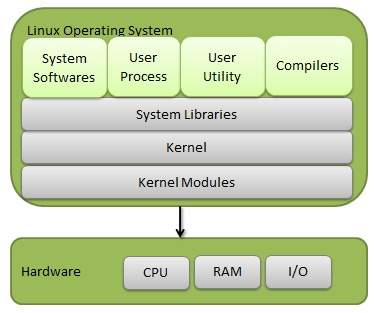
Introduction to Linux OS

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

Components of Linux System

Linux Operating System has primarily three components

* **Kernel** − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
* **System Library** − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
* **System Utility** − System Utility programs are responsible to do specialized, individual level tasks.



basic Linux commandBefore you learn about the **basic commands in Linux**, you must ensure to meet the prerequisites –

* A system that operates Linux
* Accessibility to the command line or terminal

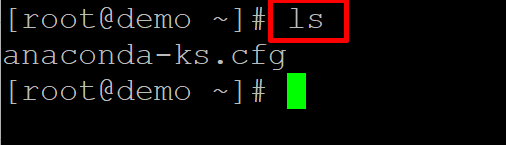
Moreover, all the **commands of Linux** can fall into one of the following four classifications:

* **Shell builtins** – These include commands that are constructed directly into the shell with the fastest execution.
* **Shell functions** – These include shell scripts that are basically grouped commands.
* **Aliases** – They incorporate custom command shortcuts.
* **Executable programs** – These include compiled and installed programs or scripts.

**1. ls**

It lists the files and directories in the current directory.

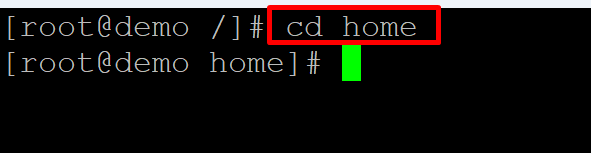
*Syntax: ls [options] [directory]*



**2. cd**

Changes the current directory.

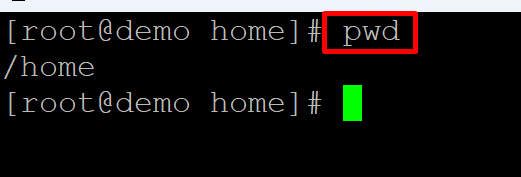
*Syntax: cd [directory]*



**3. pwd**

Shows the current working directory.

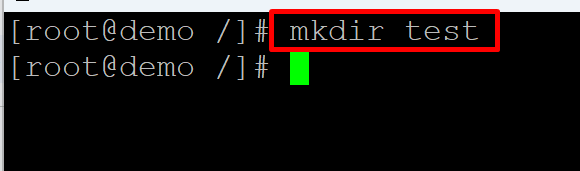
*Syntax: pwd*



**4. Mkdir**

Creates a new directory.

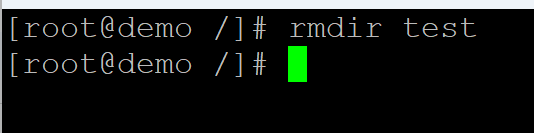
*Syntax: mkdir [directory]*



**5. rmdir**

Deletes an empty directory.

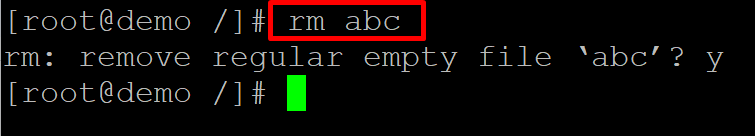
*Syntax: rmdir [directory]*



**6. rm**

Deletes a file or directory.

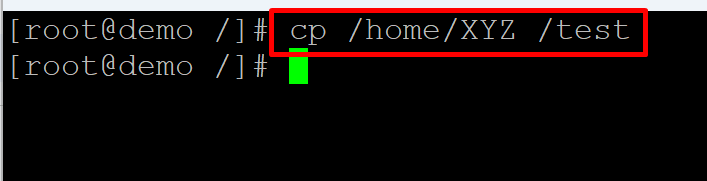
*Syntax: rm [file/directory]*



**7. cp**

Copies a file or directory.

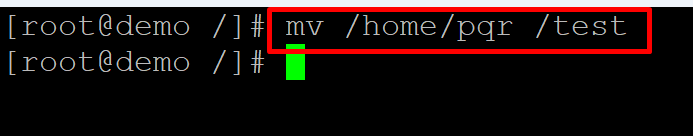
*Syntax: cp [options] [source] [destination]*



**8. mv**

Moves or renames a file or directory.

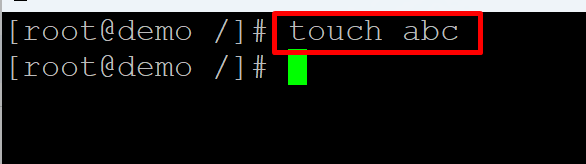
*Syntax: mv [options] [source] [destination]*



**9. touch**

Creates a new empty file.

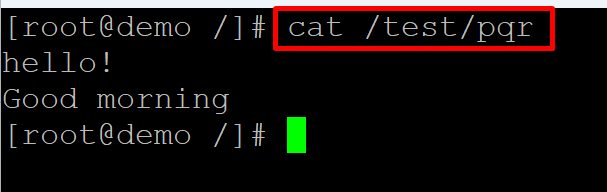
*Syntax: touch [filename]*



**10. cat**

Displays the contents of a file.

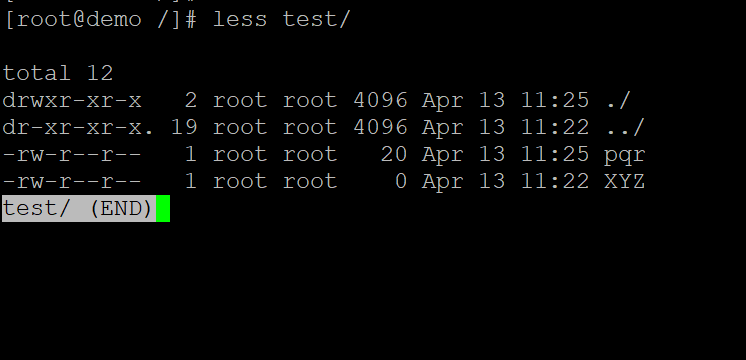
*Syntax: cat [filename]*



**11. less**

Displays the contents of a file one screen at a time.

