**<https://medium.com/@rahuldesharaj92/basic-linux-commands-used-by-devops-engineers-in-day-to-day-activities-29593253f66b>**

**https://vinodhakumara2681997.medium.com/top-linux-commands-and-tricks-for-devops-tasks-42cf93aa77e0**

[**https://medium.com/cloud-native-daily/mastering-linux-for-devops-engineers-essential-commands-and-practices-for-success-a608a718069f**](https://medium.com/cloud-native-daily/mastering-linux-for-devops-engineers-essential-commands-and-practices-for-success-a608a718069f)

[**https://medium.com/@tech\_18484/essential-linux-commands-and-for-devops-a-hands-on-guide-6e59ed4e3b92**](https://medium.com/@tech_18484/essential-linux-commands-and-for-devops-a-hands-on-guide-6e59ed4e3b92)

[**https://medium.com/@skmswetha22/linux-for-devops-engineers-9b51d4168f9b**](https://medium.com/@skmswetha22/linux-for-devops-engineers-9b51d4168f9b)

**https://vinodhakumara2681997.medium.com/cheatsheet-linux-commands-for-devops-80be32b88656**

[**https://medium.com/@thesureshyadav76/the-ultimate-guide-to-linux-networking-commands-for-devops-professionals-bc7681d87b04**](https://medium.com/@thesureshyadav76/the-ultimate-guide-to-linux-networking-commands-for-devops-professionals-bc7681d87b04)

[**https://medium.com/@medurubharathstvd/linux-roadmap-c6f82ca771f0**](https://medium.com/@medurubharathstvd/linux-roadmap-c6f82ca771f0)

[**https://medium.com/@thesureshyadav76/the-ultimate-guide-to-linux-networking-commands-for-devops-professionals-bc7681d87b04**](https://medium.com/@thesureshyadav76/the-ultimate-guide-to-linux-networking-commands-for-devops-professionals-bc7681d87b04)

[**https://razorops.com/blog/linux-commands-that-every-devops-engineer-should-know**](https://razorops.com/blog/linux-commands-that-every-devops-engineer-should-know)

**OS:-100 Linux Commands that need to be used for DevOps Engineers/Linux Administrator and Cloud Engineer**

[[](https://medium.com/@rajkumarsingh07?source=post_page---byline--294f9508f030---------------------------------------)](https://medium.com/@rajkumarsingh07?source=post_page---byline--294f9508f030---------------------------------------)

[Rajkumar Singh](https://medium.com/@rajkumarsingh07?source=post_page---byline--294f9508f030---------------------------------------)

·

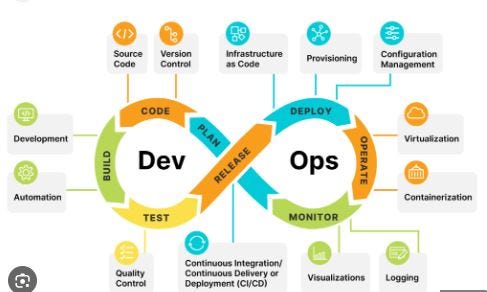
Following

3 min read

·

Oct 8, 2023

10



coped from google.com

In the world of DevOps, a profound familiarity with Linux commands is indispensable. These commands empower us to effortlessly traverse file systems, wield control over files and directories, mold data, and orchestrate an array of pivotal system operations. In this exposition, we embark on a journey to uncover the quintessential Linux commands that are indispensable for every adept DevOps Engineer.

**File and Directory Operations:**

1. **pwd**: Print Working Directory.
2. **ls**: List files and directories.
3. **cd**: Change Directory.
4. **touch**: Create an empty file.
5. **mkdir**: Create a directory.
6. **rmdir**: Remove an empty directory.
7. **rm**: Remove files or directories.
8. **cp**: Copy files or directories.
9. **mv**: Move or rename files/directories.
10. **find**: Search for files and directories.
11. **tree**: Display directory structure.
12. **ln**: Create symbolic links.

**File Viewing and Manipulation:**

1. **cat**: Concatenate and display file content.
2. **more**/**less**: View files page by page.
3. **head**/**tail**: Display the beginning/end of a file.
4. **grep**: Search for text in files.
5. **sed**: Stream Editor — Text manipulation.
6. **awk**: Text processing and pattern matching.
7. **sort**: Sort lines of text files.
8. **wc**: Word, line, character, and byte count.

**User and Permissions:**

1. **useradd**/**userdel**: Add/delete user accounts.
2. **passwd**: Change user passwords.
3. **groupadd**/**groupdel**: Add/delete user groups.
4. **id**: Display user and group information.
5. **chmod**: Change file permissions.
6. **chown**: Change file ownership.
7. **chgrp**: Change group ownership of files.
8. **sudo**: Execute commands with superuser privileges.

**Process and System Management:**

1. **ps**: List running processes.
2. **top**: Monitor system processes.
3. **kill**: Terminate processes.
4. **systemctl**: Manage system services.
5. **df**: Display disk space usage.
6. **du**: Display directory space usage.
7. **free**: Display system memory usage.
8. **uptime**: Display system uptime.
9. **nice**: Set process priority.

**Networking:**

1. **ifconfig**/**ip**: Display network configuration.
2. **netstat**: Network statistics.
3. **ss**: Socket statistics.
4. **ping**: Test network connectivity.
5. **traceroute**: Trace the route packets take to a network host.
6. **telnet**: Connect to remote hosts.
7. **ssh**: Securely connect to remote servers.
8. **scp**: Securely copy files between hosts.
9. **curl**/**wget**: Download files from the internet.

**Logs and Troubleshooting:**

1. **journalctl**: View system logs.
2. **tail**/**grep** log files for troubleshooting.
3. **dmesg**: Display kernel messages.
4. **strace**: Trace system calls and signals.
5. **lsof**: List open files and processes.
6. **nc**: Netcat — Network utility for reading/writing to network connections.
7. **tcpdump**: Network packet analyzer.

**Package Management:**

1. **yum**/**dnf**: Package management on RHEL-based systems.
2. **rpm**: Package management tool for RPM packages.
3. **yum list**/**dnf list**: List installed packages.
4. **yum install**/**dnf install**: Install packages.
5. **yum remove**/**dnf remove**: Remove packages.
6. **yum update**/**dnf update**: Update packages.
7. **yum search**/**dnf search**: Search package repositories.

**Text Processing and Manipulation:**

1. **awk**: Text processing and pattern matching.
2. **sed**: Stream Editor — Text manipulation.
3. **cut**: Remove sections from lines of files.
4. **join**: Join lines of two files on a common field.
5. **split**: Split a file into pieces.
6. **paste**: Merge lines of files.
7. **rev**: Reverse lines character-wise.
8. **uniq**: Remove duplicate lines from sorted files.
9. **diff**: Compare files line by line.

**Compression and Archiving:**

1. **tar**: Tape Archive — Compress and archive files.
2. **gzip**/**gunzip**: Compress/decompress files.
3. **bzip2**/**bunzip2**: Compress/decompress files using bzip2.
4. **zip**/**unzip**: Create/extract ZIP archives.
5. **xz**/**unxz**: Compress/decompress files using xz.

**Monitoring and Resource Usage:**

1. **iotop**: Monitor I/O usage by processes.
2. **atop**: Advanced system performance monitor.
3. **vmstat**: Virtual Memory Statistics.
4. **sar**: System activity report.
5. **nmon**: Performance monitoring tool.
6. **iftop**: Network bandwidth monitoring.

**Users and Groups:**

1. **who**: Display who is logged in.
2. **w**: Display who is logged in and what they are doing.
3. **last**: Show last logged in users.
4. **whoami**: Display the current username.
5. **su**: Switch user or become a superuser.
6. **groups**: List user’s group memberships.
7. **newgrp**: Change the effective group ID.
8. **id**: Display user and group information.
9. **chpasswd**: Change passwords in batch mode.

**Disk and Filesystem Operations:**

1. **mount**/**umount**: Mount/Unmount filesystems.
2. **fdisk**: Disk partitioning utility.
3. **parted**/**gparted**: Partition editor.
4. **mkfs**: Create a filesystem.
5. **du**: Display directory space usage.
6. **ncdu**: Disk usage analyzer with an ncurses interface.
7. **sync**: Synchronize cached writes to disk.
8. **badblocks**: Check for bad blocks on a storage device.
9. **quota**: User and group disk usage quotas.
10. **df**: Display disk space usage.
11. **system-config-firewall**: Configure the firewall settings.

I will run some of the above commands on a Linux terminal and share the details in the next post. Please follow me to get the latest update on this.

Thanks

<https://www.geeksforgeeks.org/linux-command-in-devops/>

# Linux commands for DevOps (Commands used in day-to-day activities).

[[](https://www.linkedin.com/in/chetanrakhra/)](https://www.linkedin.com/in/chetanrakhra/)

## [Chetan R](https://www.linkedin.com/in/chetanrakhra/)

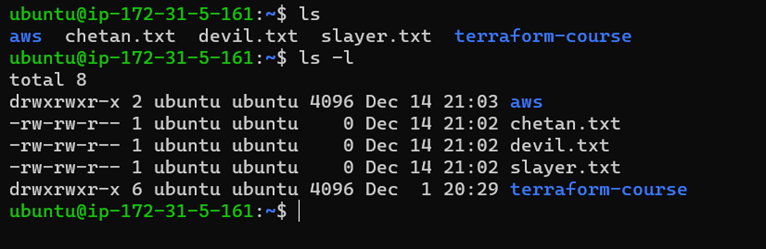
29k+ LinkedIn | DevOps Engineer | Running OkDevOps | Docker | Linux | Jenkins | Kubernetes | AWS | Git | Terraform | Open for Collaborations | Ex-HCLite | Recommended

December 16, 2022

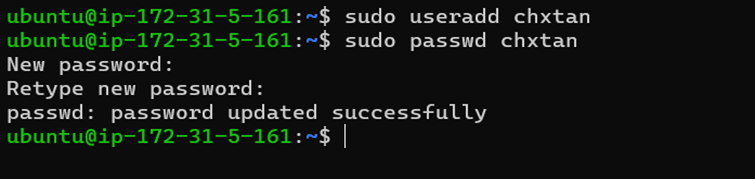
**Mostly Used Commands in DevOps**:

1. **Ls:** This command is used to list the contents of a directory.

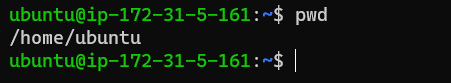
if you run ls -l, it will show the permissions, owner, size, and last modified date for each file in the directory.



2. **Sudo:** This command executes only that command with superuser privileges.



3. **Pwd:** This command will print your directory location, where you currently working.



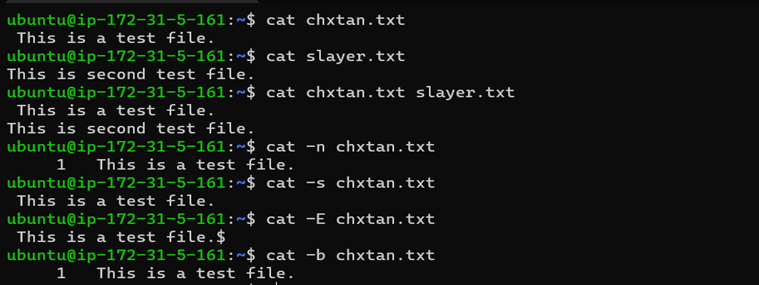
4. **Cat:** This is used to concatenate and display files on the terminal. It can also be used to modify existing ones.

***cat -b:*** This adds line numbers to non-blank lines

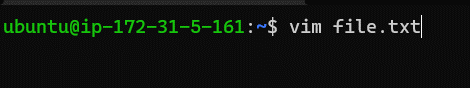
***cat -n:*** This adds line numbers to all lines

***cat -s:*** This squeezes blank lines into one line

***cat –E:*** This shows $ at the end of the line



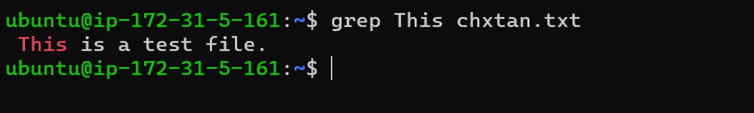
5. **Vim:** This is a text editor used in Linux. It stands for “Vi Improved”.



Mostly used modes in VIM:

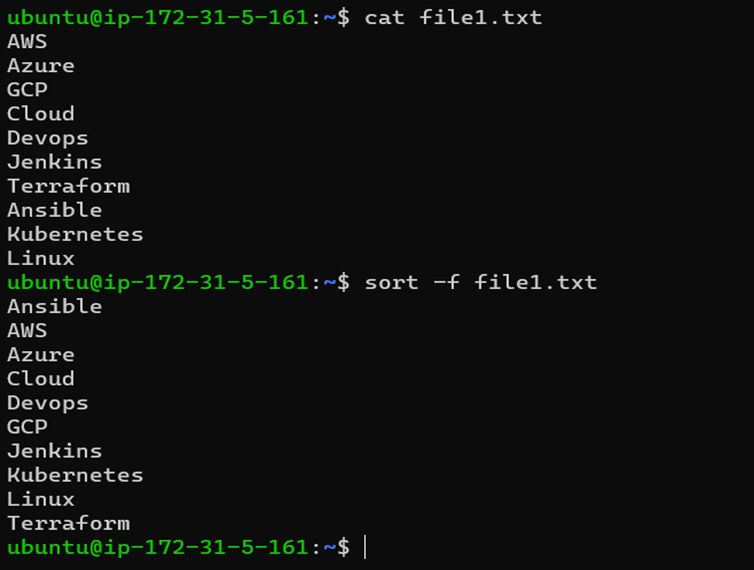
* **Normal mode:** This is the default mode in which vim starts. In normal mode, you can use various commands to navigate and edit the text.
* **Insert mode:** In insert mode, you can type text into the file. To enter insert mode, press the "i" key. To exit insert mode and return to normal mode, press the "Esc" key.
* **Command mode:** In command mode, you can enter commands to perform various actions, such as saving the file or quitting vim. To enter command mode, press the ":" key.

6. **Grep:** This command searches for a particular string/ word in a text file. This is similar to “Ctrl+F” but executed via a CLI.



This would print all of the lines in “chxtan.txt” that contain the word “This".

7. **Sort:** This command is used to sort the results of search either alphabetically or numerically. It also sorts files and directories.



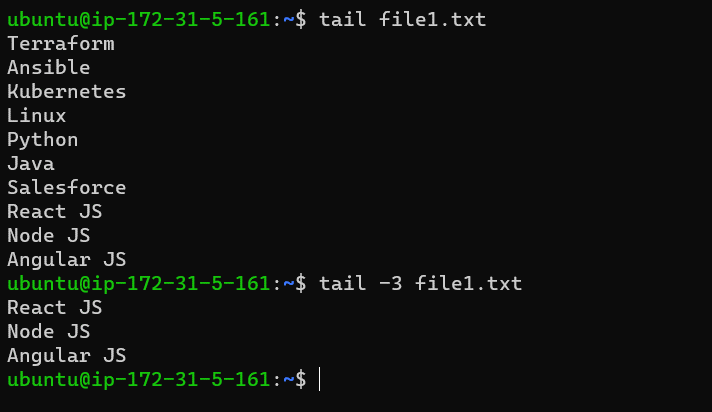
***sort -r:*** the flag returns the results in reverse order.

***sort -f:*** the flag does case-insensitive sorting.

***sort -n:*** the flag returns the results as per numerical order.

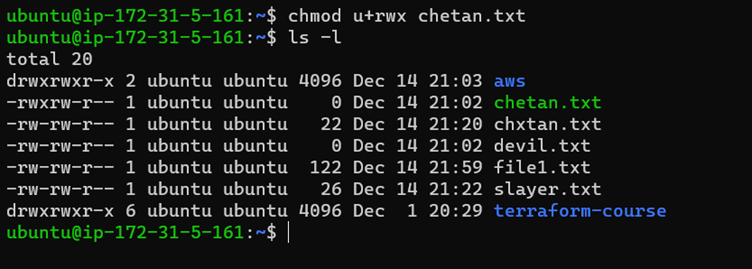
8. **Tail:** This command prints the last N number of data of the given input. By default, it prints 10 lines.

We can specify the number of lines, that we want to display.



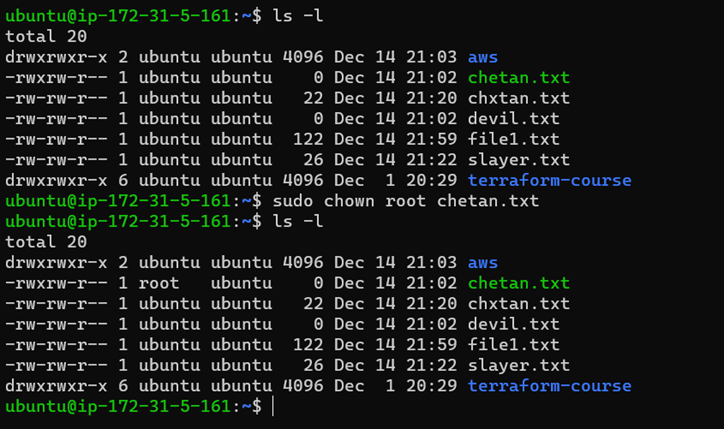
9. **Chmod:** This command is used to change the access permissions of files and directories.

For example: Following “chmod” command will give the user permission to read, write and execute a file.

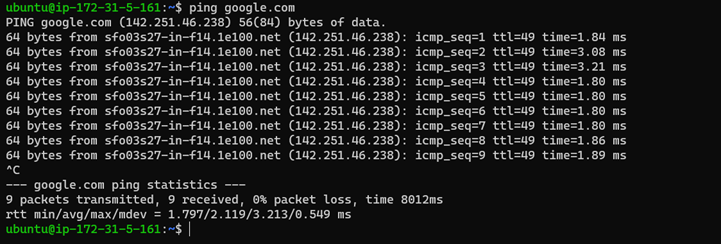


10. **Chown:** This command is used to change the file Owner or group.

