Linux Server Administration

- Linux Server Administration
  - History of linux
  - What is linux
  - Linux boot process
  - The Linux Kernel
    - Kernel Location
    - Kernel Modules
  - The linux filesystem
    - Filesystem hierarchy
    - Filesystem Types
    - Anatomy of a Disk
  - Installation linux server
  - Configuring linux server
    - system update
    - configure networking
      - Netplan
      - ifconfig
      - ip
      - route
      - DNS
    - Core Linux Commands
    - Configure SSH
    - configure firewall
    - Manage User Accounts and Permissions
    - Install and Manage Software Packages
    - Linux Processes
    - Server Monitoring and Management
  - Managing a Linux Server
    - How to Log into a Remote Linux Server
    - How to Transfer Files to/from a Linux Server
    - How to Schedule Tasks on Linux Server
    - How to Manage Terminal Sessions in Linux
  - Troubleshooting Linux Networking
    - ICMP
    - PING
    - Traceroute
    - Netstat
    - Packet Analysis
    - Logging in Linux
  - Quiz
  - Day 1: Demo
  - Day 1: Exercise
    - Day 1: Exercise Targets

- Day 2: Demo
- Day 2: Exercise
    - Day 2: Exercise Targets

History of linux

- To learn about how Linux came to be, let’s go back to the beginning to 1969 where Ken Thompson and Dennis Ritchie of Bell Laboratories developed the UNIX operating system.
- A decade or so later, Richard Stallman started working on the GNU (GNU is Not UNIX) project. The GNU General Public License (GPL), a free software license, was also created as a result of this.
- During this time other efforts such as BSD, MINIX, etc were developed to be UNIX like-systems.
- However, one thing that all these UNIX like-systems had in common was the lack of a unified kernel.

> The kernel is the most important piece in the operating system. It allows the hardware to talk to the software.

- Then in 1991, a young fellow named Linus Torvalds started developing what we now know today as the Linux kernel.

What is linux

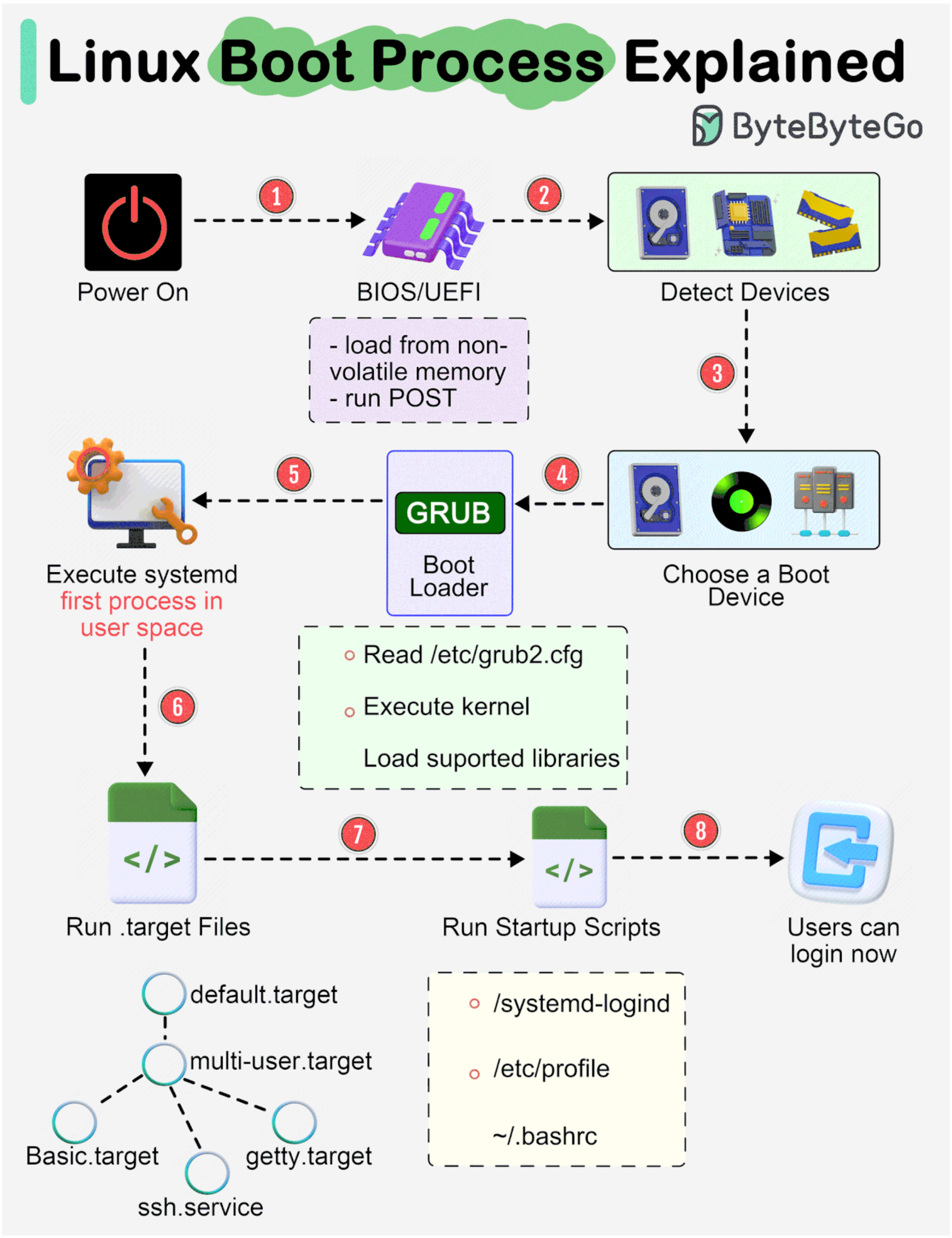
- the term Linux is actually quite a misnomer, since it actually refers to the Linux kernel.
- However, many distributions use the Linux kernel so therefore are commonly known as Linux operating systems.
- A Linux system is divided into three main parts:
    - Hardware - This includes all the hardware that your system runs on as well as memory, CPU, disks, etc.
    - Linux Kernel - As we discussed above, the kernel is the core of the operating system. It manages the hardware and tells it how to interact with the system.
    - User Space - This is where users like yourself will be directly interacting with the system.
- There are many Linux distributions to choose from, we’ll just go over the most popular options.

Distribution	Base	Package Manager	Use Case	Best For	Notable Features
Ubuntu	Debian	APT (.deb)	General-purpose computing, cloud, servers.	Beginners, system administrators.	LTS releases; strong community support; user-friendly.
Linux Mint	Ubuntu/Debian	APT (.deb)	Desktop operating system.	New Linux users, multimedia/home users.	Out-of-the-box media support; Cinnamon desktop.
Debian	Independent	APT (.deb)	Base for other distros, server workloads.	Experienced users, minimal setups.	Stability; huge repository; multi-arch.
Fedora	Independent (RHEL upstream)	DNF (.rpm)	Developer platforms, testing new tech.	Developers, security-conscious setups.	Advanced packages; SELinux; upstream to RHEL.
Arch Linux	Independent	pacman	Custom, rolling-release systems.	Advanced users who prefer manual control.	User-controlled setup; excellent documentation.
Rocky Linux	RHEL	DNF (.rpm)	Enterprise servers, CentOS replacement.	Organizations needing long-term stability.	RHEL-compatible; actively maintained; community-driven.
Alpine Linux	Independent	apk	Container-based and minimalist environments.	Advanced users, Docker users, embedded systems.	Extremely lightweight, security-focused, musl libc.
AlmaLinux	RHEL	DNF (.rpm)	Server deployments, enterprise workloads.	Cloud/server admins replacing CentOS.	RHEL-compatible; free and open-source; stable releases.
openSUSE Leap	SUSE Linux Enterprise	zypper	Enterprise desktops and servers.	Sysadmins, developers in stable environments.	YaST management tool; tested against SLE.

Distribution	Base	Package Manager	Use Case	Best For	Notable Features
openSUSE Tumbleweed	Independent	zypper	Rolling-release development environments.	Power users who want the latest software.	Always up-to-date; suitable for dev/testing.