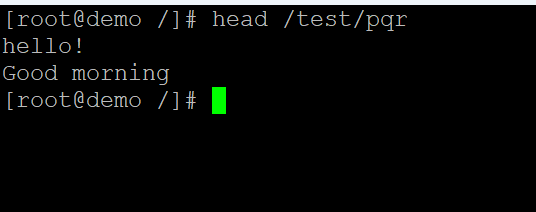
*Syntax: less [filename]*



**12. head**

Displays the first few lines of a file.

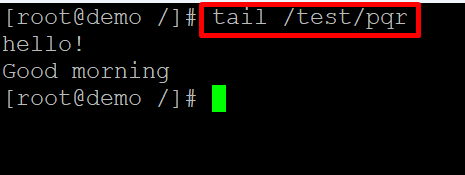
*Syntax: head [filename]*



**13. tail**

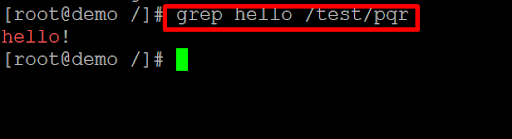
Displays the last few lines of a file.

*Syntax: tail [filename]*



**14. grep**

Searches for a pattern in a file.

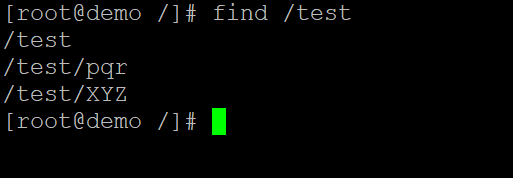


*Syntax: grep [options] [pattern] [filename]*

**15. find**

Searches for files in a directory hierarchy.

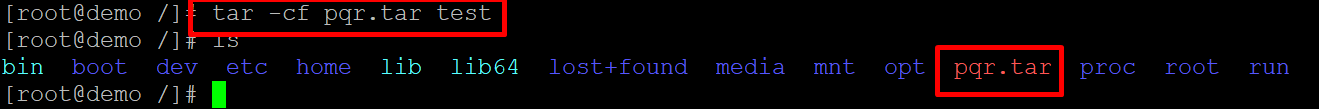
*Syntax: find [directory] [options] [expression]*



**16. tar**

Creates or extracts a compressed archive.

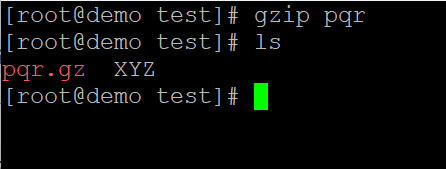
*Syntax: tar [options] [archive-filename] [files/directories]*



**17. gzip**

Compresses a file.

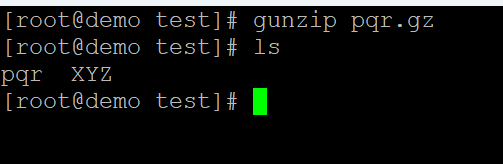
*Syntax: gzip [filename]*



**18. gunzip**

Decompresses a compressed file.

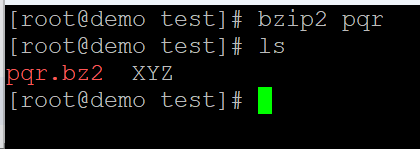
*Syntax: gunzip [filename]*



**19. bzip2**

Compresses a file.

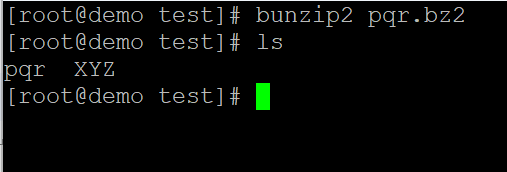
*Syntax: bzip2 [filename]*



**20. bunzip2**

Decompresses a compressed file.

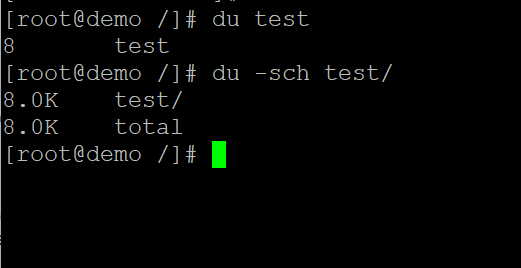
*Syntax: bunzip2 [filename]*



**du**

Shows the disk usage of files and directories.

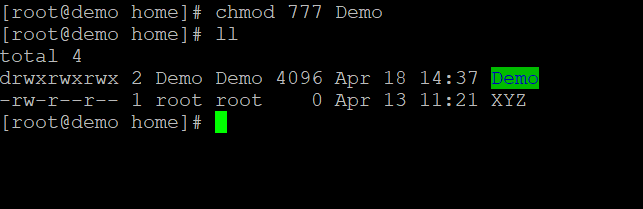
*Syntax: du [options] [directory]*



**df**

Shows the disk space usage of filesystems.

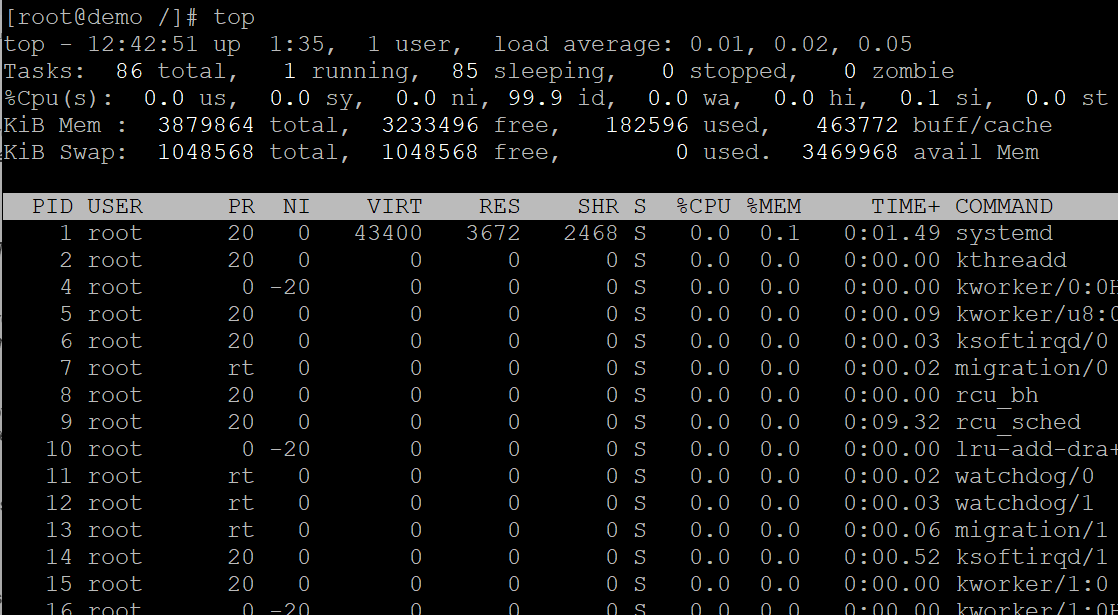
*Syntax: df [options]*



**top**

Displays the current system status.

