Based on the information provided in the sources, here are the Linux commands covered in the video transcript, along with explanations of their working:

**Understanding Internet and Servers**

The source first explains concepts like the internet working via optical fibers connecting data centers. Data centers are large locations housing computers for storing and transmitting data. This transmission happens through cables owned by companies. Servers are essentially computers whose job is to **serve information**.

* A **server** serves information, while a **client** requests information from the server.
* Examples of clients include your phone or a laptop with a browser.
* Different types of servers mentioned include **email servers** (serve emails), **file servers** (store and provide files), **database servers** (handle database interactions like inserting data), **application servers** (run applications like facebook.com or youtube.com), and **web servers** (serve static data like images and HTML pages).
* A **web server** generally serves static data, while an **application server** generally serves dynamic data that requires computation or logic. Examples given are Nginx for web servers and Django or Node.js applications running on an application server.
* Applications can be **stand-alone** (don't need internet or external servers) or **web applications** (run on the internet and require supporting servers like email, database, and application servers). DevOps engineers need to know which type of application they are working with as it affects necessary connections.

**Basic Navigation and File Management**

* pwd: **P**rint **w**orking **d**irectory. This command shows your current location in the file system hierarchy.
  + Example: pwd might output /home/ubuntu/devops.
* cd <directory>: **C**hange **d**irectory. This command allows you to move to a different directory.
  + cd /: Moves to the root directory.
  + cd ..: Moves one directory level up.
  + cd .: Refers to the current directory (demonstrated implicitly when referring to the current folder).
  + cd <absolute\_path>: Moves to the directory specified by the full path, starting from root (e.g., cd /home/ubuntu/devops).
  + cd <relative\_path>: Moves to a directory relative to your current location (e.g., cd bin from root to go to /bin).
* mkdir <directory\_name>: **M**a**k**e **dir**ectory. Creates a new folder.
  + Example: mkdir cloud creates a directory named cloud.
* touch <file\_name>: Creates an empty file.
  + Example: touch devopsfile.txt creates an empty file named devopsfile.txt. Also mentioned that redirecting echo output can create a file without touch.
* ls: **L**i**s**t. Displays the contents of the current directory.
* rm <file\_name>: **R**e**m**ove. Deletes a file.
  + Example: rm devopsfile.txt deletes the file. Once removed this way, it's not in the recycle bin; it's gone.
* rm -r <directory\_name>: **R**e**m**ove **r**ecursively. Deletes a directory and all its contents. rm alone cannot remove a directory because it's a directory, not a file. The -r flag means recursively remove everything inside the directory.
  + Example: rm -r cloud deletes the cloud directory and its contents.
* rmdir <empty\_directory\_name>: **R**e**m**ove **dir**ectory. Deletes an *empty* directory.
  + Example: rmdir devops (if devops is empty) deletes the directory.
* cat <file\_name>: Concatenate and display files. This command reads the contents of a file and prints it to the screen.
  + Example: cat demofile.txt shows the content of demofile.txt.
* echo <text>: Prints the given text to standard output (the screen).
  + Example: echo "Hello World" prints "Hello World".
* echo <text> > <file\_name>: Redirects the output of echo to a file, **overwriting** the file if it exists or creating it if it doesn't.
  + Example: echo "Hello dosto" > demofile.txt puts "Hello dosto" into demofile.txt, replacing any previous content.
* zcat <zipped\_file.gz>: Displays the content of a gzipped (.gz) file without decompressing it.
* head <file\_name>: Displays the **top** few lines of a file (default is 10, but not specified in source).
  + Example: head myfile.txt shows the top lines of myfile.txt.
* tail <file\_name>: Displays the **bottom** few lines of a file (default is 10, but not specified in source).
  + Example: tail myfile.txt shows the bottom lines of myfile.txt.
* tail -f <file\_name>: Displays the bottom lines of a file and then **monitors the file for new lines**, printing them as they appear. Useful for watching log files in real-time.
  + Example: tail -f logfile.log will show new entries in the log file. It runs until you press Ctrl+C.
* less <file\_name>: Displays file content page by page, allowing you to scroll through large files.
  + Example: less myfile.txt opens the file in a paginated view.
* more <file\_name>: Similar to less, displays file content page by page.
* cp <source> <destination>: **C**o**p**y. Copies files or directories.
  + Example: cp newfile.txt devops/ copies newfile.txt into the devops directory.
  + Can copy files from other directories by specifying the path: cp devops/devopsfile.txt cloud/ copies devopsfile.txt from the devops directory to the cloud directory.
* cp -r <source\_directory> <destination\_directory>: **C**o**p**y **r**ecursively. Copies a directory and all its contents to a destination. Necessary because cp alone cannot copy directories.
  + Example: cp -r cloud devops/ copies the entire cloud directory into the devops directory.
* mv <source> <destination>: **M**o**v**e. Moves files or directories from one location to another. Unlike copy, the source is removed after the move.
  + Example: mv newfile.txt ../cloud/ moves newfile.txt from the current directory to the cloud directory located one level up.
  + Can also be used to **rename** files or directories by moving them to a new name in the same location.
  + Example: mv devops linuxfordevops renames the directory devops to linuxfordevops.