Linux boot process

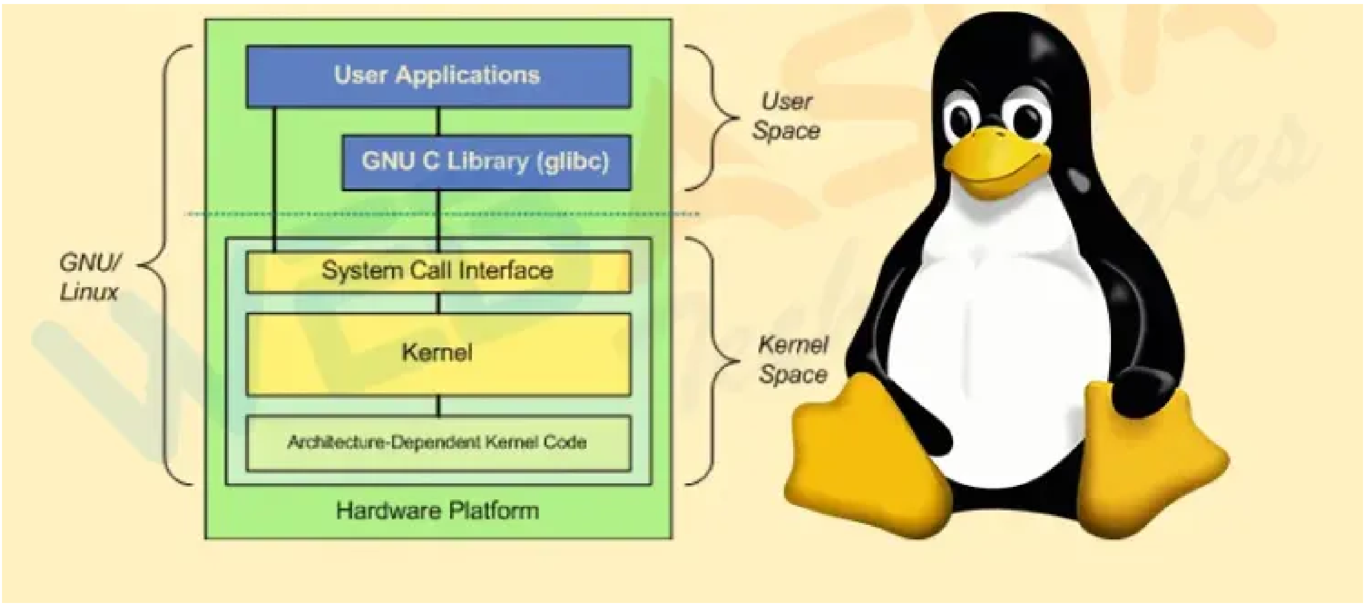
- When you turn on your machine, it does some neat things like show you the logo screen, run through some different messages and then at the end you're prompted with a login window.
- The Linux boot process can be broken down in 4 simple stages:
- BIOS: The BIOS (stands for "Basic Input/Output System") initializes the hardware and makes sure with a Power-on self test (POST) that all the hardware is good to go. The main job of the BIOS is to load up the bootloader.
- Bootloader: The bootloader loads the kernel into memory and then starts the kernel with a set of kernel parameters. One of the most common bootloaders is GRUB, which is a universal Linux standard.
- Kernel: When the kernel is loaded, it immediately initializes devices and memory. The main job of the kernel is to load up the init process.
- Init: Remember the init process is the first process that gets started, init starts and stops essential service process on the system.



The Linux Kernel

- A Linux system is divided into three main parts:
  - Hardware - This includes all the hardware that your system runs on as well as memory, CPU, disks, etc.

  - Linux Kernel - As we discussed above, the kernel is the core of the operating system. It manages the hardware and tells it how to interact with the system.
  - User Space - This is where users like yourself will be directly interacting with the system.
- Why do we have different abstraction layers for user space and kernel? Why can't you combine both powers into one layer?
  - In kernel mode, the kernel has complete access to the hardware, it controls everything. In user space mode, there is a very small amount of safe memory and CPU that you are allowed to access.
  - Basically, when we want to do anything that involves hardware, reading data from our disks, writing data to our disks, controlling our network, etc, it is all done in kernel mode.
- Why is this necessary? Imagine if your machine was infected with spyware, you wouldn't want it to be able to have direct access to your system's hardware. It can access all your data, your webcam, etc. and that's no good.
- These different modes are called privilege levels and are often described as protection rings.
  - To make this picture easier to paint, let's say you find out that a celebrity is in town at your local club, they are protected by there assistants, then their personal bodyguards, then the bouncer outside the club. You want to get their autograph (because why not?), but you can't get to them because they are heavily protected.
  - he rings work the same way, the innermost ring corresponds to the highest privilege level.
  - There are two main levels or modes in an x86 computer architecture. Ring #3 is the privilege that user mode applications run in, Ring #0 is the privilege that the kernel runs in.
  - Ring #0 can execute any system instruction and is given full trust.
- So now that we know how those privilege levels work, how are we able to write anything to our hardware? Won't we always be in a different mode than the kernel?
  - The answer is with system calls, system calls allow us to perform a privileged instruction in kernel mode and then switch back to user mode.
  - Remember the celebrity in the previous lesson? Let's say we want to see her/him and get some drinks together, how do we get from standing outside in the crowds of people to inside her innermost circle? We would use system calls. System calls are like the VIP passes that get you to a secret side door that leads directly to celebrity.
- System calls (syscall) provide user space processes a way to request the kernel to do something for us. The kernel makes certain services available through the system call API. These services allow us to read or write to a file, modify memory usage, modify our network, etc.
- the basics is that when you call a program like ls, the code inside this program contains a system call wrapper (so not the actual system call yet). Inside this wrapper it invokes the system call which will execute a trap, this trap then gets caught by the system call handler and then references the system call in the system call table. Let's say we are trying to call the stat() system call, it's identified by a syscall ID and the purpose of the stat() system call is to query the status of a file. Now remember, you were running the ls program in non-privilege mode. So now it sees you're trying to make a syscall, it then switches you over to kernel mode, there it does lots of things but most importantly it looks up your syscall number, finds it in a table based on the syscall ID and then executes the function you wanted to run. Once it's done, it will return back to user mode and your process will receive a return status if it was successful or if it had an error.
- You can actually view the system calls that a process makes with the strace command. The strace command is useful for debugging how a program executed.



Kernel Location

- What happens when you install a new kernel? Well it actually adds a couple of files to your system, these files are usually added to the /boot directory. You will see multiple files for different kernel versions:
  - vmlinuz - this is the actual linux kernel
  - initrd - as we've discussed before, the initrd is used as a temporary file system, used before loading the kernel
  - System.map - symbolic lookup table
  - config - kernel configuration settings, if you are compiling your own kernel, you can set which modules can be loaded
- If your /boot directory runs out of space, you can always delete old versions of these files or just use a package manager, but be careful when doing maintenance in this directory and don't accidentally delete the kernel you are using.

Kernel Modules

- The kernel in itself is a monolithic piece of software, when we want to add support for a new type of keyboard, we don't write this code directly into the kernel code.
- Kernel modules are pieces of code that can be loaded and unloaded into the kernel on demand. They allow us to extend the functionality of the kernel without actually adding to the core kernel code.
- View a list of currently loaded modules

```
lsmod
```