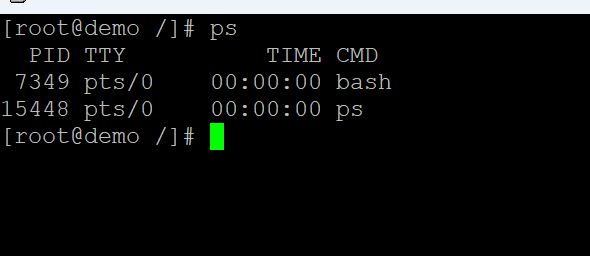
*Syntax: top*



**ps**

Lists the currently running processes.

*Syntax: ps [options]*



**kill**

Sends a signal to a process to terminate it.

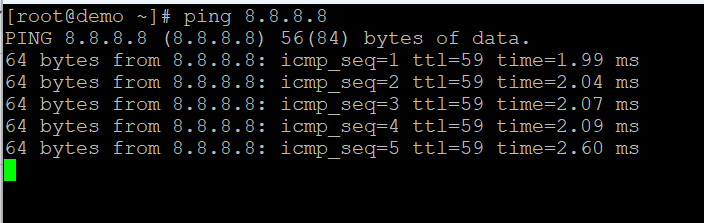
*Syntax: kill [options] [PID]*

kill command

**ping**

Tests the network connectivity to a host.

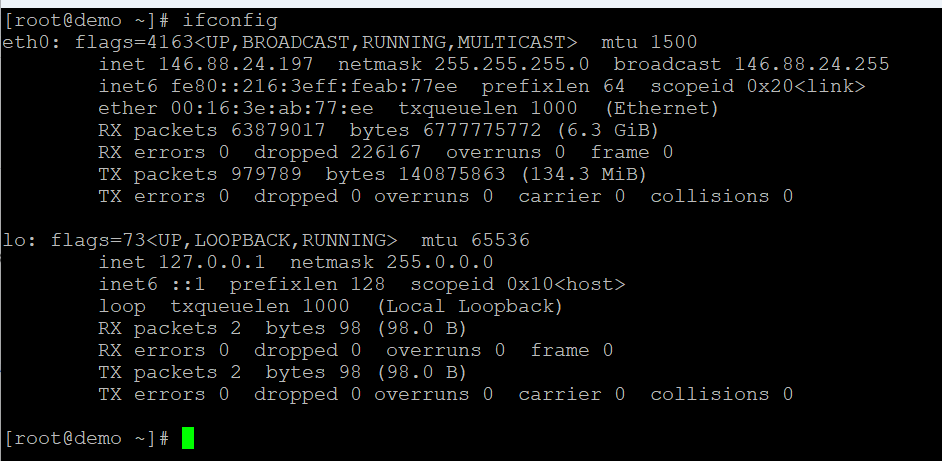
*Syntax: ping [options] [hostname/IP address]*



**ifconfig**

Configures network interfaces.

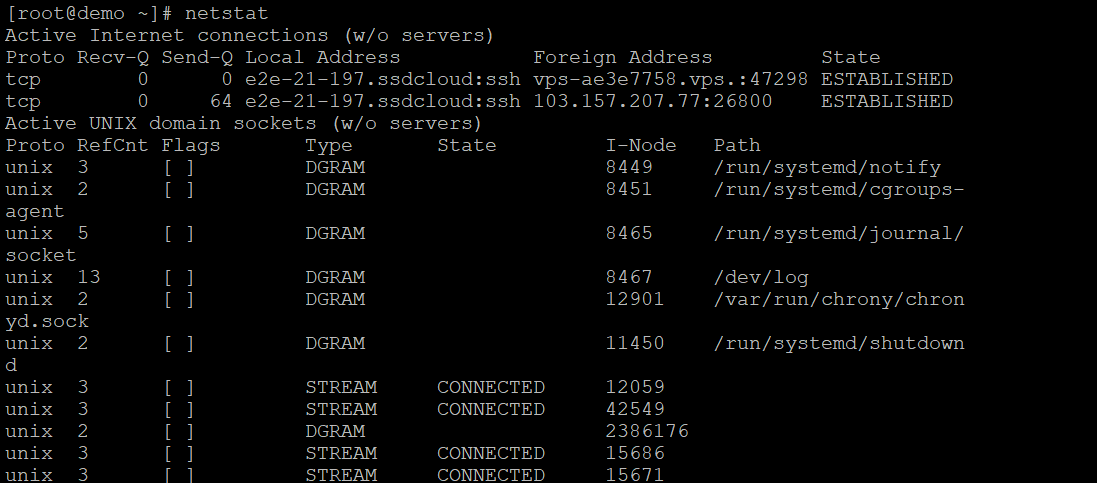
*Syntax: ifconfig [options] [interface]*



**netstat**

Shows network connections, routing tables, and network statistics.

