Linux "w" Command - Day 6 Notes

The "w" command in Linux is used to show who is logged in and what they are doing. It gives a real-time snapshot of system activity.

# Syntax

w [options] [user]

# Sample Output

14:21:42 up 2 days, 3:58, 2 users, load average: 0.00, 0.03, 0.05  
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT  
john pts/0 192.168.1.10 14:20 1:00 0.10s 0.10s bash  
root pts/1 192.168.1.11 13:45 2:00 0.20s 0.20s htop

# Fields Explained

|  |  |
| --- | --- |
| Field | Description |
| USER | Username of the logged-in user. |
| TTY | Terminal the user is using. |
| FROM | Remote host or IP address. |
| LOGIN@ | Login time. |
| IDLE | Idle time. |
| JCPU | CPU time used by all processes attached to the terminal. |
| PCPU | CPU time used by the current process. |
| WHAT | Current command/process the user is running. |

# Common Options

|  |  |
| --- | --- |
| Option | Description |
| -h | Suppress header |
| -s | Short format (displays less info) |
| -f | Show FROM field (default) |

# Use Cases

- Checking who is logged in to the system.

- Monitoring what commands users are running.

- Troubleshooting performance issues.

Linux Command: cat /proc/cpuinfo

The "cat /proc/cpuinfo" command is used in Linux to display detailed information about the CPU. This data is read from the /proc virtual filesystem, which provides system information.

# Syntax

cat /proc/cpuinfo

# Example Output (Partial)

processor : 0  
vendor\_id : GenuineIntel  
cpu family : 6  
model : 142  
model name : Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz  
cpu MHz : 1992.000  
cache size : 8192 KB  
flags : fpu vme de pse tsc ...

# Fields Explained

|  |  |
| --- | --- |
| Field | Description |
| processor | The processor number (starting from 0). |
| vendor\_id | The manufacturer of the CPU. |
| cpu family | CPU family number, used to distinguish between processor generations. |
| model | Model number within the CPU family. |
| model name | Full name of the processor. |
| cpu MHz | Current operating frequency of the CPU. |
| cache size | Size of the cache memory. |
| flags | Features and capabilities supported by the CPU. |

# Use Cases

- Check CPU architecture and model details.

- Troubleshoot or monitor CPU performance.

- Verify specs of virtual machines or physical hardware.

Linux Commands: w and uptime

# 1. w Command

The "w" command displays information about the users currently on the machine and their running processes. It also provides system uptime and load averages.

## Syntax

w [options] [user]

## Example Output

14:19:06 up 5 days, 12 min, 1 user, load average: 0.00, 0.00, 0.00  
USER TTY LOGIN@ IDLE JCPU PCPU WHAT  
ec2-user pts/0 14:14 1.00s 0.02s 0.00s w

## Explanation of Output

|  |  |
| --- | --- |
| Field | Description |
| USER | Username of the logged-in user. |
| TTY | Terminal name. |
| LOGIN@ | Login time of the user. |
| IDLE | Time since last activity. |
| JCPU | Total CPU time used by all processes attached to the terminal. |
| PCPU | CPU time used by the current process. |
| WHAT | Current command/process. |

# 2. uptime Command

The "uptime" command shows how long the system has been running, how many users are currently logged in, and the system load averages for the past 1, 5, and 15 minutes.

## Syntax

uptime

## Example Output

14:19:17 up 5 days, 12 min, 1 user, load average: 0.00, 0.00, 0.00

## Explanation of Output

|  |  |
| --- | --- |
| Field | Description |
| Time | Current system time. |
| Up | System uptime. |
| Users | Number of users logged in. |
| Load average | System load over the last 1, 5, and 15 minutes. |

Linux Commands: man and whatis (using pwd)

# 1. man Command

The "man" command is used to display the manual page for a command. It provides detailed documentation, including the command's purpose, syntax, options, and examples.

## Syntax

man [command]

## Example

man pwd

## Description

Displays the manual entry for the "pwd" command, which prints the name of the current working directory.

# 2. whatis Command

The "whatis" command provides a one-line description of a command. It's useful when you want a quick summary of what a command does.

## Syntax

whatis [command]

## Example

whatis pwd

## Expected Output

pwd (1) - print name of current/working directory

# Use Cases

- Use "man" when you want in-depth info about a command.

- Use "whatis" for a quick summary of what a command does.

Purpose of Using: date → uptime → date

This document explains the purpose of running the commands `date`, `uptime`, and then `date` again in sequence. This sequence is useful in system monitoring, logging, and scripting contexts to capture time and system status clearly.