- Load a module

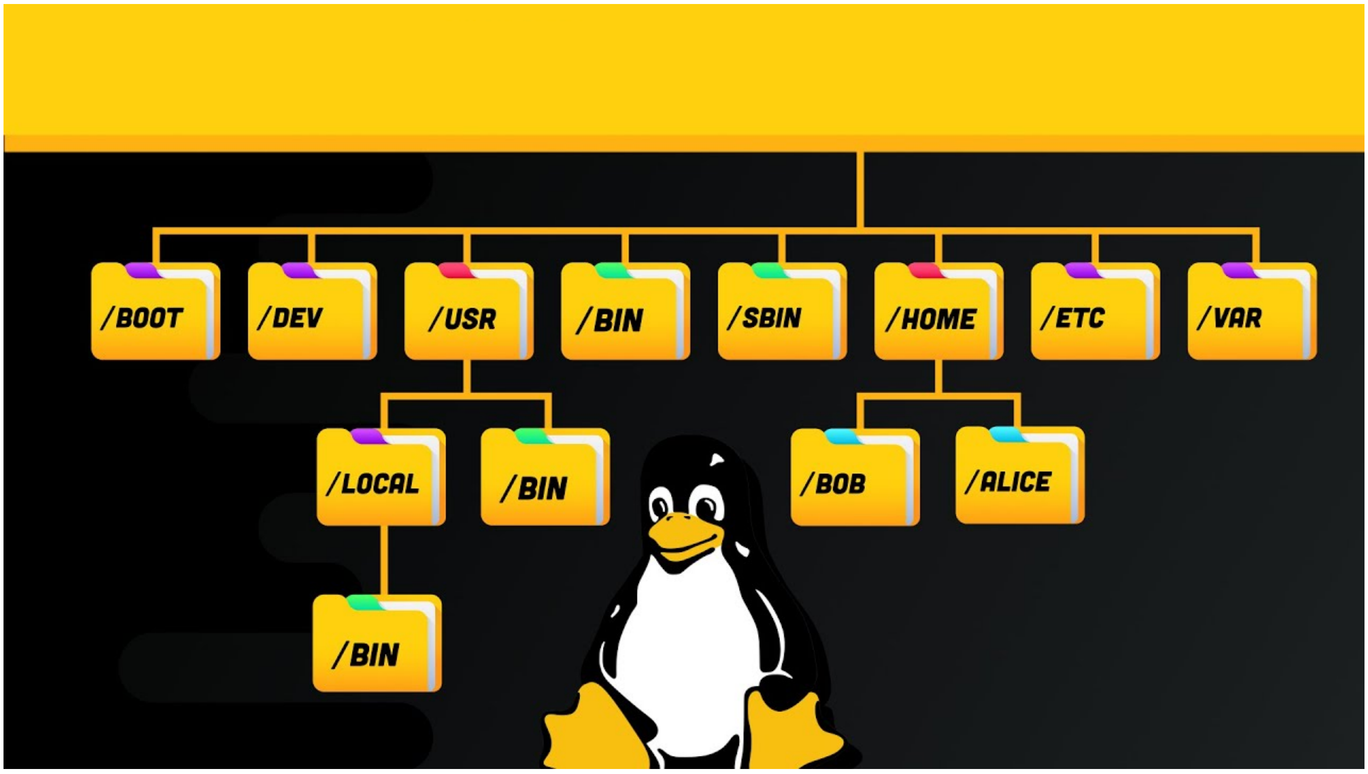
```
sudo modprobe bluetooth
```

- Remove a module

```
sudo modprobe -r bluetooth
```

- You can also load modules during system boot, instead of temporarily loading them with modprobe (which will be unloaded when you reboot). Just modify the /etc/modprobe.d directory and add a configuration file in it.

The linux filesystem



Filesystem hierarchy

- / - The root directory of the entire filesystem hierarchy, everything is nestled under this directory.
- /bin - Essential ready-to-run programs (binaries), includes the most basic commands such as ls and cp.
- /boot - Contains kernel boot loader files.
- /dev - Device files.
- /etc - Core system configuration directory, should hold only configuration files and not any binaries.
- /home - Personal directories for users, holds your documents, files, settings, etc.
- /lib - Holds library files that binaries can use.
- /media - Used as an attachment point for removable media like USB drives.
- /mnt - Temporarily mounted filesystems.
- /opt - Optional application software packages.
- /proc - Information about currently running processes.
- /root - The root user's home directory.
- /run - Information about the running system since the last boot.
- /sbin - Contains essential system binaries, usually can only be ran by root.
- /srv - Site-specific data which are served by the system.
- /tmp - Storage for temporary files

- /usr - This is unfortunately named, most often it does not contain user files in the sense of a home folder. This is meant for user installed software and utilities, however that is not to say you can't add personal directories in there. Inside this directory are sub-directories for /usr/bin, /usr/local, etc.
- /var - Variable directory, it's used for system logging, user tracking, caches, etc. Basically anything that is subject to change all the time.

Filesystem Types

- ext4 - This is the most current version of the native Linux filesystems. It is compatible with the older ext2 and ext3 versions. It supports disk volumes up to 1 exabyte and file sizes up to 16 terabytes and much more. It is the standard choice for Linux filesystems.

- Btrfs - "Better or Butter FS" it is a new filesystem for Linux that comes with snapshots, incremental backups, performance increase and much more. It is widely available, but not quite stable and compatible yet.

- XFS - High performance journaling file system, great for a system with large files such as a media server.

- NTFS and FAT - Windows filesystems

- HFS+ - Macintosh filesystem

- Check out what filesystems are on your machine:

```
df -T
lsblk -f
```

Anatomy of a Disk

- Disks are comprised of partitions that help us organize our data. You can have multiple partitions on a disk and they can't overlap each other. If there is space that is not allocated to a partition, then it is known as free space. The types of partitions depend on your partition table. Inside a partition, you can have a filesystem or dedicate a partition to other things like swap

- There are two main partition table schemes used, Master Boot Record (MBR) and GUID Partition Table (GPT).

- MBR

  - Traditional partition table, was used as the standard
  - Can have primary, extended, and logical partitions
  - MBR has a limit of four primary partitions
  - Additional partitions can be made by making a primary partition into an extended partition (there can only be one extended partition on a disk). Then inside the extended partition you add logical partitions. The logical partitions are used just like any other partition. Silly I know.
  - Supports disks up to 2 terabytes

- GPT

  - GUID Partition Table (GPT) is becoming the new standard for disk partitioning
  - Has only one type of partition and you can make many of them
  - Each partition has a globally unique ID (GUID)
  - Used mostly in conjunction with UEFI based booting (we'll get into details in another course)

- You can use the fdisk of cfdisk utility to partiton and manage your disks

```
cfdisk /dev/sda
```

- You can create a filesystem using the mkfs utility

```
sudo mkfs -t ext4 /dev/sdb2
```

- First create the mount point and mount a partition to that directory

```
mkdir /mydrive
sudo mount -t ext4 /dev/sdb2 /mydrive
```

- kernel names devices in the order it finds them. What if our device name changes for some reason after we mount it? Well fortunately, you can use a device's universally unique ID (UUID) instead of a name.

```
sudo blkid
```

- and you can mount using the uuid

```
sudo mount UUID=130b882f-7d79-436d-a096-1e594c92bb76 /mydrive
```

- When we want to automatically mount filesystems at startup we can add them to a file called /etc/fstab

```
pete@icebox:~$ cat /etc/fstab
UUID=130b882f-7d79-436d-a096-1e594c92bb76 /               ext4
relatime,errors=remount-ro 0       1
UUID=78d203a0-7c18-49bd-9e07-54f44cdb5726 /home           xfs     relatime
0       2
UUID=22c3d34b-467e-467c-b44d-f03803c2c526 none            swap    sw
0       0
```

- UUID - Device identifier

- Mount point - Directory the filesystem is mounted to
- Filesystem type
- Options - other mount options, see manpage for more details
- Dump - used by the dump utility to decide when to make a backup, you should just default to 0
- Pass - Used by fsck to decide what order filesystems should be checked, if the value is 0, it will not be checked
- the disk usage of a filesystem can be retrived using

```
df -h
```

- The fsck (filesystem check) command is used to check the consistency of a filesystem and can even try to repair it for us. Usually when you boot up a disk, fsck will run before your disk is mounted to make sure everything is ok.
- The fsck (filesystem check) command is used to check the consistency of a filesystem and can even try to repair it for us. Usually when you boot up a disk, fsck will run before your disk is mounted to make sure everything is ok.

```
sudo fsck /dev/sda
```

Installation linux server

- Step 1: Download Ubuntu Server
- Step 2: Create a Virtual Machine
- Step 3: Launch the Installer
- Step 4: Configure Network
- Step 5: Configure Storage
- Step 6: Create User Account
- Step 7: Configure Optional Features
- Step 8: Finish Installation
- Step 9: Post-Installation Check

Configuring linux server

system update

- After installing a new Linux server, ensure that security patches and system improvements are applied promptly. This reduces the probability of system vulnerabilities being exploited.

```
sudo apt update && sudo apt upgrade -y
```

Distribution	Update Package List	Upgrade Packages	Remove Unused Packages
Ubuntu/Debian	`sudo apt update`	`sudo apt upgrade -y`	`sudo apt autoremove -y && sudo apt clean`
RHEL/Rocky	`sudo dnf check-update`	`sudo dnf upgrade -y`	`sudo dnf autoremove -y`
Fedora	`sudo dnf check-update`	`sudo dnf upgrade -y`	`sudo dnf autoremove -y`
Arch Linux	`sudo pacman -Sy`	`sudo pacman -Su`	`sudo pacman -Rns $(pacman -Qdtq)`
openSUSE	`sudo zypper refresh`	`sudo zypper update -y`	`sudo zypper packages --orphaned`
Alpine Linux	`sudo apk update`	`sudo apk upgrade`	`sudo apk cache clean`

configure networking

- In Linux, network settings are managed through configuration files. The location and format of these files differ between distributions:

Distribution	Base	Package Manager	Use Case	Best For	Notable Features
Ubuntu	Debian	APT (.deb)	General-purpose computing, cloud, servers.	Beginners, system administrators.	LTS releases; strong community support; user-friendly.
Linux Mint	Ubuntu/Debian	APT (.deb)	Desktop operating system.	New Linux users, multimedia/home users.	Out-of-the-box media support; Cinnamon desktop.
Debian	Independent	APT (.deb)	Base for other distros, server workloads.	Experienced users, minimal setups.	Stability; huge repository; multi-arch.
Fedora	Independent (RHEL upstream)	DNF (.rpm)	Developer platforms, testing new tech.	Developers, security-conscious setups.	Advanced packages; SELinux; upstream to RHEL.