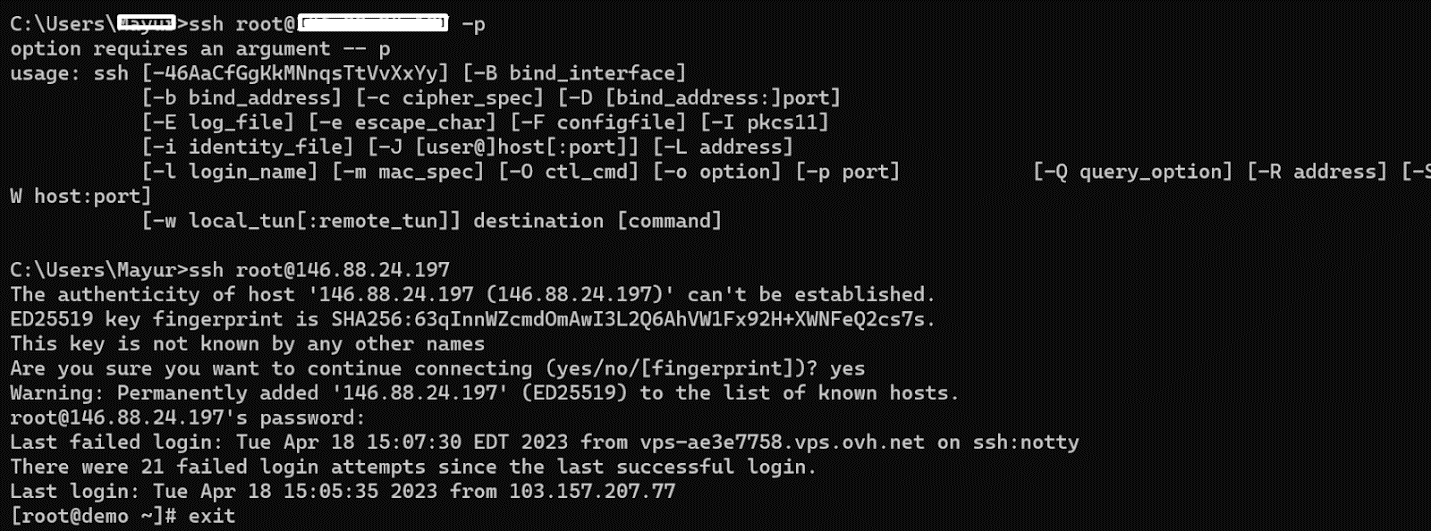
*Syntax: netstat [options]*



**ssh**

Connects to a remote host using SSH.

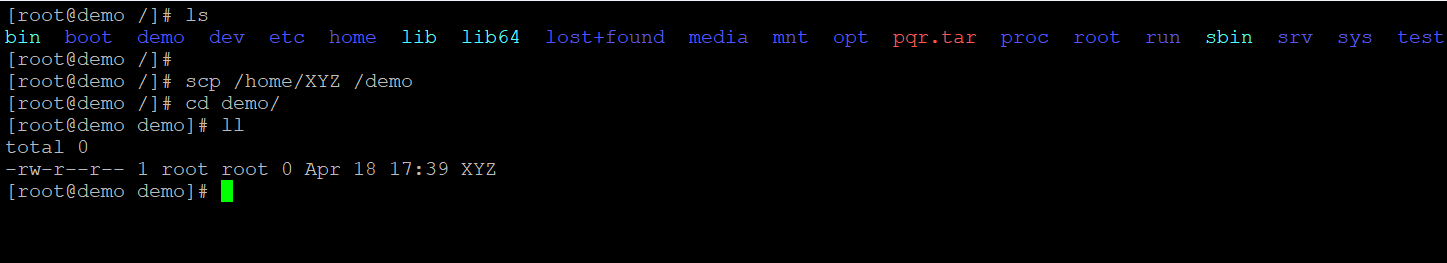
*Syntax: ssh [user@]hostname [command]*



**scp**

Copies files securely between hosts using SSH.

*Syntax: scp [options] [source] [destination]*



**ftp**

Transfers files between hosts using FTP.

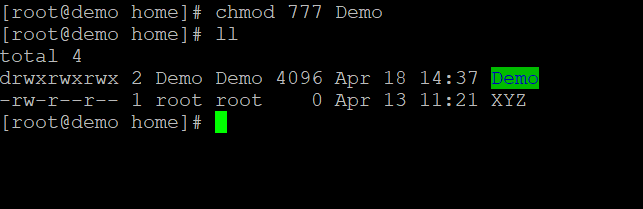
*Syntax: ftp [options] [hostname]*

ftp command

**chmod**

Changes the permissions of files and directories.

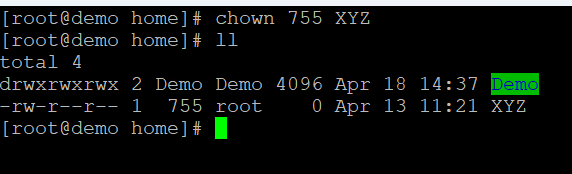
*Syntax: chmod [options] [mode] [file/directory]*



**chown**

Changes the owner of files and directories.

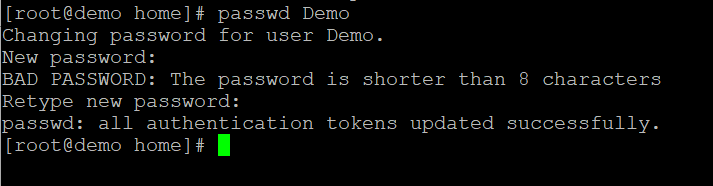
*Syntax: chown [options] [owner:group] [file/directory]*



**passwd**

Changes the password of the current user.

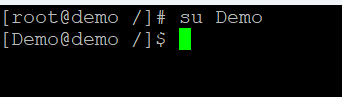
*Syntax: passwd [options] [username]*



**su**

Switches to another user account.

*Syntax: su [options] [username]*



**sudo**

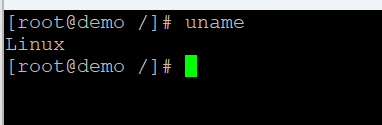
Executes a command with superuser privileges.

*Syntax: sudo [options] [command]*

**uname**

Shows system information.

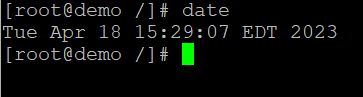
Syntax: uname [options]



**date**

Shows the current date and time.

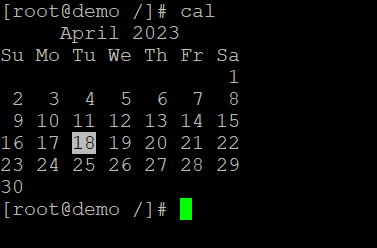
Syntax: date [options]



**cal**

Shows the calendar for the current month.