**File Content Utilities**

* wc <file\_name>: **W**ord **c**ount. Prints the number of lines, words, and bytes in a file.
  + Example: wc myfile.txt outputs something like 1 4 16 myfile.txt, indicating 1 line, 4 words, and 16 bytes.
* ln <source> <link\_name>: Creates a **hard link** to a file. A hard link is a shortcut where if the original file is deleted, the hard link still works. *Example syntax shown in source for creating hard link was slightly incomplete/had errors initially, but the concept is explained.*
* ln -s <source> <link\_name>: Creates a **soft link** (or symbolic link) to a file or directory. A soft link is a shortcut where if the original file is deleted, the soft link becomes broken and doesn't work.
  + Example: ln -s /home/ubuntu/linuxfordevops/cloud/devopsfile.txt softlinkfile creates a soft link named softlinkfile pointing to devopsfile.txt. Changes to the original file are reflected in the soft link. Removing the original file breaks the soft link. Soft links are often displayed in a different color (e.g., blue) and show the link target when listing details (ls -l).
* cut <options> <file\_name>: Extracts sections from each line of files.
  + cut -b <list>: Extracts bytes specified by the list.
  + Example: cut -b 1 myfile.txt cuts the first byte. cut -b 1-4 myfile.txt cuts bytes 1 through 4.
* echo <text> | tee <file\_name>: Uses a **pipe** (|) to send the output of the echo command as input to the teecommand. The tee command reads standard input and writes it to both standard output (the screen) and one or more files.
  + Example: echo "Hello" | tee helloworld.txt prints "Hello" to the screen and also writes "Hello" into helloworld.txt. If the file exists, it's overwritten by default.
* sort <file\_name>: Sorts the lines of a text file alphabetically.
  + Example: sort file.txt would print the lines of file.txt in sorted order.
* diff <file1> <file2>: Compares two files line by line and shows the differences.
  + Example: diff hello.txt demofile.txt shows the lines that are different between the two files.
* vi / vim: Text editors used for creating and modifying files. vim is described as more advanced and visually better. The source demonstrates using vi (or implicitly vim) to edit a file, showing how changes within the editor appear when using cat afterwards.

