cannot reach the next hop
- To remedy this situation, neighbor next-hop-self causes Router C to advertise 172.16.20.0 with next-hop set to it's 10.0.0.3.
- This proves useful in non-meshed networks (such as Frame Relay or X.25) where BGP neighbors might not have direct access to all other neighbors on the same IP subnet.

RouterC(config-router)**#neighbor 10.0.0.1 next-hop-unchanged**
Enables an eBGP multihop peer to propagate the next hop unchanged.

Do not use with route reflector setups
This command should not be configured on a route reflector, and the neighbor next-hop-self command should not be used to modify the next-hop attribute for a route reflector when this feature is enabled for a route reflector client.

eBGP Multihop
By default, eBGP neighbors exchange packets with a TTL set to 1. Even though eBGP neighbors are usually directly connected (over a WAN connection) to establish sessions, sometimes one of the directly connected routers is unable to run BGP. The command **ebgp-multihop** allows for a logical connection to be made between peer routers, even if not directly connected - up to 255 hops away and still be able to create an eBGP session. The **ebgp-multihop** is only used for eBGP sessions, and must be configured on each peer (each end). If you attempt to establish eBGP session between loopbacks, BGP packets will be dropped due to an expired TTL.



R1(config)**# ip route 10.20.20.1 255.255.255.255 209.165.201.2**
- Define a static route to Loopback0 on R2.
R1(config)**# router bgp 64496**
R1(config-router)**# neighbor 10.20.20.1 remote-as 64511**
- Identifies a peer router at 10.20.20.1

R1(config-router)# **neighbor 10.20.20.1 update-source loopback0**
R1(config-router)# **neighbor 10.20.20.1 ebgp-multihop 2**
- Allows for two routers (not directly connected) to establish an eBGP session with TTL of 2.

R2(config)# **ip route 10.10.10.1 255.255.255.255 209.165.201.1**
- Define a static route to Loopback0 on R1.

R2(config)# **router bgp 64511**
R2(config-router)# **neighbor 10.10.10.1 remote-as 64496**
R2(config-router)# **neighbor 10.10.10.1 update-souce loopback0**
R2(config-router)# **neighbor 10.10.10.1 ebgp-multihop 2**
- Allows for two routers (not directly connected) to establish an eBGP session with TTL of 2.

Loopbacks on eBGP need ebgp-multihop
If redundant links exist between two eBGP neighbors and loopback addresses are used, you must configure **ebgp-multihop** because of the default TTL of 1. Otherwise, the router decrements the TTL before giving the packet to the loopback interface, meaning that the normal IP forwarding logic discards the packet. Configuring the value to 2 solves the problem.

Default Routes, default-originate to give default to specific neighbors
For a default route to send to all neighbors/peers you just use **network 0.0.0.0** (be sure it's in the routing table)
For the default route 0.0.0.0 to only be advertised to a specific neighbor, use **neighbor 192.168.100.1 default-originate**

BGP Peer Groups
To ease the burden of configuring a large number of neighbors with identical or similar parameters (for example, route maps, filter lists, or prefix lists), the concept of peer groups was introduced. Simply configure a peer group with all the BGP parameters that are to be applied to many BGP peers. BGP neighbors are bound to the peer group, and the network administrator applies the peer group configuration on each of the BGP sessions. The result below, all four iBGP neighbors have the same basic BGP configuration assigned to them.

router bgp 65500	Starts the BGP routing process
neighbor INTERNAL peer-group	Creates a BGP peer group called INTERNAL
neighbor INTERNAL remote-as 65500	Assigns a first parameter to the peer group
neighbor INTERNAL next-hop-self	Assigns a second parameter to the group
neighbor INTERNAL update-source loopback0	Assigns a third parameter to the peer group
neighbor INTERNAL route-reflector-client	Assigns a fourth parameter to the peer group
neighbor 192.168.1.2 peer-group INTERNAL	Assigns the peer group to neighbor R2
neighbor 192.168.1.3 peer-group INTERNAL	... to neighbor R3
neighbor 192.168.1.4 peer-group INTERNAL	... to neighbor R4
neighbor 192.168.1.5 peer-group INTERNAL	... to neighbor R5

A peer group can be, among others, configured to do the following:
- Use IP of a specific interface as source address when opening the TCP session or use next-hop-self feature
- Use, or not use, things like the eBGP multihop function, or MD5 authentication on the BGP sessions.
- Filter out any incoming or outgoing routes using a prefix list, a filter list, and a route map.
- Assign a particular weight value to the routes that are received.

MP-BGP
Original BGP was designed for only IPv4. Multiprotocol BGP can run over as IPv6, multicast IPv4, and MPLS - can exchange routes for IPv4, IPv6, or both. The extensions enable NEXT_HOP to carry IPv6 addresses and NLRI (Network Layer Reachability Information) to an IPv6 prefix, thanks to TCP. The IPv4 and IPv6 routes use separate TCP connections.

R1(config)#**ipv6 unicast-routing**
R1(config)#**router bgp 65500**
R1(config-router)#**neighbor 2001:0DB8:12::2 remote-as 65501**
R1(config-router)#**neighbor 192.168.1.2 remote-as 65501**
R1(config-router)#**address-family ipv4 unicast**
- Enters IPv4 address family configuration mode for unicast prefixes (default IPv4)

R1(config-router-af)#**neighbor 192.168.1.2 activate**
- Enables the exchange of IPv4 BGP info with R2. IPv4 neighbors auto-activate, so keyword is optional.

R1(config-router-af)#**network 10.1.1.1 mask 255.255.255.255**
R1(config-router)#**address-family ipv6 unicast**
- Enters IPv6 address family configuration mode for unicast address prefixes.

R1(config-router-af)#**neighbor 2001:0DB8:12::2 activate**
- Enables the exchange of IPv6 BGP information with R2.

R1(config-router-af)#**network 2001:0DB8:1::1/64**

Router2 would have a reciprocal set of the same commands to work with R1.
Recall that the router ID is set on IPv4 addresses (**bgp router-id** *IPv4_address)*, IPv6 can still use it
Troubleshoot with **show bgp ipv6 unicast [optional: summary | neighbors]** and **show ipv6 route bgp**

Route Aggregation - Setting the Atomic Aggregate attribute

- A BGP router can transmit overlapping routes (nonidentical routes pointing to the same destination)
- When making a best path decision, a router always chooses the more specific path.

R1(config-router)**#aggregate-address 172.16.0.0 255.255.0.0**

- Creates aggregate entry in the BGP table to cover specific BGP routes in that range.
- More specific routes will still be sent unless **summary-only** is used.

R1(config-router)**#aggregate-address 172.16.0.0 255.255.0.0 summary-only**

- Creates aggregate route AND suppresses advertisement of more-specific routes.
- Specific AS_PATH info on those subnets are lost.

R1(config-router)**#aggregate-address 172.16.0.0 255.255.0.0 as-set**

- Creates an aggregate entry but the path advertised will be an AS_SET (list of AS_PATHs)

Below is a use of aggregate route as a precaution: to send an aggregate address, we only need one of the more specific routes configured. But by configuring all of them, and *not* using **summary-only**, they will all be sent, and the aggregate will be sent in case one of the networks goes down.

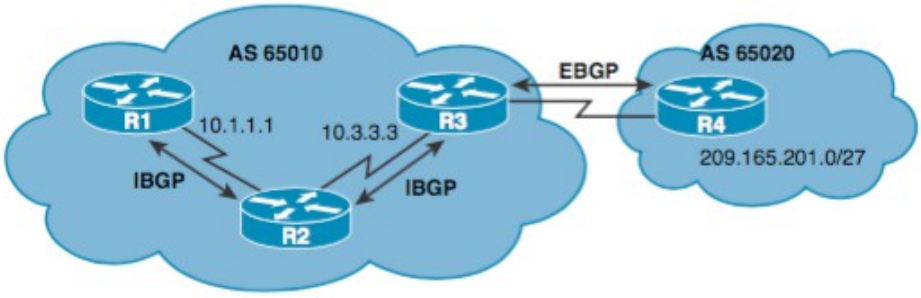
Lubbock(config)#router bgp 1
Lubbock(config-router)#neighbor 10.1.1.2 remote-as 2
Austin(config)#router bgp 2
Austin(config-router)#neighbor 10.1.1.1 remote-as 1
Austin(config-router)#network 172.16.0.0 mask 255.255.255.0
Austin(config-router)#network 172.16.1.0 mask 255.255.255.0
Austin(config-router)#network 172.16.2.0 mask 255.255.255.0
Austin(config-router)#aggregate- address 172.16.0.0 255.255.252.0

Thus, both Lubbock and Austin will have all the specific routes *and* the aggregate address in its BGP table

```
Lubbock#show ip bgp 172.16.0.0 255.255.252.0
BGP routing table entry for 172.16.0.0/22, version 18
Paths: (1 available, best #1)
<text omitted>
 Origin IGP, localpref 100, valid, external, atomic-aggregate,
  best
```

Route Reflectors

By default, a router that receives an eBGP route advertises it to its eBGP and iBGP peers.
If it receives it through iBGP, it won't advertise it to its iBGP peers, as loop-prevention
One way for all iBGP routers to receive a route after it is originated into the AS is to have a full mesh of iBGP peers, which can get complex with a large number of peers. Route reflectors eliminate resorting to that.



The objective is to allow R2 to advertise to R1 the 209.165.201.0/27 network learned from R3. Without these commands, R1 would never learn the 209.165.201.0/27 network without a full-mesh iBGP topology.

R2(config)**#router bgp 65010**	Enters BGP routing configuration mode
R2(config-router)**#neighbor 10.1.1.1 route-reflector-client**	Designates a BGP route reflector and the specified neighbor as a client
R2(config-router)**#neighbor 10.3.3.3 route-reflector-client**	Designates a BGP route reflector and the specified neighbor as a client

Verifying and Troubleshooting BGP Connections

show ip bgp	Displays entries in the BGP routing table - AS_PATH with the last ASN; Internal is 'i'. '*' is valid, and '>' is best route
show ip bgp *prefix [subnet-mask]*	List possible routes, per prefix [e.g., for default 0.0.0.0 0.0.0.0]
show ip bgp <ipsubnet/cidr> **longer-prefixes**	Show which route is older by placing its entry later
show ip bgp neighbors	Info about BGP/ TCP connections to neighbors, neighbor info
show ip bgp neighbors <address> **[routes | received-routes | advertised-routes]**	Monitor routes received/ learned from a specific neighbor.
show ip bgp rib-failure	List networks not put into the Routing Information Base (RIB) and the reason not used
show ip bgp summary	Status of all BGP connections, memory use of BGP databases, activity stats and list of BGP neighbors
show ip route bgp	Displays the BGP entries from the routing table

show tcp brief to display the TCP connection
Use **neighbor <ipaddr> shutdown** to shut down that connection, move to Idle (keeps neighbor config)
show ip bgp [peer-group | regexp | community-list]
show access-lists and **show ip access-lists**
debug ip bgp, debug ip bgp events, **debug ip bgp ipv4 unicast**, **debug ip bgp updates**

clear ip bgp * - Forces BGP to clear its table and the * resets all BGP sessions. Put IP for a specific neighbor
Using the form **clear ip bgp *** is both processor and memory intensive - use only in smaller environments.
However, you may need to use this form when the following changes occur: Additions or changes to the BGP-related access lists, weights, distribution lists, BGP timer's specifications, administrative distance, or route maps

clear ip bgp 10.1.1.2 soft out - The **soft** keyword forces the remote router to resend all BGP info **without** resetting the connection. Routes from this neighbor are not lost. Works when changing outbound policy, does not help if you are changing an inbound policy. The keyword **out** means for all outbound updates (inbound is **in**).

Hard reset can be harsh - the underlying TCP connection is zapped so neighborship drops, removes all BGP table entries from that neighbor. It's a long recovery that interrupts routing, causes flapping with disassociated peers, and the full set of routing updates generates tons of traffic. Soft resets don't bring down the neighborship or TCP connection, and just resends adjusted outgoing updates, which then adjust the BGP table. It is supported with route refresh capability included in all modern IOS versions

neighbor 10.1.1.2 soft-reconfiguration inbound Causes the router to store all updates from this neighbor. This is memory intensive, but is a safeguard before running **clear ip bgp** that a copy of the routing info is indeed saved first (arguably paranoid since there is the **clear ip bgp x.x.x.x soft** option)

Routing Information Base failures (IP routing table omissions)
When the BGP best-path algorithm has chosen a best route for a prefix, it gets handed to the Routing Table Manager (RTM) that weighs all routes by administrative distance before adding to the routing table [aka Routing Information Base (RIB)] Generally, a prefix learned with BGP should not conflict with a connected or IGP-learned route. Sometimes it arises when implementing MPLS VPNs with BGP/IGP redistribution. The **show ip bgp rib-failures** command lists routes which BGP has chosen as best, but the RTM function has not placed into the IP routing table.

As an example, consider two routers R1 and R2 finding their route to 185.0.0.0/8 - R1 learns about it from eBGP (AD of 20), but inside the AS, R2 sees it from both OSPF and iBGP. R2's BGP determined a best route, but the RTM instead chose the lower-AD OSPF route (AD 110) rather than the higher-AD iBGP route (AD 200). Running show ip bgp rib-failure shows one line for each best BGP route that the RTM does not place into the IP routing table. In this case, this command on 2 lists the route for 185.0.0.0/8.

```
E2# show ip bgp rib-failure
Network            Next Hop                  RIB-failure   RIB-NH Matches
185.0.0.0/8        10.100.1.1          Higher admin distance            n/a
```

Contents of show ip bgp - A Deeper Look

The command **show ip bgp** shows the BGP topology database, which includes all the networks BGP knows about, the next hop, some of the attributes, and the AS path for each route.

```
route-server>show ip bgp
BGP table version is 22285573, local router ID is 12.0.1.28
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
       r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

  Network     Next Hop         Metric LocPrf Weight Path
* 3.0.0.0     12.123.137.124                 0     7018 2914 9304 80 i
*>            12.123.1.236                   0     7018 2914 9304 80 i
* 3.51.92.0/23 12.123.137.124                0     7018 ?
*             12.122.125.4     2366          0     7018 ?
*>            12.123.1.236                   0     7018 ?
* 8.6.6.0/24  12.123.137.124                 0     7018 701 14744 14744 14276 i
*             12.123.145.124                 0     7018 701 14744 14744 14276 i
*>            12.123.1.236                   0     7018 701 14744 14744 14276 i
```

Networks are listed in numerical order.
The first three columns list each route's status. They are one-character fields, squeezed together

The asterisk (*) in the first column means that the route has a valid next hop.
Other options:

- s - suppressed: BGP knows about the network but is not advertising it, often if part of a summarized route.
- d - dampened: BGP stopped advertising a network that flaps too often until it is stable for a period of time.
- h - history: BGP knows about this network but does not currently have a valid route to it.
- r - RIB failure: advertised route but not put in routing table. Another protocol may have route with a better AD.
- S - stale: Used with nonstop forwarding - route needs to be refreshed when the peer is reestablished.

The second column has a greater-than sign (>) beside the route that was selected as the best path to that network. In the example, the second route was selected for network 3.0.0.0.

The third column is blank in the example, which means that the router learned all the routes from an external neighbor. A route learned from an IBGP neighbor would have an "I" in the third column. (a - aggregate, i - internal)

The fourth column lists the networks. Those without a subnet mask, such as network 3.0.0.0, use their classful mask. As seen in the example, when the router learns about the same network from multiple sources, it lists only the network once.

The fifth column lists the next-hop address for each route. This might or might not be a directly connected router. A next-hop of 0.0.0.0 means that the local router originated the route (and/or, the router has non-BGP routes to the network).

Inter-autonomous system metric - If a Med value was received with the route, it is listed in the Metric column. Notice that the advertisement for network 3.51.92.0/23 from the router at 12.122.125.4 has a large Med value of 2366. Because the default Local Preference is used for each of the routes shown, no local preference value is displayed. The default Weight value of 0 is listed, however.

Toward the end is the AS path for each network. Reading this field from left to right, the first AS number shown is the adjacent AS this router learned the route from. After that, the AS paths that this route traversed are shown in order. The last AS number listed is the originating AS. In the example, our router received an advertisement about network 3.0.0.0 from its neighbor AS 7018, which heard about it from AS 2914, which heard about it from AS 9304. And AS 9304 learned the route from AS 80, which originated it. A blank AS path means that the route was originated in the local AS.

In the AS Path column, note that network 8.6.6.0 shows AS 14744 twice in its AS path list. Most likely AS 14744 has prepended an extra copy of its AS number to make the path through it less attractive than the path through other autonomous systems. In this case it did not work because the only paths to 8.6.6.0 this router knows about all go through AS 14744.

The last column shows how BGP originally learned about the route.

- Networks 3.0.0.0 and 8.6.6.0 show an "i" for their origin codes. This means that the originating router had a network statement for that route.
- Network 3.51.92.0 shows a "?" as its origin. This means that the route was redistributed into BGP; BGP considers it an "incomplete" route.
- You will likely never see the third possibility, an "e," because that means BGP learned the route from the Exterior Gateway Protocol (EGP), which is no longer in use.