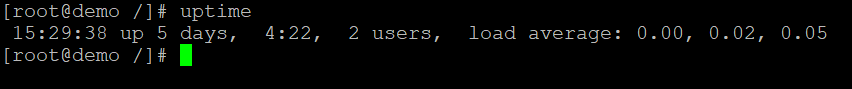
Syntax: cal [options]



**uptime**

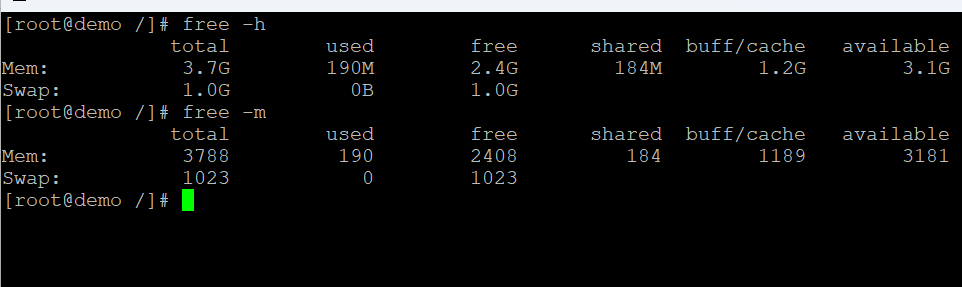
Shows the system uptime and load average.

Syntax: uptime [options]



**free**

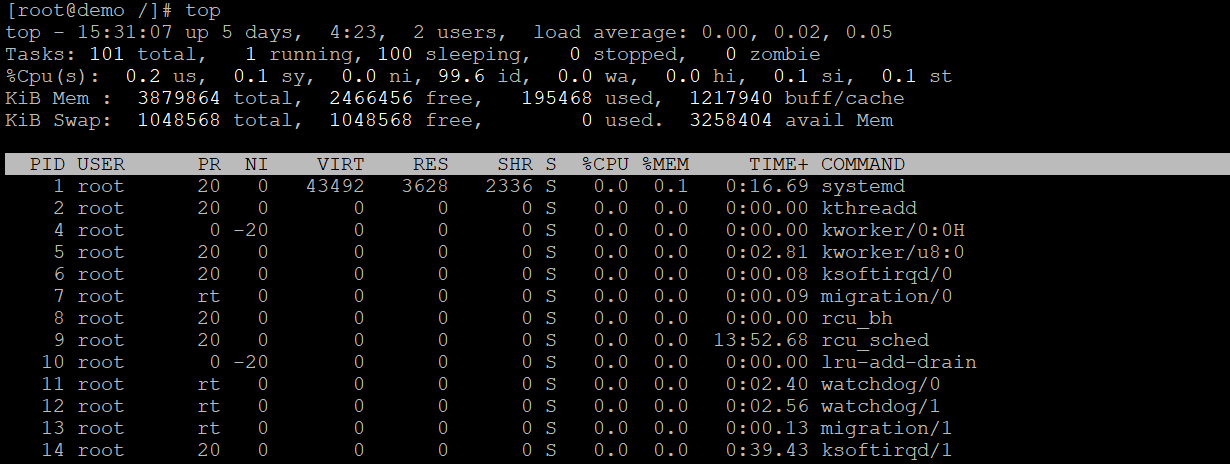
Shows the memory usage.



* **top**

Shows the system resource usage.

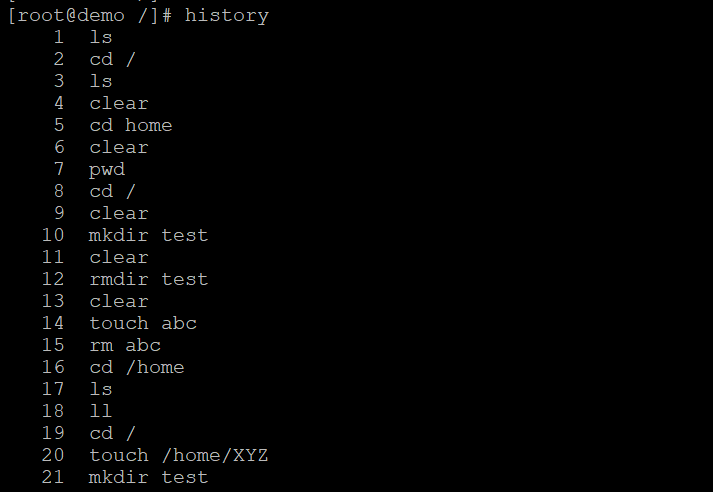
Syntax: top [options]



**history**

Shows the command history.

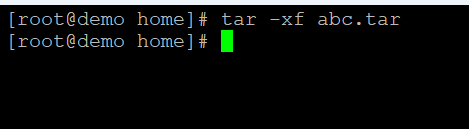
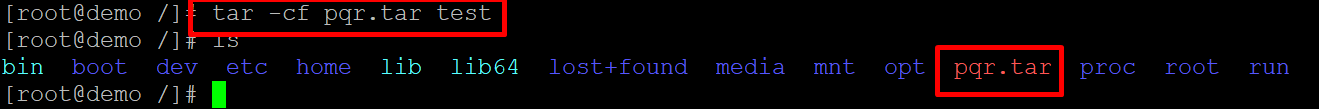
Syntax: history [options]



**tar**

Compresses or extracts files from an archive.