**System Information and Process Management**

* df: Reports file system disk space usage. Shows total blocks, used, and available space for mounted file systems.
  + df -h: Displays the information in a **h**uman-readable format (e.g., GB, MB).
* du <path>: **D**isk **u**sage. Estimates file space usage for a given path.
  + du .: Shows disk usage for the current directory.
* ls -a: **L**i**s**t **a**ll. Lists all files in a directory, including hidden files which start with a dot (.).
* ps: Reports a snapshot of current processes.
  + ps aux: A common variation (though options not explicitly explained) used to list all processes running on the system for all users.
* top: Displays Linux processes in real-time, showing PID, user, CPU usage, memory usage, etc.. Allows monitoring system processes.
  + q: Pressing q quits the top command.
* fuser: Used to identify processes using files or sockets. Mentioned but not demonstrated in detail.
* kill <PID>: Sends a signal to terminate a process identified by its **P**rocess **ID**. Requires permission to kill a process.
  + Example: kill 162 attempts to kill the process with PID 162 (permission denied in the example).
* free: Displays the amount of free and used physical and swap memory.
  + free -h: Displays memory information in a **h**uman-readable format (e.g., MB, GB).
* nohup <command>: Runs a command immune to hangup signals (HUP), automatically redirecting standard output to a file named nohup.out. Useful for running commands in the background or after disconnecting from a session. The output is appended to nohup.out if it exists.
  + Example: nohup df -h runs df -h and saves its output to nohup.out.
* uname: Prints system information, such as the operating system name or kernel.
  + Example: uname shows Linux on the Ubuntu EC2 instance and Darwin on a macOS system.
* uptime: Shows how long the system has been running and how many users are currently logged in.
* date: Prints the current system date and time.
* who: Shows who is logged on to the system and when they logged in. Provides a list if multiple users are logged in.
* whoami: Prints the effective user ID name of the current user. Returns only one user name.
* which <command>: Locates the executable file associated with a command. Useful for finding where a program is installed.
  + Example: which python shows the path to the python executable. which docker shows /usr/bin/dockerafter it's installed.
* sudo <command>: **S**uper**u**ser **do**. Executes a command with superuser (root) privileges. Necessary for actions requiring administrative permissions, like shutting down the system or installing software.
  + Example: sudo shutdown. Attempting shutdown without sudo fails due to lack of permission.
* shutdown: Halts or reboots the system. Requires superuser privileges (sudo).

**Package Management**

* sudo apt update: Updates the list of available packages from the configured repositories. Necessary before installing new software to ensure you get the latest versions and dependencies.
* sudo apt install <package\_name>: Installs software packages using the APT package manager. Requires superuser privileges.
  + Example: sudo apt install docker installs the docker package.
  + Also used to install tools like traceroute, whois, jq, zip, unzip (implied), wireless-tools (implied), net-tools (implied).
* sudo apt remove <package\_name>: Removes software packages. Requires superuser privileges.
  + Example: sudo apt remove docker uninstalls the docker package.
* Ctrl+R: Keyboard shortcut for reverse search in the command history. Allows you to search for previously typed commands.

**User and Group Management**

* id <username>: Prints user and group information for the specified user, or the current user if no username is given. Shows the User ID (**UID**) and the Group IDs (**GID**s) the user belongs to.
  + Example: id jethalal shows the UID and GIDs for the user jethalal.
* su <username>: **S**witch **u**ser. Switches the current user to the specified username. Often requires the password of the target user.
  + Example: su jethalal switches to the user jethalal.
  + sudo su: Switches to the root user with superuser privileges without needing the root password (if your user is in the sudoers list).
* exit: Exits the current user session or shell. Used to switch back from a different user (e.g., root or Jethalal) to the previous user (e.g., ubuntu).
* sudo useradd <username>: Creates a new user account on the system. Requires superuser privileges.
  + Example: sudo useradd jethalal.
* sudo userdel <username>: Deletes a user account. Requires superuser privileges.
  + Example: sudo userdel jethalal.
* sudo groupadd <groupname>: Creates a new group. Requires superuser privileges.
  + Example: sudo groupadd devops.
* cat /etc/group: Displays the contents of the /etc/group file, which lists all groups on the system and their members.
* sudo gpasswd -a <username> <groupname>: Adds a user to a specified group. Requires superuser privileges.
  + Example: sudo gpasswd -a jethalal devops adds the user jethalal to the devops group.
* sudo groupdel <groupname>: Deletes a group. Requires superuser privileges.
  + Example: sudo groupdel tester deletes the tester group.