# 1. date (Before uptime)

Command:

date

## Purpose:

To capture the current system date and time before checking system uptime.

## Use Case:

Helpful for timestamping logs or recording when the monitoring started.

## Sample Output:

Mon Mar 24 02:29:09 PM UTC 2025

# 2. uptime

Command:

uptime

## Purpose:

To check how long the system has been running, how many users are logged in, and the system load averages.

## Use Case:

Used to monitor system performance and stability.

## Sample Output:

14:29:19 up 5 days, 22 min, 1 user, load average: 0.00, 0.00, 0.00

# 3. date (After uptime)

Command:

date

## Purpose:

To reconfirm the current date and time after checking the uptime. Useful for ensuring accurate logs, especially if the uptime check is part of a longer script or session.

## Use Case:

Ensures accurate and traceable time references in logs or reports.

## Sample Output:

Mon Mar 24 02:29:22 PM UTC 2025

# Summary Table

|  |  |  |
| --- | --- | --- |
| Step | Command | Purpose |
| 1 | date | Log or record the starting time |
| 2 | uptime | Check system uptime and load |
| 3 | date | Confirm or log end time of the check |

Linux Commands: df -h and du -h demo123

# 1. df -h

The "df" command is used to report file system disk space usage. The "-h" option displays the output in a human-readable format (e.g., KB, MB, GB).

## Syntax

df -h

## Sample Output

Filesystem Size Used Avail Use% Mounted on  
/dev/xvda1 30G 12G 17G 42% /  
tmpfs 497M 0 497M 0% /dev/shm

## Explanation of Columns

|  |  |
| --- | --- |
| Column | Description |
| Filesystem | The name of the disk or partition. |
| Size | Total size of the filesystem. |
| Used | Amount of space used. |
| Avail | Amount of free space available. |
| Use% | Percentage of space used. |
| Mounted on | The mount point of the filesystem. |

## Use Cases

- Check available disk space.

- Monitor storage usage on servers.

- Troubleshoot full disk issues.

# 2. du -h demo123

The "du" command is used to estimate file space usage. The "-h" option displays the output in a human-readable format. The argument "demo123" is the name of the directory whose usage is being checked.

## Syntax

du -h demo123

## Sample Output

4.0K demo123/subdir1  
8.0K demo123/subdir2  
12K demo123

## Explanation

Each line shows the size of a subdirectory. The last line shows the total size of the "demo123" directory.

## Use Cases

- Identify which folders use the most space.

- Monitor directory-level storage usage.

- Troubleshoot space issues within directories.

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Purpose of the Command: systemctl status sshd

# Command

systemctl status sshd

# Primary Purpose

The purpose of the "systemctl status sshd" command is to check the current status of the SSH (Secure Shell) server daemon (sshd) on a Linux system managed by systemd.

# What It Shows

- Whether the SSH service is running, stopped, or failed

- If it is enabled to start at boot

- Process ID (PID) and resource usage (CPU, memory)

- Time since the service was last started

- Recent SSH connection logs (valid/invalid attempts, errors)

# Use Cases

|  |  |
| --- | --- |
| Use Case | Description |
| Check connectivity | Verify if SSH service is active so remote logins work. |
| Monitor access | View recent login attempts and errors. |
| Troubleshooting | Diagnose issues like failed starts or unexpected shutdowns. |
| Admin auditing | Check when the service was started and track access. |

# Example Scenario

You try to SSH into a server and it fails. Running "systemctl status sshd" lets you immediately check:  
- Is the SSH service running?  
- Are there any recent errors or failed login attempts?  
- Has someone tampered with the configuration?

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The **last command you used** was:

systemctl status sshd

### ✅ Summary of Its Purpose:

* Checks if the **SSH service (sshd)** is **running**, **enabled**, and **functioning correctly**
* Displays **when it started**, **resource usage**, and **recent SSH log activity**
* Useful for **troubleshooting**, **security monitoring**, and **admin auditing**

If you're referring to the last Linux command (used to show login history), here's a quick rundown of that too:

## 📘 last Command (Optional Clarification)

last

### 🔍 Purpose:

* Shows a list of **last logged-in users**
* Includes **login times**, **IP addresses**, and **session durations**

Let me know which one you meant — the last command you typed or the last command in Linux — and I can generate a DOCX accordingly.  
  
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Purpose of systemctl Command

# What is systemctl?

"systemctl" is the primary command-line tool used to interact with the "systemd" init system, which is responsible for initializing and managing system services (also called daemons), targets, sockets, and other components on modern Linux distributions.