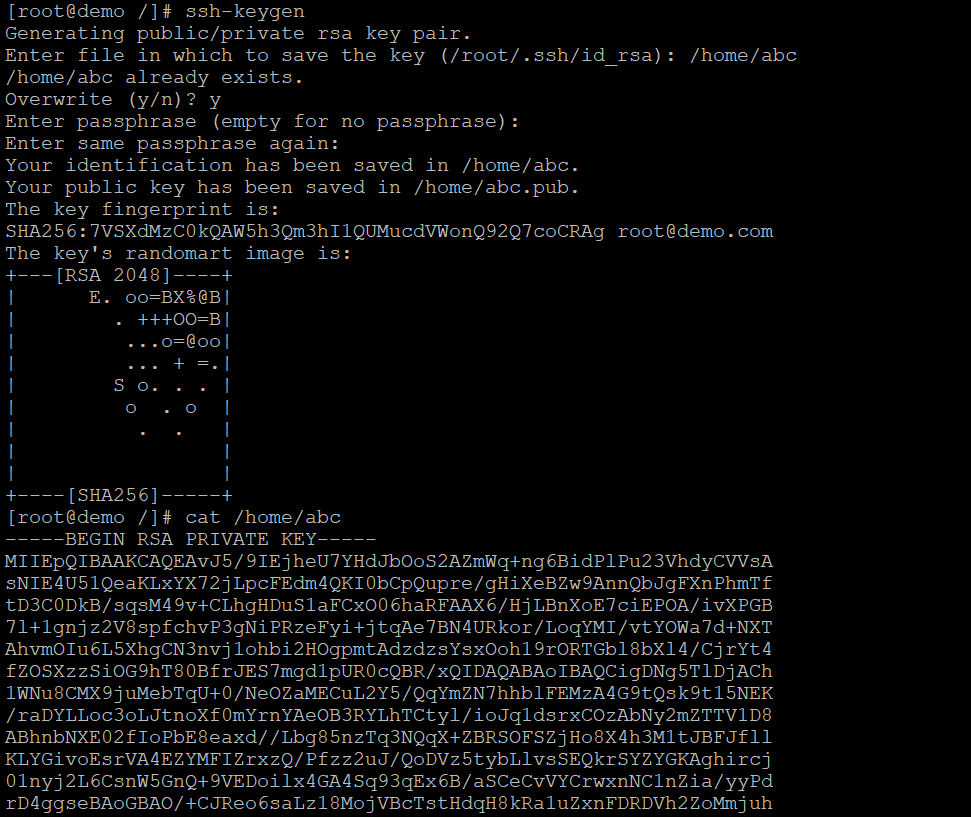
Syntax: tar [options] [archive-filename] [files/directories]



**ssh-keygen**

Generates SSH keys for authentication.

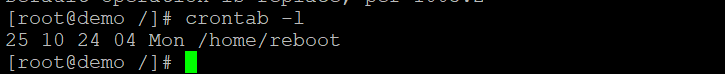
Syntax: ssh-keygen [options] [keyfile]



**crontab**

Schedules commands to run at specified times.

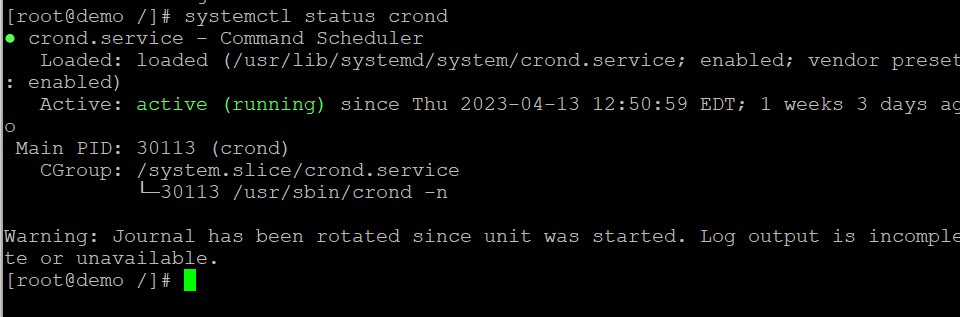
Syntax: crontab [options] [filename]



**systemctl**

Controls the system and service manager.

Syntax: systemctl [options] [command]



**ping6**

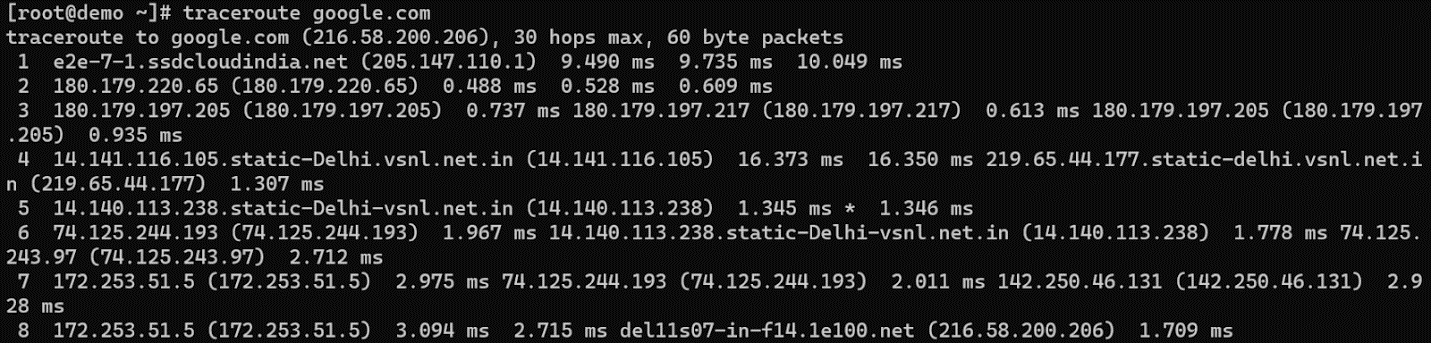
Tests the network connectivity to a host using IPv6.

Syntax: ping6 [options] [hostname/IP address]

**traceroute**

Shows the network path to a host.

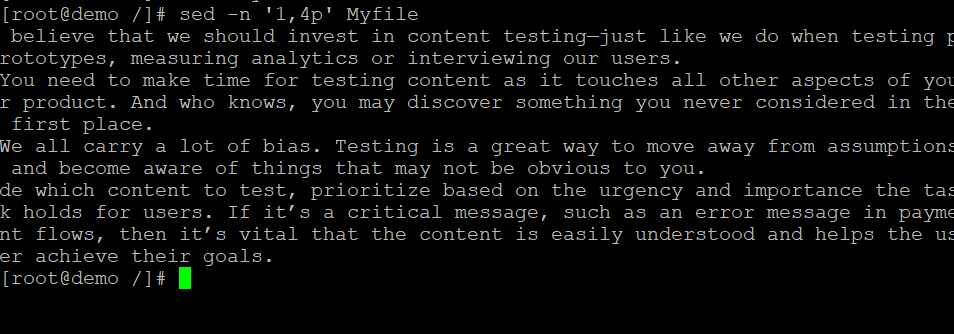
Syntax: traceroute [options] [hostname/IP address]



**sed**

A stream editor for modifying files.

Syntax: sed [options] [script] [filename]



**awk**

A versatile tool for working with text files.

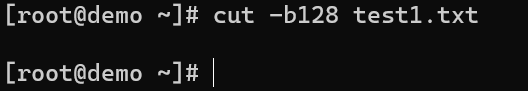
Syntax: awk [options] [script] [filename]

awk command

**cut**

Cuts out sections from a file.