**File Permissions**

* ls -l: Lists directory contents in a **l**ong format, providing details including file type, permissions, number of hard links, owner, group, size, and modification time.
  + The first character indicates the file type (d for directory, - for regular file).
  + The next nine characters represent permissions for the **u**ser (owner), **g**roup, and **o**thers (all other users), in sets of three (rwx).
  + **r**: Read permission (view file content, list directory content).
  + **w**: Write permission (modify file content, create/delete files in a directory).
  + **x**: Execute permission (run an executable file, enter a directory).
* chmod <permissions> <file\_or\_directory>: **Ch**ange **mod**e. Changes the permissions of a file or directory. Permissions can be specified numerically using octal notation.
  + Numeric values for permissions: **r=4**, **w=2**, **x=1**. Dash (-) for no permission is 0.
  + Permissions for user, group, and others are represented by a three-digit octal number. Example: **777** means rwx (4+2+1=7) for user, group, and others. **664** means rw (4+2=6) for user and group, and r (4) for others. **700** means rwx (7) for the user, and no permissions (0) for group and others.
  + Example: chmod 777 cloud changes permissions for the cloud directory to read, write, and execute for user, group, and others. chmod 700 devopsfile.txt sets rwx for the owner and no permissions for group or others.
* umask: **U**ser file creation **mask**. Displays or sets the default permissions for newly created files and directories. It's a four-digit octal number (the first digit often relates to ACLs and is often 0). The mask bits are *subtracted* from the maximum possible permissions (666 for files, 777 for directories) to determine the default permissions.
  + Example: A umask of 002 means 666-002=664 (rw-rw-r--) for files and 777-002=775 (rwxrwxr-x) for directories. A umask of 022 means 666-022=644 (rw-r--r--) for files and 777-022=755 (rwxr-xr-x) for directories.
* sudo chown <new\_owner> <file\_or\_directory>: **Ch**ange **own**er. Changes the ownership of a file or directory to a new user. Requires superuser privileges.
  + Example: sudo chown jethalal demofile.txt changes the owner of demofile.txt to jethalal.
* sudo chgrp <new\_group> <file\_or\_directory>: **Ch**ange **gr**ou**p**. Changes the group ownership of a file or directory to a new group. Requires superuser privileges.
  + Example: sudo chgrp devops demofile.txt changes the group of demofile.txt to devops.

**Compression and Archiving**

* zip <archive\_name>.zip <files\_or\_directories>: Creates a .zip archive. Requires the zip utility to be installed (sudo apt install zip).
  + Example: zip cloud.zip cloud/\* archives the contents of the cloud directory into cloud.zip.
* unzip <archive\_name>.zip: Extracts files from a .zip archive. Requires the unzip utility (installation implied).
  + Example: unzip cloud.zip extracts the contents of cloud.zip.
* gzip <file\_name>: Compresses a file using gzip compression, adding a .gz extension.
  + gunzip <file\_name>.gz: Decompresses a file compressed with gzip.
* tar <options> <archive\_name> <files\_or\_directories>: Creates, views, or extracts files from .tar archives. Can be combined with compression (like gzip).
  + Common options:
    - -c: Create an archive.
    - -x: Extract files from an archive.
    - -v: Verbose output, showing files being processed.
    - -z: Use gzip compression (.tar.gz).
    - -f <filename>: Specify the archive file name.
  + Example to create a .tar.gz archive: tar -cvzf cloud.tar.gz cloud/ creates a gzipped tar archive named cloud.tar.gz from the cloud directory. (Note: The example typed was tar cvzaf cloud cloud.tar.gzwhich was corrected during explanation). The correct order is often tar -cvzf archive\_name source\_file(s). The example shown that worked was tar cvzaf cloud cloud.tar.gz followed by correction tar cvzf cloud.tar.gz cloud. Let's use the working syntax shown and explained.
  + Example to extract: tar -xvzf cloud.tar.gz extracts the contents from cloud.tar.gz.