# Primary Purpose

To manage system services and states. This includes starting, stopping, restarting, enabling, disabling, and checking the status of services on the system.

# Capabilities of systemctl

|  |  |  |
| --- | --- | --- |
| Task | Command Example | Description |
| Start a service | systemctl start nginx | Starts the Nginx web server |
| Stop a service | systemctl stop docker | Stops the Docker service |
| Restart a service | systemctl restart sshd | Restarts the SSH server |
| Check status | systemctl status apache2 | Checks if Apache is running and shows logs |
| Enable at boot | systemctl enable mysql | Ensures MySQL starts automatically on boot |
| Disable at boot | systemctl disable mysql | Prevents MySQL from starting on boot |
| View logs | journalctl -u nginx | Displays logs for the Nginx service |
| List units | systemctl list-units | Lists all currently active services and system units |

# Real-World Use Cases

- A DevOps engineer needs to restart Jenkins after a configuration update.

- A system administrator needs to check if SSH is enabled and running for remote access.

- During server hardening, unnecessary services are disabled from starting on boot using systemctl.

- Logs for failed services are reviewed using journalctl to troubleshoot startup issues.

# Summary

"systemctl" provides unified, consistent management for services and system resources in systemd-based Linux. It has become the standard tool for managing service states, and is far more powerful and flexible than older tools like "service" and "chkconfig".

Managing Services with systemctl: Jenkins & Docker Example

This document explains how to manage services such as Jenkins and Docker on different servers using the "systemctl" and "service" commands in Linux. These services can be started, stopped, checked for status, or restarted as needed.

# Services and Servers

- Jenkins is running on server1

- Docker is running on server2

# Common Service Management Actions

- start

- stop

- status

- restart

# Using systemctl

Syntax:

systemctl [action] [service-name]

## Examples:

systemctl start jenkins

systemctl stop docker

systemctl status jenkins

systemctl restart docker

# Alternative (Older Syntax): Using service

Syntax:

service [service-name] [action]

## Examples:

service jenkins start

service docker stop

service jenkins status

service docker restart

# Which Command is Most Preferable?

The "systemctl" command is the preferred and modern way to manage services on systems that use systemd (most modern Linux distributions). It provides more detailed information, supports enabling services at boot, and is more powerful than the older "service" command.

|  |  |  |
| --- | --- | --- |
| Feature | systemctl | service |
| Controls both SysVinit and systemd services | ✅ Yes | 🚫 Limited |
| Provides detailed status info | ✅ Yes | ❌ Basic |
| Enable/disable at boot | ✅ Yes | ❌ No |
| Modern default in most distros | ✅ Yes | ⚠️ Deprecated |
| Secure and script-friendly | ✅ Yes | 🚫 Less preferred |

🧠 In short: Use "systemctl" for managing services on modern Linux systems. Use "service" only if you are on older systems that don’t support systemd.

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Linux Command: ps and grep for Process Monitoring (with Updated Screenshot)

This document explains how to use the "ps" command in Linux to monitor running processes, and how to enhance its effectiveness using "grep" to filter specific users or tasks. Screenshots are included to provide visual examples of the command usage and output.

# 1. ps Command Variations

## a) ps

Displays processes running in the current shell/session.

Use Case: Check if your command/script is currently running.

## b) ps -a

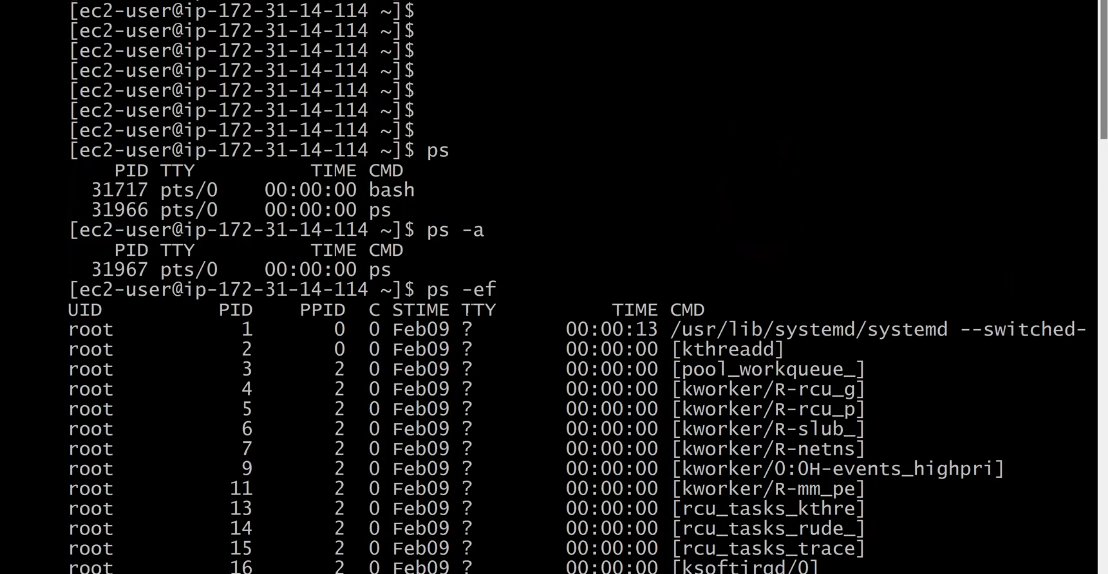
Shows processes associated with terminals, excluding session leaders.