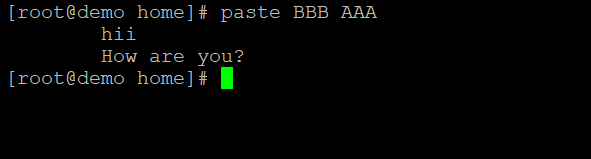
Syntax: cut [options] [filename]



**paste**

Combines lines from multiple files.

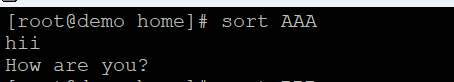
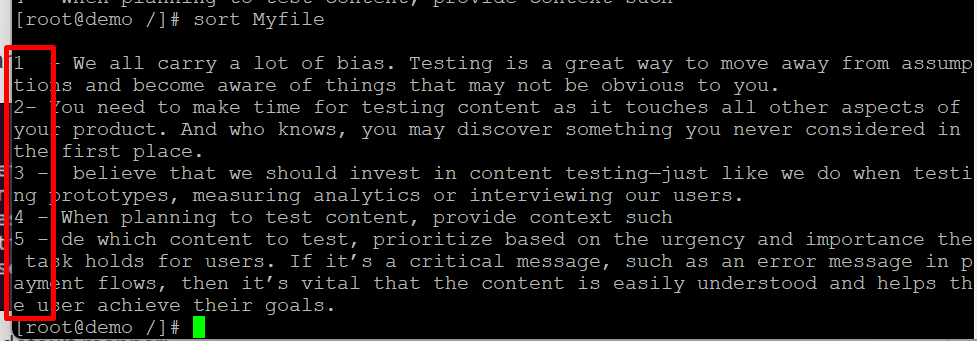
Syntax: paste [options] [filename1] [filename2]



**sort**

Sorts lines of text.

Syntax: sort [options] [filename]



**uniq**

Removes duplicate lines from a file.

Syntax: uniq [options] [filename]

uniq command

**diff**

Compares two files and shows the differences.

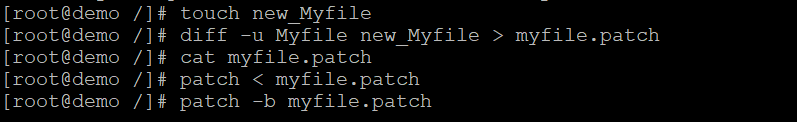
Syntax: diff [options] [file1] [file2]



**patch**

Applies a patch file to a file.

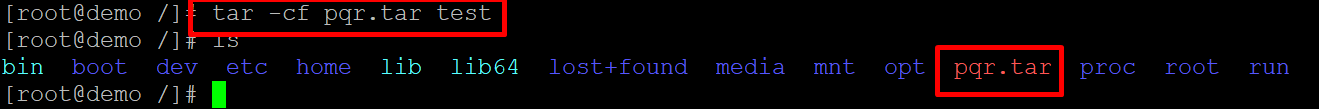
Syntax: patch [options] [original-file] [patch-file]



**tar**

Archives files and directories into a single file.

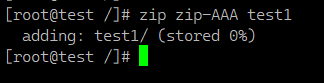
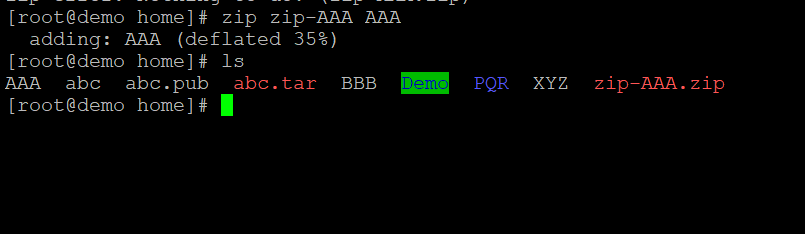
Syntax: tar [options] [archive-filename] [files/directories]



**zip**

Compresses files into a zip archive.

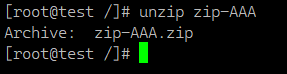
Syntax: zip [options] [zip-filename] [files/directories]



**unzip**

Extracts files from a zip archive.

Syntax: unzip [options] [zip-filename]



**curl**

Transfers data from or to a server.

Syntax: curl [options] [url]

curl command

**wget**

Downloads files from the web.

Syntax: wget [options] [url]

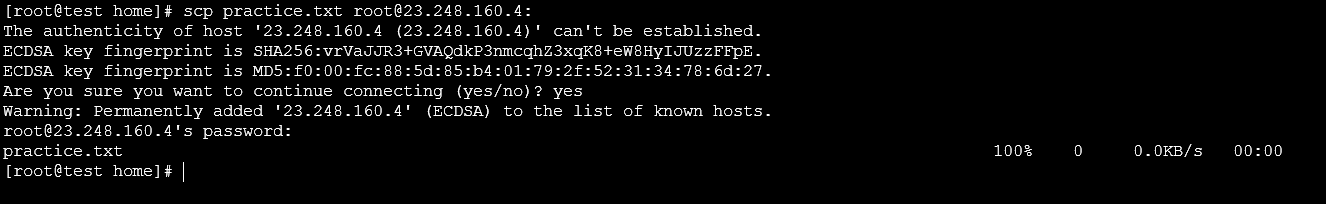
wget -N http://files.virtualizor.com/install.sh

wget command

**scp**

Copies files between hosts securely.

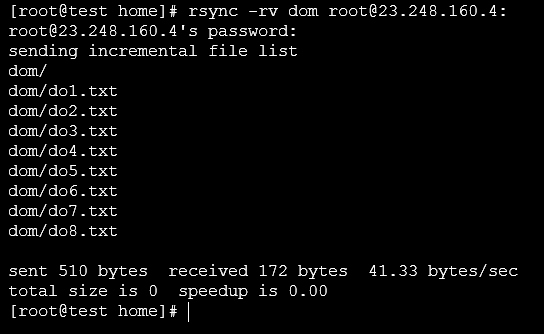
Syntax: scp [options] [source] [destination]



**rsync**

Syncs files and directories between hosts.