If you specify an address this is how it will display:

router**#show ip bgp 66.66.66.66**

```
BGP routing table entry for 66.66.66.66/32
Paths: (2 available, best #1)
  66.66.66.66/32
  3 i comm 65535:65281
  172.16.23.3 from 172.16.23.3 (peer 3.3.3.3)
    Origin IGP, local pref 100, weight 0, valid, best
     BGP Path Selection Criterion: Lowest BGP Neighbor Router-ID
    IGP Metric: 0 IGP Pref: 0 IGP Protocol: DIRECT
    IGP Next Hop: 0.0.0.0 Route Age: 0:03:22
  66.66.66.66/32 (Second best)
  77 i comm 65535:65281
  172.16.12.1 from 172.16.12.1 (peer 172.16.12.1)
    Origin IGP, local pref 100, weight 0, valid
    IGP Metric: 0 IGP Pref: 0 IGP Protocol: DIRECT
    IGP Next Hop: 0.0.0.0 Route Age: 0:02:21
```

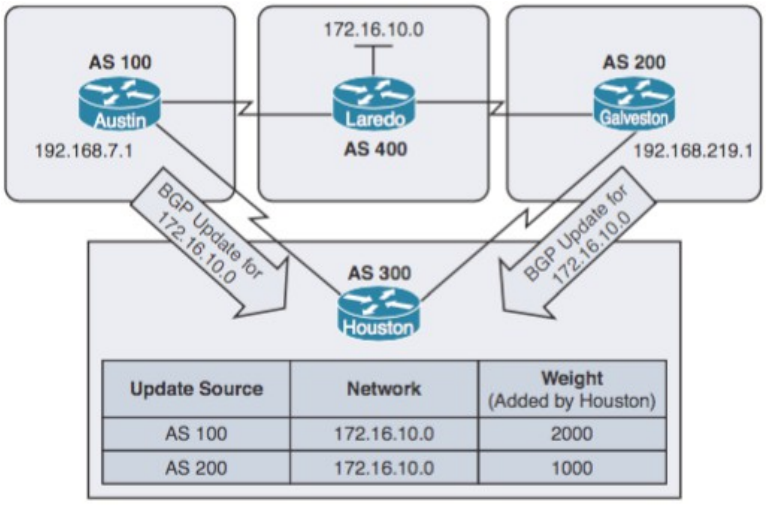
Originator	Specifies the router ID of the originator of the route in the local AS.
Cluster list	A sequence of cluster ID values for the reflection path that the route has passed.
Path	Autonomous system path to the destination network. One entry in this field per autonomous system in the path.
Origin codes	Identifies the origin of the entry. Valid values previously mentioned (i, e, ?)
Origin	Identifies the origin of the entry as: IGP, EGP, Not clear, or Best.
LocPrf	Local preference value. See the set local-preference route map configuration command. Default value is 100.
Weight	Weight of the route as defined by set weight and router bgp route map
BGP path selection criteria	Displays tie-breaking criterion for best path selection.
IGP Metric	Specifies the IS-IS metric or OSPF cost value.
IGP Protocol	IGP Protocol: IS-IS or OSPF.
IGP Next-Hop	IP address of the next system used when forwarding packets to the destination network. 0.0.0.0 in this field indicates router has non-BGP routes to the network.
Route Age	Specifies the time in *hours*:*minutes*:*seconds* that a route has been valid.
Second best	Displays second best path information.

Overview of Specific BGP Attributes

Weight Attribute

The weight attribute is a proprietary Cisco attribute
- configured locally on a router and is not propagated to any other routers.
- applies when one router is used with multiple exit points out of an autonomous system
- (local preference attribute is used when two or more routers provide multiple exit points)

Routes with a higher weight are preferred when there are multiple routes to the same destination.
[value ranges from 0 to 65,535, 0 for learned routes, and 32,768 for locally injected routes]
By default, the weight attribute is 32,768 for paths that the router originates, and 0 for other paths.



Update Source	Network	Weight (Added by Houston)
AS 100	172.16.10.0	2000
AS 200	172.16.10.0	1000

Houston(config)**#router bgp 300**
Houston(config-router)**#neighbor 192.168.7.1 remote-as 100**
Houston(config-router)**#neighbor 192.168.7.1 weight 2000**
Houston(config-router)**#neighbor 192.168.219.1 remote-as 200**
Houston(config-router)**#neighbor 192.168.219.1 weight 1000**

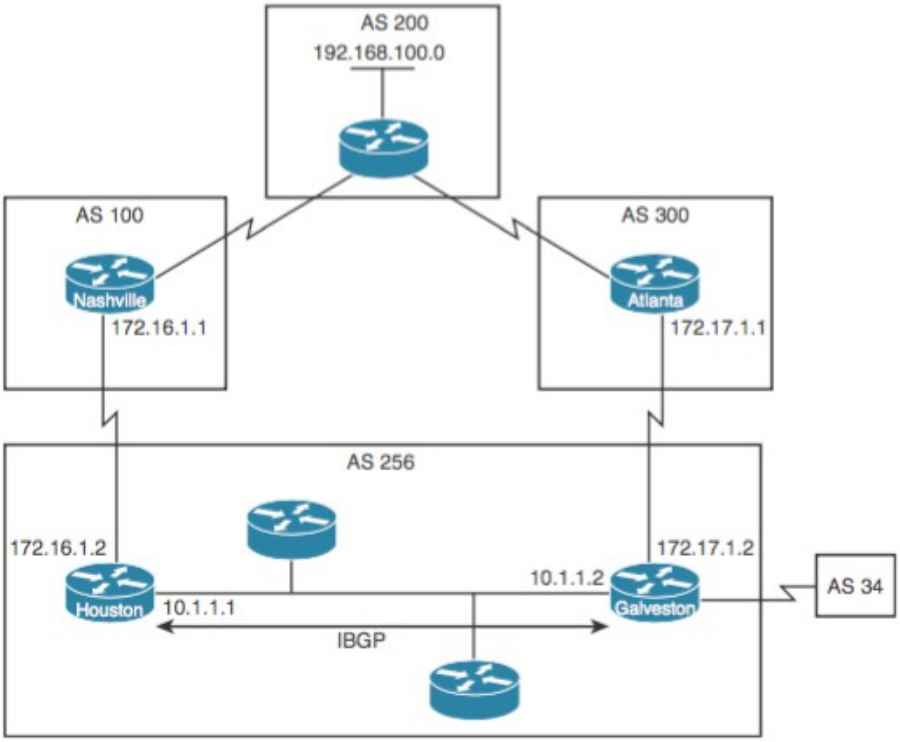
Has Houston forward traffic to the 172.16.10.0 network through AS 100, because the routes entering AS 300 from AS 100 have a higher weight attribute set compared to that same routes advertised from AS 200.
When a router receives a BGP Update, that router can set the Weight either selectively, per route, using a route map, or for all routes learned from a single neighbor

Local Preference Attribute

- use to force BGP routers to prefer one exit point over another for leaving the AS
- value can be between 0 and 429,496,729. Higher is preferred, default is 100
- It is local to the AS; it is exchanged between iBGP peers - ***not*** *advertised to eBGP peers*.
- Recall also that BGP does not allow a router to advertise iBGP-learned routes to iBGP peers

Houston(config)#router bgp 256
Houston(config-router)#neighbor 172.16.1.1 remote-as 100
Houston(config-router)#neighbor 10.1.1.2 remote-as 256
Houston(config-router)#bgp default local-preference 150
Galveston(config)#router bgp 256
Galveston(config-router)#neighbor 172.17.1.1 remote-as 300
Galveston(config-router)#neighbor 10.1.1.1 remote-as 256
Galveston(config-router)#bgp default local-preference 200

Based on these two configurations, traffic destined for a remote network that can be reached through autonomous system 256 will be routed through Galveston.



Using AS_PATH Access Lists with Route Maps to Manipulate the Local Preference Attribute
Route maps provide more flexibility than the **bgp default local-preference** router configuration command.
Galveston(config)**#router bgp 256**
Galveston(config-router)**#neighbor 172.17.1.1 remote-as 300**
Galveston(config-router)**#neighbor 172.17.1.1 route-map SETLOCAL in**
- reference route map SETLOCAL.

Galveston(config-router)**#neighbor 10.1.1.1 remote-as 256**
Galveston(config)**#ip as-path access-list 7 permit ^300$**
- BGP ACL 7 says permit updates whose AS_PATH starts and ends with 300 (^ and $)

Galveston(config)**#route-map SETLOCAL permit 10**
- creates route map SETWEIGHT, sequence number of 10

Galveston(config-route-map)**#match as-path 7**
- condition when policy routing is allowed- match BGP ACL 7.

Galveston(config-route-map)**#set local-preference 200**
- assigns local pref of 200 to any update coming from autonomous system 300

Galveston(config-route- map)**#route-map SETLOCAL permit 20**
- close route map SETLOCAL with an allow to accept all other routes.

In this example, using the **route-map** command, only updates received from AS 300, as specified in the **ip as_path access-list** command, will have a local preference set to 200.
Through the **neighbor route-map** command; **in** option is required for updates from an eBGP peer

AS_PATH Attribute Prepending

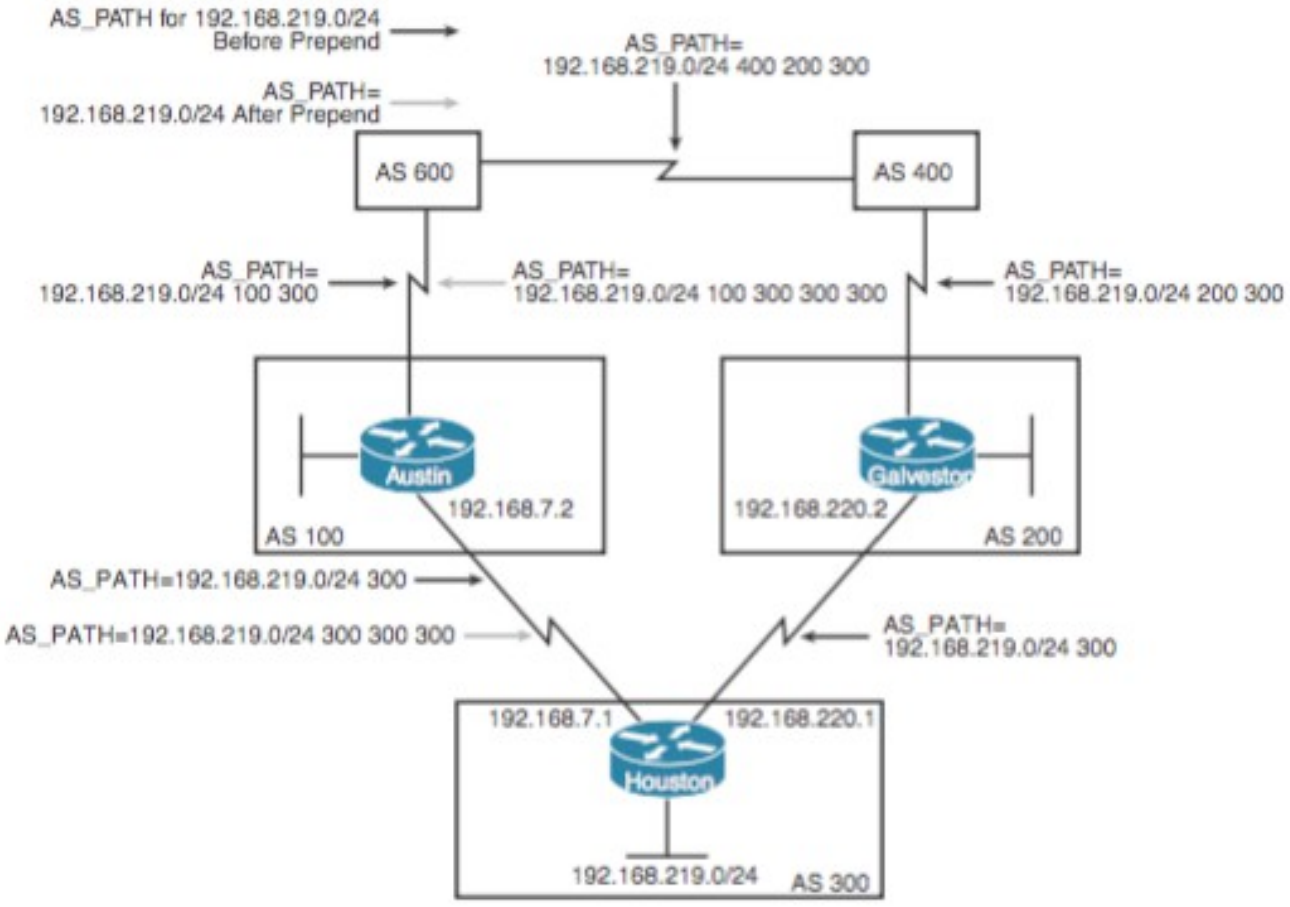
AS paths can be manipulated by prepending, or adding, extra ASNs to the AS_PATH. By this attribute's criteria, the shortest AS_PATH attribute is preferred; making it longer advertises it as a less viable route.

Houston(config)**#router bgp 300**
Houston(config-router)**#network 192.168.219.0**
Houston(config-router)**#neighbor 192.168.220.2 remote-as 200**
Houston(config-router)**#neighbor 192.168.7.2 remote-as 100**
Houston(config-router)**#neighbor 192.168.7.2 route-map SETPATH out**
- all routes going out to 192.168.7.2 are filtered with SETPATH route map

Houston(config)**#route-map SETPATH permit 10**
- create route map SETPATH; sequence number of 10

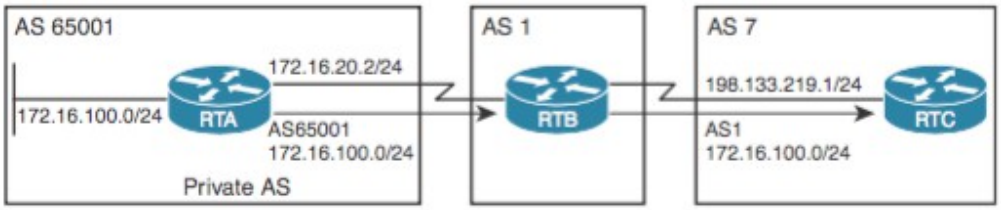
Houston(config-route-map)**#set as-path prepend 300 300**
- add 300 twice to the AS_PATH before sending to192.168.7.2

In this scenario, you want to use the configuration of Houston to influence the choice of paths in AS 600. Currently, the routers in AS 600 have reachability information to the 192.168.219.0/24 network via two routes: via AS 100 with an AS_PATH attribute of (100, 300), and via AS 400 with an AS_PATH attribute of (400, 200, 300). Assuming that the values of all other attributes are the same, the routers in AS 600 will pick the shortest AS_PATH attribute: the route through AS 100. Thus, prepend, or add, extra AS numbers to the AS_PATH attribute for routes that Houston advertises to AS 100 to have AS 600 select AS 400 as the preferred path of reaching the 192.168.219.0/24 network.



The result of this configuration is that the AS_PATH attribute of updates for network 192.168.219.0 that AS 600 receives via AS 100 will be (100, 300, 300, 300), which is longer than the value of the AS_PATH attribute of updates for network 192.168.219.0 that AS 600 receives via AS 400 (400, 200, 300).
AS 600 will choose AS 400 (400, 200, 300) as the better path.

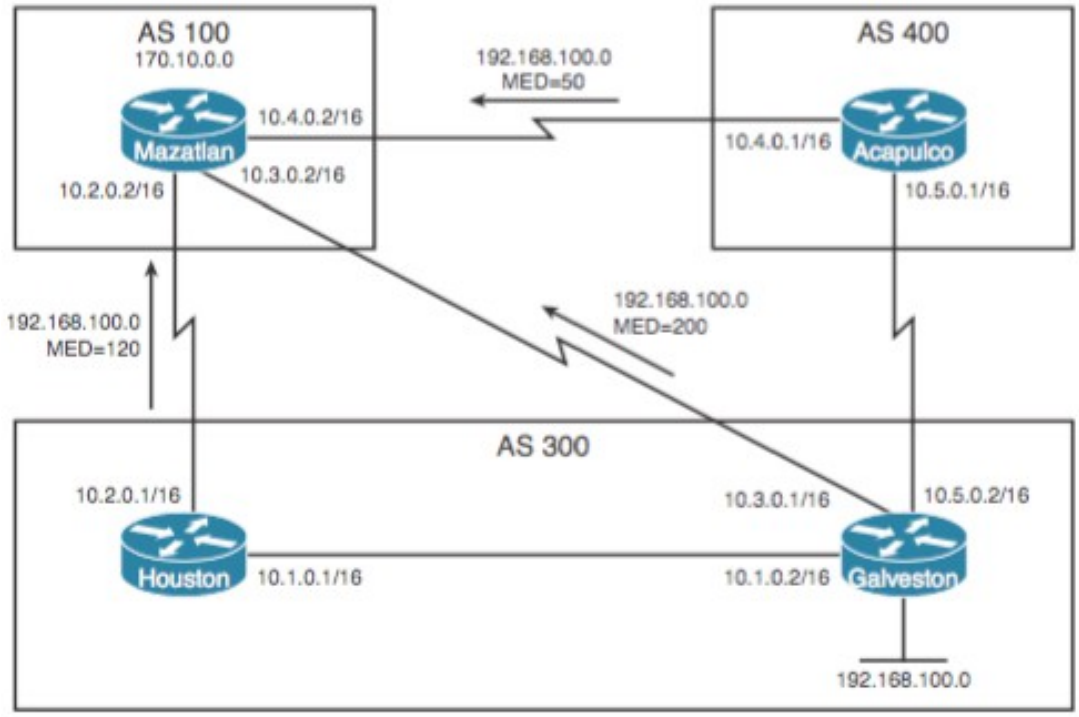
AS_PATH: Removing Private Autonomous Systems
Private ASNs (64,512 to 65,535) cannot be passed on to the Internet because they are not unique. Use **remove-private-as**, to strip private ASNs out of the AS_PATH list before advertised



RTB(config)**#router bgp 1**
RTB(config-router)**#neighbor 172.16.20.2 remote-as 65001**
RTB(config-router)**#neighbor 198.133.219.1 remote-as 7**
RTB(config-router)**#neighbor 198.133.219.1 remove-private-as**

<u>MED Attribute</u> (Multi-Exit Discriminator)
Also called ***the BGP metric***, indicates the preferred path is into an AS between two eBGP neighbors.
- AS to AS specific value describing that hop only- is not later handed to the next peer.
- A lower value is preferred over a higher value.
- The default value is 0. Change the default using the **default-metric**
- By default, only for paths from neighbors in the same AS. Override with **bgp always-compare-med**

Below, the routes for AS 300 to choose from are from Houston and Galveston. This is where MED is used by default (what it is intended for), so it won't be looked at in the case of AS 400 (Acapulco) unless it is specifically told to



Example: influence Mazatlan router to choose Houston as the entry point for AS 300 to reach 192.168.100.0.