**Networking**

* ping <hostname\_or\_IP>: Sends network packets (ICMP echo requests) to a host to check connectivity and measure the time taken for responses. Reports the number of packets transmitted and received. Useful for diagnosing network reachability.
  + Example: ping trainwithshubham.com sends packets to the server hosting the website.
* netstat: Displays network connections (inbound and outbound), routing tables, interface statistics, protocol statistics, etc.. Useful for seeing active connections and network configuration.
* ifconfig: Configures network interfaces and displays their configuration (IP address, MAC address, status, etc.).
  + Example: ifconfig shows details for interfaces like docker0, eth0 (Ethernet), and lo (loopback). Explains eth0 relates to the EC2 instance's network interface card (NIC) and lo represents the loopback interface used for localhost (127.0.0.1) for internal communication within the server.
* ip addr show: Similar to ifconfig, displays address information for network interfaces.
* traceroute <hostname\_or\_IP>: Traces the route that packets take from the source system to a destination host, showing the IP addresses of the routers (hops) along the path. Useful for diagnosing where network latency or connectivity issues occur. Requires installation (sudo apt install traceroute).
  + Example: traceroute youtube.com shows the path to youtube.com.
* tracepath <hostname\_or\_IP>: Similar to traceroute, but might be faster or provide different information. Also shows the path taken by packets. Can show No reply if packets are lost or delayed.
  + Example: tracepath trainwithshubham.com.
* mtr <hostname\_or\_IP>: **M**y **T**race**r**oute. Combines the functionality of ping and traceroute, continuously displaying packet loss and latency to each hop along the path. Described as more efficient for diagnosing issues.
  + Example: mtr trainwithshubham.com.
* nslookup <domain\_name>: Queries domain name servers to obtain domain name or IP address mapping. Can show the server used for the lookup and the corresponding IP address(es) for the domain. Can sometimes reveal different IPs for HTTP (port 80) and HTTPS (port 443) requests.
  + Example: nslookup trainwithshubham.com.
* telnet <hostname\_or\_IP> <port>: Used to check connectivity to a specific port on a remote host.
  + Example: telnet trainwithshubham.com 80 attempts to connect to port 80 (HTTP). telnet trainwithshubham.com 443 attempts to connect to port 443 (HTTPS).
* hostname: Prints the system's hostname.
  + Example: hostname might output the server's IP address acting as its hostname, or localhost for 127.0.0.1.
* iwconfig: Displays information about wireless network interfaces. Requires installation (sudo apt install wireless-tools).
* ss: Another utility to investigate sockets and network connections, often considered a replacement for netstat. Provides similar information.
* dig <domain\_name>: **D**omain **I**nformation **G**roper. A more flexible tool than nslookup for querying DNS servers. Can retrieve various DNS record types (like A, MX, NS).
  + Example: dig trainwithshubham.com shows DNS information for the domain, including the A (address) record.
* whois <domain\_name>: Retrieves registration information about a domain name (registrar, registration date, expiration date, contact info). Requires installation (sudo apt install whois).
  + Example: whois trainwithshubham.com.
* curl <URL\_or\_API\_endpoint>: A command-line tool for transferring data with URLs. Primarily used here to make HTTP requests to APIs and retrieve data.
  + curl -X GET <API\_URL>: Makes a GET request to the specified API endpoint.
  + Example: curl -X GET http://dummy.restapiexample.com/api/v1/employees fetches data from the employees API.
* jq: A lightweight and flexible command-line JSON processor. Used with curl and a pipe (|) to format messy JSON output into a readable structure. Requires installation (sudo apt install jq).
  + Example: curl ... | jq formats the JSON output from the curl command.
* wget <URL>: A command-line utility for downloading files from the web. Can download files specified by a URL.
  + Example: wget <sample\_file\_url> downloads the file from the URL.
* nmap <target>: **N**etwork **map**per. A security scanner used to discover hosts and services on a computer network, building a "map" of the network. Can scan a target (IP or hostname) to find open ports.
  + nmap -v <target>: Verbose output mode.
  + Example: nmap -v google.com scans google.com and shows open ports like 80. Note: Scanning systems you don't own permission for is generally illegal.
* route: Shows or manipulates the IP routing table. Displays the network destination, gateway, and interface used for routing traffic. Relates to routing tables seen in cloud environments like AWS VPCs.