Distribution	Base	Package Manager	Use Case	Best For	Notable Features
Arch Linux	Independent	pacman	Custom, rolling-release systems.	Advanced users who prefer manual control.	User-controlled setup; excellent documentation.
Rocky Linux	RHEL	DNF (.rpm)	Enterprise servers, CentOS replacement.	Organizations needing long-term stability.	RHEL-compatible; actively maintained; community-driven.
Alpine Linux	Independent	apk	Container-based and minimalist environments.	Advanced users, Docker users, embedded systems.	Extremely lightweight, security-focused, musl libc.
AlmaLinux	RHEL	DNF (.rpm)	Server deployments, enterprise workloads.	Cloud/server admins replacing CentOS.	RHEL-compatible; free and open-source; stable releases.
openSUSE Leap	SUSE Linux Enterprise	zypper	Enterprise desktops and servers.	Sysadmins, developers in stable environments.	YaST management tool; tested against SLE.
openSUSE Tumbleweed	Independent	zypper	Rolling-release development environments.	Power users who want the latest software.	Always up-to-date; suitable for dev/testing.

Netplan

- Open the Netplan configuration file

```
sudo nano /etc/netplan/00-installer-config.yaml
```

```
network:
  version: 2
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.1.50/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

```
sudo netplan apply
```

ifconfig

- Another tool to configure networking in linux is ifconfig
- To create an interface and bring it up

```
ifconfig eth0 192.168.2.1 netmask 255.255.255.0 up
```

- This assigns an IP address and netmask to the eth0 interface and also turns it up.
- To bring up or down an interface

```
ifup eth0
ifdown eth0
```

ip

- The ip command also allows us to manipulate the networking stack of a system. Depending on the distribution you are using it may be the preferred method of manipulating your network settings.

- To show interface information for all interfaces

```
ip link show
```

- To show the statistics of an interface

```
ip -s link show eth0
```

- To show ip addresses allocated to interfaces

```
ip address show
```

- To bring interfaces up and down

```
ip link set eth0 up
ip link set eth0 down
```

- To add an IP address to an interface

```
ip address add 192.168.1.1/24 dev eth0
```

route

- To view the routing table of a linux system

```
ip route
route -n
```

- Add a new route

```
ip route add 192.168.2.1/23 via 10.11.12.3
or
sudo route add -net 192.168.2.1/23 gw 10.11.12.3
```

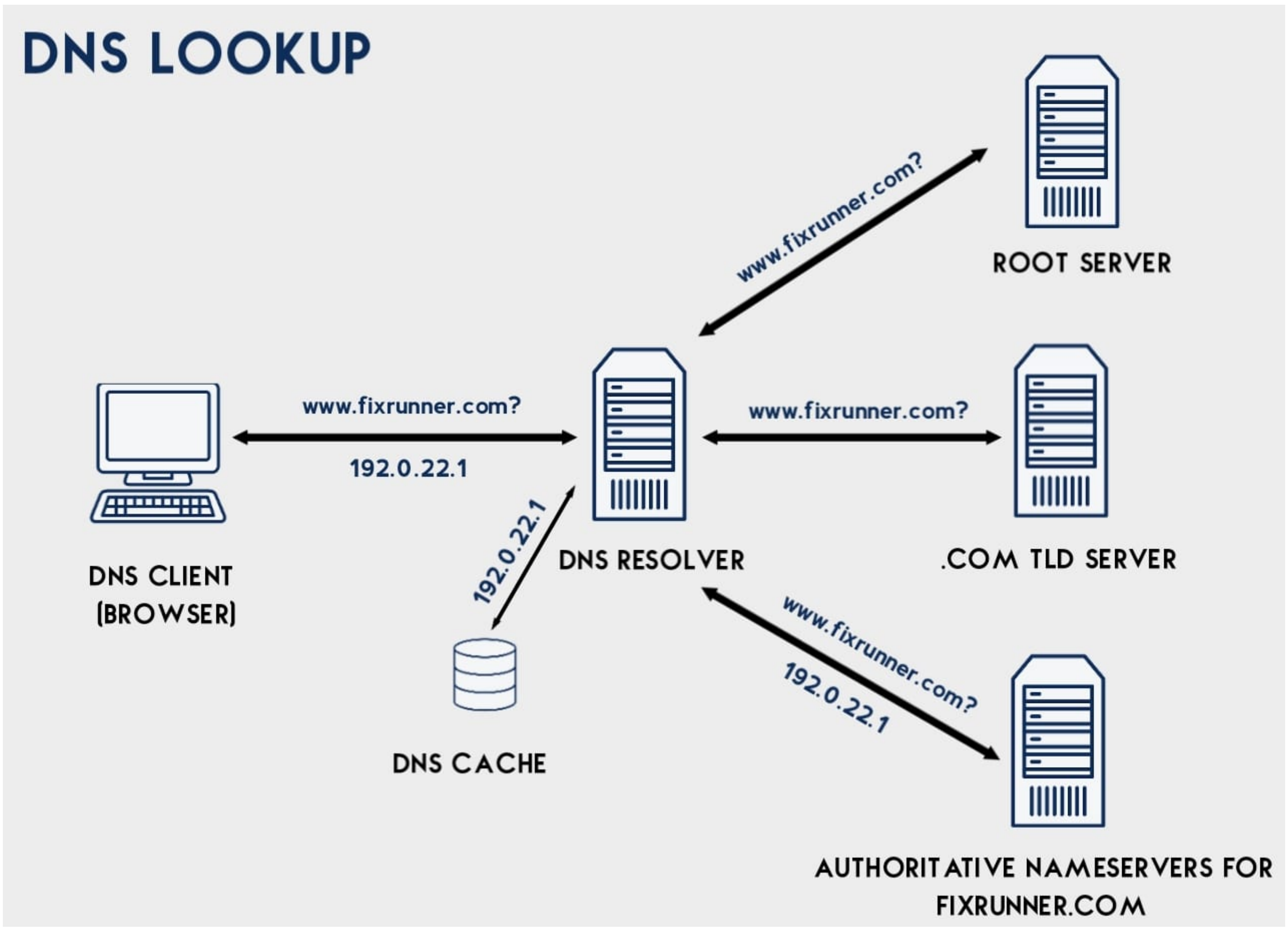
- Delete a route

```
ip route delete 192.168.2.1/23 via 10.11.12.3
or
sudo route del -net 192.168.2.1/23
```

- Obtain a fresh ip

```
sudo dhclient
```

DNS



- We setup DNS via "name servers", the name servers load up our DNS settings and configs and answers any questions from clients or other servers that want to know things like "Who is google.com?".
- If the name server doesn't know the answer to that query, it will redirect the request to other name servers. Name servers can be "authoritative", meaning they hold the actual DNS records that you're looking for, or "recursive" meaning they would ask other servers and those servers would ask other servers until they found an authoritative server that contained the DNS records.
- Recursive servers can also have the information we want cached instead of reaching an authoritative server.
- DNS Process
  - First our host asks, "Where is zergaw.com?", our local DNS server doesn't know so it goes and starts from the top of the funnel to ask the Root Servers. Keep in mind that our host is not making these requests to find zergaw.com directly, most users talk to a recursive DNS server provided by their ISPs and that server is then tasked to find the location of zergaw.com.
  - There are 13 Root Servers for the Internet, they are mirrored and distributed around the world to handle DNS requests for the Internet, so there are really hundreds of servers that are working, they are controlled by different organizations and they contain information about Top-Level Domains. Top-level domains are what you know as .org, .com, .net, etc addresses. So the Root Server doesn't know where zergaw.com is, so it tells us ask the .com Top-Level Domain DNS Server at an IP address it gives us.

  - So now we send another request to the name server that knows about ".com" addresses and asks if it knows where zergaw.com is? The TLD doesn't have the zergaw.com in their zone files, but it does see a record for the name server for zergaw.com. So it gives us the IP address of that name server and tells us to look there.
  - Now we send a final request to the DNS server that actually has the record we want. The name server sees that it has a zone file for zergaw.com and there is a resource record for 'www' for this host. It then gives us the IP address of this host and we can finally see some cats on the Internet.

- Before our machine actually hits DNS to do a query, it first looks locally on our machines.

- The /etc/hosts file contains mappings of some hostnames to IP addresses. The fields are pretty self explanatory, there is one for the IP address, the hostname and then any alias's for the host.

- The "name server lookup" tool is used to query name servers to find information about resource records. Let's find where the name server for google.com is:

```
pete@icebox:~$ nslookup www.google.com

Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
Name:   www.google.com
Address: 216.58.192.4
```

- Dig (domain information groper) is a powerful tool for getting information about DNS name servers, it is more flexible than nslookup and great for troubleshooting DNS issues.