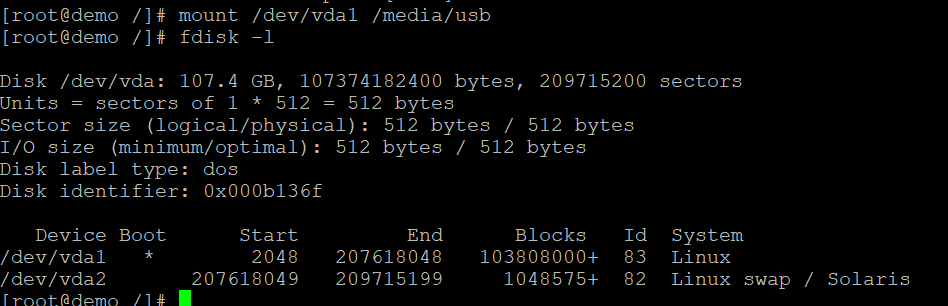
Syntax: rsync [options] [source] [destination]



**mount**

Mounts a filesystem.

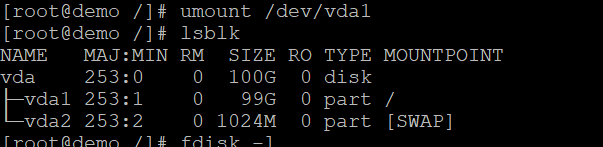
Syntax: mount [options] [device] [mountpoint]



**umount**

Unmounts a filesystem.

Syntax: umount [options] [mountpoint]



**killall**

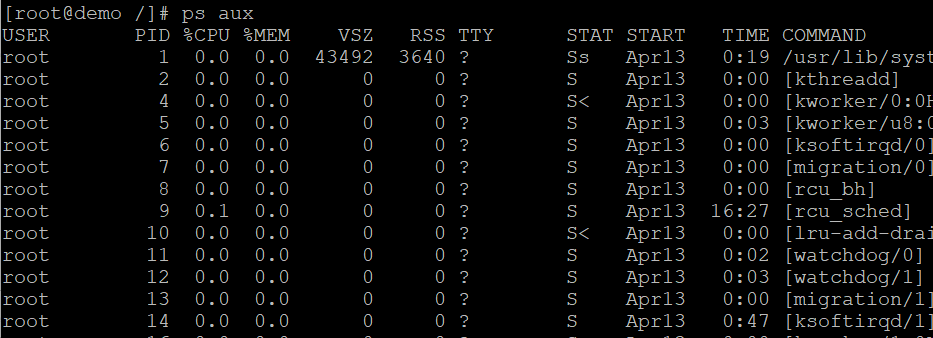
Sends a signal to all processes with a given name.

Syntax: killall [options] [process name]

**ps aux**

Shows all running processes.

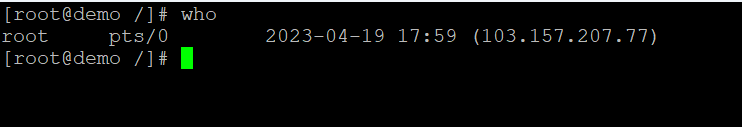
Syntax: ps aux



**who**

Shows all logged-in users.

Syntax: who



shell scripting fundamentals

Shell scripting is a powerful way to automate tasks in a Unix-like operating system, such as Linux. It involves writing a series of commands in a file, which the shell can execute as a program. Here are some fundamentals to get you started:

**What is a Shell Script?**

A shell script is a text file containing a sequence of commands for a Unix-based operating system. It is executed by the shell, which is a command-line interpreter.

**Basic Components of a Shell Script:**

1. **Shebang (**#!**)**: The first line of a script usually starts with #! followed by the path to the shell that will execute the script, e.g., #!/bin/bash.
2. **Comments**: Lines starting with # are comments and are not executed. They are used to explain the code.
3. **Commands**: These are the actual commands that the shell will execute, such as echo, ls, cd, etc.
4. **Variables**: Used to store data that can be used later in the script. For example, name="John".
5. **Control Structures**: Includes loops (for, while) and conditionals (if, else).

**Writing Your First Shell Script:**

1. **Create a File**: Use a text editor to create a file, e.g., script.sh.
2. **Add Shebang**: Add #!/bin/bash at the top of the file.
3. **Write Commands**: Add your commands below the shebang. For example:

#!/bin/bash

echo "Hello, World!"

1. **Make the Script Executable**: Use the command chmod +x script.sh to make the script executable.
2. **Run the Script**: Execute the script by typing ./script.sh in the terminal.

**Example Script:**

Here’s a simple script that prints a message and lists files in the current directory:

#!/bin/bash

# This is a simple shell script

echo "Starting the script..."

echo "Listing files in the current directory:"

ls

echo "Script completed."

Fullstack: The Big Picture

**What is Full-Stack Development?**

Full-stack development refers to the development of both the front end and back end portions of a web application. [A full-stack developer is proficient in both areas and can handle the entire development process, from designing user interfaces to managing databases and server-side logic](https://dev.to/olanetsoft/full-stack-web-development-the-big-picture-4fn8).

**Front-End Development**

The front end is the part of the application that users interact with directly. It involves:

* **HTML**: The structure of web pages.
* **CSS**: Styling and layout.
* **JavaScript**: Interactivity and dynamic content.
* [**Frameworks/Libraries**: Tools like React, Angular, and Vue.js that help streamline development](https://dev.to/olanetsoft/full-stack-web-development-the-big-picture-4fn8)[1](https://dev.to/olanetsoft/full-stack-web-development-the-big-picture-4fn8).

**Back-End Development**

The back end is the server side of the application, responsible for:

* **Server Management**: Handling requests and responses.
* **Database Management**: Storing and retrieving data.
* **Server-Side Languages**: Languages like Node.js, Python, Ruby, Java, and PHP.
* [**Frameworks**: Tools like Express.js, Django, and Ruby on Rails that facilitate back-end development1](https://dev.to/olanetsoft/full-stack-web-development-the-big-picture-4fn8)

**Full-Stack Developer Skills**

A full-stack developer needs to be proficient in:

* **Front-End Technologies**: HTML, CSS, JavaScript, and related frameworksa.
* **Back-End Technologies**: Server-side languages, databases (SQL and NoSQL), and server management.
* **Version Control**: Tools like Git for managing code changes.
* [**DevOps**: Basic knowledge of deployment, CI/CD, and cloud services](https://dev.to/olanetsoft/full-stack-web-development-the-big-picture-4fn8).