Use Case: View interactive user processes.

## c) ps -ef

Displays all running processes in a full-format list. Useful for system-wide process monitoring.

Below is a screenshot showing these variations:



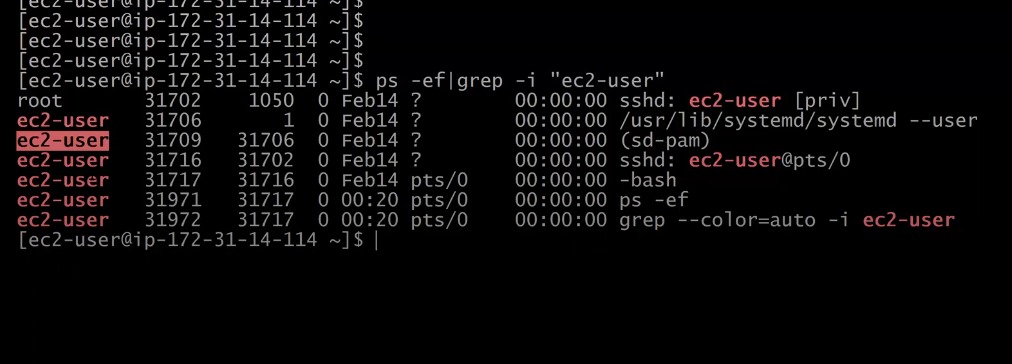
# 2. Filtering Processes with grep

You can combine "ps -ef" with "grep" to search for specific keywords, such as a username. This is especially useful for tracking all processes related to a specific user.

Example:

ps -ef | grep -i "ec2-user"

Screenshot:



## Explanation of ps -ef | grep -i "ec2-user"

- Shows all processes on the system.

- Filters output to include only lines containing "ec2-user" (case-insensitive).

- Useful for identifying user sessions, command executions, or background tasks.

## Use Cases

- Monitor your own processes using: ps -ef | grep $USER

- Find background processes or jobs by keyword.

- Debug login sessions or shell commands.

- Kill a process by first finding its PID: ps -ef | grep <name>

# Summary

The "ps" command is essential for monitoring processes on a Linux system. When combined with "grep", it becomes a powerful tool to filter and find processes for debugging, system monitoring, or automation.

# 3. Filtering by Multiple PIDs with grep

This section demonstrates how to use "ps -ef" combined with "grep -iE" to filter for multiple process IDs (PIDs). This is especially useful when you want to track related processes like parent and child or multiple important processes simultaneously.

Command Used:

ps -ef | grep -iE "31005|31717"

## Explanation:

- "ps -ef" shows a full list of all running processes.

- "grep -iE" enables extended regex search, where "|" means OR.

- This command filters for lines that match either PID 31005 or 31717.

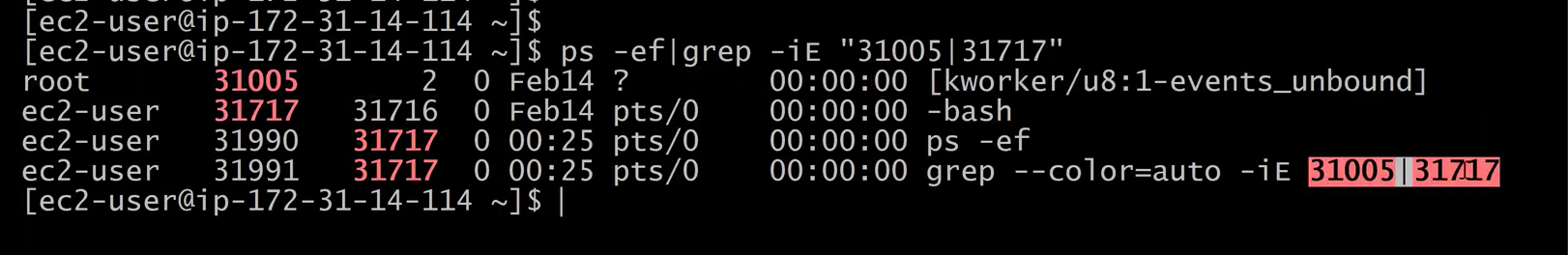
## Use Cases:

- Debugging or monitoring parent-child process chains.