Mazatlan(config)**#router bgp 100**
Mazatlan(config-router)**#neighbor 10.2.0.1 remote-as 300**
Mazatlan(config-router)**#neighbor 10.3.0.1 remote-as 300**
Mazatlan(config-router)**#neighbor 10.4.0.1 remote-as 400**
Acapulco(config)**#router bgp 400**
Acapulco(config-router)**#neighbor 10.4.0.2 remote-as 100**
Acapulco(config-router)**#neighbor 10.4.0.2 route-map SETMEDOUT out**
- reference route map SETMEDOUT.
Acapulco(config-router)**#neighbor 10.5.0.2 remote-as 300**
Acapulco(config)**#route-map SETMEDOUT permit 10**
- create route map SETMEDOUT sequence number of 10
Acapulco(config-route-map)**#set metric 50**
- set MED to 50

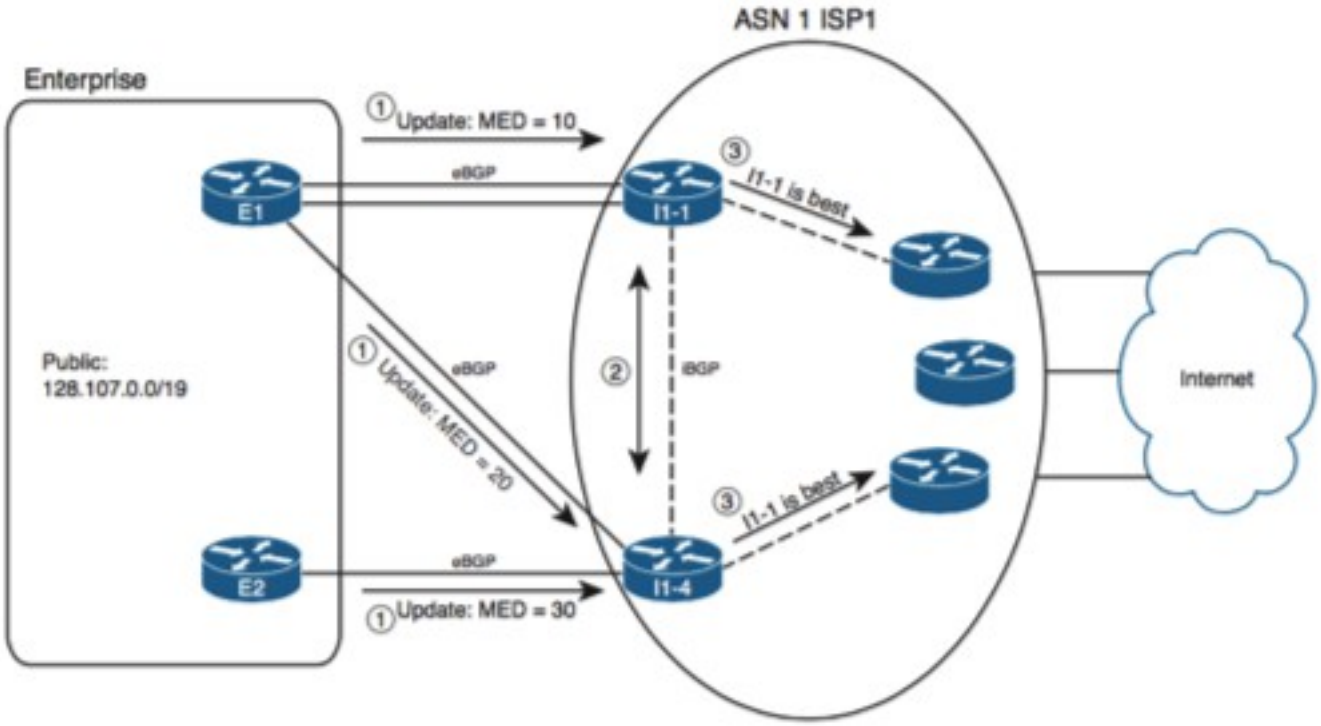
Houston and Galveston have same setup with relative neighbor statements. Below are MEDs they set:
Houston(config-router)**#neighbor 10.2.0.2 route-map SETMEDOUT out**
Houston(config-router)**#neighbor 10.1.0.2 remote-as 300**
Houston(config-route-map)**#set metric 120**
Galveston(config-router)**#neighbor 10.3.0.2 remote-as 100**
Galveston(config-router)**#neighbor 10.3.0.2 route-map SETMEDOUT out**
Galveston(config-route-map)**#set metric 200**

Mazatlan can only compare the MED attribute coming from Houston (120) to the MED attribute coming from Galveston (200) even though the update coming from Acapulco has the lowest MED value. Mazatlan will choose Houston as the best path for reaching network 192.168.100.0.

To force Mazatlan to include updates for network 192.168.100.0 from Acapulco in the comparison, use the **bgp always-compare-med** router configuration command on Mazatlan:
Mazatlan(config)**#router bgp 100**
Mazatlan(config-router)**#neighbor 10.2.0.1 remote-as 300**
Mazatlan(config-router)**#neighbor 10.3.0.1 remote-as 300**
Mazatlan(config-router)**#neighbor 10.4.0.1 remote-as 400**
Mazatlan(config-router)**#bgp always-compare-med**
Assuming that all other attributes are the same, Mazatlan will choose Acapulco as the best next hop for reaching network 192.168.100.0.

The most recent IETF decision about BGP MED assigns a value of infinity to a missing MED, making that route the least preferred. The Cisco IOS default is the opposite - treating routes without the MED attribute as having a MED of 0, making the route that is lacking the MED variable the **most** preferred.
- To configure the router to conform to the IETF standard, use **bgp bestpath missing-as-worst**.

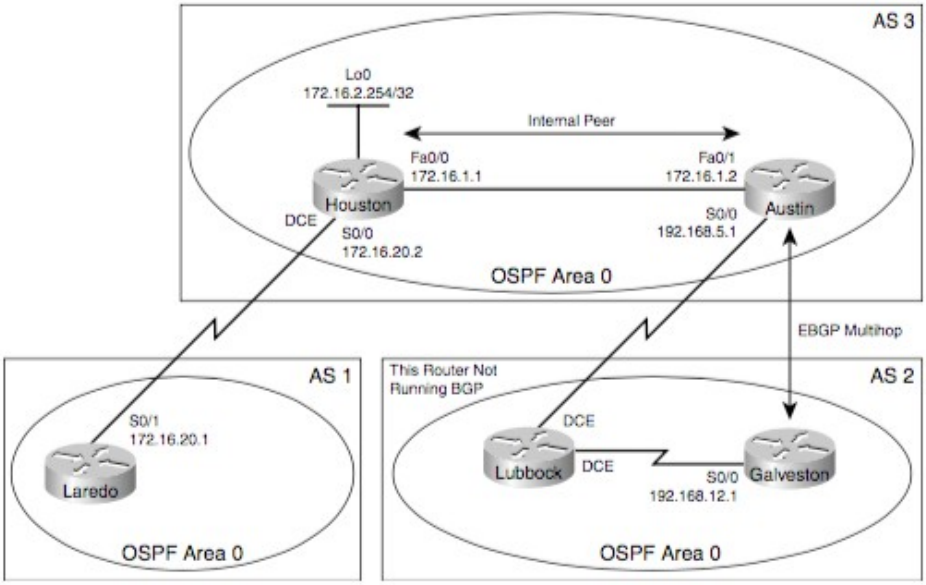
MED with a dual-homed single ISP
The enterprise can announce a MED that tells the ISP which path into the enterprise is best. As a result, the ISP can discriminate between the multiple exit points from that ISP to the enterprise. Those inbound routes into the enterprise from the ISP typically consist of either one, or a few, public IP address ranges. Say an enterprise engineer prefers the top BGP neighborship as the best path to use for inbound routes (MED 10), the middle link next (MED 20), and the bottom connection last (MED 30)



Step 1. E1 and E2 advertise 128.107.0.0/19, setting MED with an outbound route map, to various settings: MED 10 sent by E1 to I1-1, MED 20 sent by E1 to I1-4, and MED 30 sent by E2 to I1-4.
Step 2. I1-1 and I1-4 have an iBGP connection, so they learn each other's routes and agree as to which route wins based on MED.
Step 3. I1-1 and I1-4 also tell the other routers inside ISP1, causing all inbound traffic to funnel toward Router I1-1.
(Remember MED is advertised by one AS into another, propagated inside the AS, but not sent to another AS.

Note that Routers I1-1 and I1-4 in this example could have chosen a better route based on all the earlier best-path steps. However, a brief analysis of the steps tells us that unless someone makes an effort to override the effects of MED, these routers' best-path algorithms will use MED. Assuming that the enterprise and ISP agree to rely on MED, the earlier best-path steps should not matter.

BGP Configuration Example



Houston Router

Router(config)#hostname Houston	Sets the router name to Houston.
Houston(config)#interface loopback 0	Moves to loopback interface mode.
Houston(config-if)#ip address 172.16.2.254 255.255.255.255	Assigns an IP address and netmask.
Houston(config-if)#interface fastethernet 0/0	Moves to interface configuration mode.
Houston(config-if)#ip address 172.16.1.1 255.255.255.0	Assigns an IP address and netmask.
Houston(config-if)#no shutdown	Enables the interface.
Houston(config-if)#interface serial 0/0/0	Moves to interface configuration mode.
Houston(config-if)#ip address 172.16.20.2 255.255.255.0	Assigns an IP address and netmask.
Houston(config-if)#clock rate 56000	Assigns the clock rate.
Houston(config-if)#no shutdown	Activates the interface.
Houston(config-if)#exit	Returns to global configuration mode.
Houston(config)#router ospf 1	Starts the OSPF routing process.
Houston(config-router)#network 172.16.0.0 0.0.255.255 area 0	Any interface with 172.16.*x*.*x* to be placed into OSPF area 0.
Houston(config-router)#exit	Returns to global configuration mode.
Houston(config)#router bgp 3	Starts the BGP routing process.
Houston(config-router)#no synchronization	Turns off route synchronization.
Houston(config-router)#neighbor 172.16.1.2 remote-as 3	Identifies a peer router at 172.16.1.2.
Houston(config-router)#neighbor 172.16.1.2 update-source loopback 0	Use any interface for TCP connections, as long as Loopback0 is configured.
Houston(config-router)#neighbor 172.16.20.1 remote-as 1	Identifies a peer router at 172.16.20.1.
Houston(config-router)#no auto- summary	Disables auto-summarization.
Houston(config-router)#exit	Returns to global configuration mode.
Houston(config)#exit	Returns to privileged mode.
Houston#copy running-config startup- config	Saves the configuration to NVRAM.

Laredo Router

Router(config)#hostname Laredo	Sets the router name to Laredo.
Laredo(config)#interface serial 0/0/1	Moves to interface configuration mode.
Laredo(config-if)#ip address 172.16.20.1 255.255.255.0	Assigns an IP address and netmask.
Laredo(config-if)#no shutdown	Activates the interface.
Laredo(config-if)#exit	Returns to global configuration mode.
Laredo(config)#router bgp 1	Starts the BGP routing process.
Laredo(config-router)#no synchronization	Turns off route synchronization.
Laredo(config-router)#neighbor 172.16.20.2 remote-as 3	Identifies a peer router at 172.16.20.2.

Laredo(config-router)#no auto-summary	Disables auto-summarization.
Laredo(config-router)#exit	Returns to global configuration mode.
Laredo(config)#exit	Returns to privileged mode.
Laredo#copy running-config startup- config	Saves the configuration to NVRAM.

Galveston Router

Router(config)#hostname Galveston	Sets router name to Galveston.
Galveston(config)#interface serial 0/0/0	Moves to interface configuration mode.
Galveston(config-if)#ip address 192.168.12.1 255.255.255.0	Assigns an IP address and netmask.
Galveston(config-if)#no shutdown	Activates the interface.
Galveston(config-if)#exit	Returns to global configuration mode.
Galveston(config)#router ospf 1	Starts the OSPF routing process.
Galveston(config-router)#network 192.168.12.0 0.0.0.255 area 0	Any interface with 192.168.12.x to be placed into OSPF Area 0.
Galveston(config-router)#exit	Returns to global configuration mode.
Galveston(config)#router bgp 2	Starts the BGP routing process.
Galveston(config-router)#neighbor 192.168.5.1 remote-as 3	Identifies a peer router at 192.168.5.1.
Galveston(config-router)#neighbor 192.168.5.1 ebgp-multihop 2	Two routers not directly connected need an eBGP session.
Galveston(config-router)#no auto-summary	Disables auto-summarization.
Galveston(config-router)#exit	Returns to global configuration mode.
Galveston(config)#exit	Returns to privileged mode.
Galveston#copy running-config startup- config	Saves the configuration to NVRAM.

Austin Router

Router(config)#hostname Austin	Sets the router name to Austin.
Austin(config)#interface serial 0/0/0	Moves to interface configuration mode.
Austin(config-if)#ip address 192.168.5.1 255.255.255.0	Assigns an IP address and netmask.
Austin(config-if)#no shutdown	Activates the interface.
Austin(config-if)#interface fastethernet 0/1	Moves to interface configuration mode.
Austin (config-if)#ip address 172.16.1.2 255.255.255.0	Assigns an IP address and netmask.
Austin(config-if)#no shutdown	Activates the interface.
Austin(config-if)#exit	Returns to global configuration mode.
Austin(config)#router ospf 1	Starts the OSPF routing process.
Austin(config-router)#network 172.16.0.0 0.0.255.255 area 0	Any interface with 172.16.*x*.*x* to be placed into OSPF area 0.
Austin(config-router)#network 192.168.5.0 0.0.0.255 area 0	Any interface with 192.168.5.*x* to be placed into OSPF area 0.
Austin(config-router)#exit	Returns to global configuration mode.
Austin(config)#router bgp 3	Starts the BGP routing process.
Austin(config-router)#no synchronization	Turns off route synchronization.
Austin(config-router)#neighbor 172.16.2.254 remote-as 3	Identifies a peer router at 172.16.2.254.
Austin(config-router)#neighbor 192.168.12.1 remote-as 2	Identifies a peer router at 192.168.12.1.
Austin(config-router)#neighbor 192.168.12.1 ebgp-multihop 2	Two routers not directly connected need eBGP session.
Austin(config-router)#no auto-summary	Turns off auto-summarization.
Austin(config-router)#exit	Returns to global configuration mode.
Austin(config)#exit	Returns to privileged mode.
Austin#copy running-config startup- config	Saves the configuration to NVRAM.

Syntax for BGP Filtering Methods

Traditional access lists not great for granular BGP filters
The old-school IOS ACLs we know might be seem like a good way to match traffic by prefix/subnet, but it can get to be a mess. Cisco IOS 12.0 came up with a better way with **prefix-list**, but it helps to know why this was done.

Here is an easy scenario: declare BGP neighbors, make a simple access list to match one subnet, and apply it with **distribute-list** to the updates going **out**, so we can say "don't send routing updates from that **access-list 1** to this neighbor". This works fine.

R1(config)# **router bgp 3**
R1(config-router)# **neighbor 172.16.1.2 remote-as 3**
R1(config-router)# **neighbor 172.16.20.1 remote-as 1**
R1(config-router)# **neighbor 172.16.20.1 distribute-list 1 out**
R1(config-router)# **exit**
R1(config)# **access-list 1 deny 192.168.10.0 0.0.0.255**
R1(config)# **access-list 1 permit any**

Problems come up when advertising the aggregate address of 172.16.0.0/16 but not the individual subnet- you need an extended ACL. The way those work with BGP route filters, the ACL will first match the network address and *then* match the subnet mask of the prefix. To do this, both network and netmask are paired with their own wildcard bitmask and it looks like this confusing, ugly mess:

access-list 101 permit ip 172.16.0.0 0.0.255.255 255.255.0.0 0.0.0.0

That is why IOS 12.0 gave us the **prefix-list** option.