**Text Processing**

* sed <options> <script> <file\_name>: **S**tream **ed**itor. Used for performing basic text transformations on an input stream or a file.
  + s/<pattern>/<replacement>/g: Substitution command: find pattern, replace with replacement, globally on the line.
  + Example: sed 's/Info/Log/g' Application\_log.json replaces all occurrences of "Info" with "Log" in Application\_log.json and prints the result to the screen.
  + = : Prints the line number. Requires options like -n (suppress default output) and -e (execute a script). Example: sed -n -e '/Info/=' Application\_log.json prints the line numbers where "Info" appears.
  + Combining = and p (print): sed -n -e '/Info/=' -e '/Info/p' Application\_log.json prints both the line number and the line containing "Info".
  + Applying commands to a range of lines: sed '1,10s/Info/Log/g' Application\_log.json applies the substitution only to lines 1 through 10.
* grep <options> <pattern> <file\_name>: **G**lobally search for a **r**egular **e**xpression and **p**rint matching lines. Searches for lines that match a given pattern in files and prints those lines.
  + Example: grep Info Application\_log.json prints all lines in Application\_log.json that contain the word "Info".
  + grep -i <pattern> <file\_name>: Performs a **c**ase-**i**nsensitive search.
  + grep -c <pattern> <file\_name>: Counts the number of lines that match the pattern.
  + Using with pipes: ps aux | grep ubuntu sends the output of ps aux to grep, which then filters the lines to show only those containing "ubuntu".
* awk <script> <file\_name>: A powerful pattern scanning and processing language. Can perform complex text manipulation and analysis.
  + /pattern/ { action }: Basic awk structure: for lines matching pattern, perform action.
  + Example: awk '/Info/' Application\_log.json prints lines containing "Info" (similar to grep).
  + print $N: Prints the Nth field (column) of a line.
  + Example: ps aux | awk '{print $2}' takes the output of ps aux and prints only the second column (usually the PID). awk is more complex for simple tasks like counting compared to grep.