- Watching multiple services or jobs simultaneously.

- Cleaning up or tracking resource usage of known PIDs.

Screenshot:



Linux Command: kill Usage (Accurate to Screenshot)

This section focuses specifically on the commands visible in the screenshot related to the "kill" command. It covers how to gracefully and forcefully terminate processes, along with verifying running processes and listing signal options.

# 1. kill 31717

- Sends SIGTERM (signal 15 by default) to request the process to stop gracefully.

# 2. ps

- Used to verify whether the target process is still running after sending SIGTERM.

# 3. kill -l

- Displays a list of all signals that can be sent using the "kill" command.

- Common examples include:

- 9: SIGKILL — forcefully stops the process.

- 15: SIGTERM — gracefully stops the process.

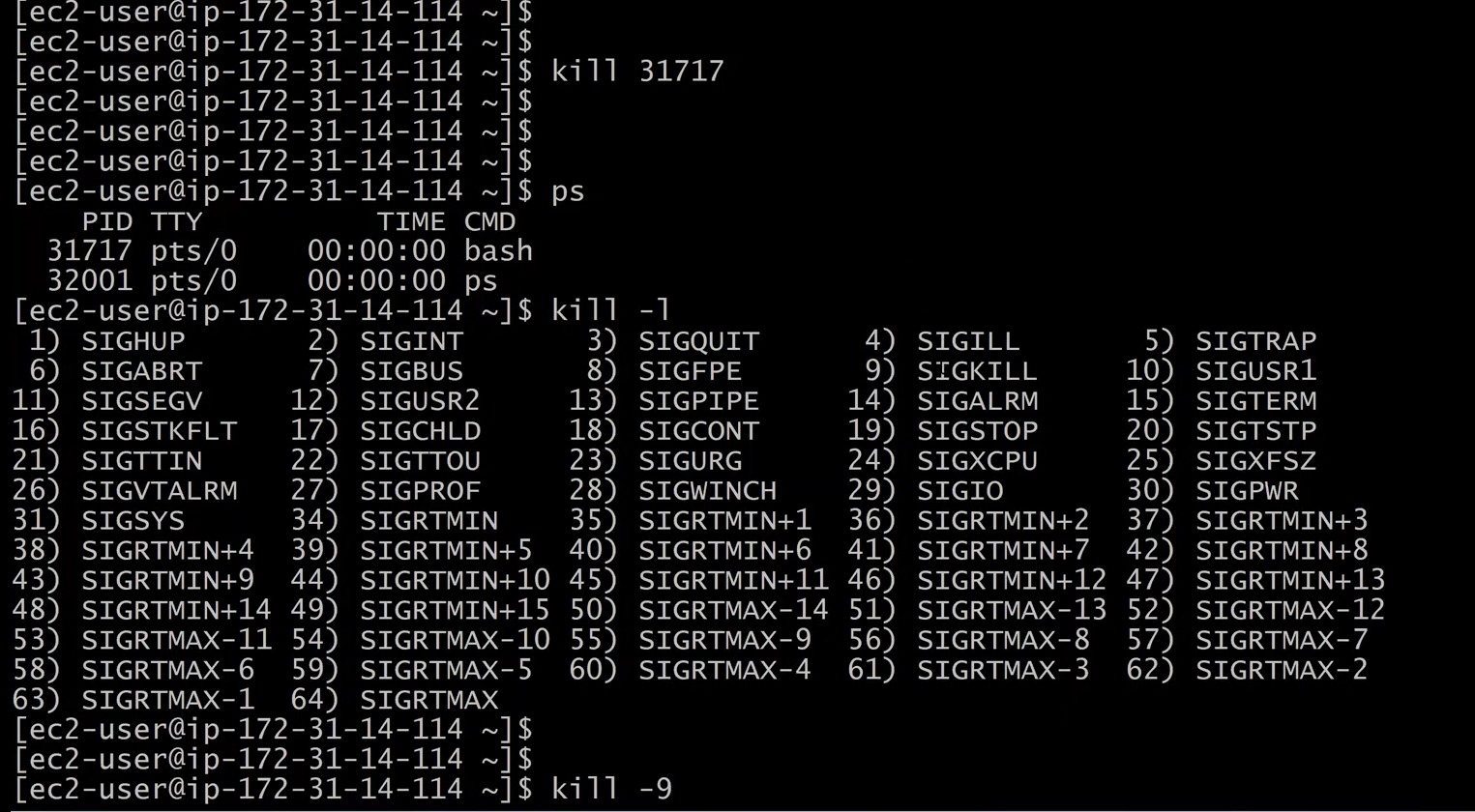
- 1: SIGHUP — often used to reload configurations.