Prefix-List Syntax
This example limits updates sent to the peer to only be routing info for this specific matching subnet:
R1(config)# **ip prefix-list UPDATE172 permit 172.16.0.0/16**
R1(config)# **router bgp 100**
R1(config-router)# **neighbor 192.168.1.1 remote-as 200**
R1(config-router)# **neighbor 192.168.1.1 prefix-list UPDATE172 out**
Applies UPDATE172 to routing updates sent to 192.168.1.1. Permits update of 172.16.0.0/16, but has implicit deny at end of the list for others like 172.16.0.0/17 or 172.16.20/24

ip prefix-list list-name **[seq** seq-value**] deny | permit** network/cidr **[ge** ge-value**] [le** le-value**]**

Parameter	Description
list-name	The name of the prefix list.
seq	(Optional) Applies a sequence number to the entry being created or deleted.
seq-value	(Optional) Specifies the sequence number.
deny	Denies access to matching conditions.
permit	Permits access for matching conditions.
network/cidr	(Mandatory) The network number and length (in bits) of the netmask.
ge	(Optional) Applies *ge-value* to the range specified.
ge-value	(Optional) Specifies the beginning or "from" value of a range
le	(Optional) Applies *le-value* to the range specified.
le-value	(Optional) Specifies the end or "to" value of a range

- **-** There is an **implicit deny** statement at the end of each prefix list.
- The range of sequence numbers that can be entered is from 1 to 4,294,967,294.
- A router tests for prefix list matches from the lowest sequence number to the highest.
- By numbering your **prefix-list** statements, you can add new entries at any point in the list.
- If no seq # is given, default is applied: 5 is applied to the first, next unnumbered entries are incremented by 5.

ip prefix-list NY_ROUTES permit 192.0.0.0/8 le 24
Permit routes with a netmask of up to 24 bits in 192.0.0.0/8 - No seq number given- gets the default 5
ip prefix-list NY_ROUTES deny 192.0.0.0/8 ge 25
Deny routes with netmask of 25 bits or greater in 192.0.0.0/8 - No seq num given- 10 is applied (+5)
ip prefix-list RENO permit 10.0.0.0/8 ge 16 le 24
Permit routes 10.0.0.0/8 with a netmask between 16 to 24 bits - No seq number given- gets the default 5
ip prefix-list HOUSTON seq 3 deny 0.0.0.0/0
Assigns a sequence number of 3 to this statement.
no ip prefix-list TORONTO seq 10
Removes sequence number 10 from the TORONTO list.

AS_PATH Access Lists

The AS_PATH attribute can be searched with regular expressions to match routing updates, and slap with an access-list label to filter or modify. Most of these should look familiar. They can also be tested with **show ip bgp**

^	Matches the beginning of the input string.
$	Matches the end of the input string.
^$	Matches an empty string in the AS_PATH field (the local device's AS is blank, so this matches that)
_	Matches a space, comma, left brace, right brace, beginning or end of an input string
.	Matches any single character.
*	Matches 0 or more single- or multiple-character patterns.

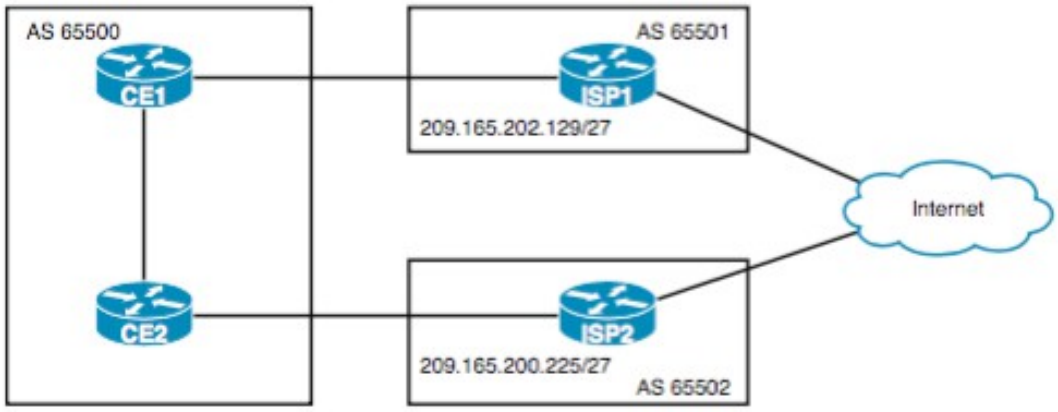
To find all subnets reachable via autonomous system 65002 (AS_PATH begins with 65002):

R1**#show ip bgp regexp ^65002_**

Network	Next Hop	Metric	LocPrf	Weight	Path
*>i172.16.0.0	192.168.28.1	100	0		65002 65003 i
* i172.24.0.0	192.168.28.1	100	0		65002 65003 65004 65005 i

show ip bgp regexp _65004$	... originating from autonomous system 65004 (AS_PATH ends with 65004):
show ip bgp regexp ^65002_	... reachable via autonomous system 65002 (AS_PATH begins with 65002):
show ip bgp regexp _65005_	... transiting through autonomous system 65005 (AS_PATH contains 65005):
show ip bgp regexp 2150	Will match 2150, 12150 or 21507. This is where "_" helps
show ip bgp ^$	Originates from THIS autonomous system (AS_PATH is blank)

This simple example demonstrates using prefix lists and AS_PATH access-lists together. Notice how the AS_PATH list is referred to with the **filter-list** keyword. The first job is to allow CE1 and CE2 to only learn ISP routes with a mask greater than / 15 (ge 16) and less than /25 (le 24). The second job is to ensure that AS 65000 does not become a transit autonomous system for ISP1 to reach ISP2 (and vice versa).



CE1 Configuration

ip prefix-list ISP1 permit 0.0.0.0 ge 16 le 24	Prefix list which only permits routes with a mask between 16 and 24
ip as-path access- list 1 permit ^$	Creates AS_PATH ACL matching routes that only originate from within autonomous system 65500
router bgp 65000	
neighbor 209.165.202.129 prefix-list ISP1 in	Assigns ISP1 prefix list to neighbor 209.165.202.129 (ISP1) for all routes learned from that neighbor
neighbor 209.165.202.129 filter-list 1 out	Assigns AS_PATH ACL to neighbor 209.165.202.129 (ISP1) for all routes sent to that neighbor

CE2 Configuration

ip prefix-list ISP2 permit 0.0.0.0 ge 16 le 24	Creates a prefix list that only permits routes with a mask between 16 and 24
ip as-path access- list 1 permit ^$	Creates an AS_PATH access list matching routes that only originate from within AS 65500
router bgp 65000	
neighbor 209.165.200.225 prefix-list ISP2 in	Assigns ISP2 prefix list to neighbor 209.165.200.225 (ISP2) for all routes learnt from that neighbor
neighbor 209.165.200.225 filter-list 1 out	Assigns AS_PATH ACL to neighbor 209.165.200.225 (ISP2) for all routes sent to that neighbor

Route Maps
This example uses a prefix-list to catch updates matching 172.16.10.0/24 and label with "AS400_ROUTES". Then a route-map is created so we can basically make a conditional statement that says, if it sees those, make a change to the weight attribute. If it doesn't match and falls through to match the second rule, set a different weight. This same mechanism can be used to make other changes, or specify what updates to use or drop.

Finally, at the end make the BGP process, add the neighbors and directs to check the route-map for the rules. Adding the keyword "in" refers to routing updates from 192.168.7.1, and the keyword "out" is used for the routing updates we send out 192.168.7.1

Houston(config)**#ip prefix-list AS400_ROUTES permit 172.16.10.0/24**
Creates a prefix list that matches the 172.16.10.0/24 network belonging to AS 400.
Houston(config)**#route-map SETWEIGHT permit 10**
Creates a route map called SETWEIGHT. A sequence number of 10 is assigned.
Houston(config-route-map)**#match ip address prefix-list AS400_ ROUTES**
Specifies the condition under which policy routing is allowed, matching the AS400_ROUTES prefix list.
Houston(config-route-map)**#set weight 200**
Assigns a weight of 200 to any route update that meets the condition of prefix list AS400_ROUTES.
Houston(config-route-map)**#route- map SETWEIGHT permit 20**
Creates a second statement for the route map. A sequence number of 20 is assigned.
Houston(config-route-map)**#set weight 100**
Assign weight of 100 to all route updates/networks learned from outside (didn't match) AS400_ROUTES
Houston(config-route-map)**#exit**
Returns to global configuration mode.
Houston(config)**#router bgp 300**
Starts the BGP routing process.
Houston(config-router)**#neighbor 192.168.7.1 route-map SETWEIGHT in**
Uses the route map SETWEIGHT to filter all routes learned from neighbor 192.168.7.1.

Route-map can employ regular and extended ACLs, prefix-lists, and AS_PATH access-lists.

BGP subcommand	Commands to create	What can be matched
neighbor distribute-list (standard ACL)	access-list, ip access-list	Prefix, with WC mask
neighbor distribute-list (extended ACL)	access-list, ip access-list	Prefix and prefix length, with WC mask for each
neighbor prefix-list	ip prefix-list	Exact or "first N" bits of prefix, plus range of prefix lengths
neighbor filter-list	ip as-path access-list	AS_Path contents; all NLRI whose AS_Paths are matched considered to be a match
neighbor route-map	route-map	Prefix, prefix length, AS_ Path, and/or any other PA matchable within a BGP route map

BGP Communities
BGP communities are used to group networking devices that share common properties, regardless of network, autonomous system, or any physical boundaries. In large networks applying a common routing policy through prefix lists or access lists requires individual peer statements on each networking device. Using the BGP community attribute BGP neighbors, with common routing policies, can implement inbound or outbound route filters based on the community tag rather than consult large lists of individual permit or deny statements. Consider a community a group of prefixes that should be treated the same way and an alternative to matching numerous prefixes using an access-list or prefix-list"

Communities are created with **set community** as below. The aa:nn identifier is usually the AS followed by an arbitrary identifier (it would look something like 100:250) several can be entered on the same line. The options in brackets are for built in communities with specific rules built in.

set community aa : nn [additive | local-as | no-advertise | no-export]
additive - Adds to existing community (one or more keywords)
local-as (aka "well-known community) advertised to only peers in local AS or within a sub-AS of a confederation.
no-advertise - these are not advertised to any peer (internal or external).
no-export routes are like local-as but more strictly not advertised to external peers.

When declaring neighbors, you can share communities with it with **neighbor x.x.x.x send-community**. Here are some snippets demonstrating the use of communities:

```
ip bgp-community new-format
access-list 101 permit ip host 6.6.6.0 host 255.255.255.0
route-map Peer-R1 permit 10
 match ip address 101
 set community 100:300 109:02 33:40

ip community-list 1 permit 100:300
route-map Peer-R3 permit 10
 match community 1
 set local-preference 130
```

ip community-list -- Creates a community list for BGP and control access to it.
match community -- Matches a BGP community.
set comm-list delete -- Removes communities from the community attribute of an inbound or outbound update.
show ip bgp community -- Displays routes that belong to specified BGP communities.

BGP Confederations
A BGP confederation divides our AS into sub-ASes to reduce the number of required IBGP peerings. Within a sub-AS we still require full-mesh IBGP but between these sub-ASes we use something that looks like EBGP but behaves like IBGP (called confederation BGP)

```
R2(config)#router bgp 24
R2(config-router)#bgp confederation identifier 2
R2(config-router)#bgp confederation peers 35
R2(config-router)#neighbor 4.4.4.4 remote-as 24
R2(config-router)#neighbor 4.4.4.4 update-source loopback 0
R2(config-router)#neighbor 3.3.3.3 remote-as 35
R2(config-router)#neighbor 3.3.3.3 update-source loopback 0
R2(config-router)#neighbor 3.3.3.3 ebgp-multihop 2
```

When you start the BGP process you have to use the AS number of the sub-AS. Secondly, you have to use the **bgp confederation identifier** command to tell BGP what the main AS number is.
We also have to configure all other sub-AS numbers with the **bgp confederation peers** command, in this case that's only AS 35. R4 is in the same sub-as so you can configure this neighbor just like any other IBGP neighbor. R3 is a bit different though…since it's in another sub-AS we have to use the same rules as EBGP, that means configuring multihop if you are using loopbacks.

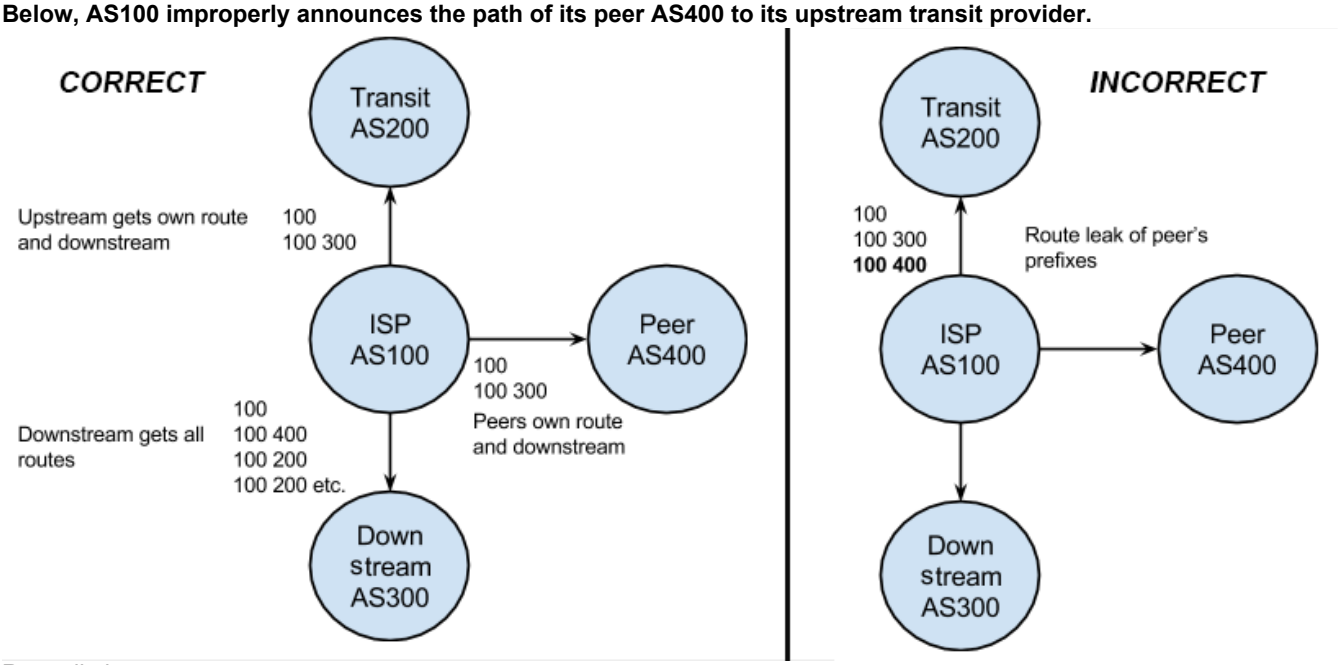
When you look up sh ip bgp x.x.x.x the route is tagged with either **confed-internal** (which means that it came from an IBGP router within the same sub-AS) or **confed-external.** With confed-external the sub-AS number simply appears in parenthese above the given route info. BGP confederations use an attribute called AS_CONFED_SET. This "confederation set" prepends the list with the sub-AS's. Generally speaking, when routes are sent to another AS, all the sub-AS numbers are removed

```
R5#show ip bgp 11.11.11.11
BGP routing table entry for 11.11.11.11/32, version 6
Paths: (2 available, best #2, table Default-IP-Routing-Table)
  Advertised to update-groups:
     2
  (24) 1
    192.168.12.1 (metric 3) from 4.4.4.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, confed-external
  (24) 1
    192.168.12.1 (metric 3) from 3.3.3.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, confed-internal, best
```

BGP Route Leaks and BGP Hijacking

Route leaks and hijacks are where illegitimate prefixes/address blocks are wrongly propagated
- illegitimate advertisement of prefixes, blocks of IP addresses
- propagate across networks and lead to incorrect or suboptimal routing.
- Can happen from an AS originating a prefix that it does not actually own
- Also happens when AS announces it can deliver traffic through a route that should not exist
- Are particularly prone to propagation when
    - a more specific prefix is advertised (BGP prefers the most specific block of addresses)
    - a path is advertised that is shorter than the currently available paths (BGP prefers shortest AS Path).
- Usually happen when BGP advertisements are not properly filtered using the no-export community.

Below, AS100 improperly announces the path of its peer AS400 to its upstream transit provider.



Remediation
- Maintain communication and work with administrators of neighboring AS'es to collaborate on fixes
- BGP-advertise routes more preferable than the leaked route (a more specific prefix length and/or shortest path)
- For example, if the problematic route advertises a /17, counter with advertising an /18
- Advertising a shorter path is not as effective
- Last resort: consider changing your prefixes entirely by changing DNS records. Only feasible if alternate locations are available (i.e., other data center). The TTL on DNS records also becomes an issue.