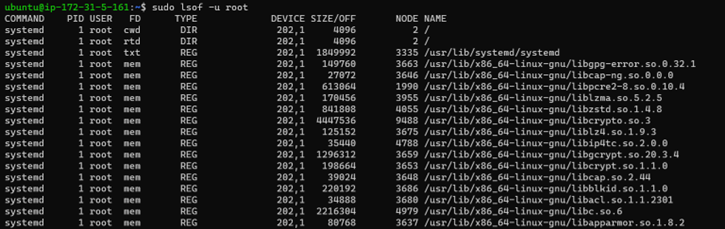
Here, below the ownership of “Chetan.txt” file got changed to root.



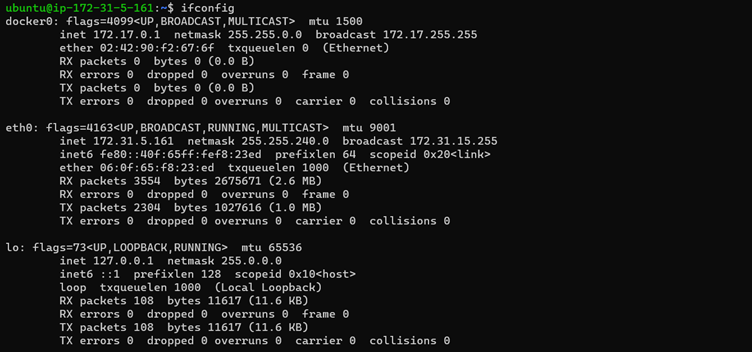
11. **Ping –** This command will ping a host and check if it is responding.



12. **Lsof:** It is used to display a list of all the open files on a Linux system.

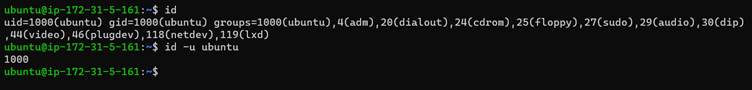


13. **Ifconfig:** This is used to configure the kernel-resident network interfaces.



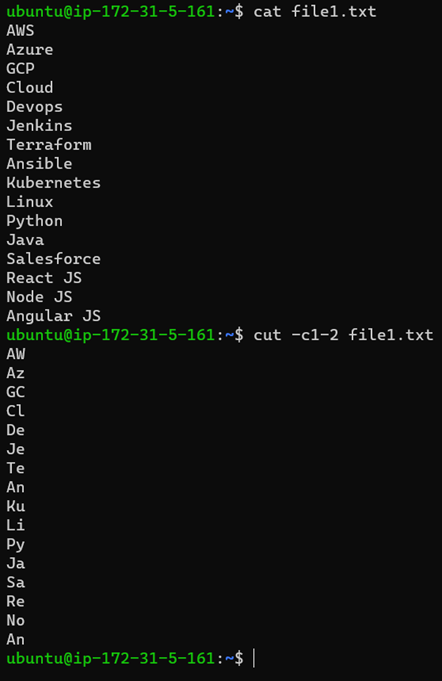
14. **ID:** This is used to find out user and group names and numeric ID’s (UID or group ID) of the current user or any other user in the server.

**Syntax:** id <option> <user\_name>

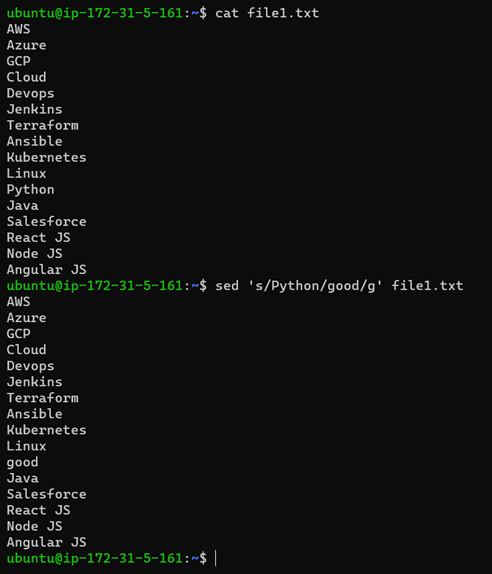


15. **Cut:** This command is used to extract specific fields or columns from a file or standard input.

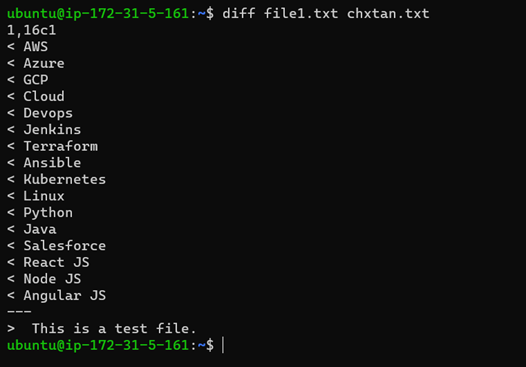
It is often combined with other commands, such as sort, uniq, and grep, to perform more complex text-processing tasks.



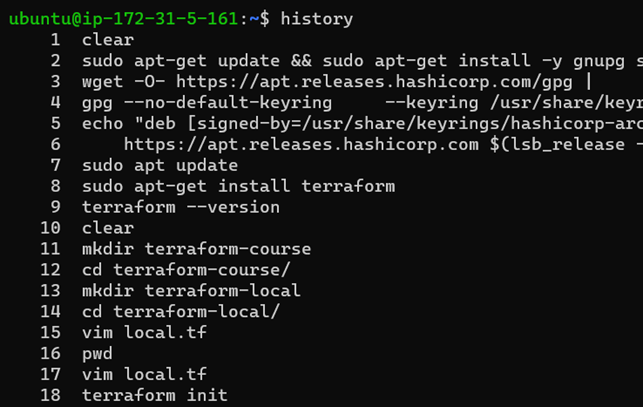
16. **Sed:** This is used to perform basic text transformations on an input file. It stands for "stream editor" and is a powerful tool for editing text files or streams in a Linux environment.



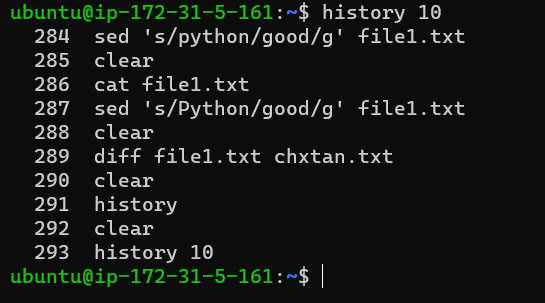
17. **Diff:** This command is used to find the difference between two files.



18. **History:** This command is used to view the previously executed command.

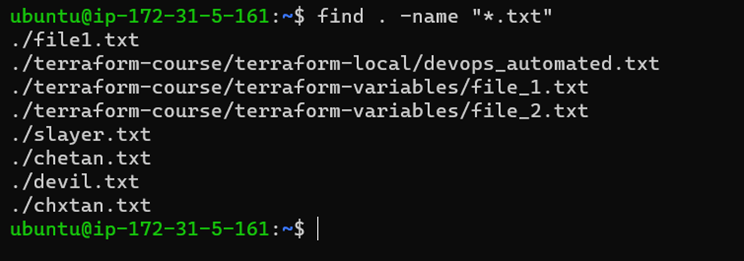


“History 10” – Will give you the last 10 executed commands.



19. **Find:** This is used to find files and directories and perform subsequent operations on them.

In the below command, It will search in the present working directory and its subdirectories, and print the name of the file that have “.txt” file extension.



20. **Free:** This command displays the total amount of free space available along with the amount of memory used and swap memory in the system, and also the buffers used by the kernel.

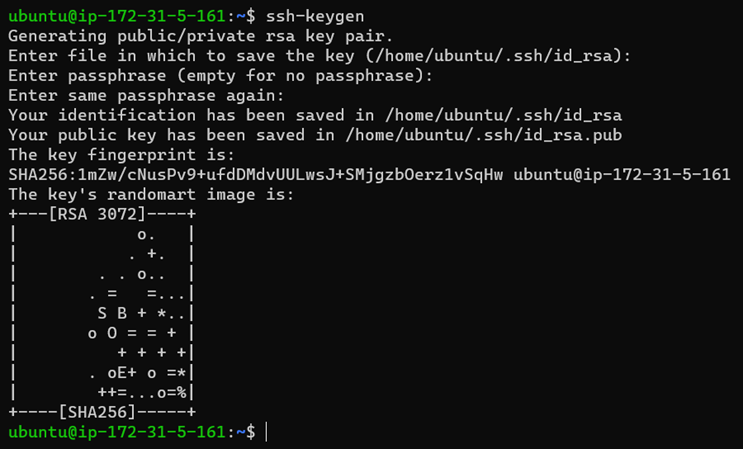


21. **ssh user@host –** connect to the host as a user.

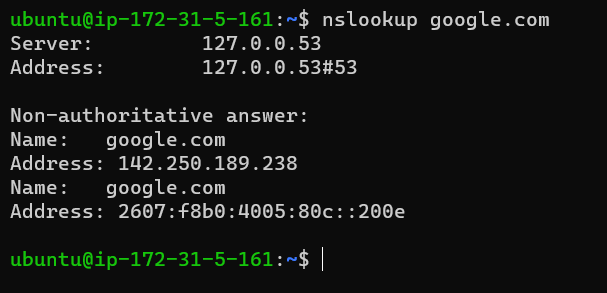
22. **Ssh-keygen:** This command is used to generate a public/private authentication key pair.

This process of authentication allows the user to connect remote server without providing a password.

(This authentication method we will use to authenticate the server with Jenkins while deploying CI/CD pipelines).

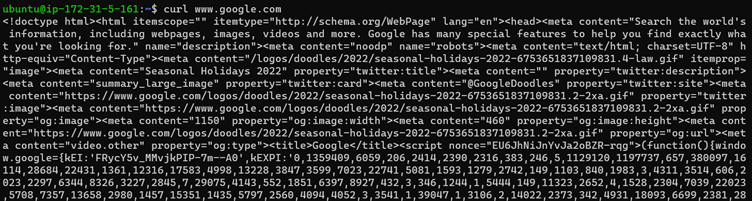


23. **Nslookup:** This stands for “Name server Lookup”. This is a tool for checking DNS hostname to Ip or Ip to Hostname. This is very helpful while troubleshooting.



24. **Curl:** Curl is a tool used for transferring data to or from a server, using various protocols, such as HTTP, HTTPS, FTP, and more. Basic example:

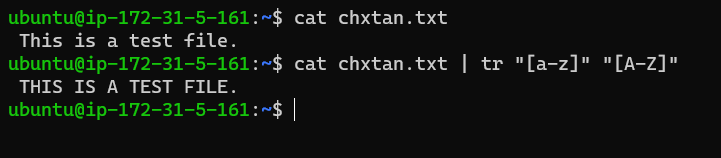
**Syntax:** curl <url>



**-o:** It will save downloaded file on the local machine with the name provided in parameters.



25. **Tr: Tr** stands for translation. This command is for translating or deleting characters.



26. **Apt-get:** This command is used to install, update, and remove packages, as well as to manage the package repository sources.

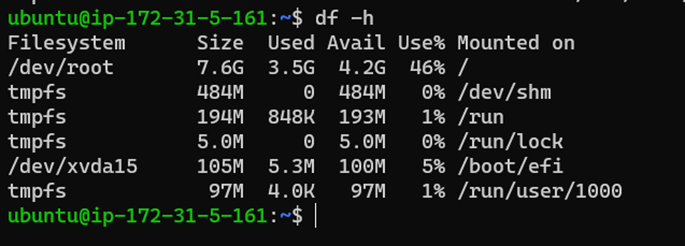
Here are some common apt-get commands:

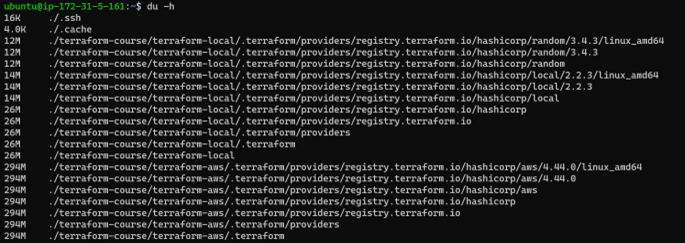
* apt-get update: This updates the package index files from the package repositories listed in the /etc/apt/sources.list file. This is usually the first command you should run after adding a new repository or package to your system.
* apt-get upgrade: This installs newer versions of packages that are already installed on the system. It will also remove any packages that are no longer required.
* apt-get install: This installs one or more packages. For example, to install the nano text editor, you would run apt-get install nano.
* apt-get remove: This removes one or more packages, but it does not remove the configuration files for the package.
* apt-get purge: This removes one or more packages and their configuration files.
* apt-get autoremove: This removes packages that were installed as dependencies but are no longer needed.

27. **Df, du:**

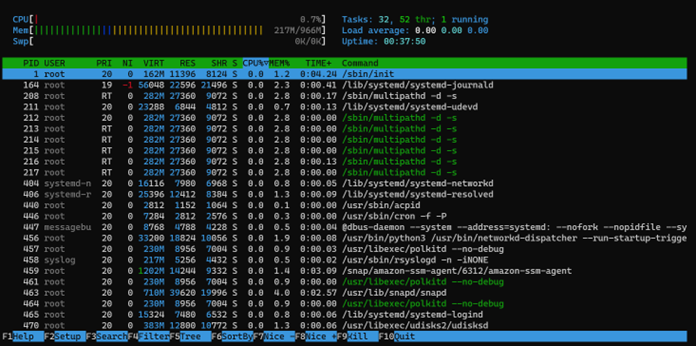
Df (disk free) command will have an account of available disk space, used by file system.

Du (disk usage) command reports the size of directory tree including all the content.



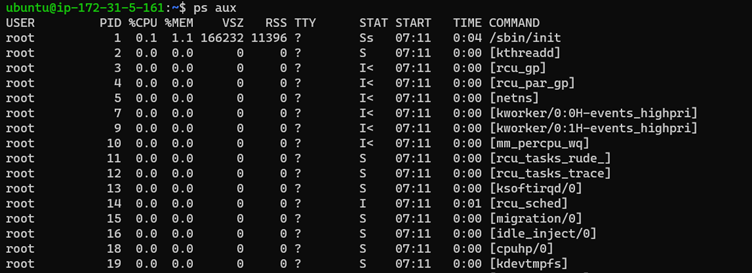


28. **Htop:** This command is used to monitor the system’s resources and processes that are running in real time.

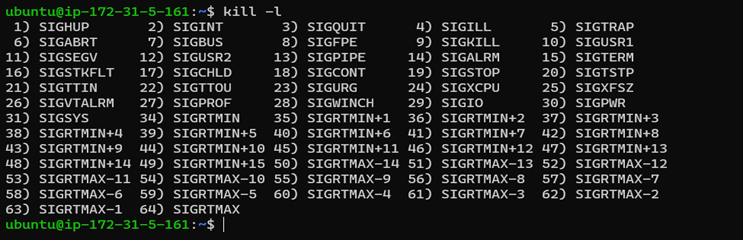


29. **Ps:** We use ps command to check the unique id behind every process.

* **a** = show processes for all users
* **u** = display the process’s user/owner
* **x** = also show processes not attached to a terminal



30. **Kill:** This command is used to terminate processes manually. This command basically, will send a signal that terminates it.



31. **TNC:** This is “Test Network Connection” command. Mostly used command while troubleshooting. It displays diagnostic information for a connection.

**Command:** tnc <server\_name> -port <port>

<https://medium.com/@osagienelson24/essential-scripts-for-devops-engineer-86ddc27b5c1d>

# Basic Linux commands used by DevOps engineers in day-to-day activities

[[](https://medium.com/@rahuldesharaj92?source=post_page---byline--29593253f66b---------------------------------------)](https://medium.com/@rahuldesharaj92?source=post_page---byline--29593253f66b---------------------------------------)

[Rahul Desharaj](https://medium.com/@rahuldesharaj92?source=post_page---byline--29593253f66b---------------------------------------)

·

Following

7 min read

·

Apr 25, 2024

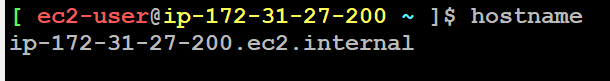
485

4

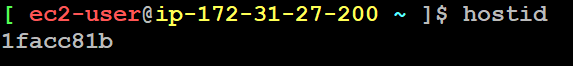
This article will help in understanding most of the important and majorly used Linux commands that would be required for a DevOps Engineer.

# System Info Commands :-

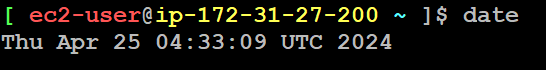
**hostname**- shows the name of the system host.



**hostid**- shows the host id of the system assigned by the OS



**date**- shows the current date and time in UTC format.



**whoami**- shows the currently logged-in username of the terminal.

https://miro.medium.com/v2/resize:fit:586/1*MV8xrXAFJgKKfDtzndgBYQ.png

**uptime**- shows the elapsed time duration since the machine logged in.

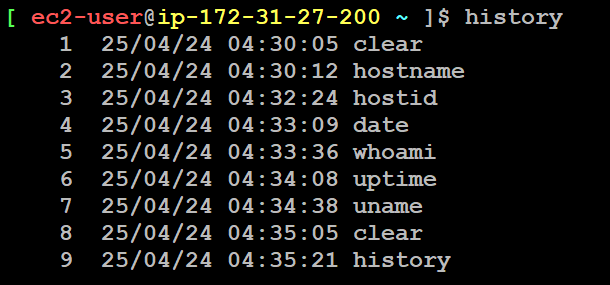
https://miro.medium.com/v2/resize:fit:700/1*i40eFwJTgqpelycULGoUHw.png

**uname**- unix name.

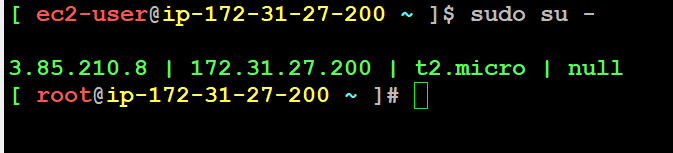
https://miro.medium.com/v2/resize:fit:610/1*QJxZEImCZ6D09T8JoaMcHg.png

**clear**- clears the screen.

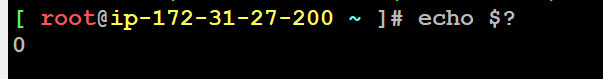
**history**- lists all the commands executed until now.