```
pete@icebox:~$ dig www.google.com
; <<>> DiG 9.9.5-3-Ubuntu <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 42376
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; MBZ: 0005 , udp: 512
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         5       IN      A       74.125.239.147
www.google.com.         5       IN      A       74.125.239.144
www.google.com.         5       IN      A       74.125.239.146
www.google.com.         5       IN      A       74.125.239.145
www.google.com.         5       IN      A       74.125.239.148

;; Query time: 27 msec
;; SERVER: 127.0.1.1#53(127.0.1.1)
;; WHEN: Sun Feb 07 10:14:00 PST 2016
;; MSG SIZE  rcvd: 123
```

Core Linux Commands

- Some of the first linux commands you will start using include:

Command	Description
`ls`	List files and directories in the current location.
`cd`	Change the current directory.
`pwd`	Print the current working directory.
`cp`	Copy files or directories.

Command	Description
`mv`	Move or rename a file or directory.
`rm`	Delete files or directories.
`mkdir`	Create a new directory.
`touch`	Create an empty file or update the timestamp.
`cat`	Display the content of a file.
`less`	View the file content one screen at a time.
`head`	Show the first few lines of a file.
`tail`	Show the last few lines of a file.
`echo`	Print text or variables to the terminal.
`grep`	Search for patterns in text.
`find`	Search for files and directories recursively.
`chmod`	Change file or directory permissions.
`chown`	Change file or directory ownership.
`df`	Show disk space usage.
`du`	Show file/directory size.
`top`	Real-time view of system processes and resource usage.
`htop`	Enhanced version of `top` with a user-friendly interface
`ps`	List running processes.
`kill`	Send a signal to a process (often to terminate it).
`sudo`	Run a command with superuser (admin) privileges.
`apt`	Manage packages on Debian-based systems.
`dnf`	Manage packages on RHEL/Fedora systems.
`pacman`	Manage packages on Arch-based systems.
`wget`	Download files from the internet.
`curl`	Transfer data to/from URLs (supports multiple protocols).
`man`	Display the manual for a command.
`history`	Show previously run commands.

Configure SSH

- Default SSH settings may leave a server vulnerable to brute-force attacks or scanning bots. To improve security, users can customize SSH settings in the /etc/ssh/sshd_config file, including:

    - Changing the default port number.
    - Disabling remote root access to the server.
    - Enabling key-based SSH authentication. (Optional)
    - Restricting access to specific IP addresses or users.

- To configure SSH on your Ubuntu Server:

- Enter the following command to install the OpenSSH service:

```
sudo apt install openssh-server -y
```

- Use a preferred text editor, such as Nano, to access the SSH config file:

```
sudo nano /etc/ssh/sshd_config
```

- In the configuration file, find the following line:

```
#Port 22
```

- Uncomment the line and change it to the port you want. For example, set the service to listen on TCP port 2477:

```
Port 2477
```

- Changing the default SSH port number in Linux.

- Ensure the root user cannot be used remotely via a password. Find the following line:

```
#PermitRootLogin prohibit-password
```

- If you uncomment the line, password-based root logins will be disabled, and root users can only log in if they have the corresponding SSH key.

```
PermitRootLogin prohibit-password
```

- To prohibit root logins entirely, enter:

```
PermitRootLogin no
```

- Change root login setting in Linux.
- This setting forces users to log in with regular user accounts and elevate privileges with sudo. It provides maximum security but is slightly less flexible.
    - Press CTRL+X, then Y, followed by Enter to save the changes and exit the file.
    - Restart the SSH service to apply the changes with the following command

```
sudo systemctl restart sshd.service
```

configure firewall

- A firewall monitors and controls incoming and outgoing traffic to and from the server. It is designed to protect systems from unauthorized access, particularly from untrusted external networks.

- Most Linux distributions support firewall tools built on top of iptables or nftables. The table below lists recommended firewall tools for major Linux distributions, along with commands to install and enable them:

Distribution	**Recommended Tool**	**Install Command**	**Enable and Start Command**
Ubuntu/Debian	ufw (Uncomplicated Firewall)	`sudo apt install ufw`	`sudo ufw enable`
RHEL/Rocky	firewalld	`sudo dnf install firewalld`	`sudo systemctl enable --now firewalld`
Fedora	firewalld	Preinstalled (usually)	`sudo systemctl enable --now firewalld`
Arch Linux	iptables or ufw	`sudo pacman -S ufw` or `sudo pacman -S iptables`	`sudo systemctl enable --now ufw` (or iptables)
openSUSE	firewalld	`sudo zypper install firewalld`	`sudo systemctl enable --now firewalld`
Alpine Linux	iptables	Built-in (via the iptables package)	Rules applied via `/etc/network/interfaces`

- Ubuntu Server ships with UFW preinstalled. To confirm the firewall is installed, run:

```
sudo ufw version
```

- Check UFW version in Ubuntu. By default, UFW is disabled. When enabled, it denies all incoming connections to protect the server and allows all outgoing connections so the server can communicate with external services.

- To view the firewall status and active rules, enter:

```
sudo ufw status verbose
```

- Detailed status of UFW in Ubuntu.
- The output shows the status, default settings, and open ports.
- Before enabling UFW, define which incoming connections are necessary for your server setup. For example, if the system uses IPv4 and IPv6, modify the UFW configuration file to support both protocols:
- Open the default UFW configuration file:

```
sudo nano /etc/default/ufw
```

- Check the IPV6 value. If the value is no, change it to yes to enable IPv6 use.

```
sudo ufw allow ssh
```

- Enable SSH in Ubuntu UFW.
- The command adds a rule for IPv4 (and IPv6 if enabled) to allow incoming and outgoing traffic from SSH connections.
- Optionally, specify the custom port defined in the SSH section using the following command:

```
sudo ufw allow 2477/tcp
```

- After configuring the settings, enable the UFW firewall to apply the changes:

```
sudo ufw enable
```

Manage User Accounts and Permissions

- Linux enforces access controls using a permission model built into the file system. Every file and directory has an owner, an associated group, and permissions for three user categories:
  - User (u). The owner of the file or directory.
  - Group (g). Other users in the owners group.
  - Others (o). Everyone else on the system.
- Each of these user categories can be granted the following permissions:

Permission	Symbol	Description
Read	r	A user can view file contents or list directory contents.

Permission	Symbol	Description
Write	w	A user with write permissions can create, move, and delete files or directories, as well as modify or remove file contents.
Execute	x	Allows a user to run script/executable files or enter/traverse directories.

- Use the ls command to view permissions for a file or directory:

```
ls -l
```

- When you install Ubuntu Server, the default user created typically has sudo privileges, which allow them to perform administrative tasks. Most Linux distributions use the same set of commands for managing users and file permissions.
- The following table lists the most commonly used commands:

Command	Function
`adduser` or `useradd`	Creates a new user.
`usermod`	Modifies user properties (e.g., add to group).
`passwd`	Sets or updates the user password.
`ls -l`	Displays file permissions.
`chmod`	Changes file permissions.
`chown`	Changes the file owner and group.

- Enter the following command on your Ubuntu Server to create a non-root admin user:

```
sudo adduser newuser
```

- Follow the prompts to set a password and optional user details.
- Next, add the new user to the sudo group:

```
sudo usermod -aG sudo newuser
```

- This grants the user administrative privileges while keeping root access disabled.

Install and Manage Software Packages

- Each distribution has its default package manager:

Distribution	Package Manager	Example Commands
Ubuntu/Debian	APT	`sudo apt install nginx`
RHEL/Fedora	DNF	`sudo dnf install nginx`
Arch Linux	Pacman	`sudo pacman -S nginx`
openSUSE	Zypper	`sudo zypper install nginx`
Alpine Linux	apk	`sudo apk add nginx`

- One of the key benefits of a package manager is that you can install multiple tools or even entire software stacks with a single command:

```
sudo apt update && sudo apt install apache2 mysql-server php libapache2-
mod-php php-mysql ufw -y
```

Linux Processes

- Processes are the programs that are running on your machine. They are managed by the kernel and each process has an ID associated with it called the process ID (PID).
- This PID is assigned in the order that processes are created.