Preventative measures
- publish Route Origin Authorizations (ROAs) in the various regional Internet registries (RIRs)
- verifies that a given origin AS is authorized to announce its prefixes, (incl. max prefix lengths)
- networks using Resource Certification (RPKI) can validate the origin AS and verify routes legitimacy

Best practices: Proper route filtering uses a set of robust filtering rules, likely including:
- Filter out Bogon prefixes and routes with Bogon ASNs anywhere in the AS path. Reserved or unallocated IP spaces should never be advertised.
- Filter out routes with more than two Tier 1 (“transit-free”) networks in the AS path. When more are present, at least one of the networks is providing transit to another.
- If you don’t sell transit to large networks (like Tier 1 networks), filter out routes from them in the AS path. Use with a whitelist of prefixes that each of your customers may announce to you.
- Use peer locking- ask peers for all possible upstream networks, and only allow those as intermediate networks (See Snijders NANOG67 - https://www.nanog.org/sites/default/files/Snijders_Everyday_Practical_Bgp.pdf
- Use BGP Maximum-Prefix to set the maximum number of prefixes that can be announced from your peers. This helps during an episode of rapid propagation of problematic AS'es.

2008- Pakistan's ISPs hijacked YouTube’s routes with a more specific prefix pointing to a null interface, The covered prefix was mistakenly leaked to the rest of the Internet, blocking YouTube for everyone else too. The practice of null routing commonly appears as a part of censorship efforts, including the Great Firewall of China (https://blog.thousandeyes.com/deconstructing-great-firewall-china/).

http://dyn.com/blog/mitm-internet-hijacking/
Bigger real-world examples: https://blog.thousandeyes.com/finding-and-diagnosing-bgp-route-leaks/

Setting up GRE with IPSEC for a Typical VPN

I made this from a transcription I made of Doug Suida's video at:
https://www.youtube.com/watch?v=2PtK8HgkRvM

A followup that uses the same technique to set up DMVPN (Dynamic Multipoint VPN) is here:
https://www.youtube.com/watch?v=WEzo1UvMpg0 - My summarization is on the next pages.

On network mapping it looks like these aren't directly connected but "sh cdp nei only shows f/0/0 - doesn't show as a directly connected neighbor. Neighborship won't happen until you add to OSPF network 10.10.1.0 0.0.0.3 area 0

Outbound connection:
R1 ==> 162.27.193.128/30 ==>========INTERNET========<== 45.12.153.200/30 <== R2
Tunnel inside
10.10.1.1 ==>Tunnel0 << ------> ====== 10.10.1.0/30 ====== < ------- >> Tunnel0 <== 10.10.1.2

Setting up the LANs, uplinks, and tunnel

Local: 192.168.1.0/24
```
R1> int Loopback0
R1> ip address 192.168.1.1
R1> int f0/1
R1> ip address 192.168.1.2
R1> router ospf 123
R1> network 192.168.1.0 0.0.0.255 area 0
```
Uplink
```
R1> int f0/0
R1> ip address 162.27.193.130
R1> exit
```
Tunnel
```
R1> int tunnel0
R1> tunnel source f0/0
R1> tunnel destination 45.12.153.202
R1> ip address 10.10.1.1 255.255.255.252
```

Local: 10.1.1.0/24
```
R2> int Loopback0
R2> ip address 10.1.1.1
R2> int f0/1
R2> ip address 10.1.1.2
R2> router ospf 123
R2> network 10.1.1.0 0.0.0.255 area 0
```
Uplink
```
R2> int f0/0
R2> ip address 45.12.153.202
```
Tunnel
```
R2> int tunnel0
R2> tunnel source f0/0
R2> tunnel destination 162.27.193.130
R2> ip address 10.10.1.2 255.255.255.252
R2>
```

These are also added on both routers but not needed
> tunnel path-mtu-discovery
> ip ospf mtu-ignore

The first one gets the tunnel to do it's own MTU discovery since it is over the internet
The second one is to do the same with OSPF to make sure it stays neat
You can put in a keepalive on the tunnel but don't need it since it's a logical interface
If you are skiddish about OSPF go ahead.
On OSPF: no tunnel = no multicast and no neighborship

So now, after testing and making sure it's all working, we can add standard VPN mechanisms with IPSEC.
You might otherwise have many policies and transform sets but here just one.

1. Define traffic to be encrypted

```
R1> ip access-list extended MY-IPSEC-TRAFFIC
R1> remark VPN Traffic
R1> permit gre host 162.27.193.130 host 45.12.153.202
```

You'll do the same on R2 with the address reversed

2. Set up Phase I, with a preshared key rather than a certificate

```
R1> crypto isakmp policy 1
R1> authentication pre-share
####### that is where you'd put cert-based rsa-encr, rsa-sig
R1> encryption aes 128
####### other options 3des des aes and 128, 192, 256
R1> hash sha
####### you can also choose to put md5
R1> group 2
####### choose DH group 1, 2 or 5
R1> lifetime 86400
####### 60-86400 seconds (86400 sec = 1 day, is the default)
R1> crypto isakmp key 0 mypassword address 45.12.153.202
####### If the password you are giving here is encrypted or not:  0 is plaintext, 6 is encrypt
```

3. Set up Phase II - make the IPSEC transform set

```
R1> crypto ipsec transform-set TRANSFM-TUN esp-aes 128 esp-sha-hmac
####### There are tons of transforms to use.  The above are the defaults.
R1> mode tunnel
####### Or transport
```

4. Create crypto-map

```
R1>  crypto map CRYPTOMAPPY 1 ipsec-isakmp
R1(config-cryptomap)> description DESC to R2
R1(config-cryptomap)> match address MY-IPSEC-TRAFFIC
####### Name of the ACL made in step 1
```

5. Apply to interfaces

```
R1> set peer 45.12.153.202
R1> set transform-set TRANSFM-TUN

R1> int f0/0
R1> crypto map CRYPTOMAPPY
R1> int tunnel 0
R1> crypto map CRYPTOMAPPY
```

Test

```
R1> show crypto ipsec sa
```

Show stats of # crypted/ decrypted - tells us if it really being encrypted
If you do a sh run config you won't see a lot of the stuff that were used here which were default settings

Valid Encryption Methods
esp-des
esp-3des (default)
esp-aes (128-bit encryption)
esp-aes-192
esp-aes-256
esp-null

Valid Authentication Methods
esp-md5-hmac
esp-sha-hmac (default)

DMVPN - Dynamic Multipoint VPN with mGRE (multipoint GRE), IPSec and NHRP

5 sites (routers), full mesh
R1 has individual tunnels to R2, R3, R4, and R5, and each of those routers have similar tunnels to eachother.
10 tunnels? DMVPN can automate the process. Sort of a hub and spoke configuration mashed in with a frame relay-style network with one hub site (not full mesh FR). With DMVPN, it turns out you don't have to configure all the tunnels after doing those coming out of R1. Those tunnels among the "spokes" routers are built dynamically as-needed by the DMVPN configuration.

If some host off of R4 needs something off of the network hosted by R2, it can dynamically create the tunnel so it can be used, wait for it to time out, and then tear it down again. Say R4 queries the hub router (R1), which provides the next hop information it needs to build the tunnel; it does so, and when they finish tear the tunnel down. In this scenario you don't need anything but interior routing protocols
On routing, lets say R3 adds a network and sends out an advertising update. The update only goes to the hub router- not the others, and it's the hub router that sends the update to the others. Sortof like a NBMA setup, and have to turn off split horizon, add Next Hop Resolution Protocol (NHRP), etc. Also- participant routers are never neighbors with any other router than the "hub" router, even when a tunnel is created. R1 is depended on for information needed to build the tunnels and propagate the routing updates, and no other participating router can communicate with another without this "hub" router as it's resource.

Here we set up R1, then just R2 and R3 - it will become apparent after one spoke is done, that it's a replication.
After setting up the "hub router" and the first spoke router, the rest of the spoke routers can use the same stuff with only the tunnel and gateway IPs needing to be set, and EIGRP tuned on.

Overview: EIGRP 10 with:
R1 - f0/0 - 54.45.12.1 to default GW .2/30 - internal network (loopback): 192.168.17.0/24 DMVPN IP: 192.168.1.1
R2 - f0/0 - 54.45.12.5 to 6/30 - internal network (loopback): 192.168.200.0/24 DMVPN IP: 192.168.1.12
R3 - f0/0 - 54.45.12.9 to 10/30 - internal network (loopback): 10.1.1.0/24 DMVPN IP: 192.168.1.3
Set up all of this before starting the exercise. Ping-test all the links and you are ready:

Differences from previous: we create a transform, no cryptomap with ACL. Create ipsec profile instead
1. Set ISAKMP policy and define pre-shared key
R1> crypto isakmp policy 10
R1> authentication pre-share
R1> encryption aes 192
R1> hash md5
R1> group 2
R1> crypto isakmp key 0 ISAKEY address 0.0.0.0 0.0.0.0 <---said isakey was name - no password?
####### Here we don't have a specific IP address like the other example. Put ANY with matching key (with 0's)

2. Phase II - make the IPSEC transform set
R1> crypto ipsec transform-set DMVPN-TRANSFM esp-aes 192 esp-md5-hmac

3. Make a profile (instead of the cryptomap for single tunnel)
R1> ipsec profile DMVPN-PROFILE
R1(ipsec profile)> set security-association lifetime seconds 120
####### 120 sec is the minimum. Is lifetime before tunnel gets torn down!
R1(ipsec profile)> set transform-set DMVPN-TRANSFM
####### DONE WITH IPSEC

Tunnel
R1> int tunnel0
R1> ip address 192.168.1.1 255.255.255.0
R1> no ip redirects
####### MTU for encryption on top of packets
R1> ip mtu 1440

####### Next Hop Resolution Protocol (NHRP) stuff
R1> ip nhrp authentication ISAKEY
R1> ip nhrp network-id 1

####### Allow multicast protocols to use dynamic tunnels, shut off split horizon and next-hop-self
R1> ip nhrp map multicast dynamic
R1> no ip split-horizon eigrp 10
R1> no next-hop-self
R1> tunnel source fastEthernet 0/0
R1> tunnel mode gre multipoint
R1> tunnel key 0
R1> tunnel protect profile DMVPN-PROFILE
R!> router eigrp 10
R1> network 192.168.1.0

So it should be understood at this point:
R2 and R3 both use tunnel0 to get to R1; tunnel0 goes from R2 to R3. It is the SAME tunnel for ALL connected routers and that is what multipoint GRE provides to us. Very much like FR- here one tunnel for the whole mess instead of 5, or 10 tunnels.

R2 - needs the same IPSEC config we just did. On R1, do a sh running config and copy ipsec material. Just copy and paste it into the R2 command line after config t is entered. Done.
Like the IPSEC config, the tunnel config is almost the same... don't copy and paste this quite yet- this is the configuration that is going to be copied, pasted into all the other spoke routers.

R2> int tunnel0
R2> ip address 192.168.1.2 255.255.255.0
R2> no ip redirects
R2> ip mtu 1440
R2> ip nhrp authentication ISAKEY
R2> ip nhrp map multicast dynamic
######## Set the hub for next hop info and map with it's gateway address:
R2> ip nhrp nhs 192.168.1.1
R2> ip nhrp map 192.168.1.1 54.45.12.1
R2> ip nhrp map multicast 54.45.12.1
R2> ip nhrp network-id 1
R2> tunnel source fastEthernet 0/0
R2> tunnel mode gre multipoint
R2> tunnel key 0
R2> tunnel protection profile DMVPN-PROFILE
R2> router eigrp 10
R2> network 192.168.1.0

Other spokes will be the same as R2 except for the router's ip address:
####### config t and copy IPSEC info from sh running confg on R2, paste.
R3> int tunnel0
R3> ip address 192.168.1.3 255.255.255.0
####### Copy and paste the rest of the tunnel from first running config here, then make EIGRP declaration:
R3> router eigrp 10
R3> network 192.168.1.0
That's all. Do this for as many spokes as you want and it "just works"

sh crypto isakmp sa -- this is really just to show phase 1 info - does show tunnel is up or not
sh crypto ipsec sa -- this is for actual tunnel stats - shows traffic data and encryption %'s

So, what happens when the "hub" goes down? Is there a way to have failover/redundancy? It turns out, when naming NHS's you're not limited to 1 IP addy- you just need map commands for all added. This addresses the SPOF problem.

TCP Ports - Interactions and Scanning

SYN	Synchronize. Initiates a connection between hosts.
ACK	Acknowledge. Established connection between hosts.
PSH	Push. System is forwarding buffered data.
URG	Urgent. Data in packets must be processed quickly.
FIN	Finish. No more transmissions.
RST	Reset. Resets the connection.

TCP Connect and SYN stealth (aka half-open) scans
TCP connect (-sT) is the most reliable scan type but also the noisiest and most detectable since it attempts the standard 3-way TCP handshake to a port. If that port is open and unfiltered, it will think there is a connection attempt it is waiting for. When a SYN stealth scan (-sS) receives an ACK response it will shoot back a RST to slam the connection shut. It is also gives more clear results than inverse scans. To Nmap, SYN/ACK means port is open, RST means port is closed and no response (even on retrying) or ICMP back means it is filtered en route.

Packet fragmentation can evade IDS packet filtering and detection. A stealth SYN can be used in a way that splits the TCP header over several IP packets to mess with firewalls. Some firewalls may have rule sets that block IP fragmentation queues, but might not due to the adverse effect on network performance
Example: nmap -sS -T4(time delay) -A -f -v 192.168.0.2

Inverse TCP Flag scanning sends packets with various TCP flags (or no flags) enabled.
- *When the port is open (or firewall-filtered), no response comes from the host.*
- *When the port is closed, a RST/ACK is received from the target host.*

It's often said these are called inverted since responses are sent back only by closed ports. Inverse TCP scans are named for the flags they use, NMap has -sF (FIN scan, with only the FIN flag set), -sN (Null scan with no flags set) and -sX for an Xmas scan setting all three FIN, URG, PUSH flags. Nmap doesn't have a URG and PSH similar individual shortcut (like --flag URG), Hping has -U and -P, but these don't yield different results than other inverse scans (NUL, FIN and Xmas all you really need). A maimon scan sends packets with both the FIN and ACK flags set. It was considered more useful but not in modern times.

NULL, FIN and Xmas scans: if we get a RST, we know the port is closed. If it's not closed, the remote host shouldn't respond so it's either open or filtered

Caveats: According to RFC 793, a RST/ACK packet must be sent for connection reset, when the port is closed on host side. RFC 793 is completely ignored in Windows, so you won't get a RST/ACK response when trying to interact with the closed port. So, is only effective when target is not a Windows OS. If an "ICMP unreachable" is returned Nmap considers the port firewall-filtered. These might get through some firewalls (FIN most likely), and sometimes no response doesn't mean a port is open, since instead the packet was dropped by the firewall.
Finally, for ALL types of scans, just because a method can pass through a firewall does NOT necessarily mean it is slipping through undetected!

ACK scans (are not considered inverse)
Sent with just the ACK flag set, these never determines ports are closed or open (or even open/filtered). It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered, which usually means running other scans and cross-referencing the results. The simplest usage shown below, determines getting ***a RST back means "unfiltered"*** (RFC 793 again), and either an ICMP unreachable or no result at all is "filtered" (firewalls that block usually make no response or send back an ICMP destination unreachable)

```
# nmap -sA -T4 scanme.nmap.org
        Not shown: 994 filtered ports
        22/tcp  unfiltered ssh
        25/tcp  unfiltered smtp …
```

Forbidding anything but outbound connections is fine, but blocking ACK packets without state info doesn't tell which side started the connection. To block unsolicited ACK packets and allow packets belonging to legitimate connections, firewalls must stateful. Either way, the stateless approach is still quite common (netfilter/iptables, basic zone-based, etc). Cross-reference a SYN scan below shows us both getting responses; SYN scan says 98/100 ports are filtered, ACK says all are filtered (all getting RST back). This is a stateless use of "iptables -A INPUT -m multiport -p tcp --destination-port 22,80 -j ACCEPT" on the firewall.

 # nmap -sS -p1-100 -T4 para
Not shown: 98 filtered ports
22/tcp open ssh
80/tcp closed http
Nmap done: 1 IP address (1 host up) scanned in 3.81 seconds

 # nmap -sA -p1-100 -T4 para
All 100 scanned ports on para (192.168.10.191) are: unfiltered

Nmap done: 1 IP address (1 host up) scanned in 0.70 seconds
The ACK scan shows some packets are probably reaching the destination host- firewall forgery is always possible. Other scan types, such as FIN scan, may even be able to determine which ports are open and thus infer the purpose of the hosts. While you may not be able to establish TCP connections to those ports, they can be useful for determining which IP addresses are in use, OS detection tests, certain IP ID shenanigans, and as a channel for tunneling commands to rootkits installed on those machines. Such hosts may be useful as zombies for an IP ID idle scan.

Two ways of interpreting other info from ACK scans can be more revealing: checking the time-to-live (TTL) field and the WINDOW field of received packets

It doesn't tell us if a port is open or closed, but it does try to tell us if the
firewall is stateful (keeps tracks of connections) or not (probably just denies
incoming SYN packets).
If the firewall is non-stateful and just drops SYN packets, an ACK will get in
because it looks like a reply to something from the other side.
If an open OR closed port receives an unexpected ACK, it should send a RST back.
So if we get a RST back, then it means the firewall is non-stateful (or there's
just not one in place). If we don't get a response, or some ICMP unreachable is
sent, it's most likely filtered.