**sudo** - Super User Do.



**echo $?** - shows the exit status of the last executed command (0 - success, 1–127 - error/failure).



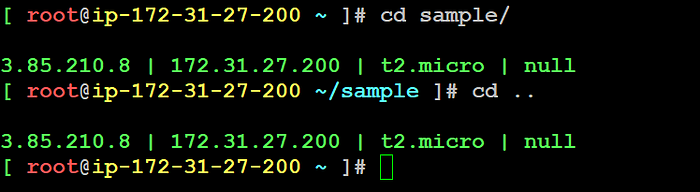
# Directory Commands :-

**pwd**- To check where you are currently in the system we use pwd command.

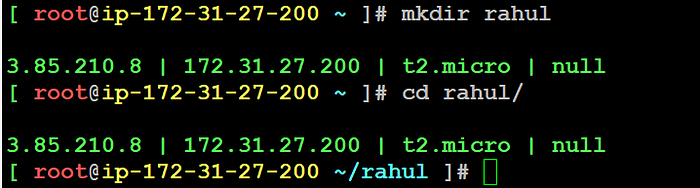
https://miro.medium.com/v2/resize:fit:504/1*DwbtA3ep4gNoqvO6FK8JBA.png

**cd**To change the working directory from one location to another we use cd command

**cd ..**In Linux, we refer the parent directory with .. , So in case if we want to go to parent path then simply we can go with.



**mkdir**- make directory.

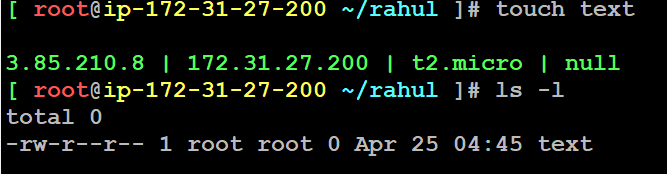


# File Commands :-

**touch**This Command by default creates an empty file.

https://miro.medium.com/v2/resize:fit:675/1*SlPadtfr5zZv2at40w1cFw.png

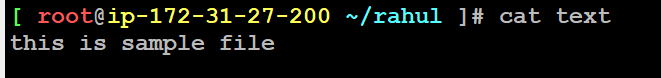
**ls -l**To check the file created.



**vim**: This is a text editor used in Linux. It stands for “Vi Improved”.

* Normal mode: This is the default mode in which vim starts. In normal mode, you can use various commands to navigate and edit the text.
* Insert mode: In insert mode, you can type text into the file. To enter insert mode, press the “i” key. To exit insert mode and return to normal mode, press the “Esc” key.
* Command mode: In command mode, you can enter commands to perform various actions, such as saving the file or quitting vim. To enter command mode, press the “:” key.

**cat**- concatenates and displays the contents of files.

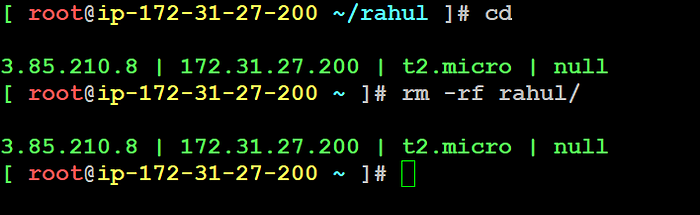


**rm**- remove command.

* **rm -f <fileName>** - removes the file.

https://miro.medium.com/v2/resize:fit:661/1*PSG_GdnxbtaTn4AV9B4TjQ.png

* **rm -rf <dirName>** - force remove the files & folders of directory recursively (-f force).

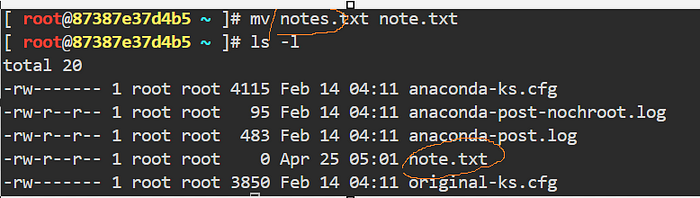


**cp**- copy command.

**cp <source> <destination>** - copy the files and folders from source to destination.

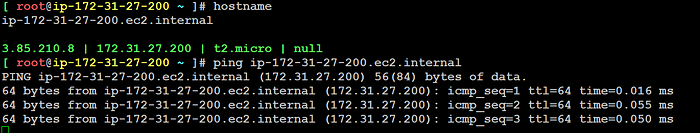
**mv**- move or rename command.

**mv <fileName> <newFileName>** - renames the file to new name.

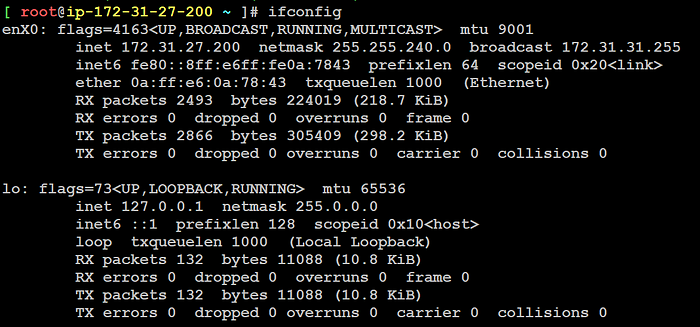


# Network Commands :-

**ping <hostName>** - tests the reachability & responsiveness of the remote host.



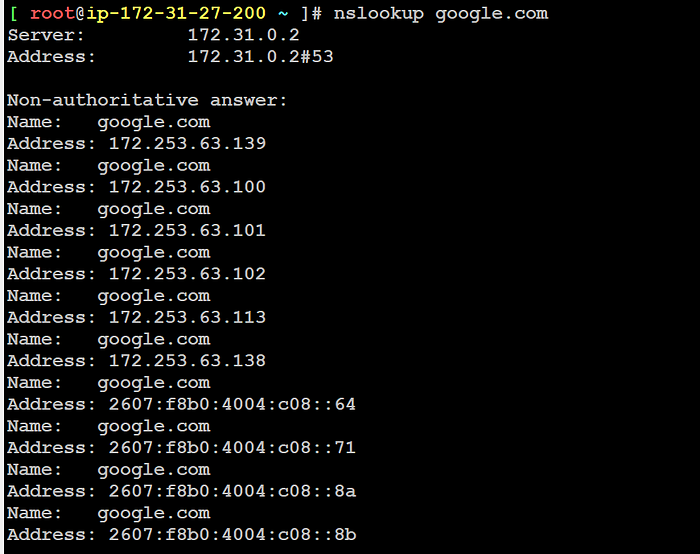
**ifconfig**- display available network interfaces.



**netstat -lntp**- shows all tcp open ports (-a all, t-tcp, n-active, p protocol).

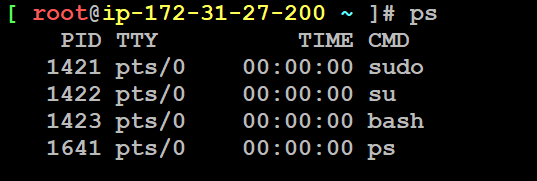


**Nslookup**: This stands for “Name server Lookup”. This is a tool for checking DNS hostname to Ip or Ip to Hostname. This is very helpful while troubleshooting.

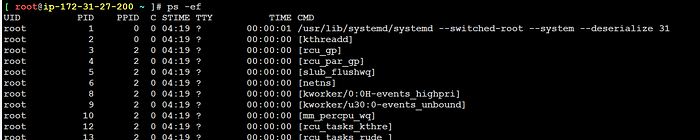


# Process Information Commands :-

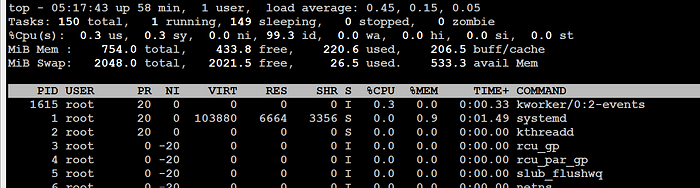
* **ps**- shows the currently running process.



* **ps -ef** - shows all the processes of the system.



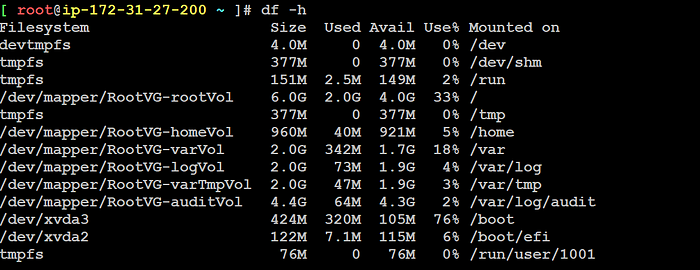
* **top**- shows the real-time, dynamic view of the running processes of a system.



* **kill <pid>** - gracefully terminates the process pid.

https://miro.medium.com/v2/resize:fit:577/1*3LcjjKAJMbf48s7U9NW5_A.png

* **df -h**(disk free) command will have an account of available disk space, used by file system.

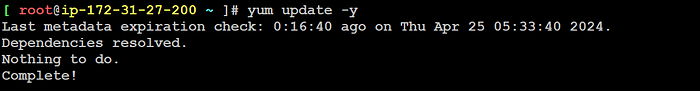


# Red Hat Package related Commands

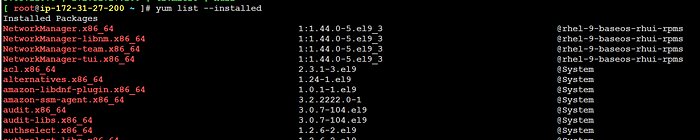
Yum - Package Manager for RHEL Linux distributions.

* **yum**- a newer version of the package manager with colorized output, progress bar and additional functions.

**yum update -y** - updates the package list.

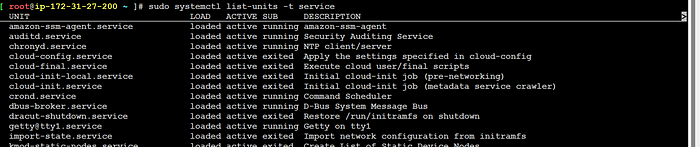


**yum list --installed** - lists all the installed packages.



# Service Management

**sudo systemctl list-units -t service** : To list all the services which are running in the Operating System



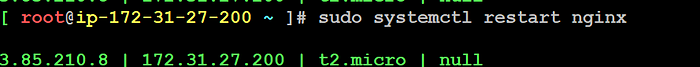
**sudo systemctl start nginx** : To start a service  
**syntax**: **sudo systemctl start <service name>**

https://miro.medium.com/v2/resize:fit:700/1*B2DYzNbvbsHFuIeklJz8rg.png

**sudo systemctl status nginx :**check the status of the service



**sudo systemctl restart nginx :**To restart a service



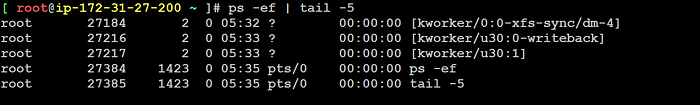
**Other Commands :-**

**Grep**: This command searches for a particular string/ word in a text file. This is similar to “Ctrl+F” but executed via a CLI.

https://miro.medium.com/v2/resize:fit:700/1*khiVmnOco1_nZZ1y_H7KyQ.png

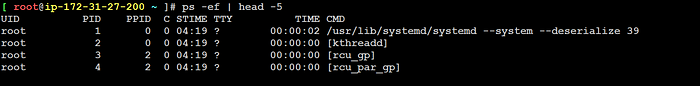
**Tail**: This command prints the last N number of data of the given input. By default, it prints 10 lines.

We can specify the number of lines, that we want to display.

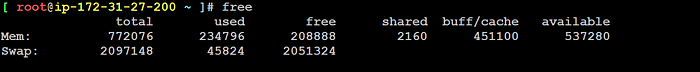


**Head**: This command prints the first N number of data of the given input. By default, it prints 10 lines.

We can specify the number of lines, that we want to display.

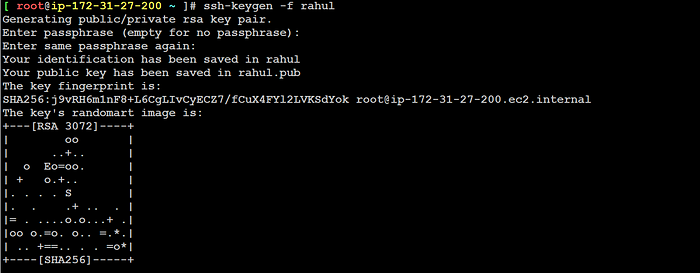


**Free**: This command displays the total amount of free space available along with the amount of memory used and swap memory in the system, and also the buffers used by the kernel.



**ssh-keygen**: This command is used to generate a public/private authentication key pair.

This process of authentication allows the user to connect remote server without providing a password.



**CURL**: Curl is a tool used for transferring data to or from a server, using various protocols, such as HTTP, HTTPS, FTP, and more. Basic example:

Syntax: curl <url>

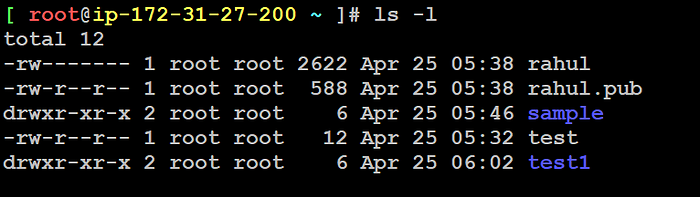


**tar**: is used to create, maintain, modify, and extract files from archives, often called “tarballs”. It can bundle multiple files and directories into a single archive file, preserving permissions, timestamps, and directory structures.

**cron and crontab**: cron is a system daemon used to execute scheduled tasks at predefined times. crontab is the command-line interface to manage these scheduled tasks for individual users.  
Example Usage:  
Editing user’s crontab: crontab -e  
Specifying a cron job to run every hour: 0 \* \* \* \* /path/to/[script.sh](http://script.sh/)

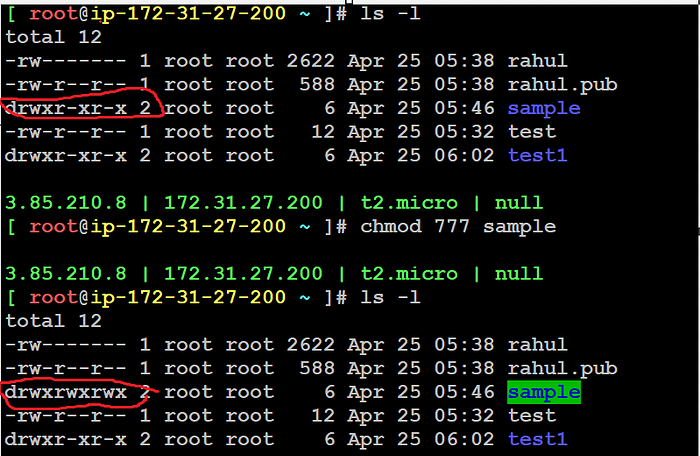
# File Permission Octal Numbers





**chmod <octalNumber> <fileName>** - changes mode/permissions of the file.

Syntax : chmod 777 test1.txt



This blog page covers most of the majorly used Linux commands for DevOps Warriors and is a growing document. Commands will be added as the days go and when I find some interesting commands on the same.

Do follow me for more such content related to DevOps world.

**Reach me 👨‍💻**

* Linkedin -[🟦](https://www.linkedin.com/in/rahul-desharaju-299b872a7/)

# Top Linux Commands And Tricks For DevOps Tasks

[[](https://vinodhakumara2681997.medium.com/?source=post_page---byline--42cf93aa77e0---------------------------------------)](https://vinodhakumara2681997.medium.com/?source=post_page---byline--42cf93aa77e0---------------------------------------)

[Vinodha kumara](https://vinodhakumara2681997.medium.com/?source=post_page---byline--42cf93aa77e0---------------------------------------)

·

Following

10 min read

·

Jan 19, 2024

1,125

5



This article will help in understanding most of the important and majorly used Linux commands that would be required for a DevOps Engineer.

To execute these commands one can either use any Linux machine / virtual machine / online Linux terminal to quickly start working with the commands.

As a DevOps professional, mastering the Linux command line is essential for efficient server management, automation, and troubleshooting. This comprehensive guide explores essential Linux commands that every DevOps user should know. Each command is accompanied by a clear explanation and practical examples to help deepen your Linux proficiency.

# System Info Commands

**hostname**- shows the name of the system host.

➜ ~ hostname  
localhost

**hostid**- shows the host id of the system assigned by the OS.

➜ ~ hostid  
0a123456

**date**- shows the current date and time in UTC format.

➜ ~ date  
Wed Jan 19 12:34:56 UTC 2024

**uptime**- shows the elapsed time duration since the machine logged in.

➜ ~ uptime  
12:34:56 up 1 day, 3:45, 2 users, load average: 0.25, 0.20, 0.18

**uname**- unix name.

➜ ~ uname  
Linux

**clear**- clears the screen.

➜ ~ clear

**history**- lists all the commands executed until now.

➜ ~ history  
 1 ls  
 2 cd Documents  
 3 nano file.txt  
 4 gcc program.c -o program  
 5 ./program  
 6 history

**sudo** - Super User Do.

➜ ~ sudo su - USERNAME

**echo $?** - shows the exit status of the last executed command (0 - success, 1-255 - error/failure).

➜ ~ echo $?  
127

**shutdown -r now** - restart the machine immediately (-r restart).

➜ ~ sudo shutdown -r now  
Broadcast message from user@hostname  
 (/dev/pts/0) at 12:34 ...  
  