# 4. kill -9

- The user is preparing to forcefully terminate the process with signal 9 (SIGKILL).

- This signal cannot be caught or ignored and will immediately stop the process.

Screenshot:



Linux Command: top

The "top" command in Linux is a powerful utility that provides real-time information about system performance. It displays CPU and memory usage, running tasks, and detailed information on each active process.

# Purpose of top Command

The main purpose of the "top" command is to monitor system performance, including CPU load, memory usage, and running processes. It is especially useful for identifying resource-hogging processes and understanding system behavior in real time.

# Key Sections in Output

## 1. Header Area

- Current time, system uptime, number of users, and system load averages over 1, 5, and 15 minutes.

## 2. Task Summary

- Shows total number of processes and their current states: running, sleeping, stopped, or zombie.

## 3. CPU Usage

- us: User-space CPU usage  
- sy: System (kernel) usage  
- id: Idle time  
- wa: Waiting for I/O

## 4. Memory Usage

- Shows total, used, free memory and cache/buffer memory details.

## 5. Process Table

- PID: Process ID  
- USER: Owner  
- %CPU and %MEM: CPU and Memory usage  
- COMMAND: Name of the command/process

# How to Exit top

To exit the top command, simply press the "q" key.

# Common Use Cases

- Monitor system resource usage

- Identify high CPU or memory consuming processes

- Debug performance issues in real time

Linux Command: Zipping a Directory with zip

This document demonstrates how to zip a directory using the "zip" command in Linux. It covers the installation of the zip utility, directory inspection using tree, zipping a folder, and transferring/unzipping it.

# 1. Viewing Directory Structure with tree

Command used:

tree demo123/

Purpose: To view the structure and contents of the demo123 folder before compressing it.

# 2. Attempting to Zip (Failed Due to Missing zip Utility)

Command used:

zip -r demo123.zip demo123/

Error received: zip: command not found

The system did not have the zip utility installed, so the command failed.

# 3. Troubleshooting zip Utility

Commands used:

man zip  
whereis zip

These confirmed that zip was not installed or available.

# 4. Installing zip Utility

Command used:

sudo yum install zip -y

This installs the zip utility using the yum package manager on RHEL/CentOS-based systems.

# 5. Zipping the Folder

After installation, the zip command was run again successfully to compress demo123.

[Screenshot: Zip installation and zipping process]

# 6. Verifying the Created zip File

Command used:

ls -lrth

This confirms that demo123.zip was created and is visible in the directory.

[Screenshot: demo123.zip verification]

# 7. Transferring and Unzipping the Zip File

## A. Transferring the zip File

Option 1: Using scp to Transfer to Another System

scp demo123.zip username@remote\_host:/target/directory/  
Example:  
scp demo123.zip ec2-user@192.168.1.10:/home/ec2-user/

Option 2: Move to Another Local User

sudo mv demo123.zip /home/otheruser/  
sudo chown otheruser:otheruser /home/otheruser/demo123.zip

## B. How to Unzip the File

Basic unzip command:

unzip demo123.zip

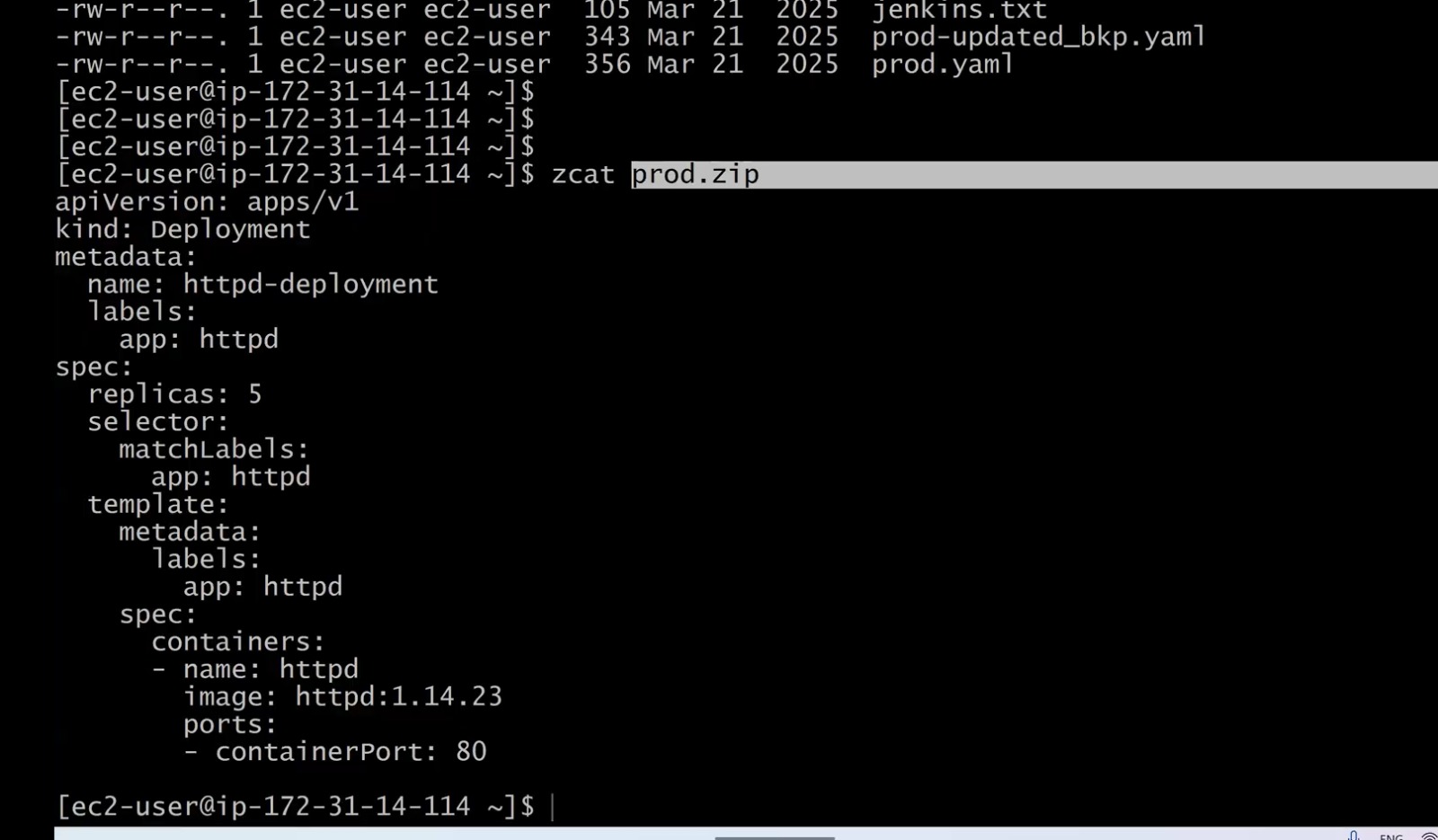
To install unzip if not available:

sudo yum install unzip -y # For RHEL/CentOS/Amazon Linux  
sudo apt install unzip -y # For Debian/Ubuntu

To unzip into a specific directory:

unzip demo123.zip -d target\_directory/

Clarifying zcat vs zip: Why zcat prod.zip Might Work



This screenshot shows the result of running `zcat prod.zip`, which surprisingly printed YAML content. This is likely due to the file being misnamed or formatted differently than a standard ZIP archive.

# 🧠 Command Shown:

zcat prod.zip

# ⚠️ Important Clarification:

`zcat` is intended for viewing the contents of `.gz` (gzip-compressed) files — not `.zip` files.  
If "zcat prod.zip" works, it is likely because:

- The file is actually a `.gz` file renamed as `.zip`  
- The file was compressed using gzip but misnamed  
- The content is plain text and zcat is able to display it, though this is not reliable behavior

# 🔎 How to Verify File Type:

Use the `file` command to check what kind of file it really is:

file prod.zip

If the output is:  
- "gzip compressed data" → then it is a `.gz` file  
- "Zip archive data" → then zcat should NOT be used