TTL-Base Analysis:
TTL value can be used as a marker of how many systems the packet has hopped through
Below, the value on the RST packets returned by port 22 is 50, whereas the other ports return a value of 80. This suggests that port 22 is open on the target host because the TTL value returned is smaller than the TTL boundary value of 64*.
1: host 192.168.0.12 port 21: F:RST TTL:80 WIN:0
2: host 192.168.0.12 port 22: F:RST **TTL:50** WIN:0
The *firewalk* assessment tool is similar http://www.packetfactory.net/projects/firewalk/.

Window-based Analysis:
If window field has non-zero value then port is open, no response, presumed filtered
1: host 192.168.0.20 port 22: F:RST -> ttl: 64 **win: 512**
2: host 192.168.0.20 port 23: F:RST -> ttl: 64 win: 0

The advantage of using ACK flag probe scanning is detection is difficult (for both IDS and host-based systems). The disadvantage is it relies on TCP/IP stack implementation bugs, which are prominent in BSD-derived systems but not in many other modern platforms.

hping3 -A 72.14.207.99 -p 80 -c 1

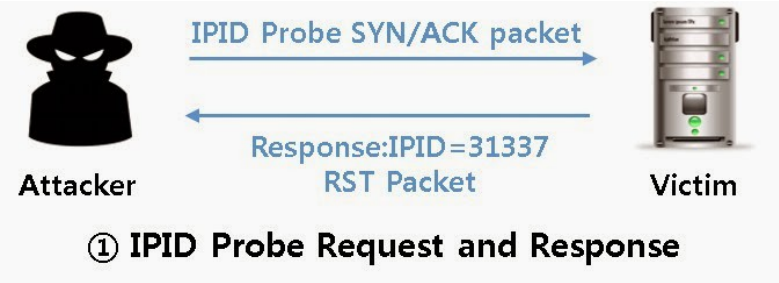
nmap -sW

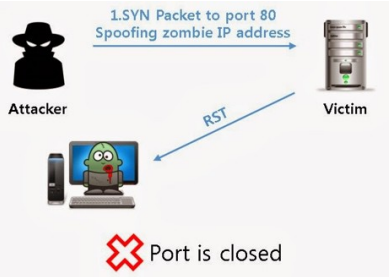
Some systems use a positive Window size for open ports, and zero for closed. Fewer systems do the exact opposite. If you scan and get tons of open ports and just a few closed ones, chances are it's the opposite. And some systems don't do either, so you can't always trust it.

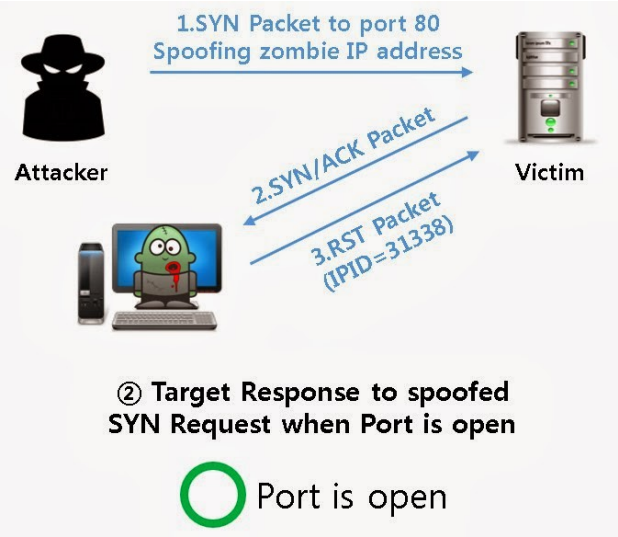
IDLE scan/ IP ID header scan - IP ID (IP fragment ID number)
Sends a spoofed source address to a computer to find out what services are available for complete blind scanning of a remote host. This is accomplished by spoofing another computer's IP address. No packet is send from your own IP address; instead, another host is used to scan the remote host and determine the open ports. This is done by expecting the sequence number of the zombie host and if the remote host checks the IP of the scanning party, the IP of the zombie machine will show up. You don't need access to the zombie machine to do any of this.

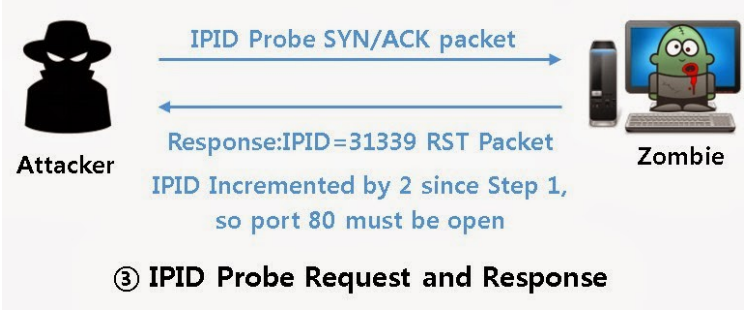
As noted, to determine if port is open, SYN scan full or stealth, get back an RST if closed, SYNACK if open.
Consider:
- An unsolicited SYNACK is responded to with a RST; An unsolicited RST will be ignored
- Each IP packet has a fragment ID (IPID), incremented for each sent. Check to get # of packets sent since probe

Step 1) Send an unsolicited SYNACK to the target to get a RST containing IP ID (say it's 31337)
Step 2) Spoof the IP address of your zombie machine, and send a SYN packet to the target (port 80)
- If the port is open, the target will send a SYNACK to the zombie and it will shoot a RST back to the target. (IP ID predicted is 31338)
- If the port is closed, the target will send a RST to the zombie host, and it won't respond
Step 3) Probe the target again with another SYNACK. The IP ID should have incremented by 2 from the last RST obtained in step 1 if the tested port on the target is open.









nmap -Pn -p- -sI www.host123.com www.host345.com
Idlescan using zombie www.host123.com (192.130.18.124:80); Class: Incremental
Nmap scan report for 198.182.30.30.110
(the 40321 ports scanned but not shown below are in the state: closed)

Port	State	Service
21/tcp	open	ftp
25/tcp	open	smtp
80/tcp	open	http

Nmap done: 1 IP address (1 host up) scanned in 1931.23 seconds

ICMP Echo scanning/List scan
ICMP echo scanning is used to discover live machines by pinging all the machines in the target network. ICMP probes sent to the broadcast or network address are relayed to all the host addresses in the subnet. The live systems will send ICMP echo reply message to the source of ICMP echo probe.
nmap -P 192.168.0.0/24 --OR-- nmap -sn 192.168.0.2 --OR-- nmap -sL -v 192.168.2.5

UDP Scanning -sUV
Sends UDP packets to each target port, often empty but some commonly known port services need a specific payload, so Nmap tries add it in (the -V option checks Nmap's database). Some listening applications will still discard empty packets as invalid, firewalls also drop packets without responding, and UDP might just drop packets or time out. So, a UDP response means it is open, no response can be either open or filtered (can't tell!). Nmap running with application version detection (-sUV) helps there a bit.
If you get a UDP response back, it's open
If the port is closed, it will likely shoot back an ICMP port unreachable (type 3/code 3)
If it is another ICMP unreachable error (type 3, code 1, 2, 9, 10, or 13) it's likely filtered.

Some OSs limit the frequency of ICMP Port Unreachable messages, Nmap adjusts accordingly. Microsoft-based OSs do not usually implement any type of ICMP rate limiting. Not usually useful for most types of attack, but it can reveal information about some exploitable services (DHCP, SNMP, NFS, etc). Malware often uses UDP
nmap -sU -v 192.168.0.2

Some flags available in Nmap (see https://nmap.org/book/man-briefoptions.html)

-sT	TCP connect scan	-sR	RPC scan	-oN	Normal output
-sS	SYN scan	-sL	List/DNS scan	-oX	XML output
-sF	FIN scan	-sI	Idle scan	-oG	Greppable output
-sX	XMAS tree scan	-Po	Don't ping	-oA	All output
-sN	Null scan	-PT	TCP ping	-T Paranoid	Serial scan; 300 sec between scans
-sP	Ping scan	-PS	SYN ping	-T Sneaky	Serial scan; 15 sec between scans
-sU	UDP scan	-PI	ICMP ping	-T Polite	Serial scan; .4 sec between scans
-sO	Protocol scan	-PB	TCP and ICMP ping	-T Normal	Parallel scan
-sA	ACK scan	-PB	ICMP timestamp	-T Aggressive	Parallel, 300 sec timeout, 1.25 sec/probe
-sW	Windows scan	-PM	ICMP netmask	-T Insane	Parallel, 75 sec timeout, 0.3 sec/probe

IP PROTOCOL SCAN
Looks for supported IP protocols rather than open ports; sends raw IP packets with different values in the protocol field of the header. Instead of looking for “ICMP Port Unreachable”, it looks for ICMP Protocol Unreachables to tell if it's unsupported (somewhat “closed”). If we get a response back in the same protocol, it's supported (somewhat “open”). If we get some different ICMP unreachable, it's probably filtered. If we don't get anything back, it's either open (and didn't reply) or filtered.
nmap -sO 192.168.10.1
hping3 -c 1 --rawip --ipproto 0 192.168.10.1
hping3 -c 1 --icmp 192.168.10.1
hping3 -c 1 --rawip --ipproto 2 192.168.10.1

FTP bounce scanning
Hosts running outdated FTP services can relay numerous TCP attacks, including port scanning. There is a flaw in the way many FTP servers handle connections using the PORT command (see RFC 959 or technical description) that allows data to be sent to user-specified hosts and ports. In their default configurations, the FTP services running on the following older Unix-based platforms are affected
The FTP bounce attack can have a far more devastating effect if a writable directory exists because a series of commands or other data can be entered into a file and then relayed via the PORT command to a specified port of a target host. For example, some- one can upload a spam email message to a vulnerable FTP server and then send this email message to the SMTP port of a target mail server. Figure 4-9 shows the parties involved in FTP bounce scanning.

1. The attacker connects to the FTP control port (TCP port 21) of the vulnerable FTP server that she is going to bounce her attack through and enters passive mode, forcing the FTP server to send data to a specific port of a specific host:
QUOTE PASV
    227 Entering Passive Mode (64,12,168,246,56,185).
2. A PORT command is issued, with an argument passed to the FTP service telling it
to attempt a connection to a specific TCP port on the target server; for example, TCP port 23 of 144.51.17.230:
    PORT 144,51,17,230,0,23
    200 PORT command successful.
3. After issuing the PORT command, a LIST command is sent. The FTP server then
attempts to create a connection with the target host defined in the PORT command issued previously:
LIST
    150 Opening ASCII mode data connection for file list
    226 Transfer complete.
If a 226 response is seen, then the port on the target host is open. If, however, a 425 response is seen, the connection has been refused:
LIST
    425 Can't build data connection: Connection refused
Nmap supports FTP bounce port scanning with the –P0 and –b flags used in the following manner:
  nmap –P0 –b username:password@ftp-server:port <target host>
The –P0 flag must be used to suppress pinging of the target host, as it may not be accessible from your location (e.g., if you are bouncing through a multihomed FTP server). Also, you may not want your source IP address to appear in logs at the target site.

Proxy bounce scanning
Attackers bounce TCP attacks through open proxy servers. Depending on the level of poor configuration, the server will sometimes allow a full-blown TCP port scan to be relayed. Using proxy servers to perform bounce port scanning in this fashion is often time-consuming, so many attackers prefer to abuse open proxy servers more efficiently by bouncing actual attacks through to target networks.
ppscan.c, a publicly available Unix-based tool to bounce port scans, can be found in source form at:
http://examples.oreilly.com/networksa/tools/ppscan.c http://www.phreak.org/archives/exploits/unix/network-scanners/ppscan.c

---

(NMAP) When scanning hardened environments, you should use the -Pn flag to force scanning of each address within scope. A slower timing policy (such as -T2) is also useful, as an aggressive policy will trigger SYN flood protection by firewalls

SYN probes - four response variants: a packet with SYN/ACK flags indicating an open port, RST/ACK denoting closed, no response or an ICMP type 3 message implying a filter

This runs three Nmap scans to identify accessible hosts across TCP, SCTP, and UDP. Optionally, load a list of targets into Nmap from a file using the -iL flag.
nmap –T4 –Pn –v –n –sS –F –oG /tmp/tcp.gnmap 192.168.0.0/24
nmap –T4 –Pn –v –n –sY –F –oG /tmp/sctp.gnmap 192.168.0.0/24
nmap –T4 –Pn –v –n –sU –p53,111,123,137,161,500 -oG /tmp/udp.gnmap 192.168.0.0/24
These scans generate output in /tmp with gnmap file extensions. UDP results may contain false positives. If the UDP dataset looks noisy (i.e. all the hosts are reporting to have open ports), then simply disregard it. Once you're happy with the contents of these files, use grep and awk to generate a refined list of targets, as follows:
grep open /tmp/*.gnmap | awk '{print $2}' | sort | uniq > /tmp/targets.txt
This list should then be fed into four subsequent scans: A fast TCP scan of common services
nmap -T4 -Pn -v --open -sS -A -oA tcp_fast -iL /tmp/targets.txt
A TCP scan of all ports:
nmap -T4 -Pn -v --open -sS -A –p0-65535 -oA tcp_full -iL /tmp/targets.txt
An SCTP scan of all ports:
nmap -T4 -Pn -v --open -sY –p0-65535 -oA sctp -iL /tmp/targets.txt
A UDP scan of common services:
nmap -T3 -Pn -v --open –sU -oA udp -iL /tmp/targets.txt

The -oA flag will generate multiple output files for each scan type, including a gnmap file that you can easily parse, and a full text file (i.e. tcp_fast.nmap) that is human- readable. These scanning modes do not perform service fingerprinting or deep analysis of the exposed network services

Same procedure for IPv6 - first sweeping IPv6 address space for hosts running common network services, and then perform full scanning of that subset). When TCP sweeping large IPv6 networks, I recommend reducing the port list to increase speed, from -F (100 common ports) to -p22,25,53,80,111,139,443.
Upon preparing a list of targets (e.g. /tmp/targets.txt) from host discovery and sweeping, run the same four scans as before, using the -6 flag to perform the scanning over IPv6:
nmap -6 -T4 -Pn -v --open -sS -A -oA ipv6_tcp_fast -iL /tmp/targets.txt
nmap -6 -T4 -Pn -v --open -sS -A –p0-65535 -oA ipv6_tcp_full -iL /tmp/targets.txt
nmap -6 -T4 -Pn -v --open -sY –p0-65535 -oA ipv6_sctp -iL /tmp/targets.txt
nmap -6 -T3 -Pn -v --open -sU -oA ipv6_udp -iL /tmp/targets.txt

---

SCTP
Stream Control Transmission Protocol (SCTP) is a transport protocol that sits alongside TCP and UDP. Intended to provide transport of telephony data over IP, the protocol duplicates many of the reliability features of SS7, and underpins a larger family of protocols known as SIGTRAN. SCTP is supported by operating systems including IBM AIX, Oracle Solaris, HP-UX, Linux, Cisco IOS, and VxWorks.

SCTP chunk types

ID	*Value*	*Description*
0	DATA	Payload data
1	INIT	Initiation
2	INIT ACK	Initiation acknowledgement
3	SACK	Selective acknowledgement
4	HEARTBEAT	Heartbeat request
5	HEARTBEAT ACK	Heartbeat acknowledgement
6	ABORT	Abort
7	SHUTDOWN	Shutdown
8	SHUTDOWN ACK	Shutdown acknowledgement
9	ERROR	Operation error
10	COOKIE ECHO	State cookie
11	COOKIE ACK	Cookie acknowledgement
12	ECNE	Explicit congestion notification echo
13	CWR	Congestion window reduced
14	SHUTDOWN COMPLETE	Shutdown complete

Tools such as Nmap and SING don't identify these responses from private addresses (behind NAT), as low- level stateful analysis of the traffic flowing into and out of a network is required. A quick and simple example of this behavior can be seen in the ISS BlackICE personal firewall event log in Figure 4-1 as a simple ICMP ping sweep is performed.

It is beneficial to run a network sniffer such as Ethereal or tcpdump during testing to pick up on unsolicited ICMP responses, including “ICMP TTL exceeded” (type 11 code 0) messages, indicating a routing loop, and “ICMP administratively prohibited” (type 3 code 13) messages, indicating an ACL in use on a router or firewall.

OS Fingerprinting Using ICMP
Ofir Arkin’s Xprobe2 utility performs OS fingerprinting primarily by analyzing responses to ICMP probes.

Another SYN port scanner worth mentioning is Scanrand, a component of the Paketto Keiretsu suite. Paketto Keiretsu contains a number of useful networking utilities that are available at http://www.doxpara.com/read.php/code/paketto.html. For Windows, Foundstone’s SuperScan is an excellent port scanning utility with good functionality, including banner grabbing. SuperScan is available from http:// examples.oreilly.com/networksa/tools/superscan4.zip
---
Scanrand is well designed, with distinct SYN probing and background listening components that allow for very fast scanning. Inverse SYN cookies (using SHA1) tag out- going probe packets, so that false positive results become nonexistent, as the listening component only registers responses with the correct SYN cookies. Scanrand is much faster than bulkier scanners, such as Nmap.