The system is going down for reboot NOW!

**printenv**- displays all the environment variables of the Linux system.

➜ ~ printenv  
TERM=xterm-256color  
SHELL=/bin/bash  
USER=your\_username  
...

**last**- shows previous logins in the Linux system.

➜ ~ last  
root pts/0 Wed Jan 19 12:34 still logged in  
reboot system boot 5.4.0-96-generic Wed Jan 19 12:33 still running

**systemctl**— System Control: Manage system services using systemd.

➜ ~ systemctl status sshd  
● sshd.service - OpenBSD Secure Shell server  
 Loaded: loaded (/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)  
 Active: active (running) since Wed 2024-01-19 12:34:56 UTC; 1 day 3h ago  
 Docs: man:sshd(8)  
 man:sshd\_config(5)  
 Process: 1234 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)  
 Main PID: 5678 (sshd)  
 Tasks: 1 (limit: 1234)  
 Memory: 2.3M  
 CPU: 12ms  
 CGroup: /system.slice/sshd.service  
 └─5678 /usr/sbin/sshd -D  
  
Jan 19 12:34:56 hostname systemd[1]: Starting OpenBSD Secure Shell server...  
Jan 19 12:34:56 hostname sshd[5678]: Server listening on 0.0.0.0 port 22.  
Jan 19 12:34:56 hostname sshd[5678]: Server listening on :: port 22.  
Jan 19 12:34:56 hostname systemd[1]: Started OpenBSD Secure Shell server.

# File Commands

**touch**- creates an empty file or updates timestamp of the existing file.

* **touch <fileName>** - creates a single empty file.
* **touch <file1> <file2>** - creates file1, file2 empty files.

**cat**- concatenates and displays the contents of files.

* **cat <fileName>** - displays the contents of the file.
* **cat > <fileName>** - creates a new file, allows to input content interactively and redirects inputted content to the created file (> redirection operator).

**head <fileName>** - displays first 10 lines of the file by default.

* **head -n 5 <fileName>** - displays first 5 lines of the file (-n number)

➜ ~ head -n 5 help.txt  
 1. Commands shortcut  
....  
 5. huddle - Connect to Syncup Call

**tail <fileName>** - displays the last 10 lines of the file by default.

* **tail -F <fileName>** - displays contents of the file in real-time even when the file is rotated or replaced (used for log file monitoring).

➜ ~ tail -F mySystem.logs  
echo "I love DevOps"  
echo "Best Linux commands"  
....

**less <fileName>** - used to view large files (log files) in a paginated manner.

**rm**- remove command.

* **rm <fileName>** - removes the file.
* **rm -r <dirName>** - removes files & folders of directory recursively (-r recursive).
* **rm -rf <dirName>** - force remove the files & folders of directory recursively (-f force).
* Example: rm -r ./test

**cp**- copy command.

* **cp <source> <destination>** - copy the files and folders from source to destination.
* **cp -r <dir1> <dir2>** - copy dir1 directory to dir2 directory recursively (-r recursive).
* Example: cp -r ./sourceDir ./destiDir

# File Permission Commands

**ls -l <pathOfFileName>** - shows the permissions of the file.

➜ ~ ls -l .  
total 1016  
-rw-r--r-- 1 vinodhakumaral staff 48 Jan 19 21:06 crazy.sh  
-rw-r--r-- 1 vinodhakumaral staff 2463 Jan 2 11:25 help  
-rw-r--r-- 1 vinodhakumaral staff 48 Jan 19 22:14 mySystem.logs  
drwxr-xr-x@ 8 vinodhakumaral staff 256 Dec 20 12:51 observability-signoz

**ls -ld <dirNamePath>** - shows the permissions of the directory.

➜ ~ ls -ld Downloads  
drwx------@ 53 vinodhakumaral staff 1696 Jan 19 21:00 Downloads

**chmod <octalNumber> <fileName>** - changes mode/permissions of the file.

* Example: chmod 742 test.txt

**chmod <octalNumber> -R <dirName>** - changes mode/permissions of the directory recursively.

**chown <newUser> <fileName>** - changes the user ownership of a file.

* Example: chown rocky test.txt

**chown <newUser>:<newGroup> <fileName>** - changes the user & group ownerships of a file.

**chgrp <groupName> <fileName/dirName>** - updates the group name for file/directory.

* Example: chgrp mygroup ./test

**getfacl <fileName/dirName>** - shows the file/directory access control list.

➜ ~ getfacl filename.txt  
# file: filename.txt  
# owner: user1  
# group: group1  
user::rw-  
group::r--  
other::r--

**setfacl -m u:<userName>:rwx <fileName/dirName>** - modifies the current acl of the file/directory.

**setfacl -x u:<userName>: <fileName/dirName>** - removes the acl permissions for the file/directory.

**setfacl -m g:<groupName>:rwx <fileName/dirName>** - modifies the group acls for the file/directory.

**setfacl -x g:<groupName>: <fileName/dirName>** - removes the group acl permissions for the file/directory.

# File Permission Octal Numbers

read (r) — 4, write (w)- 2, execute (x) — 1

* Sum the numbers to generate an octal number for setting permissions on a file or directory.

# User Management Commands

[**ac**](https://man7.org/linux/man-pages/man7/man-pages.7.html)**—**Total connect time for all users or specified users.

* The ac command reads the /var/log/wtmp file, which contains binary data about every login, logout, system event, and current status on the system. It gets its data from the wtmp file.
* Display total login time of a specific user.  
  **ac john**
* Display total login time for each user.  
  **ac -p**
* Display total login time for each day.  
  **ac -d**
* Display total login time for the current day.  
  **ac -d -p**
* Display login time from a specific log file.  
  **ac -f /var/log/wtmp**

**useradd**- Creates a user account.

* **useradd <userName>** - Creates user account without home & mail spool directories.
* Example: useradd bot
* **useradd -m <userName>** - Creates user account with home & mail spool directories.
* Example: useradd -m bot

**passwd <userName>** - The system generates a password for the user and then stores it in the /etc/shadow file.

**userdel** - Deletes User Account.

* **userdel <userName>** - deletes the user from the system.
* **userdel -r <userName>** - deletes the user from the system along with home and mail spool directories.
* Example: userdel -r bot

**/etc/passwd** - Stores information about user accounts.

* **cat /etc/passwd** - displays the complete list of users on that machine.

**/etc/shadow** - stores the password for users in an encrypted format.

* **cat /etc/shadow** - displays the complete list of user passwords on that machine.

**su**- substitute user.

* **su <userName>** - switches to the user mentioned.
* **exit**- to logout from that user.
* Example: su - ram

**usermod**- modify user.

* **usermod -aG <groupName> <userName>** - adds the user to another group (-aG append the user to the group without removing from other groups).
* Example: usermod -aG mygroup ram

**chsh**- change shell.

* **chsh -s /bin/bash <user>** - changes the shell to bash for the user.
* **chsh -s /bin/sh <user>** - changes the shell to sh for the user.
* Example: chsh -s /bin/sh ubuntu

# Group Management Commands

**groupadd <groupName>** - creates the group.

**groupdel <groupName>** - delete the group.

**/etc/group** - stores the information of the groups.

* **cat /etc/group** - displays the complete list of groups on that machine.

**gpasswd <groupName>** - creates a password for the group.

* **gpasswd -a <userName> <groupName>** - adds the user to the group.
* **gpasswd -d <userName> <groupName>** - removes the user from the group.
* **gpasswd -M <userName1>,<userName2>,<userName3> <groupName>** - adds multiple users to the group and removes the existing ones of the group.

# Searching Commands

[**find**](https://www.geeksforgeeks.org/find-command-in-linux-with-examples/) — Search for files/directories based on their names.

* **find . -name <fileName>** - finds the mentioned file if available in the current directory (.(period) represents current directory).
* **find <dirName> -name <fileName>** - finds the mentioned file in the directory.
* **find <dirName> -perm 754** - finds the files in the directory having 754 permission.
* **locate** is faster for finding files by name due to its pre-built database, while **find** is more versatile, allowing complex searches based on various criteria in real-time.

[**locate**](https://ioflood.com/blog/locate-linux-command/)- Search for files/directories based on their names.

* **locate <fileName/dirName>** - locates the file/directory and displays the path.
* Example: locate crazy.txt

# GREP Command — Global Regular Expression Print

* **grep <textToSearch> <fileName>** - used to find text patterns within files.
* **grep -i <textToSearch> <fileName>** - used to find text patterns within the file ignoring the case (-i ignore case).
* **grep -v <textToSearch> <fileName>** - used to find non matching lines of text patterns (-v invert-match).
* **grep -l <textToSearch> <fileNames>** - used to display the matching string file names.
* Example: grep -l welcome crazy.txt
* There are additional commands related to grep.
* [egrep (or grep -E)](https://www.geeksforgeeks.org/fgrep-command-in-linux-with-examples/)
* [fgrep (or grep -F)](https://www.geeksforgeeks.org/fgrep-command-in-linux-with-examples/)
* [zgrep (for compressed files)](https://www.geeksforgeeks.org/zgrep-command-in-linux-with-examples/#:~:text=The%20zgrep%20command%20is%20used,applies%20to%20the%20zgrep%20command.&text=Options%3A,matching%20lines%20for%20each%20file.)
* [zegrep (or zgrep -E for compressed files)](https://www.commandlinux.com/man-page/man1/zegrep.1.html#:~:text=Zgrep%20invokes%20grep%20on%20compressed,necessary%20and%20fed%20to%20grep.)
* [bzgrep (for compressed files)](https://www.geeksforgeeks.org/bzgrep-command-in-linux-with-examples/#:~:text=Syntax%20of%20bzgrep%20command%20in,perform%20a%20case%2Dinsensitive%20search.)
* [ack-grep (Ack)](https://manpages.ubuntu.com/manpages/trusty/man1/ack-grep.1p.html#:~:text=%2D%2Dhelp%2Dtypes%2C%20%2D%2Dhelp,want%20to%20customize%20ack%2D%20grep.)

# Hardware Infomation Commands

**free -h** - Display system memory information in human-readable format (-h).

**df -h** - It displays the disk space usage of mounted file systems.

**du**- Disk usage.

* **du -h** - Display disk usage information in human-readable format.
* **du -sh** - Display the total size of the directory in human-readable format, summarizing the size instead of listing individual file sizes.
* **du -sh <fileName/dirName>** - Displays the total size of the file/directory.

# Connection To Remote System

**ssh** - Secure Shell: Connect to a remote server securely.  
Example: ssh user@remote\_host

**scp** - Securely Copy Files: Copy files between local and remote systems using SSH.  
Example: scp file.txt user@remote\_host:/path

**rsync** - Remote Sync: Synchronize files and directories between systems.  
Example: rsync -avz local\_folder/ user@remote\_host:remote\_folder/

# Network Commands

[**nc**](https://www.computerhope.com/unix/nc.htm) — Simple tcp proxy, network daemon testing

* Example: nc -vz google.com 443

[**ping**](https://www.ibm.com/docs/en/aix/7.2?topic=p-ping-command)**<hostName>** - tests the reachability & responsiveness of the remote host.

* Example: ping google.com -c 2 (-c pings 2 times)

[**dig**](https://www.geeksforgeeks.org/dig-command-in-linux-with-examples/)**<domainName>** - Shows DNS information of the domain.

* Example: dig medium.com

**wget <url>**- Used to retrieve/download files from the internet.

**curl**- client URL.

* **curl <url>** - Used to retrieve/download files from the internet.

**ifconfig**- Display available network interfaces.

**ip addr** - Display and manipulate network interface info.

**ip route —**To show or manipulate the IP routing table. Shows clear ip tables rules.

[**curl ifconfig.me**](https://www.linuxtrainingacademy.com/determine-public-ip-address-command-line-curl/) - Shows the public ip address of the machine.

**netstat -antp**- shows all tcp open ports (-a all, t-tcp, n-active, p protocol).

[**traceroute <url>**](https://www.varonis.com/blog/what-is-traceroute#:~:text=You%20can%20do%20this%20by,%E2%80%9Ctracert%20varonis.com%E2%80%9D.) - traces the route using packets from source to destination host.

# Process Information Commands

**ps**- Process status.

* **ps**- Displays the currently running process.
* **ps -u <userName>**- Displays the process of the username
* **ps -ef** - Displays all the processes of the system.

**top**- Shows the real-time, dynamic view of the running processes of a system.

**kill <pid>** - Gracefully terminates the process pid(-9 forcefull).

**pgrep <processName>** - Shows process id of processes based on name/other criteria.

**bg**- background, sends the process to the background & continues execution without interruption.

**fg** - foreground, brings the process to the foreground and makes it an active process.

**nohup**- no hangup, runs command/script in the background even after the terminal is closed or the user logs out.

* Example: nohup ./script.sh

**<command> &** — Using in last of command runs in background, allowing you to continue using the terminal while the command runs asynchronously.

* Example: ./script.sh &

# Archiving File Commands

**tar**- tape archive.

* **tar -cvf <fileName> <directory>** - creates the tar file with the fileName for the directory mentioned (-c create, -v verbose, -f output file name).
* **tar -xvf <sourceTarFileName> -C <destinationDir>** - puts the extracted files into the destination directory (-x extract, -v verbose, -f source tar file name, -C change the folder and download to destination dir).

# Ubuntu Package related Commands

**apt** - Package Manager for Debian-based Linux distributions Eg: Ubuntu.

* **apt**- Anewer version of the package manager with colorized output, progress bar and additional functions.
* **apt-get** - Older version and basic package manager.

**apt update** - Updates the package list.

**apt list --installed** - Lists all the installed packages.

* **apt list --installed <packageName>** - shows the package name if it's installed.

**apt show <packageName>** - shows information about a package mentioned.

**apt search <packageName>** - searches and shows the list of packages.

**apt install <packageName>** - installs the required package.

**apt remove <packageName>** - removes the required package.

**apt purge <packageName>** - removes the required package along with its config files.

**Note:**For other package manager just replace “**apt**” with other package manager

# Directory Commands

**pwd**- shows the present working directory (abbr. Print Working Directory).

**cd**- change directory.

* **cd ..** - changes to its parent directory (i.e) one level up.
* **cd <dirName>** - change to the directory mentioned.
* **cd ~** or **cd**- changes to the currently logged in user's home directory.
* **cd ../..** - changes the directory two levels up.
* **cd -** - changes to the last working directory.

**mkdir**- make directory.

* **mkdir <dirName>** - creates the directory.
* **mkdir -p <pathOftheDir>** - creates directory with its parent directories if it does not exists (-p parent).

**ls**- lists the files & folders of the directory you are in.

* **ls -a** - lists all files & folders along with hidden files (-a all).
* **ls -al** - lists all files & folders along with hidden files in a formatted manner (-l long listing format).

# Misc Commands

* **man** - Displays the manual page for a specific command. Provides detailed information and usage instructions.
* **sed** - Edits a stream of text by substituting occurrences of a pattern with another.
* **awk** - A powerful programming language for text processing.
* **wc** -(Word Count)
* **ln** -(Create Links):
* **stat <fileName/dirName>** - shows detailed information about the file or directory.
* **cron**- system daemon for managing scheduled tasks.
* **crontab**-Used to create, edit, and manage cron jobs.
* **tree**- Representation of files and directories of a specific directory.
* **echo "sample text" | grep text** - The output of the first command is passed as an input to the second command using the pipe (|) symbol.
* **ls -l | tee file.txt** - Redirects the list to the file.txt and simultaneously displays it in the terminal.
* **echo "sample text" > <fileName>** - Write the content to the file mentioned by overwriting the existing content (> redirection operator).
* **echo "new sample text" >> <fileName>** - Appends the contents to the file mentioned without overwriting the existing content (>> redirection operation).

Linux Commands for DevOps CheatSheet [read more](https://vinodhakumara2681997.medium.com/cheatsheet-linux-commands-for-devops-80be32b88656)

# Conclusion

DevOps professionals often rely on a set of essential Linux commands to manage systems, automate tasks, and ensure the smooth operation of infrastructure. These commands are foundational for DevOps tasks and are used in various contexts, from system administration to deployment automation.

# Mastering Linux for DevOps Engineers: Essential Commands and Practices for Success

## **Explore the fundamental concepts, tools, and techniques that every DevOps engineer should be familiar with when working with Linux.**

[[](https://medium.com/@imran-1997?source=post_page---byline--a608a718069f---------------------------------------)](https://medium.com/@imran-1997?source=post_page---byline--a608a718069f---------------------------------------)

[[Cloud Native Daily](https://medium.com/cloud-native-daily?source=post_page---byline--a608a718069f---------------------------------------)](https://medium.com/cloud-native-daily?source=post_page---byline--a608a718069f---------------------------------------)

[MD AL IMRAN](https://medium.com/@imran-1997?source=post_page---byline--a608a718069f---------------------------------------)

·

Follow

Published in

[Cloud Native Daily](https://medium.com/cloud-native-daily?source=post_page---byline--a608a718069f---------------------------------------)

·

5 min read

·

May 31, 2023

44

1



## **Introduction**

In today’s fast-paced and highly competitive tech industry, DevOps has become a crucial methodology for organizations looking to streamline their software development and deployment processes. At the heart of this methodology lies Linux, the open-source operating system that empowers DevOps engineers to achieve scalability, flexibility, and automation. In this blog post, I will explore the fundamental concepts, tools, and techniques that every DevOps engineer should be familiar with when working with Linux.