# ✅ Correct Ways to Handle zip Files:

To view files inside a .zip archive:

unzip -l prod.zip

To read content of a file inside without extracting:

unzip -p prod.zip

# 📋 Summary: zcat vs unzip

|  |  |  |
| --- | --- | --- |
| Command | File Type | Purpose |
| zcat file.gz | .gz | View file contents without unzipping |
| gunzip file.gz | .gz | Decompress gzip file |
| unzip -l file.zip | .zip | List contents of a zip archive |
| unzip -p file.zip | .zip | Print contents without extracting |

--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

# 🔍 Using zgrep on Compressed Files

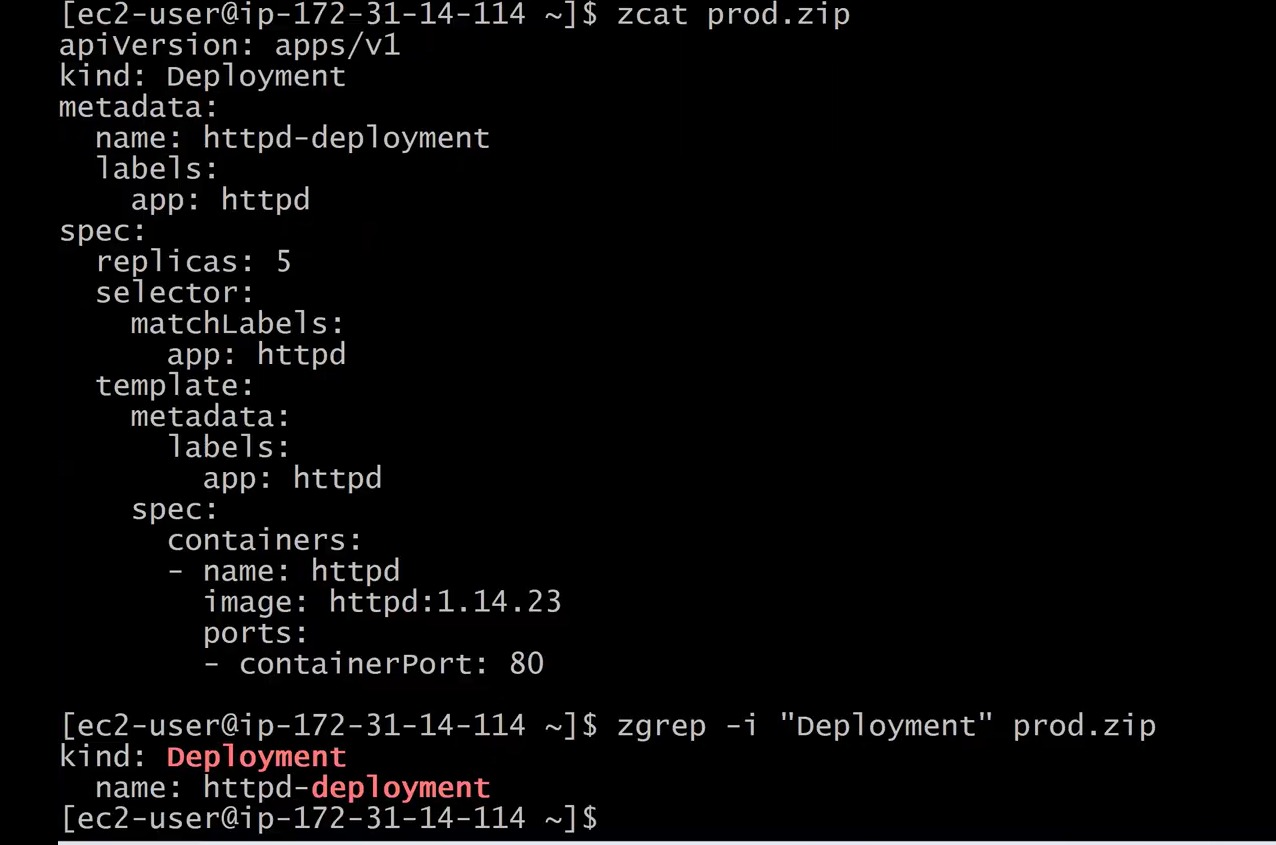
`zgrep` is used to search for patterns within `.gz` compressed files without extracting them first. It works just like `grep` but is built to work with compressed files.

Command shown:

zgrep -i "Deployment" prod.zip

In this case, even though `prod.zip` has a `.zip` extension, the command still worked, which suggests that the file may have been compressed using gzip format.

`-i` makes the search case-insensitive. It found occurrences of the word "Deployment" and highlighted them in the output.



The screenshot above demonstrates the use of `zgrep` to locate 'Deployment' inside the compressed file.

========================================================================================================================================================================================================================================  
completed the class 6 Next watch class 7