Unicornscan (http://www.unicornscan.org) is another tool that performs fast half- open scanning. It has some unique and very useful features, and it is recommended for advanced users.
(UDP payload scan) against the 10.3.0.1 candidate within my environment, results are as follows:

root@kali:~# unicornscan -mU 10.3.0.1
UDP open domain[53] from 10.3.0.1 ttl 128
UDP open netbios-ns[137] from 10.3.0.1 ttl 128

UDP scanning results across tools may vary. Nmap provides a comprehensive option with -sV, but testing of a single host using the -F option (scanning 100 ports) takes around 10 minutes to complete.

Using malformed TCP flags to probe a target is known as an inverted technique because responses are sent back only by closed ports. RFC 793 states that if a port is closed on a host, an RST/ACK packet should be sent to reset the connection. To take advantage of this feature, attackers send TCP probe packets with various TCP flags set.

Vscan is another Windows tool you can use to perform inverse TCP flag scanning. The utility doesn’t require installation of WinPcap network drivers; instead it uses raw sockets within Winsock 2 (present in Windows itself). Vscan is available from http://examples.oreilly.com/networksa/tools/vscan.zip.

SING (Send ICMP Nasty Garbage) - http://sourceforge.net/projects/sing
Ability to transmit and receive spoofed packets, send MAC-spoofed packets, and support the transmission of many other message types, including ICMP address mask, timestamp, and information requests, as well as router solicitation and router advertisement messages

ICMPScan - http://www.bindshell.net/tools/icmpscan
Bulk scanner derived from Nmap that sends type 8, 13, 15, and 17 ICMP; can process inbound responses by placing the network interface into promiscuous mode, thereby identifying internal IP addresses and machines that respond from probes sent to subnet network and broadcast addresses. Because ICMP is a connectionless protocol, it is best practice to resend each probe (using –r 1) and set the timeout to 500 milliseconds (using -t 500). We also set the tool to listen in promiscuous mode for unsolicited responses (using the –c flag).

SSH Tunnels

Shortened version of what's at https://robotmoon.com/ssh-tunnels/

Port forwarding - Forwards a port from one system (local or remote) to another

Local port forwarding- Forwards connections from a port on a local system to a port on a remote host.
To forward traffic on the SSH client to some destination through an SSH server. This lets you access remote services over an encrypted connection as if they were local services.

Example use cases:

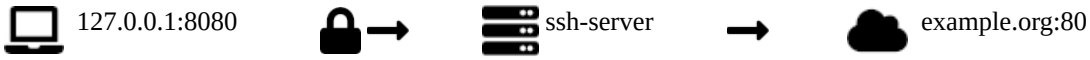
- Accessing a remote service (redis, memcached, etc.) listening on internal IPs
- Locally accessing resources available on a private network
- Transparently proxying a request to a remote service

If you want to use a secure connection to access a remote service that communicates over plaintext. For example, redis and memcached all use plaintext protocols. If you securely access one of these services on a remote server over public networks, you can tunnel a connection from your local system to the remote server instead of having it listen over the public internet.

SSH -L is FROM-INSIDE-IP:port:TO-OUTSIDE-ACCESS-IP:port SSHServerName

ssh -L 127.0.0.1:8080:example.org:80 ssh-server

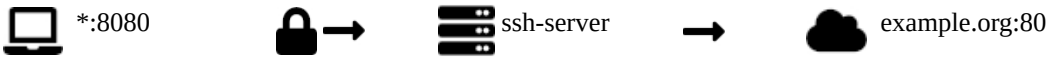
Forward connections to 127.0.0.1:8080 on your local system to port 80 on example.org through ssh-server.



The traffic between your local system and ssh-server is wrapped in an SSH tunnel, but the traffic between ssh-server and example.org is not. From the perspective of example.org the traffic originates from ssh-server.

ssh -L 8080:example.org:80 ssh-server
ssh -L *:8080:example.org:80 ssh-server

Forward connections to port 8080 on all interfaces on your local system to example.org:80 through a tunnel to ssh-server.



ssh -L 192.168.0.1:5432:127.0.0.1:5432 ssh-server

Forward connections to 192.168.0.1:5432 on your local system to 127.0.0.1:5432 on ssh-server. Note that 127.0.0.1 here is localhost from the viewpoint of ssh-server.

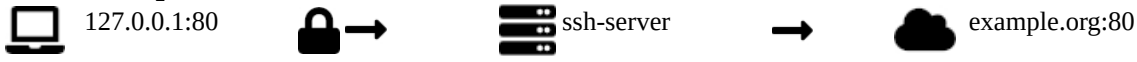
SSH configurations. Make sure:

TCP forwarding is enabled on the SSH server. In /etc/ssh/sshd_config set "AllowTcpForwarding yes"
If you're forwarding ports on interfaces other than 127.0.0.1 then you'll need to enable GatewayPorts on your local system, either within ssh_config or as a command-line option
In /etc/ssh/ssh_config set "GatewayPorts yes"

Forwarding from privileged ports

If you want to open a privileged port (ports 1-1023) to forward traffic, you'll need to run SSH with superuser privileges on the system that opens the port.

sudo ssh -L 80:example.com:80 ssh-server



Remote port forwarding - Forwards a port on a remote system to another system

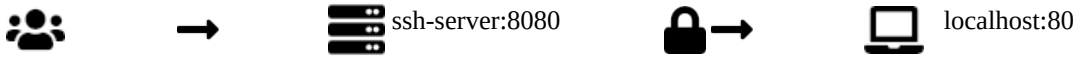
To forward traffic on an SSH server to a destination through either the SSH client or another remote host. This gives users on public networks access to resources on private networks. Example use cases:

Making a local development server available over a public network
Granting IP-restricted access to a remote resource on a private network

SSH -R is OUTSIDE-ACCESS-IP:port:TO-INSIDE-IP:port SSHServerName

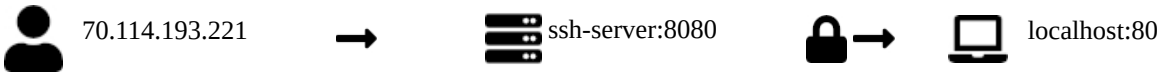
ssh -R 8080:localhost:80 ssh-server

Forwards traffic to all interfaces on port 8080 on ssh-server to localhost port 80 on your local computer. If one of these interfaces is available to the public internet, traffic connecting to port 8080 will be forwarded to your local system.



ssh -R 70.114.193.221:8080:localhost:80 ssh-server

Forwards traffic to ssh-server:8080 to localhost:80 on your local system but only allows access to the SSH tunnel entrance on ssh-server from IP address 70.114.193.221. This requires the GatewayPorts clientspecified option in the server's sshd_config.



ssh -R 8080:example.org:80 ssh-server

Forwards traffic to all interfaces on ssh-server:8080 to localhost:80 on your local system. From your local system, traffic is then forwarded to example.org:80. From the perspective of example.org the traffic is originating from your local system.

SSH server configurations set in /etc/ssh/sshd_config

By default, forwarded ports are not accessible to the public internet. You'll need to add this to your sshd_config on your remote server to forward public internet traffic to your local computer.

GatewayPorts yes

If you want only *specific* clients to be allowed access, you can use this instead:

GatewayPorts clientspecified

Dynamic port forwarding- To forward traffic from a range of ports to a remote server

Opens a SOCKS proxy on the SSH client that lets you forward TCP traffic through the SSH server to a remote host.

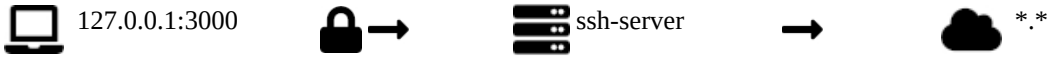
ssh -D 3000 ssh-server

Opens a SOCKS proxy on port 3000 of all interfaces on your local system. This allows you to forward traffic sent through the proxy to the ssh-server on any port or destination host. By default, SSH will use the SOCKS5 protocol, which forwards TCP and UDP traffic.

ssh -D 127.0.0.1:3000 ssh-server

Opens a SOCKS proxy on 127.0.0.1:3000 on your local system. When you have a SOCKS proxy running, you can configure your web browser to use the proxy to access resources as if connections were originating from ssh-server. For example, if ssh-server had access to other servers within a private network, you could access those other servers locally as if you were on the network, without a VPN. Test the proxy like this:

curl -x socks5://127.0.0.1:12345 https://example.org



SSH client configuration - /etc/ssh/ssh_config

If you want the SOCKS proxy to be available to more interfaces than just localhost, this needs to be set: GatewayPorts yes

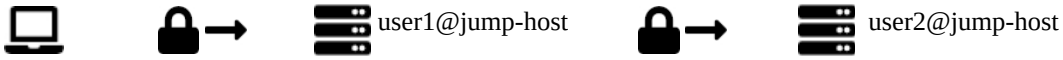
When configuing the SSH client, you can also configure it with a command instead of editing manually:

ssh -o GatewayPorts=yes -D 3000 ssh-server

Jump hosts and proxy commands - Transparently connecting to a remote host through intermediate hosts

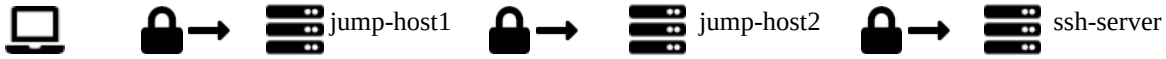
ssh -J user1@jump-host user2@remote-host
ssh -o "ProxyJump user1@jump-host" user2@remote-host

Establishes a SSH connection with jump-host and forwards TCP traffic to remote-host, connecting to remote-host. The -J command should work out of the box if jump-host already has SSH access to remote-host. If it does not, you can use agent forwarding to forward the SSH identity of your local computer to remote-host.



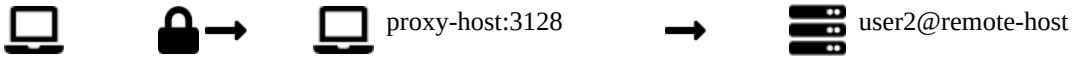
ssh -J jump-host1,jump-host2 ssh-server

To specify multiple comma-separated jump hosts.



ssh -o ProxyCommand="nc -X 5 -x localhost:3000 %h %p" user@remote-host

Connecting to a remote server through a SOCKS5 proxy using netcat. From the perspective of the server, the originating IP is from proxy-host. However, the SSH connection itself is end-to-end encrypted so proxy-host only sees an encrypted stream of traffic between the local system and remote-host.



SSH client configuration

To enable agent forwarding, you can use ssh-add to add your local SSH identity to your local ssh agent.
The command ssh-add will simply add the currently logged-in user, while adding configuration to /etc/ssh/ssh_config will be for system-wide SSH client configuration settings

Reliable SSH Tunnels - Keeping SSH tunnels open through network failures

The commands listed above work on an ad-hoc basis, but if you want to maintain SSH tunnels through network outages or unreliable connections, you'll have to do some additional setup. Another issue is, by default, the TCP connection used to establish an SSH tunnel may time out after a period of inactivity. To prevent timeouts, you can configure the server and/or the client to send heartbeat messages by adding this into /etc/ssh/sshd_config:

On the server, add:

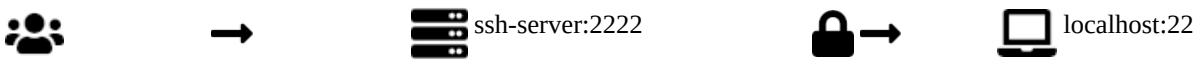
```
ClientAliveInterval 15
ClientAliveCountMax 4
```

On the client add:

```
ServerAliveInterval 15
ServerAliveCountMax 4
```

Using AutoSSH to restqablish dropped connections

While the above options may prevent a connection from dropping due to inactivity, they will not re-establish dropped connections. To ensure that an SSH tunnel will be re-established, you can use autossh, which builds an SSH tunnel and monitors its health.
AutoSSH accepts the same arguments for port forwarding as SSH.



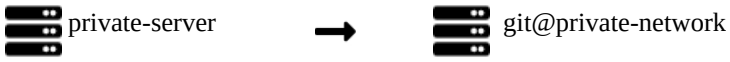
This establishes a reverse tunnel that comes back after network failures. By default, AutoSSH will open extra ports on the SSH client and server for health checks. If traffic appears to no longer pass between the health check ports, AutoSSH will restart the SSH tunnel.

autossh -R 2222:localhost:22 -M 0 -o "ServerAliveInterval 10" -o "ServerAliveCountMax 3" remote-host

Using the -M 0 flag disables the health check ports and allows the SSH client to handle the health checks. In this example, the SSH client expects the server to send a heartbeat every 10 seconds. If 3 heartbeats fail in a row, the SSH client exits, and AutoSSH will re-establish a new connection.

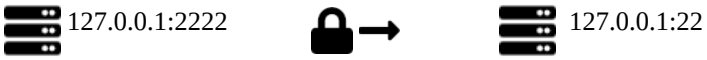
Transparent access to remote resource on a private network

Let's say there's a git repository on a private network that's only accessible through a private server on the network.
This server is not accessible to the public internet. You have direct access to the server, but don't have VPN access to the private network.



For convenience, you'd like to access this private git repository as if you were connecting to it directly from your local system. If you have SSH access to another server that's accessible from both your local system and the private server, you can accomplish this by establishing an SSH tunnel and using a couple of ProxyCommand directives.

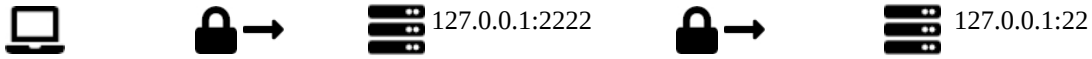
ssh -L 127.0.0.1:22:127.0.0.1:2222 intermediate-host
This forwards port 2222 on intermediate-host to port 22 on the private server.



Now, if you SSH to port 2222 from intermediate-host, you're connecting to the SSH server on the private server despite the private server not being accessible by the public internet.

ssh -p 2222 user@localhost

Now you can now access the private git repository as if you were on the private network.



If you'd like to make the backdoor even more convenient, you can add some directives to your local ~/.ssh/config:

```
Host git.private.network
    HostName git.private.network
    ForwardAgent yes
    ProxyCommand ssh private nc %h %p

Host private
    HostName localhost
    Port 2222
    User private-user
    ForwardAgent yes
    ProxyCommand ssh tunnel@intermediate-host nc %h %p
```

SSH command-line flags

These are some useful SSH command-line flags when establishing tunnels. For simplicity, the examples given here leave these out.

-f	Forks the ssh process into the background
-n	Prevents reading from STDIN
-N	Do not run remote commands. Used when only forwarding ports
-T	Disables TTY allocation

Here's an example of a command you would run to create an SSH tunnel in the background that forwards a local port through the ssh server:

ssh -fnNT -L 127.0.0.1:8080:example.org:80 ssh-server

In some misconfigured systems supporting SSH you may not be able to gain a shell via ssh, but you can port forward and such. There is an article discussing that here:
https://www.pixelstech.net/article/1328532389-SSH-Security-and-You---bin-false-is-%2Anot%2A-security

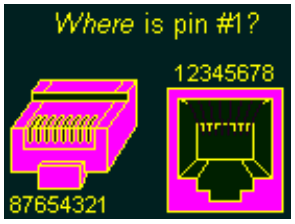
Socat to Netcat Commands Task Quick Reference

NOTE: In BSD/ MacOS versions of Netcat, using -l and -p in the same directive together throws an error. Required is "nc -l <port>..." the -l option immediately followed by port number. In that case -lABCD are acceptable as long as any of ABCD dont also require a value follow them, example: "nc -lks <port>" won't be acceptable, but "nd -lk <port> -s <device>" will.