## What is Linux and why is it important for DevOps?

Linux is an open-source operating system that was initially developed as a free alternative to proprietary Unix-based systems. It has grown into a robust and widely adopted operating system used in various domains, including servers, embedded systems, desktops, and mobile devices. Linux is known for its stability, security, flexibility, and scalability, making it an ideal choice for DevOps practices. Here’s why Linux is important for DevOps:

1. **Open-source nature:** Linux’s open-source nature allows DevOps engineers to access and modify the source code, enabling customization and optimization to meet specific requirements. This flexibility is crucial for tailoring the operating system to fit the needs of different software development and deployment workflows.
2. **Stability and reliability:** Linux has a reputation for stability and reliability, making it an ideal choice for critical systems and large-scale deployments. Its robust architecture and continuous improvements through community-driven development ensure a solid foundation for DevOps workflows.
3. **Command-line interface (CLI):** The command-line interface is a vital aspect of Linux, providing powerful tools and utilities for managing and automating various tasks. DevOps engineers often leverage the CLI to write scripts, automate processes, and perform system administration, enabling efficient and streamlined operations.
4. **Extensive software ecosystem:** Linux boasts a vast ecosystem of open-source software and tools that align perfectly with DevOps principles. From configuration management tools like Ansible and Puppet to containerization platforms like Docker and Kubernetes, Linux provides a rich environment for implementing DevOps practices.
5. **Compatibility and portability:** Linux offers broad hardware and software compatibility, making it adaptable to diverse environments. DevOps engineers can easily deploy applications on different Linux distributions and cloud platforms, ensuring portability and minimizing compatibility issues during the development and deployment stages.

## Different Linux distributions

When it comes to Linux distributions, there is a wide range of options available, each with its own characteristics and strengths. Here are some popular distributions that are commonly used in the DevOps community:

1. **Ubuntu:** Ubuntu is one of the most widely known and user-friendly Linux distributions. It is based on Debian and offers a balance between ease of use and robustness. Ubuntu has a strong focus on regular releases and provides Long Term Support (LTS) versions, ensuring stability and security updates for an extended period. Ubuntu has a large community and extensive documentation, making it an excellent choice for beginners and experienced users alike.
2. **CentOS:** CentOS, derived from Red Hat Enterprise Linux (RHEL), is known for its stability, security, and long-term support. CentOS aims to provide a free and open-source alternative to RHEL, making it suitable for production environments. It is widely used in enterprise setups, web servers, and cloud environments. CentOS follows the same release cycle as RHEL and offers a stable platform for DevOps operations.
3. **Fedora:** Fedora is a community-driven Linux distribution sponsored by Red Hat. It is known for its cutting-edge features, frequent releases, and focus on the latest software advancements. Fedora serves as a testing ground for technologies that eventually make their way into Red Hat Enterprise Linux. It is a popular choice for developers and enthusiasts who want access to the latest software and features.
4. **Arch Linux:** Arch Linux is a lightweight and highly customizable distribution that follows a “do-it-yourself” approach. It provides a minimalist base and allows users to build their desired system from the ground up. Arch Linux emphasizes simplicity, elegance, and user-centric design. It offers a rolling release model, providing continuous updates and access to the latest software versions.
5. **SUSE Linux Enterprise Server (SLES):** SUSE Linux Enterprise Server is a commercial Linux distribution designed for mission-critical workloads and enterprise environments. It offers advanced features like high availability, clustering, and enterprise-level support. SLES is widely used in large-scale deployments and industries such as finance, healthcare, and manufacturing.

## linux Commands That Every DevOps Engineer Should Know

As a DevOps engineer, mastering certain Linux commands is essential for efficient system administration, automation, and troubleshooting. Here are some crucial Linux commands that every DevOps engineer should know:

1. **File and Directory Management**

* ls: List files and directories.
* cd: Change the current directory.
* pwd: Print the current working directory.
* mkdir: Create a new directory.
* rm: Remove files and directories.
* cp: Copy files and directories.
* mv: Move or rename files and directories.
* find: Search for files and directories.
* chmod: Change file permissions.
* chown: Change file ownership.
* chgrp: Change file group ownership.

**2. Text Manipulation and Viewing**

* cat: Concatenate and display file contents.
* grep: Search for patterns in files.
* head: Display the beginning of a file.
* tail: Display the end of a file.
* less: View file contents interactively.
* sed: Stream editor for text manipulation.
* awk: Text processing and data extraction tool.

**3. Process and System Management**

* ps: View running processes.
* top: Monitor system resources and processes in real-time.
* kill: Terminate processes.
* systemctl: Manage system services.
* service: Control system services (older Linux distributions).
* df: Display disk space usage.
* du: Estimate file and directory disk usage.
* free: Display system memory usage.
* uptime: Show system uptime and load averages.

**4. Networking**

* ping: Send ICMP echo requests to a host.
* curl or wget: Download files from the web.
* ssh: Securely connect to remote systems.
* scp: Securely copy files between systems.
* netstat: Network statistics and connections.
* ifconfig or ip: Network interface configuration.
* iptables or ufw: Firewall configuration.

**5. Package Management**

* apt (Debian/Ubuntu) or yum (CentOS/RHEL): Package management commands for installing, updating, and removing software packages.
* dpkg (Debian/Ubuntu) or rpm (CentOS/RHEL): Package management commands for querying package information and managing individual packages.

**6. Compression and Archiving**

* tar: Create and extract tar archives.
* gzip, gunzip, bzip2, unzip: Compress and decompress files.
* zip: Create and extract ZIP archives.

## Conclusion

Linux has emerged as the backbone of modern-day DevOps practices, enabling organizations to achieve scalability, agility, and automation in their software development and deployment processes. By mastering Linux and its related tools and technologies, DevOps engineers can streamline workflows, improve efficiency, and deliver high-quality applications with ease. Whether you are a seasoned DevOps professional or just starting your journey, investing time and effort in understanding Linux is a crucial step toward becoming a proficient DevOps engineer.

# Essential Linux Commands and SSH for DevOps: A Hands-On Guide

[[](https://medium.com/@tech_18484?source=post_page---byline--6e59ed4e3b92---------------------------------------)](https://medium.com/@tech_18484?source=post_page---byline--6e59ed4e3b92---------------------------------------)

[Rapidcode Technologies](https://medium.com/@tech_18484?source=post_page---byline--6e59ed4e3b92---------------------------------------)

·

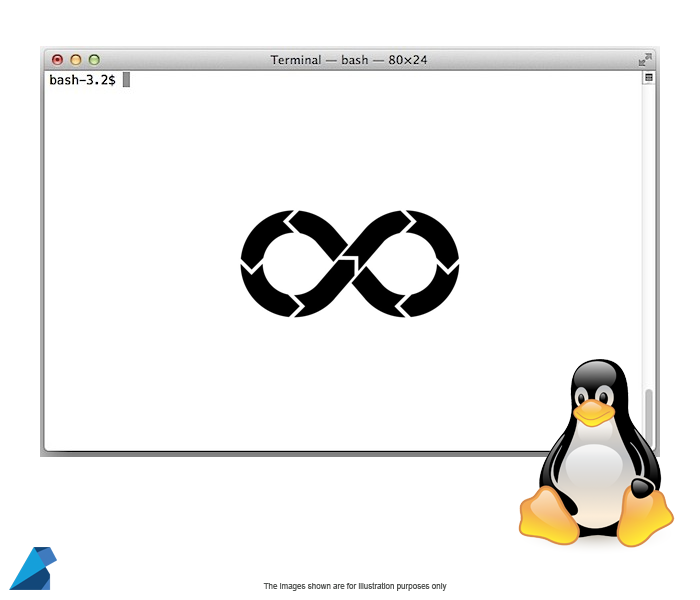
Follow

6 min read

·

Aug 22, 2023

1



# Introduction:

In the world of DevOps, proficiency in Linux commands is essential for the efficient management and deployment of applications and systems. Whether you’re a seasoned DevOps engineer or just starting your journey, mastering these fundamental Linux commands can greatly enhance your productivity. In this article, we’ll explore some indispensable Linux commands and how you can use ssh for DevOps.

**1. ls — List Files and Directories**:  
The**`ls`** command is used to list the contents of a directory.

ls  
ls -l  
ls -a

**2. cd — Change Directory:**  
The **`cd`** command allows you to change your current directory.

cd /path/to/directory  
cd ..

**3. mkdir — Create Directory:**  
Use the **`mkdir`** command to create new directories.

mkdir new\_directory

**4. cp — Copy Files and Directories:**  
The **`cp`** command is used to copy files and directories.

cp file.txt /path/to/destination/  
cp -r directory/ /path/to/destination/

**5. mv — Move and Rename Files:**  
The **`mv`** command moves or renames files and directories.

mv file.txt new\_location/  
mv old\_name.txt new\_name.txt

**6. rm — Remove Files and Directories:**  
The **`rm`**command deletes files and directories.

rm file.txt  
rm -r directory/

**7. touch — Create Empty Files:**  
The **`touch`** command creates empty files.

touch new\_file.txt

**8. grep — Search Text:**  
Use **`grep`** to search for specific text within files.

grep "keyword" file.txt

**9. ps — Process Status:**  
The **`ps`** command displays information about running processes.

ps aux

**10. top — Monitor System Activity:**  
Use the **`top`**command to monitor system activities in real-time.

top

**11. df — Disk Space Usage:**  
The **`df`** command shows disk space usage.

df -h

**12. chmod — Change File Permissions:**  
Use **`chmod`** to modify file permissions.

chmod +x script.sh

**13. chown — Change File Ownership:**  
The **`chown`** command changes file ownership.

chown new\_owner:new\_group file.txt

**14. wget — Download Files:**  
Use **`wget`** to download files from the internet.

wget https://example.com/file.zip

**15. curl — Transfer Data with URLs:**  
The **`curl`** command is used for transferring data with URLs.

curl -O https://example.com/file.txt

**16. tar — Archive and Extract Files:**  
The **`tar`** command is used to create and extract archive files.

tar -czvf archive.tar.gz directory/  
tar -xzvf archive.tar.gz

**17. git — Version Control:**  
The **`git`** command manages source code versioning.

git clone repository\_url  
git add .  
git commit -m "Commit message"  
git push origin main

**18. systemctl —**System Services:  
Control and manage system services.

systemctl start service\_name  
systemctl enable service\_name

**19.journalctl — System Journal:**  
View and analyze system logs.

journalctl -u service\_name

**20. curl — Data Transfer with URLs:**  
Transfer data using URLs, useful for APIs and web services.

curl -X GET https://api.example.com/data

**21. jq — JSON Processor:**  
Process and manipulate JSON data from the command line.

curl -X GET https://api.example.com/data | jq '.property'

**22. awk — Text Processing:**  
Process and manipulate text data using patterns.

cat logfile.txt | awk '/error/ {print $1}'

**23. sed — Stream Editor:**  
Edit text streams using regular expressions.

cat file.txt | sed 's/old/new/g'

**24. nc — Network Connection:**  
Create network connections for debugging and monitoring.

nc -vz example.com 80

**25. tmux — Terminal Multiplexer:**  
Manage multiple terminal sessions within a single window.

tmux new-session -s session\_name

**26. htop — Interactive System Monitoring:**  
Monitor system resources interactively.

htop

**27. tail — Display Last Lines of a File:**  
Display the last lines of a log file.

tail -f /var/log/syslog

**28. cron — Schedule Jobs:**  
Schedule repetitive tasks using the cron daemon.

crontab -e

**29. ncdu — Disk Usage Analyzer:**  
Analyze disk space usage interactively.

ncdu /path/to/directory

# SSH for DevOps

In the world of DevOps, secure and efficient communication between servers is a cornerstone. One tool that has played a pivotal role in enabling this communication is SSH (Secure Shell).

## **Booming into SSH: A Quick Introduction**

**SSH**stands for **Secure Shell** or **Secure Socket Shell**. It is a protocol in which the traffic is encrypted over the network thus called a **Cryptographic Network Protocol**.

This is used to establish a secure network connection between two computers over the internet.

**SSH**is mainly used for the following:

**Secure Remote Login:**

**Secure File Transfers:**

**Tunneling:**

**Public and Private Keys**

We need to understand the two keys that are involved in the connection process. So what are the two keys, let’s go further reading

When we wanted to access another computer we generally use passwords that are not secure and can be hacked by anyone. So the much better, secure alternative to using passwords is the SSH Keys.

With these Keys, we can make a secure connection and this is also called **passwordless authentication**.

So in SSH, we have a key pair named Public key and Private key.

**Public Key:**

* Distributable to anyone who needs to communicate with you.
* Stored on the server you want to access.
* Used for encrypting messages.
* Identified by the “.pub” extension.

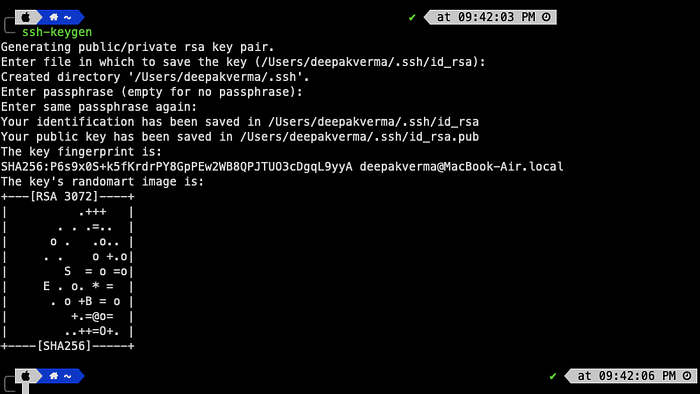
**Private Key:**

* Must be kept confidential and never shared.
* Used to decrypt messages encrypted with the associated public key.
* The cornerstone of passwordless authentication.

## Generating Your Key Pair

Generating an SSH key pair is a crucial step in using SSH. Follow these steps to create your key pair:

1. Launch your terminal.
2. Enter the command ssh-keygen.
3. Follow the prompts; you can keep default values or customize as needed.
4. Two keys are generated: id\_rsa (Private Key) and id\_rsa.pub (Public Key).



## **Copy Public Key to Target Server**

In order to make a connection to the target Server (the server which we wanted to login) we have to copy the Public key to the target server’s .ssh folder.

**Manual Copy:**

1. Navigate to the target server’s .ssh folder.
2. If absent, create the authorized\_keys file.
3. Add your public key’s contents to the file.

**Using ssh-copy-id:**

1. Run ssh-copy-id <target-host-ip>.
2. The public key will be appended to the target server’s authorized\_keys file.

## **Establishing Connections**

1. Attempt to connect from the client to the target server.
2. The server sends an encrypted message using the public key.
3. The client decrypts the message with its private key and sends it back.
4. The server confirms the message, establishing trust and the connection.

## **Finally, the SSH Command**

To effortlessly connect using SSH:

ssh -i <path-to-private-key> <userName>@<target-ip-address>

* ssh: Initiates the secure shell connection.
* -i: Specifies the identity (private key) to use.
* userName: Your username on the target server.
* target-ip-address: The IP address of the target server.

# Conclusion:

In the world of DevOps, learning essential Linux commands isn’t just something you do; it’s like adding fuel to a powerful engine. It helps manage computer systems smoothly and put applications in place without any bumps. It doesn’t matter if you’re already really good at DevOps or just starting out — understanding and using these basic Linux commands can make you much better at what you do.

# LINUX For DevOps Engineers

[[Skmswetha](https://medium.com/@skmswetha22?source=post_page---byline--9b51d4168f9b---------------------------------------)](https://medium.com/@skmswetha22?source=post_page---byline--9b51d4168f9b---------------------------------------)

[Skmswetha](https://medium.com/@skmswetha22?source=post_page---byline--9b51d4168f9b---------------------------------------)

·

Follow

12 min read

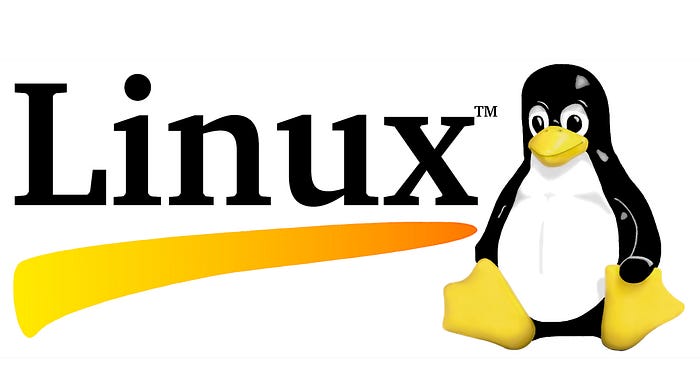
·

Aug 18, 2023

54

1

Linux is an open-source operating system kernel that serves as the foundation for various Linux-based operating systems (distributions). Linux is known for its stability, security, and flexibility. It is based on the Unix operating system and follows the Unix philosophy of small, modular, and reusable components. Linux is highly customizable and can be tailored to suit various needs, ranging from desktop computers and servers to embedded systems and supercomputers.



*Linux distributions, such as Ubuntu, Fedora, Debian, CentOS, and many others, take the Linux kernel and combine it with additional software packages, utilities, and graphical interfaces to create complete operating systems that are ready to be used by end-users.*

Linux offers a command-line interface (CLI) that allows users to interact with the system through text commands, as well as graphical user interfaces (GUIs) that provide a more user-friendly experience. It supports a wide range of applications, software development tools, and server services, making it suitable for various purposes, including general computing, web servers, cloud infrastructure, networking, and more.