**Volume Management (LVM and AWS EBS Context)**

* lsblk: **L**i**s**t **b**loc**k** devices. Lists available block devices (disks, volumes, partitions) and their hierarchical relationships. Shows their names (e.g., xvda, xvdf, xvdg, xvdh), sizes, and mount points if mounted. Useful for seeing which volumes are attached to the system.
* df -h: (Already mentioned in System Info, but crucial here) Shows **d**isk **f**ree space of mounted file systems in a **h**uman-readable format. Important for seeing which attached volumes are actually available for use by being mounted to a directory.
* sudo su: Switches to the root user. Necessary for most volume and LVM operations as they require administrative privileges.
* pvs: **P**hysical **V**olume**s**. Reports information about physical volumes. Shows devices that have been initialized by LVM for use in volume groups.
  + Example: After running pvcreate, pvs shows /dev/xvdf, /dev/xvdg, /dev/xvdh as PVs with their size and free space.
* sudo pvcreate <device\_path>: Initializes a block device (like /dev/xvdf) or file for use as a **p**hysical **v**olume by LVM. Requires superuser privileges. This converts a raw block device into an LVM-manageable PV.
  + Example: sudo pvcreate /dev/xvdf /dev/xvdg /dev/xvdh initializes these three devices as PVs.
* vgs: **V**olume **G**roup**s**. Reports information about volume groups. Shows existing volume groups, their size, free space, and the number of physical and logical volumes within them.
* sudo vgcreate <group\_name> <pv\_path1> <pv\_path2> ...: Creates a **v**olume **g**roup by combining one or more physical volumes. Requires superuser privileges.
  + Example: sudo vgcreate TWS\_VG /dev/xvdf /dev/xvdg creates a volume group named TWS\_VG using the physical volumes /dev/xvdf and /dev/xvdg. The total size of the VG is the sum of the sizes of the PVs.
* lvs: **L**ogical **V**olume**s**. Reports information about logical volumes. Shows logical volumes, their volume group, size, and status.
* sudo lvcreate -L <size> -n <lv\_name> <vg\_name>: Creates a **l**ogical **v**olume from a volume group. Requires superuser privileges.
  + -L <size>: Specifies the size of the logical volume (e.g., 10g for 10 GB).
  + -n <lv\_name>: Specifies the name for the logical volume (e.g., TWS\_LV).
  + <vg\_name>: Specifies the volume group from which to create the logical volume (e.g., TWS\_VG).
  + Example: sudo lvcreate -L 10g -n tws\_lv tws\_vg creates a 10GB logical volume named tws\_lv from the tws\_vg volume group. Logical volumes are often found under /dev/<vg\_name>/<lv\_name> (e.g., /dev/TWS\_VG/tws\_lv).
* pvdisplay: Provides detailed information about physical volumes.
* vgdisplay: Provides detailed information about volume groups.
* lvdisplay: Provides detailed information about logical volumes.
* sudo mkfs.<filesystem\_type> <device\_path>: **M**a**k**e **f**ile **s**ystem. Formats a block device (a physical volume or a logical volume) with a specific file system type (like ext4 or xfs). This prepares the volume to be mounted and store files. Requires superuser privileges.
  + Example for Logical Volume: sudo mkfs.ext4 /dev/mapper/TWS\_VG-TWS\_LV formats the logical volume with the ext4 filesystem. (/dev/mapper/VG\_name-LV\_name is the path to an LV).
  + Example for Physical Volume/Block Device: sudo mkfs.xfs /dev/xvdh formats the /dev/xvdh device with the xfs filesystem. Note: Formatting a device already containing an LVM partition might require confirmation.
* sudo mount <device\_path> <mount\_point>: Attaches a formatted volume (with a filesystem) to a specific directory in the file system tree. This makes the storage on the volume accessible through that directory. Requires superuser privileges.
  + <device\_path>: The path to the device or logical volume (e.g., /dev/mapper/TWS\_VG-TWS\_LV or /dev/xvdh).
  + <mount\_point>: The directory where the volume's content will be accessible (e.g., /mnt/TWS\_LV\_mount). The mount point directory must exist before mounting.
  + Example: sudo mount /dev/mapper/TWS\_VG-TWS\_LV /mnt/TWS\_LV\_mount makes the logical volume accessible under /mnt/TWS\_LV\_mount.
  + Example: sudo mount /dev/xvdh /mnt/TWS\_Disk\_mount makes the physical disk accessible under /mnt/TWS\_Disk\_mount.
* sudo lvextend -L +<size> <logical\_volume\_path>: Extends the size of a logical volume using free space from its volume group. Requires superuser privileges.
  + -L +<size>: Specifies the amount to increase the LV size by (e.g., +5G adds 5 GB).
  + <logical\_volume\_path>: The path to the logical volume (e.g., /dev/TWS\_VG/tws\_lv).
  + Example: sudo lvextend -L +5G /dev/TWS\_VG/tws\_lv extends the logical volume tws\_lv by 5GB, changing its size from 10GB to 15GB. Note: After extending the logical volume, you typically need to resize the filesystem *on* the volume to make the extra space usable, but this step was mentioned conceptually and not explicitly demonstrated with a command like resize2fs or xfs\_growfs in the source.

This comprehensive list covers the commands and their explanations as detailed in the provided video transcript excerpts.