```
$ ps
PID         TTY      STAT    TIME           CMD
41230     pts/4     Ss         00:00:00      bash
51224     pts/4     R+         00:00:00      ps
```

```
ps aux
```

- The a displays all processes running, including the ones being ran by other users. The u shows more details about the processes. And finally the x lists all processes that don't have a TTY associated with it, these programs will show a ? in the TTY field, they are most common in daemon processes that launch as part of the system startup.

- You'll notice you're seeing a lot more fields now, no need to memorize them all, in a later course on advanced processes, we'll go over some of these again:

    - USER: The effective user (the one whose access we are using)
    - PID: Process ID
    - %CPU: CPU time used divided by the time the process has been running

  - %MEM: Ratio of the process's resident set size to the physical memory on the machine
  - VSZ: Virtual memory usage of the entire process
  - RSS: Resident set size, the non-swapped physical memory that a task has used
  - TTY: Controlling terminal associated with the process
  - STAT: Process status code
  - START: Start time of the process
  - TIME: Total CPU usage time
  - COMMAND: Name of executable/command

- You can send signals that terminate processes, such a command is aptly named the kill command. By default it sends a TERM signal.

```
kill 12445
```

```
kill -9 12445
```

- This will run the SIGKILL signal and kill the process.
- These signals all sound reasonably similar, but they do have their differences.
  - SIGHUP - Hangup, sent to a process when the controlling terminal is closed. For example, if you closed a terminal window that had a process running in it, you would get a SIGHUP signal. So basically you've been hung up on
  - SIGINT - Is an interrupt signal, so you can use Ctrl-C and the system will try to gracefully kill the process
  - SIGTERM - Kill the process, but allow it to do some cleanup first
  - SIGKILL - Kill the process, kill it with fire, doesn't do any cleanup
  - SIGSTOP - Stop/suspend a process

Server Monitoring and Management

- System administrators must continuously monitor key server metrics. This includes memory usage, CPU load, hard disk activity, application performance, and network traffic to:
  - Detect performance bottlenecks.
  - Prevent unexpected outages.
  - Identify unusual behavior or potential security intrusions.
  - Plan for future resource scaling and capacity management.

Tool	Purpose	Available On	Install Method
top	Real-time CPU/memory usage	All Linux systems	Preinstalled
htop	Enhanced top with color & UI	Debian, Ubuntu, Fedora, Arch	apt, dnf, pacman
glances	All-in-one system overview	Most major distros	Python-based (apt, dnf, pip)

Tool	Purpose	Available On	Install Method
vmstat	Memory and process statistics	All Linux systems	Usually preinstalled
iostat	CPU and I/O device stats	Most distros (via sysstat)	apt install sysstat
ss	Socket statistics (modern netstat)	All modern Linux systems	Preinstalled
dstat	Combined CPU, disk, and network usage	Debian, RHEL	apt, dnf
Netdata	Web-based real-time dashboard	All major Linux systems	Shell script installer or packages
Nagios	Enterprise-grade alerting and metrics	Debian, RHEL, SUSE	Manual install, package managers
Prometheus+Grafana	Metrics+Visualization	Cloud/server monitoring stacks	Manual or automated (Docker/K8s)

Managing a Linux Server

- Linux has a wide range of beginner-friendly utilities to help users interact with and administer servers. The following section introduces frequently used tools and commands.

How to Log into a Remote Linux Server

- The most common way to access a remote Linux server is via SSH (Secure Shell).
- Step 1: Install SSH Client and Server
- To use SSH, you must install the necessary packages on both the client and server machines. OpenSSH, discussed in an earlier section, includes both the client and server components.
  - The server machine, the one you are connecting to, needs openssh-server.
  - The client machine, the one you are connecting from, needs openssh-client.
  - If OpenSSH is not installed, use the corresponding command for your distribution:

Distribution	Command to Install SSH Client and Server
Debian/Ubuntu	`sudo apt install openssh-client openssh-server`
RHEL/Fedora	`sudo dnf install openssh-clients openssh-server`
Arch Linux	`sudo pacman -S openssh`
openSUSE	`sudo zypper install openssh`
Alpine Linux	`sudo apk add openssh-client openssh`

- The SSH service (sshd) starts automatically after installation. To confirm it is active, enter the following command:

```
sudo systemctl status sshd
```

- Step 2: Connect to Remote Server via SSH
- After installing and setting up the SSH client and server on both machines:
- Establish a secure remote connection using the SSH command:

```
ssh [username]@[host_ip_address]
```

- Replace [username] with your actual Linux username and [host_ip_address] with the IP address of the remote server. If the username is the same on both machines, you can omit it from the command:

```
ssh host_ip_address
```

- When connecting to the server for the first time, the following message appears:

```
The authenticity of host '192.168.1.10' can't be established.
ECDSA key fingerprint is SHA256:...
Are you sure you want to continue connecting (yes/no)?
```

- Type yes and press Enter to trust the server and save its key to your local system.
- Provide the password when prompted (characters do not appear as you type) and press Enter.

How to Transfer Files to/from a Linux Server

- Rsync is a free command-line utility for transferring files and directories both locally and between local and remote systems. It is preinstalled on most modern Linux distributions.
- Use the command for your distribution to verify the installation or install rsync:

Distribution	**Command to Install rsync**
Debian/Ubuntu	`sudo apt install rsync`
RHEL/Fedora	`sudo dnf install rsync`
Arch Linux	`sudo pacman -S rsync`
openSUSE	`sudo zypper install rsync`
Alpine Linux	`sudo apk add rsync`

- When performing remote data transfers, you must specify the address of the remote host. Use the following syntax to upload local files and directories to a remote server:

```
rsync [OPTION] [SOURCE] [USERNAME]@[HOSTNAME_OR_IP]:[DESTINATION]
```

- You can also download files and directories from a remote system to your local machine:

```
rsync [OPTION] [USERNAME]@[HOSTNAME_OR_IP]:[SOURCE] [DESTINATION]
```

- Rsync is a versatile synchronization tool that can be customized for specific use cases. The -a option enables archive mode (preserving symbolic links, permissions, timestamps, etc.) and -v enables verbose output.

- The following examples cover the most common scenarios.

Description	Command
Upload a file to a remote server.	`rsync -av /home/user/file.txt user@192.168.1.10:/home/user/remote/`
Upload a directory to a remote server.	`rsync -av /home/user/project user@192.168.1.10:/home/user/remote/`
Upload as a different user.	`rsync -av /home/user/test_project test@192.168.1.10:/home/test/remote/`
Upload multiple local directories.	`rsync -av /dir1 /dir2 user@192.168.1.10:/home/user/remote/`
Download a file from a remote server.	`rsync -av user@192.168.1.10:/home/user/file.txt /home/user/local/`
Download a directory from a remote server.	`rsync -av user@192.168.1.10:/home/user/project /home/user/local/`
Download multiple remote directories.	`rsync -av user@192.168.1.10:{/dir1,/dir2} /home/user/local/`
Use rsync over SSH	`rsync -av -e ssh /home/user/file.txt user@192.168.1.10:/home/user/remote/`

- Other simple file sharing tool is the scp command. The scp command stands for secure copy, it works exactly the way the cp command does, but allows you to copy from one host over to another host on the same network. It works via ssh so all your actions are using the same authentication and security as ssh.
- To copy a file over from local host to a remote host

```
scp myfile.txt username@remotehost.com:/remote/directory
```

- To copy a file from a remote host to your local host

```
scp username@remotehost.com:/remote/directory/myfile.txt /local/directory
```

- To copy over a directory from your local host to remote host

```
scp -r mydir username@remotehost.com:/remote/directory
```

- Python has a super useful tool for serving files over HTTP. This is great if you just want to create a quick network share that other machines on your network can access. To do that just go to the directory you want to share and run:

```
python3 -m http.server
```

- This sets up a basic webserver that you can access via the localhost address. So grab the IP address of the machine you ran this on and then on another machine access it in the browser with: http://IP_ADDRESS:8000.

How to Schedule Tasks on Linux Server

- Updating software, creating backups, or clearing caches are essential server management tasks. Linux provides a built-in utility for automating these tasks, called cron.
- The cron utility reads the crontab (cron table) file and executes scheduled commands or scripts at specified times and intervals. A cron job line in a crontab file must follow this format:

```
MIN HOUR DOM MON DOW CMD
```

Field	Description	Example Value
MIN	Minute (0–59)	0
HOUR	Hour (0–23)	9
DOM	Day of Month (1–31)	1
MON	Month (1–12)	1 (January)
DOW	Day of Week (0–7, Sun=0)	* (any day)

Field	Description	Example Value
CMD	Command or script to run (script must be executable)	/path/to/script.sh

- For example, the following line prompts Cron to execute a script on January 1st at 9 AM:

```
0 9 1 1 * /path/to/your_script.sh
```

- The asterisk (*) wildcard in the DOW field instructs Cron to run the job regardless of the day of the week. For a more detailed explanation of wildcards and scheduling syntax, refer to our complete guide on cron expressions.
- Other examples of cron jobs include:

Task	Schedule	Crontab Line
Run a backup script every day at 2 AM	Daily	`0 2 * * * /home/user/scripts/backup.sh`
Clear temp files every Sunday night	Weekly	`0 23 * * 0 rm -rf /tmp/*`
Run a system update on the 1st of every month	Monthly	`0 4 1 * * apt update && apt upgrade -y`

- To view the current user's scheduled cron jobs, enter:

```
crontab -l
```

- To edit the crontab file:

```
crontab -e
```

- This opens crontab in the system's default text editor, where you can add, remove, or change scheduled tasks.
- To remove your crontab and all scheduled jobs, enter:

```
crontab -r
```

- Use this command with caution as it permanently deletes all cron jobs for the current user.
- You can combine cron with session management tools like Byobu to monitor logs or watch script output in real-time.

How to Manage Terminal Sessions in Linux

- Administrators often need to manage multiple terminal sessions from a single interface. This is especially useful for server administration, scripting, or working over SSH connections. If the session drops or you need to step away, the tasks keep running in the background.

- Terminal multiplexers like Byobu, tmux, and screen allow you to:

    - Run multiple terminal sessions within a single SSH connection.
    - Split the terminal into resizable horizontal and vertical panes.
    - Keep tasks running even if your session disconnects.
    - Detach and reattach to sessions without losing state.

- The table lists commonly used commands and shortcuts for Byobu, tmux, and screen:

Function	Byobu	tmux	screen
Start a session.	`byobu`	`tmux`	`screen`
Detach a session.	`Ctrl+A`, then `D` (or `byobu detach`)	`Ctrl+B`, then `D`	`Ctrl+A`, then `D`
Reattach a session.	`byobu attach`	`tmux attach`	`screen -r`
Create a new window.	`Ctrl+A`, then `C`	`Ctrl+B`, then `C`	`Ctrl+A`, then `C`
Next window.	`Ctrl+A`, then `N`	`Ctrl+B`, then `N`	`Ctrl+A`, then `N`
Previous window.	`Ctrl+A`, then `P`	`Ctrl+B`, then `P`	`Ctrl+A`, then `P`
Horizontal split.	`Ctrl+A`, then `Shift+S`	`Ctrl+B`, then `"` (double quote)	No built-in split support
Vertical split.	`Ctrl+A`, then `` ` `` (backtick) | `Ctrl+B`, then `%`	Not applicable	
Move between panes.	`Ctrl+A`, then arrow keys	`Ctrl+B`, then arrow keys	Not applicable
Close the current window/pane.	`exit`	`exit` or `Ctrl+D`	`exit` or `Ctrl+D`

Troubleshooting Linux Networking

ICMP

- The Internet Control Message Protocol (ICMP) is part of the TCP/IP protocol suite, it used to send updates and error messages and is an extremely useful protocol used for debugging network issues such as a failed packet delivery.
- Common ICMP types
    - Type 0 - Echo Reply
    - Type 3 - Destination Unreachable
    - Type 8 - Echo Request

    - Type 11 - Time Exceeded
- When a packet can't get to a destination, Type 3 ICMP message is generated, within Type 3 there are 16 code values that will further describe why it can't get to the destination:
    - Code 0 - Network Unreachable
    - Code 1 - Host Unreachable etc

PING

- One of the most simplest networking tools ping, it's used to test whether or not a packet can reach a host. It works by sending ICMP echo request (Type 8) packets to the destination host and waits for an ICMP echo reply (Type 0). Ping is successful when a host sends out the request packet and receives a response from the target. For example

```
pete@icebox:~$ ping -c 3 www.google.com

PING www.google.com (74.125.239.112) 56(84) bytes of data.

64 bytes from nuq05s01-in-f16.1e100.net (74.125.239.112): icmp_seq=1
ttl=128 time=29.0 ms

64 bytes from nuq05s01-in-f16.1e100.net (74.125.239.112): icmp_seq=2
ttl=128 time=23.7 ms

64 bytes from nuq05s01-in-f16.1e100.net (74.125.239.112): icmp_seq=3
ttl=128 time=15.1 ms
```

- In this example, we are using ping to check if we can get to www.google.com. The -c flag (count) is used to stop sending echo request packets after the count has been reached.
- The first part says that we are sending 64-byte packets to 74.125.239.112 (google.com) and the rest show us the details of the trip. By default it sends a packet per second.
- The icmp_seq field is used to show the sequence number of packets sent, so in this case, I sent out 3 packets and we can see that 3 packets made it back. If you do a ping and you get some sequence numbers missing, that means that some connectivity issue is happening and not all your packets are getting through. If the sequence number is out of order, your connection is probably very slow as your packets are exceeding the one second default.
- The Time To Live (ttl) field is used as a hop counter, as you make hops, it decrements the counter by one and once the hop counter reaches 0, our packet dies. This is meant to give the packet a lifespan, we don't want our packets travelling around forever.
- The roundtrip time it took from you sending the echo request packet to getting an echo reply.

Traceroute

- The traceroute command is used to see how packets are getting routed. It works by sending packets with increasing TTL values, starting with 1. So the first router gets the packet, and it decrements the TTL value by one, thus dropping the packet.
- The router sends back an ICMP Time Exceeded message back to us. And then the next packet gets a TTL of 2, so it makes it past the first router, but when it gets to the second router the TTL is 0 and it

returns another ICMP Time Exceeded message.
- Traceroute works this way because as it sends and drops packets it is build a list of routers that the packets traverse, until it finally gets to its destination and gets an ICMP Echo Reply message.

```
traceroute google.com
traceroute to google.com (216.58.216.174), 30 hops max, 60 byte packets

 1  192.168.4.254 (192.168.4.254)  0.028 ms  0.009 ms  0.008 ms
 2  100.64.1.113 (100.64.1.113)  1.227 ms  1.226 ms 0.920 ms
 3  100.64.0.20 (100.64.0.20)  1.501 ms 1.556 ms  0.855 ms
```

Netstat

- An extremely useful tool to get detailed information about your network is netstat. Netstat displays various network related information such network connections, routing tables, information about network interfaces and more
- We will focus mostly on one feature netstat has and that's the status of network connections.
- The netstat -a command shows the listening and non-listening sockets for network connections, the -t flag shows only tcp connections.

```
pete@icebox:~$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 icebox:domain           *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 icebox.lan:44468        124.28.28.50:http
TIME_WAIT
tcp        0      0 icebox.lan:34751        124.28.29.50:http
TIME_WAIT
tcp        0      0 icebox.lan:34604        economy.canonical.:http
TIME_WAIT
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
tcp6       1      0 ip6-localhost:35094     ip6-localhost:ipp
CLOSE_WAIT
tcp6       0      0 ip6-localhost:ipp       ip6-localhost:35094
FIN_WAIT2
```

- The columns are as follows from left to right:
  - Proto: Protocol used, TCP or UDP.
  - Recv-Q: Data that is queued to be received
  - Send-Q: Data that is queued to be sent
  - Local Address: Locally connected host
  - Foreign Address: Remotely connected host
  - State: The state of the socket

Packet Analysis

- There are two extremely popular packet analyzers, Wireshark and tcpdump. These tools scan your network interfaces, capture the packet activity, parse the packages and output the information for us to see.
- install tcpdump

```
sudo apt install tcpdump
```

- Capture packet data on an interface

```
pete@icebox:~$ sudo tcpdump -i wlan0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on wlan0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:28:23.958840 IP icebox.lan > nuq04s29-in-f4.1e100.net: ICMP echo
request, id 1901, seq 2, length 64
11:28:23.970928 IP nuq04s29-in-f4.1e100.net > icebox.lan: ICMP echo reply,
id 1901, seq 2, length 64
11:28:24.960464 IP icebox.lan > nuq04s29-in-f4.1e100.net: ICMP echo
request, id 1901, seq 3, length 64
11:28:24.979299 IP nuq04s29-in-f4.1e100.net > icebox.lan: ICMP echo reply,
id 1901, seq 3, length 64
11:28:25.961869 IP icebox.lan > nuq04s29-in-f4.1e100.net: ICMP echo
request, id 1901, seq 4, length 64
11:28:25.976176 IP nuq04s29-in-f4.1e100.net > icebox.lan: ICMP echo reply,
id 1901, seq 4, length 64
11:28:26.963667 IP icebox.lan > nuq04s29-in-f4.1e100.net: ICMP echo
request, id 1901, seq 5, length 64
11:28:26.976137 IP nuq04s29-in-f4.1e100.net > icebox.lan: ICMP echo reply,
id 1901, seq 5, length 64
11:28:30.674953 ARP, Request who-has 172.254.1.0 tell ThePickleParty.lan,
length 28
11:28:31.190665 IP ThePickleParty.lan.51056 > 192.168.86.255.rfe: UDP,
length 306
```

- Understanding the output
  - The first field is a timestamp of the network activity
  - IP, this contains the protocol information
  - Next, you'll see the source and destination address: icebox.lan > nuq04s29-in-f4.1e100.net
  - seq, this is the TCP packets's starting and ending sequence number
  - length, length in bytes