# Linux Folder Structure

The folder structure in Ubuntu Linux follows the Filesystem Hierarchy Standard (FHS), which is a standard for organizing the files and directories on a Unix-like operating system. Here is an overview of the main directories you will typically find in Ubuntu:

* /bin: Contains essential command-line executable files (binaries) that are available to all users.
* /boot: Contains files related to the boot process, including the Linux kernel, initial ramdisk (initrd), and bootloader configuration.
* /dev: Contains device files that represent and allow access to various hardware devices on the system.
* /etc: Contains system-wide configuration files for various applications and services.
* /home: The home directories for individual users. Each user typically has a subdirectory here to store their personal files and settings.
* /lib and /lib64: These directories contain shared libraries needed by the system and applications. The “lib64” directory is present on 64-bit systems.
* /media: Mount point for removable media devices such as USB drives or optical discs.
* /mnt: A general-purpose mount point for temporarily mounting filesystems.
* /opt: Contains optional software packages installed on the system. Applications installed here are often self-contained in their own directories.
* /proc: A virtual filesystem that provides information about processes and system status. It is used by many system utilities to obtain runtime information.
* /root: The home directory for the root user, the administrative superuser.
* /run: A temporary filesystem that contains runtime data for various system services. It is cleared on each reboot.
* /sbin: Contains system binaries (executable files) that are primarily used by the root user for system administration tasks.
* /srv: Contains data for services provided by the system.
* /sys: A virtual filesystem that exposes kernel-related information and configuration.
* /tmp: A directory for temporary files created by applications and users. Its contents are typically cleared on each reboot.
* /usr: Contains user-related programs, libraries, documentation, and shared resources. It has subdirectories such as /usr/bin for user binaries, /usr/lib for libraries, and /usr/share for shared data.
* /var: Contains variable data that changes during the system’s operation, such as logs, databases, and spool files.

This is a high-level overview of the Ubuntu Linux folder structure. Each directory serves a specific purpose in organizing the system’s files and resources.

## Important Linux Commands for DevOps Engineer

## 1. **ls**

* Description: Lists files and directories in the current directory.
* Usage:
* ls: Lists files and directories in the current directory.
* ls -l: Lists files and directories in long format.
* ls -a: Lists all files and directories, including hidden ones

## 2. **cd**

. Description: Changes the current directory.

* Usage:
* cd /path/to/directory: Changes the current directory to the specified path.
* cd ..: Moves up to the parent directory.
* cd ~: Moves to the home directory.

## 3. **mkdir**

* Description: Creates a new directory.
* Usage:
* mkdir directory\_name: Creates a new directory with the specified name.
* mkdir -p path/to/directory: Creates nested directories if they don't exist.

## 4. **rm**

* Description: Removes files and directories.
* Usage:
* rm file\_name: Removes the specified file.
* rm -r directory\_name: Removes the specified directory and its contents recursively.
* rm -f file\_name: Forces removal of the specified file without prompting.

## 5. **cp**

* Description: Copies files and directories.
* Usage:
* cp source\_file destination\_file: Copies the source file to the destination.
* cp -r source\_directory destination\_directory: Copies the source directory to the destination recursively.

## 6. **mv**

* Description: Moves or renames files and directories.
* Usage:
* mv source\_file destination\_file: Moves the source file to the destination or renames it.
* mv source\_directory destination\_directory: Moves the source directory to the destination or renames it.

## 7. **grep**

* Description: Searches for a specific pattern in files or output.
* Usage:
* grep pattern file\_name: Searches for the specified pattern in the given file.
* grep -r pattern directory: Searches for the pattern recursively in the specified directory.
* command | grep pattern: Filters the output of a command and searches for the pattern.

## 8. **ps**

* Description: Lists running processes.
* Usage:
* ps: Lists the running processes for the current user.
* ps -ef: Lists all running processes.
* ps -eaf: Lists all running processes with full details.

## 9. **top**

* Description: Displays real-time system information and the list of processes.
* Usage:
* top: Displays real-time system information, CPU usage, memory usage, and the list of processes.
* Press q to exit the top command.

## 10. **tail**

* Description: Outputs the last part of a file.
* Usage:
* tail file\_name: Displays the last 10 lines of the specified file.
* tail -n N file\_name: Displays the last N lines of the specified file.
* tail -f file\_name: Continuously outputs new lines appended to the file.

# Here are the top 50 networking commands in Linux:

1. ifconfig: Displays or configures network interfaces. Example: ifconfig eth0
2. ip: Configures and displays network interfaces, routing tables, and more. Example: ip address show
3. ping: Sends ICMP echo requests to a specified network host. Example: ping google.com
4. traceroute: Displays the route packets take to reach a destination host. Example: traceroute google.com
5. nslookup: Queries DNS servers for DNS-related information. Example: nslookup google.com
6. dig: DNS lookup utility for querying DNS servers. Example: dig google.com
7. host: Performs DNS lookups. Example: host google.com
8. netstat: Displays network connections, routing tables, and network statistics. Example: netstat -tun
9. ss: Provides detailed socket statistics. Example: ss -tun
10. route: Configures and displays routing table information. Example: route -n
11. arp: Manipulates or displays the ARP cache. Example: arp -a
12. iptables: Manages firewall rules. Example: iptables -L
13. tcpdump: Captures network traffic. Example: tcpdump -i eth0
14. ifup: Brings a network interface up. Example: ifup eth0
15. ifdown: Brings a network interface down. Example: ifdown eth0
16. ethtool: Displays or changes Ethernet device settings. Example: ethtool eth0
17. hostname: Displays or sets the system's hostname. Example: hostname
18. ssh: Connects to a remote server using the SSH protocol. Example: ssh user@hostname
19. scp: Copies files between hosts using SSH. Example: scp file.txt user@hostname:/path/to/destination
20. rsync: Syncs files and directories between different locations. Example: rsync -avz source/ user@hostname:/path/to/destination
21. nc: Reads and writes data across network connections. Example: nc -l 8080
22. wget: Downloads files from the web. Example: wget <http://example.com/file.txt>
23. curl: Transfers data to or from a server. Example: curl [http://example.com](http://example.com/)
24. nmap: Scans ports and discovers network services. Example: nmap -p 1-1000 hostname
25. telnet: Establishes a telnet connection to a remote host. Example: telnet hostname
26. ifstat: Displays network interface statistics. Example: ifstat
27. mtr: Combines ping and traceroute functionality. Example: mtr google.com
28. route add: Adds a new route to the routing table. Example: route add -net 192.168.0.0/24 gw 192.168.1.1
29. route delete: Deletes a route from the routing table. Example: route delete default gw 192.168.1.1
30. ifconfig up: Enables a network interface. Example: ifconfig eth0 up
31. ifconfig down: Disables a network interface. Example: ifconfig eth0 down
32. ip link: Manages network interfaces. Example: ip link show
33. ip route: Manages routing tables. Example: ip route show
34. ip neigh: Manages ARP cache. Example: ip neigh show
35. ip addr: Manages IP addresses and interfaces. Example: ip addr show
36. ip link set: Modifies network interface properties. Example: ip link set eth0 mtu 1500
37. ip route add: Adds a new route to the routing table. Example: ip route add 192.168.0.0/24 via 192.168.1.1 dev eth0
38. ip route delete: Deletes a route from the routing table. Example: ip route delete 192.168.0.0/24
39. ip addr add: Adds an IP address to an interface. Example: ip addr add 192.168.0.1/24 dev eth0
40. ip addr delete: Deletes an IP address from an interface. Example: ip addr delete 192.168.0.1/24 dev eth0
41. ip tunnel add: Creates a tunnel interface. Example: ip tunnel add mytunnel mode gre remote 203.0.113.1 local 198.51.100.1
42. ip tunnel delete: Deletes a tunnel interface. Example: ip tunnel delete mytunnel
43. ip link set promisc on: Puts a network interface into promiscuous mode. Example: ip link set eth0 promisc on
44. ip link set mtu: Sets the Maximum Transmission Unit (MTU) of a network interface. Example: ip link set eth0 mtu 1500
45. iptables -A INPUT: Appends a rule to the INPUT chain of the firewall. Example: iptables -A INPUT -s 192.168.0.0/24 -j ACCEPT
46. iptables -P: Sets the default policy for a chain in the firewall. Example: iptables -P INPUT DROP
47. iptables -F: Flushes all rules from a chain in the firewall. Example: iptables -F INPUT
48. iptables-save: Saves the current firewall rules to a file. Example: iptables-save > firewall.rules

These commands cover a wide range of networking tasks and can be helpful for managing and troubleshooting network connections in Linux.

These commands provide essential functionality for network configuration, troubleshooting, and analysis in Linux. You can refer to the respective man pages or online documentation for each command to learn more about their options and usage.

*In Linux, you can change permissions and ownership of files and directories using the chmod, chown, and chgrp commands. Here's how you can use these commands:*

# Changing Permissions

## **chmod**

The chmod command is used to change permissions of files and directories. It supports two modes of operation: symbolic mode and octal mode.

1. Symbolic Mode:

Syntax: chmod [options] [permissions] file(s)

Examples:

* Grant read and write permissions to the owner: chmod u+rw file.txt
* Revoke execute permission from the group: chmod g-x script.sh
* Add read and execute permissions to others: chmod o+rx program
* Combined permissions: chmod u=rw,go=r file.txt

2.Octal Mode:

Syntax: chmod [options] [mode] file(s)

Examples:

* Set read, write, and execute permissions for owner, group, and others: chmod 755 script.sh
* Restrict permissions to the owner only: chmod 700 private.txt
* Grant full permissions to everyone: chmod 777 public\_dir

# Changing Ownership

## **chown**

The chown command is used to change the ownership of files and directories.

* Syntax: chown [options] owner:group file(s)

Examples:

* Change the owner and group of a file: chown john:users file.txt
* Recursively change ownership for a directory and its contents: chown -R alice:staff project\_dir

## **chgrp**

The chgrp command is used to change the group ownership of files and directories.

* Syntax: chgrp [options] group file(s)

Examples:

* Change the group of a file: chgrp developers script.sh
* Recursively change group ownership for a directory and its contents: chgrp -R team project\_dir

In Linux, you can change permissions using the chmod command. The permissions determine who can read, write, or execute a file. Here's how you can use chmod to change permissions:

Syntax:

chmod [options] permissions file(s)

Options:

* -R or --recursive: Change permissions recursively for directories and their contents.

Permissions:

* u (user/owner): The owner of the file.
* g (group): The group associated with the file.
* o (others): Users who are neither the owner nor part of the group.
* a (all/everyone): All users, including the owner, group, and others.

Each permission has three possible settings:

* r (read): Permission to read the file or view directory contents.
* w (write): Permission to modify or delete the file or add/remove files in a directory.
* x (execute): Permission to execute a file or access a directory.

Examples:

1. Grant read and write permissions to the owner:

chmod u+rw file.txt

2.Revoke execute permission from the group:

chmod g-x script.sh

3.Add read and execute permissions to others:

chmod o+rx program

4.Set read and write permissions for the owner, and read-only permissions for the group and others:

chmod u=rw,g=r,o=r file.txt

5.Change permissions recursively for a directory and its contents:

chmod -R u+w project/

# Top 100 LINUX COMMANDS

Sure! Here is a list of 100 commonly used Linux commands with examples:

1. ls: List files and directories in the current directory. Example: ls
2. cd: Change directory. Example: cd /path/to/directory
3. Pwd: Print the current working directory. Example: pwd
4. mkdir: Create a new directory. Example: mkdir directory\_name
5. rmdir: Remove an empty directory. Example: rmdir directory\_name
6. cp: Copy files and directories. Example: cp file.txt destination\_folder/
7. mv: Move or rename files and directories. Example: mv file.txt new\_file.txt or mv file.txt destination\_folder/
8. rm: Remove files and directories. Example: rm file.txt or rm -r directory/ (recursively)
9. cat: Concatenate and display file content. Example: cat file.txt
10. less: View file content page by page. Example: less file.txt
11. head: Display the first few lines of a file. Example: head file.txt
12. tail: Display the last few lines of a file. Example: tail file.txt
13. grep: Search for a pattern in files. Example: grep "pattern" file.txt
14. find: Search for files and directories. Example: find /path/to/search -name "filename"
15. chmod: Change file permissions. Example: chmod 755 file.txt (gives read, write, execute permissions to the owner, read and execute permissions to the group and others)
16. chown: Change file ownership. Example: chown user:group file.txt
17. tar: Archive files and directories. Example: tar -cvf archive.tar file.txt (create an archive)
18. gzip: Compress files. Example: gzip file.txt (creates file.txt.gz)
19. gunzip: Decompress files compressed with gzip. Example: gunzip file.txt.gz
20. wget: Download files from the web. Example: wget <http://example.com/file.txt>
21. curl: Transfer data from or to a server. Example: curl [http://example.com](http://example.com/)
22. ssh: Connect to a remote server securely. Example: ssh user@remote\_server
23. scp: Copy files between local and remote machines. Example: scp file.txt user@remote\_server:/path/to/destination
24. ping: Send ICMP echo requests to a network host. Example: ping example.com
25. ifconfig: Display or configure network interfaces. Example: ifconfig
26. netstat: Network statistics. Example: netstat -an
27. ip: Show or manipulate routing, devices, policy routing, and tunnels. Example: ip addr show
28. route: Show or manipulate the IP routing table. Example: route -n
29. whois: Retrieve WHOIS information for a domain. Example: whois example.com
30. uname: Print system information. Example: uname -a
31. ps: Display running processes. Example: ps aux
32. top: Monitor system processes in real-time. Example: top
33. kill: Terminate processes by ID or name. Example: kill 1234 or killall process\_name
34. df: Report file system disk space usage. Example: df -h
35. du: Estimate file and directory space usage. Example: du -sh directory
36. mount: Mount a file system or device. Example: mount /dev/sda1 /mnt
37. umount: Unmount a mounted file system. Example: umount /mnt
38. ln: Create hard or symbolic links. Example: ln -s /path/to/file link\_name
39. echo: Print a message. Example: echo "Hello, world!"
40. date: Display the current date and time. Example: date
41. history: View command history. Example: history
42. tar: Archive files and directories. Example: tar -cvf archive.tar file.txt (create an archive)
43. unzip: Extract compressed files in a ZIP format. Example: unzip file.zip
44. file: Determine file type. Example: file file.txt
45. awk: Pattern scanning and processing language. Example: awk '{print $1}' file.txt
46. sed: Stream editor for filtering and transforming text. Example: sed 's/foo/bar/' file.txt
47. wc: Count lines, words, and characters in a file. Example: wc -l file.txt
48. sort: Sort lines of text. Example: sort file.txt
49. diff: Compare files line by line. Example: diff file1.txt file2.txt
50. grep: Search for a pattern in files. Example: grep "pattern" file.txt
51. tr: Translate or delete characters. Example: tr 'a-z' 'A-Z' file.txt
52. cut: Remove sections from lines of files. Example: cut -d ',' -f 1 file.txt
53. du: Estimate file and directory space usage. Example: du -sh directory
54. scp: Copy files between local and remote machines. Example: scp file.txt user@remote\_server:/path/to/destination
55. ssh: Connect to a remote server securely. Example: ssh user@remote\_server
56. man: Display the manual page for a command. Example: man ls
57. info: View command information and documentation. Example: info command\_name
58. apt-get: Package handling utility for Debian-based systems. Example: apt-get install package\_name
59. yum: Package manager for RPM-based systems. Example: yum install package\_name
60. systemctl: Control the systemd system and service manager. Example: systemctl start service\_name
61. service: Run a System V init script. Example: service service\_name start
62. chown: Change file ownership. Example: chown user:group file.txt
63. chmod: Change file permissions. Example: chmod 755 file.txt
64. ln: Create hard or symbolic links. Example: ln -s /path/to/file link\_name
65. tee: Redirect output to multiple files or commands. Example: command | tee file.txt
66. fg: Bring a background process to the foreground. Example: fg %1
67. bg: Send a process to the background. Example: bg %1
68. alias: Create an alias for a command. Example: alias l='ls -l'
69. source: Execute commands from a file in the current shell. Example: source script.sh
70. echo: Print a message. Example: echo "Hello, world!"
71. export: Set environment variables. Example: `export VARIABLE\_NAME=value`
72. env: Display the current environment variables. Example: env
73. sleep: Delay for a specified amount of time. Example: sleep 5 (sleep for 5 seconds)
74. su: Switch user or become superuser. Example: su username or su - (switch to root user)
75. sudo: Execute a command as the superuser. Example: sudo command\_name
76. passwd: Change user password. Example: passwd username
77. useradd: Create a new user. Example: useradd username
78. usermod: Modify user account settings. Example: usermod -aG group\_name username
79. groupadd: Create a new group. Example: groupadd group\_name
80. groupmod: Modify group settings. Example: groupmod -n new\_group\_name old\_group\_name
81. crontab: Schedule commands to run at specific times. Example: crontab -e (edit cron jobs)
82. at: Execute commands at a specified time. Example: at 10:00AM (enter commands and press Ctrl+D)
83. shutdown: Shutdown or restart the system. Example: shutdown now (shutdown immediately)
84. reboot: Reboot the system. Example: reboot
85. ifconfig: Display or configure network interfaces. Example: ifconfig
86. netstat: Network statistics. Example: netstat -an
87. ip: Show or manipulate routing, devices, policy routing, and tunnels. Example: ip addr show
88. route: Show or manipulate the IP routing table. Example: route -n
89. iptables: Administration tool for IPv4 packet filtering and NAT. Example: iptables -L (list firewall rules)
90. adduser: Interactive tool for adding new users. Example: adduser username
91. deluser: Remove a user account and associated files. Example: deluser username
92. passwd: Change user password. Example: passwd username
93. crontab: Schedule commands to run at specific times. Example: crontab -e (edit cron jobs)
94. history: View command history. Example: history
95. file: Determine file type. Example: file file.txt
96. hostname: Print or set the system’s hostname. Example: hostname
97. uptime: Display the system’s uptime. Example: uptime
98. dmesg: Print or control the kernel ring buffer. Example: dmesg
99. free: Display amount of free and used memory. Example: free -h
100. htop: Interactive process viewer and system monitor. Example: htop

# CheatSheet: Linux Commands for DevOps

[[](https://vinodhakumara2681997.medium.com/?source=post_page---byline--80be32b88656---------------------------------------)](https://vinodhakumara2681997.medium.com/?source=post_page---byline--80be32b88656---------------------------------------)

[Vinodha kumara](https://vinodhakumara2681997.medium.com/?source=post_page---byline--80be32b88656---------------------------------------)

·

Following

5 min read

·

Jan 20, 2024

605

6



Absolutely! As a DevOps professional, it’s vital to be proficient in the Linux command line for effective server management, automation, and troubleshooting. In this comprehensive guide, we’ll cover **50+** essential Linux commands(**cheatsheet**) with clear explanations and practical examples. This will help you enhance your Linux skills in a straightforward and practical manner.