Connections/ Data Transfer:	**Socat example**	**Netcat example**
Open TCP connections:	socat tcp-listen:<port>,fork tcp:<host>:<port>	nc -l <port> -c "nc <host> <port>" (protocol=<protocol> supported)
Open UDP connections:	socat udp-listen:<port>,fork udp:<host>:<port>	nc -u -l <port> -c "nc -u <host> <port>" (protocol=<protocol> supported)
Connect to remote systems:	socat tcp:<host>:<port>,fork (connect to specific port)	nc <host> <port> # connect to specific port)
Connect to serial devices:	socat serial:<device>,raw,echo=1 tcp:<host>:<port>	nc -l -p <port> <device> # Serial to TCP
Send/ write files:	socat file:<file>,fork tcp:<host>:<port>	nc <host> <port> < <file>
Receive/ read files:	socat tcp-listen:<port>,fork write:<file>	nc -l <port> > <file>
Tunneling and Proxying:	***Using SSH for tunnels is a better option than nc and socat***	***See https://robotmoon.com/ssh-tunnels/***
Create basic tunnels:	socat tcp-listen:<port>,fork tcp:<remote_host>:<remote_port>user=user,password=pass	nc -l <port> -c "nc <remote_host> <remote_port>"
Establish nested tunnels:	socat tcp-listen:<port1>,fork tcp:<host1>:<port2>,fork tcp:<remote_host>:<remote_port>	nc -l -p <port1> | nc <host1> <port2> & nc <remote_host> <remote_port>
SSL/TLS tunnels:	socat openssl:<cipher>,fork tcp:<host>:<port>	nc -l <port> | openssl enc -d des3 # enlist SSL
SOCKS tunnel (firewall traversal):	socat tcp:<host>:<port>,proxy:socks:<proxy_server>: <proxy_port>,fork tcp:<remote_host>:<remote_port>	proxychains nc -l <host> <port> | nc -x <proxy_server>: <proxy_port> <remote_host> <remote_port>
PKI authentication:	socat tcp-listen:<port>,fork pty,ptyexec:/bin/bash,user= remote_user, keyfile=/path/to/private_key	ssh -i /path/to/private_key user@host "nc -l -p <port>"
SSH tunnel:	socat tcp-listen:<port>,fork exec:ssh -N -f -L <port>:localhost:<remote_port> <user>@<remote_host>	ssh -N -f -L <port>:localhost:<remote_port> <user>@<remote_host> & nc -l -p <port>
Proxy TCP connections:	socat tcp-listen:<port1>,fork tcp:<host1>:<port2>	nc -l -p <port1> | nc <host1> <port2> *(socat is better for this)*
Basic IP forwarding:	socat tcp-listen:<port1>,fork TCP:<host2>:<port2>	*nc -l <port1> | nc <host2> <port2>* *(socat is better for this)*
Port forwarding:	socat tcp-listen:<port1>,fork tcp:<host1>:<port2>	nc -l -p <port1> | nc <host1> <port2> *(socat is better for this)*
UDP hole punching:	socat udp-listen:<port>,fork udp:<host>:<port>	nc -u -l <port1> -c "nc -u <host2> <port2>
Multiplexing connections:	socat tcp-listen:<port>,fork tcp:<host1>:<port1>,fork tcp:<host2>:<port2> *-In a script you could add error checks, logging, etc.*	*Netcat isn't really designed to do this but you could use it in a script to accomplish it, since it gets out of scope for a 'one-liner)*
Other Networking:		
Serial communication:	socat serial:<device>,raw,echo=1 tcp:<host>:<port>	nc -l -p <port> <device> # TCP to serial
Unix domain sockets:	socat unix-listen:<path>,fork unix-connect:<path>	
Execute remote commands:	socat tcp:<host>:<port>,fork exec:<command>	nc -l -p <port> | bash -i ***SSH is preferable- both socat and netcat are too insecure***
Traffic shaping and bandwidth limiting:	socat tcp:<host>:<port>,fork rate:<bandwidth>,link:<delay> (<bandwidth> is desired rate and <delay> is desired latency)	*Doing this with nc would employ tc, which can do this by itself, so it's redundant)*
Load balancing and failover:	socat tcp-listen:<port>,fork tcp:<db1>:<port>,fork tcp:<db2>:<port>,roundrobin (distribute connections evenly)	nc <haproxy_host> <haproxy_port> *(inefficient since you still have to configure haproxy server to do the actual task)*
DNS resolution:	socat tcp-listen:<port>,fork tcp:dns:<dns_server>:53	*This would employ dig to use a custom DNS which makes nc irrelevant/ redundant*
Capture network traffic:	socat tcp-listen:<port>,fork tcp:<host>:<port>,fork write:<filename>	nc -l <port> > traffic.log
Inspect and modify data:	socat tcp-listen:<port>,fork tcp:<host>:<port>,fork exec:<commands>	nc -l <port> | <commands>
Port scanning:	*Not efficient. resort to nc or - better yet- just use nmap*	nc -z <host><start_port>-<end_port> *(just use nmap??)*

Some of these examples might show that you CAN use socat or netcat for these things, but that doesn't mean that's a GOOD way to do things. SSH is a good example, but so are nmap, etc. But, if you just need to mess with something like a directly connected device with no security concerns, nc and socat might be fine for tunnelling or remote commands instead of SSH!

Category 5/ 6, UTP (Unshielded Twisted Pair); RJ-45 8P8C (8 position, 8 contact) connector
Twisting of the 2 wires in each of the 4 pairs minimizes interference (one is insulated by the other).
Each pair also even have a different "twist rate" to minimize interference between them. Each pair is transmit +/-, receive +/-
Shielded twisted pair surrounds each pair in metal shielding and uses a grounding wire.



The only difference between T568A and T568B wiring standards is that pairs 2 and 3 (orange and green) are swapped. Both configurations wire the pins "straight through", i.e., pins 1 through 8 on one end are connected to pins 1 through 8 on the other end.

As you can see below, a crossover cable for both T568A and T568B is ALMOST T568A wiring on one end and T568B on the other- making it easy to remember- HOWEVER- notice that pins 7 and 8 also switch to pins 4 and 5 (below, blue and brown switch places, as well as green and orange).
So crossover is 1 and 2 to 3 and 6, and 4 and 5 to 7 and 8

Straight-Through Ethernet Cable Pin Out for T568B

RJ45 Pin #	Wire Color	10/100Base-TX Signal	1000Base-T Signal
1	White/Orange	Transmit+	BI_DA+
2	Orange	Transmit-	BI_DA-
3	White/Green	Receive+	BI_DB+
4	Blue	Unused	BI_DC+
5	White/Blue	Unused	BI_DC-
6	Green	Receive-	BI_DB-
7	White/Brown	Unused	BI_DD+
8	Brown	Unused	BI_DD-

Straight-Through and PoE Ethernet Cable Pin Out for T568A

RJ45 Pin #	Wire Color	10/100Base-TX Signal	PoE
1	White/Green	Transmit+	Mode A +
2	Green	Transmit-	Mode A +
3	White/Orange	Receive+	Mode A -
4	Blue	Unused	Mode B +
5	White/Blue	Unused	Mode B +
6	Orange	Receive-	Mode A -
7	White/Brown	Unused	Mode B -
8	Brown	Unused	Mode B -

Crossover Cable Pin Outs for T568B

Pin # (END 1)	Wire Color	Pin # (END 2)	Wire Color
1	White/Orange	1	White/Green
2	Orange	2	Green
3	White/Green	3	White/Orange
4	Blue	4	White/Brown
5	White/Blue	5	Brown
6	Green	6	Orange
7	White/Brown	7	Blue
8	Brown	8	White/Blue

One other type of wiring is a rollover/ console cable for switch and routers using the EIA-TIA 232 serial COM port. It is just straight-through in reverse order- picture cutting the end off a straight-through cable, turn it over, and put on a new connector. Often USB to serial converter and rollover cables are used.

Crossover cables are necessary when endpoints transmit on the same pin pair, they need to be crossed for compatibility. Use crossover for "like devices" not unlike:
Transmits on Pins 1,2: workstation NICs, routers, cabled non-USB WAPs
Transmits on Pins 3,6: switches and hubs
Some devices have auto-MDIX to autosense cable type and make the link work. Often not present.

Category 5 cabling is meant for 100Base-TX, but was first used as 10Base-T (wired 2-pair instead of 4) for legacy equipment. Cat 5e is an updated version of Cat5, offers reduced crosstalk, ok for 1000BASE-T
Cat6 cable is *preferred* for 1000BASE-T Ethernet networks. Some Cat6 is made of thicker wires (for example, 22 or 23 gauge instead of 24); more pair twisting gives thicker insulation for reduced crosstalk.

Copper Ethernet Cabling Types

10BASE2 (thinnet)	RG-58 coax	10 Mbps	185 meters
10BASE5 (thicknet)	RG-8 coax	10 Mbps	500 meters
10Base-T	Cat3 (POTS) or Cat5	10 Mbps	100 meters
100Base-T (fast ethernet)	Cat5	100 Mbit/s	100MHz
1000Base-T (gigabit ethernet)	Cat5e	1Gbit/s	100MHz
10GBase-T	Cat6	1Gbit/s	250MHz
10GBase-T	Cat6a	10Gbit/s	500MHz, shielded
10GBase-T	Cat7 (not TIA/EIA ratified)	10Gbit/s	600MHz, shielded

Cat 5/6/6a have different speed performance at different lengths and shorter is faster.
Cat5 can do 1000Base-T, but not guaranteed. Cat 5e/6 achieve maximum efficiency at 55 meters.
*When sizing copper ethernet cables remember that cable should not extend more than **100m (~328ft)**.*

Power over Ethernet (PoE)
Endspan means PoE is built into switch. Midspan means a Inline Power Injector is used to add power
Mode A (phantom): Power and data use same twisted pairs; Mode B: separate pairs for each power and data.
IEEE 802.3af (802.3at type1) specifies a max 15.4W, 802.3at (type 2) specifies a max of 30W (typically 25W)

Other Cabling Types and Specifications

All RG (Radio Guide) specs use a familiar CATV F-connector on coaxial cable. BASE means "baseband"
RG-6 coax is commonly used by local cable companies to connect individual homes to the distribution point. For higher frequency signals over longer distances (~70 meters). RG-6 replaced RG-9 with better shielding.
RG-59 low-freq, shorter distance, like component video; older shielding, susceptible to UHF interference, impedance of 75 Ohms. RG-6 was preferred for video.
RG-58 - early 10BASE2 max length 185 meters; impedance 50 Ohms. Better frequency range, shielding
RG-8- 10BASE5, 500 meters. RG-58 and -8 predicted to replace RG-6 and RG-59
[You may see RG-500 for heavy outdoor use (burial) - has a FAT core, 75 Ohms, super-long range]

Fiber Optics: *LX FX and S's can all do MMF - FX has full and half duplex (below)*

100Base-SX	100Mbps	200-550m	MMF (short wavelength laser)
100Base-FX	100Mbps	2km full duplex, 400m half-duplex	MMF
1000Base-LX	1000Mbps	10km SMF or 550m MMF	(both SMF and MMF)
1000Base-LH	1000Mbps	10km	SMF
1000Base-ZX	1000Mbps	70km	SMF
10GBase-SR and -SW	10 Gbps	300m (short reach)	MMF
10GBase-LW	10 Gbps	10km	SMF
10GBase-LR	10 Gbps	25km (long reach)	SMF
10GBase-ER and -EW	10 Gbps	40km (extended reach)	SMF
100GBASE-ER4	100 Gbps	40km	SMF
100GBASE-SR10	100 Gbps	100-125km	MMF

Advantages of fiber: difficult to monitor (tap), no RF interference, support long distances
Multimode (MMF) refers that different light frequencies bouncing along a fiber bundle; often uses LEDs (less expensive), so signal degrades at a shorter distance. The core is typically 62.5 microns or larger in diameter
Singlemode (SMF) uses a single fiber with single frequency of light; more expensive- lasers to make the distance and increase signal strength. Typically uses a small light carrying core of 8 to 10 microns in diameter.

Fiber Connectors:

The SC connector as called a subscriber, standard, or square connector.
The MTRJ goes by two names, the "Media Termination Recommended Jack" and "Mechanical Transfer Registered Jack". It has two fiber strands (that is, a transmit strand and a receive strand) included in a single connector. Despite the second nickname, it doesn't lock into place or anything.
The LC (Lucent/Local/Little Connector) connects to a terminating device by pushing the connector into the terminating device with a "click" and can be removed by depressing the tab on the connector to pull it out.
A Straight Tip (ST) connector is sometimes referred to as a bayonet connector; most commonly used with multimode; connects by pushing the connector in and then twisting the connector housing to lock it in place.
- Multimode Delay Distortion - on multi-mode fiber-optic cables when an initial transmission can arrive at the receiver after a second transmission
- Mode of Propagation: the path that light takes through a fiber-optic cable.
- A multiplexer combines a number of signals into a single signal for transmission over the medium.

Bulk Data Carrier Types

Bytes to bits in bandwidth conversions
"A megabit per second (abbreviated as Mbps, Mbit/s, or mbps) is a unit of data transfer rates equal to 1,000,000 bits per second (this equals 1,000 kilobits per second). Because there are 8 bits in a byte, a transfer speed of 8 megabits per second (8 Mbps) is equivalent to 1,000,000 bytes per second (approximately 976 KiB/s)"
1 Kbps = 1000/8=125 bytes; 1 Mbps = 1,000,000/8=125,000 bytes
Bytes are made up of eight bits, so one kilobyte equals eight kilobits
1KB per sec = 8 Kbps; 1MB per sec = 8 Mbps

T-carrier (T1- US/ Canada), and E-carrier (E1- Europe)
Transmission System Level 1 - For trunking, time-division multiplexing. The T-carrier system (T1 and T3) refers to copper transport corresponding to DS1 and DS3. DS0 supported twenty 2.4 kbit/s channels, ten 4.8 kbit/s channels, five 9.67 kbit/s channels, one 56 kbit/s channel, or one 64 kbit/s clear channel, which is where DSL and ISDN came in. The base DS0 represents a single voice call digitized at 8 kHz sample rate w/ 8-bit pulse-code modulation at 8000 samples/sec which comes out to 64kbit/s

T1/E1/DS1 - 24 channels (DS0's)- 64 kbit/s per channel 1.5Mbit/s total line rate, 8 bits for framing info. E1 - 32 channels 2.048 Mbit/s line rate
T3/E3/DS3 - 28 T1 circuits - 672 T1 channels - 44.736 Mbit/s. E3 = 16 E1 circuits, 512 channels, 33.368 Mbit/s
In T1/E1 more than one frame is sent at once. Two methods to grouping these frames together:
Super Frame (SF) combines 12 standard 193-bit frames into one. Extended Super Frame (ESF) 24 193-bit frames into one.

Data Capacity - The beer/soda can analogy:
Base DS0- single voice call digitized at 8 kHz sample rate w/ 8-bit PCM @ 8000 samples/sec = 64kbit/s

DS0	56/64Kbps	1 POTS line	Old modem		One can
DS1/T1	1.544 Mbps	1.536 Mbps			A case of beer (24 cans)
DS3/T3	44.736 Mbps	28 DS1's	672 DS0's		A pallet of beer with 28 cases
OC1	1 DS3/T3	***[End copper***	***and begin***	***SONET/SDH]***	A shrink-wrapped pallet
OC3	155.52 Mbps	3 DS3's	84 DS1's	2016 DS0's	A truck that can hold 3 pallets
OC12	622.08 Mbps	12 DS3's	336 DS1's	8064 DS0's	A railroad train car - 12 pallets
OC48	2488.32 Mbps	4 OC12's	48 DS3's	1344 DS1's	4 train cars x 12 pallets
OC192	9953.28 Mbps	16 OC12's	192 DS3's	5376 DS1's	16 train cars - 192,024 DS0's
OC768	39813.12 Mbps	64 OC12's	768 DS3's	21,504 DS1's	Train with 64 railroad cars

Rough calculations for downloading a 1GB file (including ISDN and DSL)

Connection	**Speed**	**(Y:D:H:M:S)**	**Difference**
56 K	56,000 bps	1:15:40:57	96% slower
128 K	128,000 bps	17:21:40	91% slower
256 K	256,000 bps	8:40:50	83% slower
512 K	512,000 bps	4:20:25	66% slower
768 K	768,000 bps	2:53:37	50% slower
T1, DS-1	1.544 Mbps	1:26:21	Baseline
T3, DS-3	44.736 Mbps	2:59	2,798% faster
OC-3	155.520 Mbps	51	9,973% faster
OC-12	622.080 Mbps	13	40,191% faster
OC-48	2.488 Gbps	3	161,040% faster
OC-192	10 Gbps	1	647,569% faster

SONET - Synchronous Transport Signals (STS)

Synchronous Optical networking (SONET) - ANSI - Synchronous Transport Signals (STS)

Synchronous Digital Hierarchy (SDH) - International equivalent - Synchronous Transport Modules (STM)

SONET	SDH	Bandwidth	Overhead
STS-1/ OC-1	STM-0	51.48Mbps	1.728Mbps
STS-3/ OC-3	STM-1	155.52Mbps	6.912Mbps
STS-12/ OC-12	STM-4	622.08Mbps	20.736Mbps
2.5G SONET/ STS-48/ OC-48	STM-16	2.488Gbps	82.944Mbps
5G SONET/ STS-96/ OC-96	STM-32	4.876Gbps	
10G SONET/ STS-192/ OC-192	STM-64	9.953Gbps	442.368 Mbps
STS-256/ OC-256	STM-128	13.271Gbps	
STS-768/ OC-768	STM-256	39.813Gbps	1.327104Gbps

ISDN - Integrated Services Digital Network - Delivered over T1/E1

ISDN has two levels of service:

BRI - Basic Rate Interface (2B+D):

2 64kbit/s bearer/ user (B) channels (throughput)

1 16kbit/s signaling/delta (D) channel (connection maintenance)

PRI - Primary Rate Interface - hooked up directly to the telco central office

- 23B + 1D on a T1 (1.544 Mbps)
- 30B + 1D + sync/alarm channel on an E1 (2.048 Mbps)
- Commonly to deliver Public Switched Telephone Network (PSTN) to digital PBX (23 lines in one)
- Fewer active B channels can be used for a fractional T1.
- Can be used flexibly and reassigned when necessary (such as video conferencing)
- More channels can be used with more T1s, within certain design limits (PRI pairing)
- PRI pairing uses NFAS (non-facility associated signalling) to accommodate itself

ISDN Terminology:

An R reference point resides between a non-ISDN device to a terminal adapter.

An S/T reference point resides between a NT1 and a terminal endpoint 1 (TE1).

A TA (terminal adapter) performs conversion between a non-ISDN device and a TE1 device.

A U reference point resides between a NT1 and the wall jack connecting back to an ISDN service provider.

A NT1 (network termination 1) device interconnects a four-wire ISDN circuit and a two-wire ISDN circuit.