```
11:28:23.958840 IP icebox.lan > nuq04s29-in-f4.1e100.net: ICMP echo
request, id 1901, seq 2, length 64
11:28:23.970928 IP nuq04s29-in-f4.1e100.net > icebox.lan: ICMP echo reply,
id 1901, seq 2, length 64
```

- Writing tcpdump output to a file

```
sudo tcpdump -w capture.pcap
```

Logging in Linux

- All Linux systems create and store information log files for boot processes, applications, and other events. These files are a helpful resource for troubleshooting system issues.
- Most Linux log files are stored in plain text files (ASCII format) in the /var/log directory and subdirectories. Logs are generated by the Linux system daemon log, syslogd, or rsyslogd. Properly managing these logs ensures essential data is readily available for analysis and auditing.
- System log files in Linux contain information about the core operating system activities, including boot processes, kernel messages, and hardware events.
    - /var/log/syslog and /var/log/messages capture a wide range of system events, including general system messages, notifications from system services, application errors, and other important events.
    - /var/log/kern.log contains messages from the kernel, including hardware-related information and kernel module logs.
    - /var/log/dmesg logs messages from the kernel ring buffer, primarily capturing information about the boot process, hardware detection, and driver initialization.
    - /var/log/daemon.log contains information about events related to running the Linux operation.
    - /var/log/faillog records failed login attempts, providing information on the number of unsuccessful login attempts by user accounts.
    - /var/log/lastlog maintains a record of the last successful login for each user.
    - /var/log/boot.log stores all information related to booting operations.
    - /var/log/auth.log stores all authentication logs, including successful and failed attempts.
    - /var/log/cron logs activities related to cron jobs, i.e., scheduled tasks running at specified intervals.
    - /var/log/debug stores detailed messages related to debugging and helps troubleshoot specific system operations.
    - /var/log/yum.log is the yum command log. Records activities related to the Yum package manager.
- Essential commands need to intract with log files
    - cat
    - less
    - more
    - tail
    - head
    - grep
- Systemd logs all Linux messages from the kernel and system processes. One of the most powerful systemd functionalities is the logging features. Systemd provides a centralized solution for logging all kernel and user processes through logs known as journals.
- The journald daemon collects all the messages the system outputs and then creates journals, regardless of the program or process. The daemon gathers data from all available system resources and stores them in a binary format.

- The journalctl command queries and manipulates the journal data collected by the journald daemon.

```
journalctl
```

- Without any parameters, the journalctl command outputs the entire journal contents starting from the oldest entry.
- To jump to the pager end and display the most recent entries, use the -e option:

```
journalctl -e
```

- To control how many lines display in the output, use the -n option followed by the number of lines. For example, to show the five most recent journal entries, use

```
journalctl -n 5
```

- To limit the logs to the current boot, use the -b tag without any parameters:

```
journalctl -b
```

- Jump to a specific boot by adding an offset parameter. For example, show the previous boot logs with:

```
journalctl -b 1
```

- An alternative way to see a specific boot is to use a boot ID. Fetch the boot IDs using --list-boots with:

```
journalctl --list-boots
journalctl -b cc07702b00884ec59312ece62604cac8
```

- Filter the journal by specifying a time limit. The two options for limiting since or until a specified time are:

```
journalctl -S 2022-04-02 -U 2022-04-22
journalctl -S "50 minutes ago"
```

- Filter the logs by the specific systemd unit using the -u tag and providing the unit name. For example, to filter only the Jenkins service unit records, run:

```
journalctl -u apache2
```

- To display only the kernel journal log messages, use the -k option:

```
journalctl -k
```

- Use the -f or --follow tag to print the most recent logs continuously:

```
journalctl -f
```

Quiz

This quiz is extracted from Compitia Linux+ Certification

1. **What does the `chmod 755 script.sh` command do?**
   A. Grants read/write to owner, read to group/others
   B. Grants full permissions to owner, read/execute to group/others
   C. Grants read/write/execute to owner, read/execute to group/others
   D. Grants execute-only to all users

2. **What is the purpose of `journalctl`?**
   A. Manage disk partitions
   B. Query systemd journal logs
   C. Configure network interfaces
   D. Schedule cron jobs

3. **Which package manager is used in RHEL-based distributions?**
   A. `apt`
   B. `dnf`
   C. `pacman`
   D. `zypper`

4. **Which partition holds the Linux kernel and boot files?**
   A. `/`
   B. `/var`
   C. `/boot`
   D. `/home`

5. **What does `ssh -i ~/.ssh/id_rsa user@host` do?**
   A. Forces password authentication
   B. Uses a specific private key for authentication
   C. Disables host key checking
   D. Enables verbose mode

6. **Which tool checks and repairs ext2/ext3/ext4 filesystems?**
   A. `fsck`
   B. `mkfs`
   C. `resize2fs`
   D. `tune2fs`

7. **Which command displays open network ports?**
   A. `netstat -tuln`
   B. `ip addr show`
   C. `route -n`
   D. `tcpdump`

8. **What does `grep -E 'error|fail' /var/log/syslog` do?**
   A. Counts errors in syslog
   B. Searches for lines containing "error" or "fail"

C. Deletes matching lines
D. Replaces "error" with "fail"

9. **What is the purpose of `cron`?**
   A. Manage system services
   B. Schedule recurring tasks
   C. Monitor disk usage
   D. Handle user authentication

10. **Which directory contains temporary files cleared at boot?**
    A. `/var/tmp`
    B. `/tmp`
    C. `/dev/shm`
    D. `/run`

11. **Which command shows disk usage per directory?**
    A. `df`
    B. `du`
    C. `mount`
    D. `free`

12. **What does `kill -9 PID` do?**
    A. Gracefully stops a process
    B. Forces immediate termination
    C. Reloads configuration
    D. Pauses the process

13. **How to add a user to a group?**
    A. `useradd -G group user`
    B. `usermod -aG group user`
    C. `groupadd user group`
    D. `chgrp user group`

14. **What is the role of `/etc/fstab`?**
    A. Defines filesystem mount points at boot
    B. Configures firewall rules
    C. Stores user passwords
    D. Manages swap space

15. **Which command tests connectivity to a port?**
    A. `ping`
    B. `traceroute`
    C. `netcat`
    D. `curl`

Day 1: Demo

- Go through the installation step of ubuntu server 24.04 from iso
    - configure networking during installation
    - configure storage during installation
    - create user account during installation
    - configure optional feature during installation
- Review the configurations once installation is finished
    - review network configuration
    - review storage configuration
    - review account configuration
    - perform system update

Day 1: Exercise

- Installation of 2 Linux Server Operating Systems
    - Installing Ubuntu Server
    - Installing Rocky/Alma Linux

Day 1: Exercise Targets

- Going through ubuntu server and rocky linux installation process
- Configure static IP during the installation
- Setup cloudflares DNS for the systems during installation
- Create a separate home partition during installation
- Review your configuration once the installation is done

Day 2: Demo

- SSH Server Configuration on linux
- Configuring firewall on linux
- Managing User Account and permissions
- Package management
- Process management
- Server monitoring
- Configuring ssh-keys and remoting into a machine
- Transfer file and directoires to remote server
- Schedule a task
- Tmux
- Troubleshooting linux server issues

Day 2: Exercise

- Configuring of Linux Server Operating Systems
- Install a Webserver
- Troubleshoot a bricked Linux system

Day 2: Exercise Targets

- Configure the Linux server to have a static ip address
- Configure google dns on the linux server
- Create a user that have root privilages
- Create a user that doesn't have root privilages
- Configure ssh daemon to listen on custom port
- Configure Firewall to allow ssh on a custom port
- Configure passwordless ssh to the user with root privilages
- Trasfer a file from you local computer to the user directory of the unprivilaged user
- Install apache webserver and make it accessable throught the linux servers ip address
- Schedule a task that logs the current time every minute
- All this configuration should be reboot persistent