1. **id** - This is used to find out user and group names and numeric ID’s (UID or group ID) of the current user or any other user in the server.  
   Example: id -u root

2. **cd** - Change Directory: Navigate to a different directory.  
Example:cd /home/user/documents

3. **pwd** - Print Working Directory: Display the current directory's full path. Example: pwd

4. **mkdir** - Make Directory: Create a new directory.  
Example: mkdir new\_folder

5. **rm** - Remove: Delete files or directories.  
Example: rm file.txt

6. **cp** - Copy: Copy files or directories.  
Example: cp file.txt /backup

7. **mv** - Move: Move files or directories.  
Example: mv file.txt /new\_location

8. **touch** - Create Empty File: Create a new empty file.  
Example: touch new\_file.txt

9. **cat** - Concatenate and Display: View the content of a file.  
Example: cat file.txt

10. **nano** - Text Editor: Open a text file for editing.  
Example: nano file.txt

11. **grep** - Search Text: Search for text patterns in files.  
Example: grep "pattern" file.txt

12. **find** - Search Files and Directories: Search for files and directories. Example: find /path/to/search -name "file\_name"

13. **chmod** - Change File Permissions: Modify file permissions.  
Example: chmod 755 file.sh

14. **chown** - Change Ownership: Change the owner and group of a file or directory.  
Example: chown user:group file.txt

15. **ps** - Process Status: Display running processes.  
Example: ps aux

16. **top** - Monitor System Activity: Monitor system processes in real-time. Example: top

17. **kill** - Terminate Processes: Terminate a process using its ID. Also can use **pkill** to terminate processes based on their name or other attributes.  
Example: kill PID  
pkill Process\_Name

18. **wget** - Download Files: Download files from the internet.  
Example: wget <https://example.com/file.zip>

19. **less** - To view the contents of a file one screen at a time, allowing for easy navigation and search within the file. Example: less test.log

20. **tar** - Archive and Extract: Create or extract compressed archive files. Example: tar -czvf archive.tar.gz folder

21. **ssh** - Secure Shell: Connect to a remote server securely.  
Example: ssh user@remote\_host

22. **scp** - Securely Copy Files: Copy files between local and remote systems using SSH.  
Example: scp file.txt user@remote\_host:/path

23. **rsync** - Remote Sync: Synchronize files and directories between systems.  
Example: rsync -avz local\_folder/ user@remote\_host:remote\_folder/

24. **df** - Disk Free Space: Display disk space usage.  
Example: df -h

25. **du** - Disk Usage: Show the size of files and directories.  
Example: du -sh /path/to/directory

26. **ifconfig** - Network Configuration: Display or configure network interfaces (deprecated, use ip).  
Example: ifconfig

27. **ip** - IP Configuration: Manage IP addresses and network settings. Example: ip addr show

28. **netstat** - Network Statistics: Display network connections and statistics (deprecated, use ss).  
Example: netstat -tuln

29. **systemctl** - System Control: Manage system services using systemd. Example: systemctl start service\_name

30. **journalctl** - Systemd Journal: View system logs using systemd's journal.  
Example: journalctl -u service\_name

31. **free** - This command displays the total amount of free space available.  
Example: free -m

32. **at** - Execute Commands Later: Run commands at a specified time. Example: echo "command" | at 15:30

33. **ping** - Network Connectivity: Check network connectivity to a host. Example: ping google.com

34. **traceroute** - Trace Route: Trace the route packets take to reach a host. Example: traceroute google.com

35. - Check Website Connectivity: Check if a website is up.  
Example: curl -Is https://example.com | head -n 1

36. **dig** - Domain Information Groper: Retrieve DNS information for a domain.  
Example: dig example.com

37. **hostname** - Display or Set Hostname: Display or change the system's hostname.  
Example: hostname

38. **who** - Display Users: Display currently logged-in users.  
Example: who

39. useradd - Add User: Create a new user account.  
Example: useradd newuser

40. **usermod** - Modify User: Modify user account properties.  
Example: usermod -aG groupname username

41. **passwd** - Change Password: Change user password.  
Example: passwd username

42. **sudo** - Superuser Do: Execute commands as the superuser.  
Example: sudo command

43. **lsof** - List Open Files: List open files and processes using them. Example: lsof -i :port

44. **nc** - Netcat: Networking utility to read and write data across network connections.  
Example: echo "Hello" | nc host port

45. **scp** - Secure Copy Between Hosts: Copy files securely between hosts. Example: scp file.txt user@remote\_host:/path

46. **sed** - Stream Editor: Text manipulation using regex.  
Example: sed 's/old/new/g' file.txt

47. **awk** - Text Processing: Pattern scanning and text processing.  
Example: awk '{print $2}' file.txt

48. **cut** - Text Column Extraction: Extract specific columns from text. Example: cut -d"," -f2 file.csv

49. **sort** - Sort Lines: Sort lines of text files.  
Example: sort file.txt

50. **diff** - File Comparison: Compare two files and show differences. Example: diff file1.txt file2.txt

51. **ls** - List Files and Directories: List the contents of a directory.  
Example: ls -la

52. **history** - This command is used to view the previously executed command.  
Example: history 10

53. **cron** - Schedule Tasks: Manage scheduled tasks.  
Example: crontab -e

54. **ssh-keygen** - This command is used to generate a public/private authentication key pair. This process of authentication allows the user to connect remote server without providing a password.  
Example: ssh-keygen

55. **nslookup** - This stands for “Name server Lookup”. This is a tool for checking DNS hostname to Ip or Ip to Hostname. This is very helpful while troubleshooting.  
Example: nslookup google.com

56. **tr** - For translating or deleting characters.

These commands cover a wide range of tasks that are essential for DevOps professionals working with Linux systems. Remember to always refer to the man pages (man command) for more detailed information about each command and its options.  
Example:cat crazy.txt | tr "[a-z]" "[A-Z]"

57. **tnc** - This is “Test Network Connection” command. Mostly used command while troubleshooting. It displays diagnostic information for a connection.  
Example:tnc google.com --port 443

58. **w**- Displays current user.

59. **su -** Switch User.  
Example: su - root

60. **ac(All Connections) —**Total connect time for all users or specified users.  
Example: ac john

61. **tail**— Displays the last part of a file, commonly used to monitor logs in real-time.  
Example: tail monitor.logs

62. **head**— Displays the first part of a file, often used to quickly see the beginning of a file’s content.  
Example: head content.txt

63. **ip route —**To show or manipulate the IP routing table. Shows clear ip tables rules.  
Example: ip rout

Linux Commands and tricks for DevOps tasks [read more](https://vinodhakumara2681997.medium.com/top-linux-commands-and-tricks-for-devops-tasks-42cf93aa77e0)

# Conclusion

DevOps professionals often rely on a set of essential Linux commands to manage systems, automate tasks, and ensure the smooth operation of infrastructure. These commands are foundational for DevOps tasks and are used in various contexts, from system administration to deployment automation.

# The Ultimate Guide to Linux Networking Commands for DevOps Professionals

[[](https://medium.com/@thesureshyadav76?source=post_page---byline--bc7681d87b04---------------------------------------)](https://medium.com/@thesureshyadav76?source=post_page---byline--bc7681d87b04---------------------------------------)

[Suresh yadav](https://medium.com/@thesureshyadav76?source=post_page---byline--bc7681d87b04---------------------------------------)

·

Follow

8 min read

·

Aug 21, 2024

352



[**Cypik.com**](https://www.cypik.com/)

**Introduction:**In the world of DevOps, networking is a cornerstone of infrastructure management. Understanding and efficiently using Linux networking commands is essential for troubleshooting, monitoring, and managing network configurations. This guide provides a comprehensive list of Linux networking commands, complete with examples and explanations, to equip you with the tools you need to excel in your role.

# Section 1: Interface and Address Management

**1. ifconfig — Display Network Interface Information**

ifconfig

* ifconfigis a traditional command used to configure and display network interface parameters. It provides information such as IP addresses, netmasks, and broadcast addresses.

**2. ip — Show/Manipulate Routing, Devices, Policy Routing, and Tunnels**

ip address show

* Theip command is the modern replacement forifconfig. It provides comprehensive options for managing network interfaces, routes, and tunnels.

**3. hostname — Show or Set the System’s Host Name**

hostname

* Thehostname command displays or sets the system's hostname, which is used to identify the system on a network.

**4. iwconfig — Configure a Wireless Network Interface**

iwconfig

* iwconfigis used for configuring wireless network interfaces. It allows you to set parameters like SSID, frequency, and encryption keys.

**5. ethtool — Display or Change Ethernet Card Settings**

ethtool eth0

* ethtoolprovides detailed information about network interfaces and allows you to modify settings such as speed and duplex mode.

**6. resolvconf — Manage DNS Information**

resolvconf -u

* resolvconfis used to manage DNS resolver information, ensuring consistent DNS settings across the system.

**7. hostnamectl — Control the System Hostname and Related Settings**

hostnamectl status

* hostnamectlis part of systemd and allows you to view or change the system’s hostname and other related settings.

# Section 2: Route and Traffic Management

**1. route — Display or Manipulate the IP Routing Table**

route -n

* Theroute command displays or modifies the IP routing table. It’s essential for managing how packets are routed through a network.

**2. ip route — Manage Routing Tables**

ip route show

* Theip route command is part of the ip suite, used for displaying and modifying the routing table.

**3. route add — Add a New Route**

route add default gw 192.168.1.1

* This command adds a default gateway to the routing table, directing traffic destined for outside the local network.

**4. route del — Delete a Route**

route del -net 192.168.2.0 netmask 255.255.255.0

* This command removes a specific route from the routing table, which can be necessary when reconfiguring network paths.

**5. arp — Display or Modify the ARP Cache**

arp -a

* arp shows the Address Resolution Protocol (ARP) cache, mapping IP addresses to MAC addresses on the local network.

**6. iptables — Administration Tool for IPv4 Packet Filtering and NAT**

iptables -L

* iptables is a powerful utility for setting up, maintaining, and inspecting the tables of IP packet filter rules in the Linux kernel.

# Section 3: Diagnostic and Monitoring Tools

**1. ping — Send ICMP ECHO\_REQUEST to Network Hosts**

ping google.com

* ping checks the reachability of a host on the network by sending ICMP echo requests and measuring the response time.

**2. traceroute — Print the Route Packets Take to Network Host**

traceroute google.com

* traceroute shows the path that packets take from your system to a specified host, useful for diagnosing routing issues.

**3. netstat — Print Network Connections, Routing Tables, Interface Statistics**

netstat -an

* netstat provides a detailed view of network connections, routing tables, and interface statistics.

**4. ss — Display Socket Statistics**

ss -tulpn

* ss is a modern replacement for netstat, providing more detailed information about TCP, UDP, and UNIX socket connections.

**5. lsof — List Open Files**

lsof -i :80

* lsof lists open files and network connections. It’s particularly useful for identifying which processes are using specific network ports.

**6. tcpdump — Dump Traffic on a Network**

tcpdump -i eth0

* tcpdump captures and displays the packets on a network interface. It’s invaluable for diagnosing network issues at the packet level.

**7. iftop — Display Bandwidth Usage on an Interface**

iftop

* iftop shows real-time bandwidth usage on a network interface, helping you identify bandwidth-intensive processes.

**8. iptraf — Interactive Color IP LAN Monitor**

iptraf

* iptrafprovides real-time monitoring of network traffic with an interactive, color-coded display.

**9. mtr — Network Diagnostic Tool**

mtr google.com

* mtr combines the functionality of ping and traceroute, providing a real-time view of the network's performance as packets travel from source to destination.

**10. nload — Visual Representation of Incoming and Outgoing Traffic**

nload

* nload provides a graphical representation of network traffic, showing the incoming and outgoing data rates.

**11. iperf — Tool for Measuring TCP and UDP Bandwidth Performance**

iperf -c server\_ip

* iperf measures the maximum TCP and UDP bandwidth performance between two hosts, making it useful for network performance testing.

**12. fping — Quickly Ping Multiple Hosts**

fping -a -g 192.168.1.1 192.168.1.254

* fpingis similar toping but allows you to ping multiple hosts simultaneously, providing a quick overview of network reachability.

# Section 4: DNS and Connectivity Utilities

**1. dig — DNS Lookup Utility**

dig google.com

* digqueries DNS name servers, providing detailed information about DNS records and helping to troubleshoot DNS issues.

**2. nslookup — Query Internet Name Servers Interactively**

nslookup google.com

* nslookupis a user-friendly tool for querying DNS servers, providing interactive DNS queries.

**3. host — DNS Lookup Utility**

host google.com

* Thehost command performs DNS lookups and returns information about the given domain name.

**4. nmap — Network Exploration Tool and Security Scanner**

nmap -sP 192.168.1.0/24

* nmapis used for discovering hosts and services on a network, providing information about the devices and their open ports.

**5. netcat (nc)—Arbitrary TCP and UDP Connections and Listens**

nc -l 8080

* netcat, also known asnc, is a versatile tool for reading from and writing to network connections using TCP or UDP. It can be used for network diagnostics, testing, and data transfer.

# Section 5: Remote Access and File Transfer

**1. sshd — OpenSSH Daemon**

service sshd restart

* sshdis the OpenSSH server process that handles incoming SSH connections. Restarting it can resolve remote access issues.

**2. ssh-keygen — Generate, Manage, and Convert Authentication Keys for SSH**

ssh-keygen -t rsa

* ssh-keygenis used to generate, manage, and convert SSH authentication keys, enabling secure SSH connections.

**3. scp — Secure Copy (Remote File Copy Program)**

scp file.txt user@remote:/path/to/destination

* scpsecurely copies files between hosts over a network using SSH. It’s a reliable method for transferring files while maintaining security.

**4. telnet — User Interface to the TELNET Protocol**

telnet google.com 80

* telnetis a protocol for accessing remote servers over a network. While largely replaced by SSH, it’s still useful for testing connections to specific ports.

**5. curl — Command Line Tool for Transferring Data with URL Syntax**

curl http://example.com/api

* curlis a versatile tool for making HTTP requests from the command line. It supports a wide range of protocols and is often used for testing APIs and downloading files.

**6. wget — Non-interactive Network Downloader**

wget http://example.com/file.zip

* wgetis a non-interactive command-line utility for downloading files from the web. It’s particularly useful for downloading large files or mirroring entire websites.

# Section 6: Advanced Network Utilities

**1. ss — Show Socket Statistics**

ss -s

* ssprovides detailed statistics on sockets, similar tonetstat, but with more advanced features and better performance.

**2. ss — Show Listening Sockets**

ss -l

* This command displays all the sockets that are currently in a listening state, which is crucial for identifying active services on a server.

**3. tcpdump — Capture and Display Packets on a Network**

tcpdump -A -i eth0

* This command captures and displays network packets in ASCII format, useful for debugging network traffic in a human-readable format.

**4. nc — Utility for Reading from and Writing to Network Connections**

nc -zv 192.168.1.1 22

* netcatcan also be used to scan for open ports and services on a network, making it a powerful tool for network diagnostics.

**5. ss — Display Socket Statistics**

ss -t -a

* This variation of thess command shows detailed statistics for TCP sockets, including both active and listening connections.

**6. nmtui — Text User Interface for Controlling NetworkManager**

nmtui

* nmtuiprovides a text-based interface for managing NetworkManager, allowing you to configure network interfaces and connections interactively.

**7. nmcli — Command-Line Client for NetworkManager**

nmcli connection show

* nmcliis the command-line interface for NetworkManager, used for managing network connections and settings.

**8. nmcli — List Available Wi-Fi Networks**

nmcli device wifi list

* This command lists all available Wi-Fi networks, providing information about SSIDs, signal strength, and security settings.

**9. iperf — Measure TCP and UDP Bandwidth Performance**

iperf -c server\_ip

* iperfis a tool for measuring the bandwidth between two hosts, supporting both TCP and UDP protocols.

**10. fping — Quickly Ping Multiple Hosts**

fping -a -g 192.168.1.1 192.168.1.254

* fpingallows you to ping multiple hosts with a single command, making it faster than the traditional methodping for checking multiple devices.

**11. nmap — Security Scanner and Network Exploration Tool**

nmap -sP 192.168.1.0/24

* Thisnmap command scans a subnet for active hosts, providing a quick overview of devices on your network.

**12. netcat — Arbitrary TCP and UDP Connections and Listens**

nc -l 8080

* netcatcan be used as a simple TCP/UDP server, allowing you to listen on a specific port and test connections.

**13. scp — Securely Copy Files Between Hosts**

scp file.txt user@host:/path/to/destination

* Another example ofscp in action is securely transferring files between different hosts over a network.

**14. tcpdump — Capture and Display Packets on a Network**

tcpdump -i eth0 tcp port 80

* Thistcpdump command captures TCP packets on port 80, which is useful for monitoring HTTP traffic specifically.

**Conclusion:**

These commands cover a wide range of networking tasks and are invaluable for DevOps professionals managing Linux-based systems. From basic connectivity checks to advanced network diagnostics, these tools empower you to maintain and optimize your network infrastructure efficiently.

Whether you are troubleshooting network issues, monitoring performance, or securing your environment, having a solid understanding of these commands will make you a more effective and confident DevOps engineer.

I hope this helps! Follow for more such stories 😁



Enjoy it! 🍻 That’s It; we are done...

For seamless Cloud Management incorporating DevOps as the core of the methodology, reach out to us at [info@cypik.com](mailto:info@cypik.com)

[**Cypik**](https://www.linkedin.com/company/cypik)

# About the author:

My name is [**Suresh Yadav**](https://medium.com/@thesureshyadav76), and I am an experienced Linux enthusiast and DevOps engineer. I’m passionate about automating and streamlining development processes, and currently, I work as a DevOps Engineer at Cypik. I specialize in cloud technologies, with a focus on Google Cloud Platform (GCP), AWS cloud services, and DevOps tools such as Terraform, Ansible, Docker, Jenkins, Python, and Kubernetes ability to streamline operations and increase efficiency.

# Linux RoadMap

[[Medurubharathstvd](https://medium.com/@medurubharathstvd?source=post_page---byline--c6f82ca771f0---------------------------------------)](https://medium.com/@medurubharathstvd?source=post_page---byline--c6f82ca771f0---------------------------------------)

[Medurubharathstvd](https://medium.com/@medurubharathstvd?source=post_page---byline--c6f82ca771f0---------------------------------------)

·

Follow

3 min read

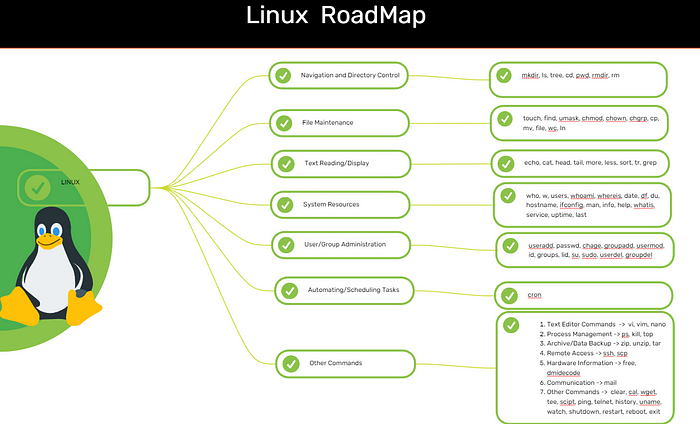
·

Feb 22, 2025

A Beginner’s Guide to Linux:

Linux is a powerful, open-source operating system valued for its flexibility, security, and widespread use across industries. Whether you’re just starting with Linux or brushing up on your skills, mastering the basics of Linux commands is crucial for efficient system management.

This article will cover key Linux concepts and commands every beginner should know.



**1. Navigation and Directory Control:**The first step in becoming comfortable with Linux is knowing how to navigate its file system. Here are some basic commands:

* **mkdir**: Creates a new directory.
* **ls**: Lists the files and directories in the current directory.
* **cd**: Changes the directory you're working in.
* **pwd**: Shows your current working directory.
* **rmdir**: Removes an empty directory.
* **rm**: Deletes files or directories.

**2. File Maintenance:**Managing files is a core part of working with Linux. Here are a few key commands:

* **touch**: Creates a new file.
* **find**: Searches for files in a directory hierarchy.
* **chmod**: Changes the permissions of a file.
* **chown**: Changes the ownership of a file.
* **cp**: Copies files or directories.
* **mv**: Moves files or directories.
* **file**: Determines the file type.
* **wc**: Counts the number of lines, words, and characters in a file.

**3. Text Reading and Display:**Linux has powerful tools for reading and displaying the contents of files. Some of the most useful commands are:

* **echo**: Displays a line of text or variables.
* **cat**: Concatenates and displays the content of files.
* **head**: Displays the first few lines of a file.
* **tail**: Displays the last few lines of a file.
* **more**: Displays a file one page at a time.
* **less**: Like more, but allows scrolling in both directions.
* **sort**: Sorts lines of text in a file.
* **tr**: Translates or deletes characters.

**4. System Resources:**Monitoring your system’s resources is key to keeping everything running smoothly. Use these commands to check the status of your system:

* **who**: Shows who is logged into the system.
* **uptime**: Displays how long the system has been running.
* **date**: Shows the current system date and time.
* **df**: Displays disk space usage.
* **du**: Shows the disk usage of files and directories.
* **top**: Displays real-time system processes and resource usage.
* **free**: Shows memory usage.

**5. User and Group Administration:**Managing users and groups is essential for system security. Here are some key commands:

* **useradd**: Adds a new user to the system.
* **passwd**: Changes the user password.
* **chage**: Changes user account aging information.
* **groupadd**: Creates a new group.
* **usermod**: Modifies a user account.
* **id**: Displays user and group ID information.
* **groups**: Shows the groups a user belongs to.
* **sudo**: Allows a permitted user to execute a command as the superuser or another user.

**6. Automating and Scheduling Tasks:**Automation is a key feature of Linux, especially in a DevOps environment. Use these commands to schedule and automate tasks:

* **cron**: Schedules tasks to run at specific times or intervals.

Cron jobs are vital for automating backups, system maintenance, and other repetitive tasks without manual intervention.

**7. Other Useful Commands**: There are a variety of other commands that come in handy when working with Linux:

* **vi, vim**: Text editors that allow you to edit files directly in the terminal.
* **ps**: Displays information about running processes.
* **kill**: Terminates a process.
* **top**: Displays real-time information about the system's resource usage.
* **scp**: Securely copies files between hosts over SSH.
* **ssh**: Connects securely to remote servers.
* **wget and curl**: Download files from the web.
* **man**: Displays the manual pages for a command

These commands round out your Linux toolkit, allowing you to do everything from editing text files to securely copying data over the network.

# The Ultimate Guide to Linux Networking Commands for DevOps Professionals

[[](https://medium.com/@thesureshyadav76?source=post_page---byline--bc7681d87b04---------------------------------------)](https://medium.com/@thesureshyadav76?source=post_page---byline--bc7681d87b04---------------------------------------)

[Suresh yadav](https://medium.com/@thesureshyadav76?source=post_page---byline--bc7681d87b04---------------------------------------)

·

Follow

8 min read

·

Aug 21, 2024

353



[**Cypik.com**](https://www.cypik.com/)

**Introduction:**In the world of DevOps, networking is a cornerstone of infrastructure management. Understanding and efficiently using Linux networking commands is essential for troubleshooting, monitoring, and managing network configurations. This guide provides a comprehensive list of Linux networking commands, complete with examples and explanations, to equip you with the tools you need to excel in your role.

# Section 1: Interface and Address Management

**1. ifconfig — Display Network Interface Information**

ifconfig

* ifconfigis a traditional command used to configure and display network interface parameters. It provides information such as IP addresses, netmasks, and broadcast addresses.

**2. ip — Show/Manipulate Routing, Devices, Policy Routing, and Tunnels**

ip address show

* Theip command is the modern replacement forifconfig. It provides comprehensive options for managing network interfaces, routes, and tunnels.

**3. hostname — Show or Set the System’s Host Name**

hostname

* Thehostname command displays or sets the system's hostname, which is used to identify the system on a network.

**4. iwconfig — Configure a Wireless Network Interface**

iwconfig

* iwconfigis used for configuring wireless network interfaces. It allows you to set parameters like SSID, frequency, and encryption keys.

**5. ethtool — Display or Change Ethernet Card Settings**

ethtool eth0

* ethtoolprovides detailed information about network interfaces and allows you to modify settings such as speed and duplex mode.

**6. resolvconf — Manage DNS Information**

resolvconf -u

* resolvconfis used to manage DNS resolver information, ensuring consistent DNS settings across the system.

**7. hostnamectl — Control the System Hostname and Related Settings**

hostnamectl status

* hostnamectlis part of systemd and allows you to view or change the system’s hostname and other related settings.

# Section 2: Route and Traffic Management

**1. route — Display or Manipulate the IP Routing Table**

route -n

* Theroute command displays or modifies the IP routing table. It’s essential for managing how packets are routed through a network.

**2. ip route — Manage Routing Tables**

ip route show

* Theip route command is part of the ip suite, used for displaying and modifying the routing table.

**3. route add — Add a New Route**

route add default gw 192.168.1.1

* This command adds a default gateway to the routing table, directing traffic destined for outside the local network.

**4. route del — Delete a Route**

route del -net 192.168.2.0 netmask 255.255.255.0

* This command removes a specific route from the routing table, which can be necessary when reconfiguring network paths.

**5. arp — Display or Modify the ARP Cache**

arp -a

* arp shows the Address Resolution Protocol (ARP) cache, mapping IP addresses to MAC addresses on the local network.

**6. iptables — Administration Tool for IPv4 Packet Filtering and NAT**

iptables -L

* iptables is a powerful utility for setting up, maintaining, and inspecting the tables of IP packet filter rules in the Linux kernel.

# Section 3: Diagnostic and Monitoring Tools

**1. ping — Send ICMP ECHO\_REQUEST to Network Hosts**

ping google.com

* ping checks the reachability of a host on the network by sending ICMP echo requests and measuring the response time.

**2. traceroute — Print the Route Packets Take to Network Host**

traceroute google.com

* traceroute shows the path that packets take from your system to a specified host, useful for diagnosing routing issues.

**3. netstat — Print Network Connections, Routing Tables, Interface Statistics**

netstat -an

* netstat provides a detailed view of network connections, routing tables, and interface statistics.

**4. ss — Display Socket Statistics**

ss -tulpn

* ss is a modern replacement for netstat, providing more detailed information about TCP, UDP, and UNIX socket connections.

**5. lsof — List Open Files**

lsof -i :80

* lsof lists open files and network connections. It’s particularly useful for identifying which processes are using specific network ports.

**6. tcpdump — Dump Traffic on a Network**

tcpdump -i eth0

* tcpdump captures and displays the packets on a network interface. It’s invaluable for diagnosing network issues at the packet level.

**7. iftop — Display Bandwidth Usage on an Interface**

iftop

* iftop shows real-time bandwidth usage on a network interface, helping you identify bandwidth-intensive processes.

**8. iptraf — Interactive Color IP LAN Monitor**

iptraf

* iptrafprovides real-time monitoring of network traffic with an interactive, color-coded display.

**9. mtr — Network Diagnostic Tool**

mtr google.com

* mtr combines the functionality of ping and traceroute, providing a real-time view of the network's performance as packets travel from source to destination.

**10. nload — Visual Representation of Incoming and Outgoing Traffic**

nload

* nload provides a graphical representation of network traffic, showing the incoming and outgoing data rates.

**11. iperf — Tool for Measuring TCP and UDP Bandwidth Performance**

iperf -c server\_ip

* iperf measures the maximum TCP and UDP bandwidth performance between two hosts, making it useful for network performance testing.

**12. fping — Quickly Ping Multiple Hosts**

fping -a -g 192.168.1.1 192.168.1.254

* fpingis similar toping but allows you to ping multiple hosts simultaneously, providing a quick overview of network reachability.

# Section 4: DNS and Connectivity Utilities

**1. dig — DNS Lookup Utility**

dig google.com

* digqueries DNS name servers, providing detailed information about DNS records and helping to troubleshoot DNS issues.

**2. nslookup — Query Internet Name Servers Interactively**

nslookup google.com

* nslookupis a user-friendly tool for querying DNS servers, providing interactive DNS queries.

**3. host — DNS Lookup Utility**

host google.com

* Thehost command performs DNS lookups and returns information about the given domain name.

**4. nmap — Network Exploration Tool and Security Scanner**

nmap -sP 192.168.1.0/24

* nmapis used for discovering hosts and services on a network, providing information about the devices and their open ports.

**5. netcat (nc)—Arbitrary TCP and UDP Connections and Listens**

nc -l 8080

* netcat, also known asnc, is a versatile tool for reading from and writing to network connections using TCP or UDP. It can be used for network diagnostics, testing, and data transfer.

# Section 5: Remote Access and File Transfer

**1. sshd — OpenSSH Daemon**

service sshd restart

* sshdis the OpenSSH server process that handles incoming SSH connections. Restarting it can resolve remote access issues.

**2. ssh-keygen — Generate, Manage, and Convert Authentication Keys for SSH**

ssh-keygen -t rsa

* ssh-keygenis used to generate, manage, and convert SSH authentication keys, enabling secure SSH connections.

**3. scp — Secure Copy (Remote File Copy Program)**

scp file.txt user@remote:/path/to/destination

* scpsecurely copies files between hosts over a network using SSH. It’s a reliable method for transferring files while maintaining security.

**4. telnet — User Interface to the TELNET Protocol**

telnet google.com 80

* telnetis a protocol for accessing remote servers over a network. While largely replaced by SSH, it’s still useful for testing connections to specific ports.

**5. curl — Command Line Tool for Transferring Data with URL Syntax**

curl http://example.com/api

* curlis a versatile tool for making HTTP requests from the command line. It supports a wide range of protocols and is often used for testing APIs and downloading files.

**6. wget — Non-interactive Network Downloader**

wget http://example.com/file.zip

* wgetis a non-interactive command-line utility for downloading files from the web. It’s particularly useful for downloading large files or mirroring entire websites.

# Section 6: Advanced Network Utilities

**1. ss — Show Socket Statistics**

ss -s

* ssprovides detailed statistics on sockets, similar tonetstat, but with more advanced features and better performance.

**2. ss — Show Listening Sockets**

ss -l

* This command displays all the sockets that are currently in a listening state, which is crucial for identifying active services on a server.

**3. tcpdump — Capture and Display Packets on a Network**

tcpdump -A -i eth0

* This command captures and displays network packets in ASCII format, useful for debugging network traffic in a human-readable format.

**4. nc — Utility for Reading from and Writing to Network Connections**

nc -zv 192.168.1.1 22

* netcatcan also be used to scan for open ports and services on a network, making it a powerful tool for network diagnostics.

**5. ss — Display Socket Statistics**

ss -t -a

* This variation of thess command shows detailed statistics for TCP sockets, including both active and listening connections.

**6. nmtui — Text User Interface for Controlling NetworkManager**

nmtui

* nmtuiprovides a text-based interface for managing NetworkManager, allowing you to configure network interfaces and connections interactively.

**7. nmcli — Command-Line Client for NetworkManager**

nmcli connection show

* nmcliis the command-line interface for NetworkManager, used for managing network connections and settings.

**8. nmcli — List Available Wi-Fi Networks**

nmcli device wifi list

* This command lists all available Wi-Fi networks, providing information about SSIDs, signal strength, and security settings.

**9. iperf — Measure TCP and UDP Bandwidth Performance**

iperf -c server\_ip

* iperfis a tool for measuring the bandwidth between two hosts, supporting both TCP and UDP protocols.

**10. fping — Quickly Ping Multiple Hosts**

fping -a -g 192.168.1.1 192.168.1.254

* fpingallows you to ping multiple hosts with a single command, making it faster than the traditional methodping for checking multiple devices.

**11. nmap — Security Scanner and Network Exploration Tool**

nmap -sP 192.168.1.0/24

* Thisnmap command scans a subnet for active hosts, providing a quick overview of devices on your network.

**12. netcat — Arbitrary TCP and UDP Connections and Listens**

nc -l 8080

* netcatcan be used as a simple TCP/UDP server, allowing you to listen on a specific port and test connections.

**13. scp — Securely Copy Files Between Hosts**

scp file.txt user@host:/path/to/destination

* Another example ofscp in action is securely transferring files between different hosts over a network.

**14. tcpdump — Capture and Display Packets on a Network**

tcpdump -i eth0 tcp port 80

* Thistcpdump command captures TCP packets on port 80, which is useful for monitoring HTTP traffic specifically.

**Conclusion:**

These commands cover a wide range of networking tasks and are invaluable for DevOps professionals managing Linux-based systems. From basic connectivity checks to advanced network diagnostics, these tools empower you to maintain and optimize your network infrastructure efficiently.

# Linux commands that every DevOps engineer should know

25 May 2024 - Shyam Mohan

Here are some essential Linux commands that every DevOps engineer should know. These commands cover a range of functionalities including file and directory management, process management, system monitoring, networking, and package management

### File and Directory Management

* ls: List directory contents.
* cd: Change the current directory.
* pwd: Print the current working directory.
* mkdir: Create a new directory.
* rm: Remove files or directories.
* cp: Copy files or directories.
* mv: Move or rename files or directories.
* touch: Create an empty file or update the timestamp of a file.
* find: Search for files and directories.
* grep: Search for patterns within files.

### File Viewing and Editing

* cat: Concatenate and display file content.
* less: View file content one page at a time.
* head: Display the first few lines of a file.
* tail: Display the last few lines of a file.
* nano / vi / vim: Text editors for file editing.

### Process Management

* ps: Display currently running processes.
* top: Display real-time system statistics and running processes.
* htop: An improved, interactive process viewer (needs to be installed separately).
* kill: Terminate processes by PID.
* killall: Terminate processes by name.
* systemctl: Manage systemd services.

### System Monitoring

* df: Display disk space usage.
* du: Display disk usage of files and directories.
* free: Display memory usage.
* uptime: Display how long the system has been running.
* iostat: Display CPU and I/O statistics (requires sysstat package).
* vmstat: Display virtual memory statistics.

### Networking

* ifconfig: Display or configure network interfaces (deprecated in favor of ip command).
* ip: Display or configure IP addresses, routes, and devices.
* ping: Check connectivity to a host.
* netstat: Network statistics (deprecated in favor of ss).
* ss: Display socket statistics.
* traceroute: Display the route packets take to a network host.
* curl: Transfer data from or to a server.
* wget: Download files from the web.

### Package Management

**Debian-based distributions (e.g., Ubuntu):**

* apt-get / apt: Package management commands.
  + sudo apt-get update
  + sudo apt-get install package\_name
  + sudo apt-get upgrade
  + sudo apt-get remove package\_name
  + sudo apt-cache search package\_name

**Red Hat-based distributions (e.g., CentOS, Fedora):**

* yum / dnf: Package management commands.
  + sudo yum install package\_name
  + sudo yum update
  + sudo yum remove package\_name
  + sudo yum search package\_name
  + sudo dnf install package\_name
  + sudo dnf update
  + sudo dnf remove package\_name
  + sudo dnf search package\_name

### User and Permission Management

* chmod: Change file permissions.
* chown: Change file owner and group.
* usermod: Modify user accounts.
* passwd: Change user password.

### Miscellaneous

* alias: Create command shortcuts.
* crontab: Schedule periodic tasks.
* tar: Archive files.
* ssh: Securely connect to remote servers.
* scp: Securely copy files between hosts.