

Project Title: Linux Server Hardening & Automation

Linux Server Hardening & Automation:

This project is about securing and setting up a Linux server using best practices for safety, automation, and monitoring. The goal is to make the server more secure against cyber threats while making it easier to manage. By strengthening security and automating tasks, the project helps reduce weaknesses in the system, lessen manual work, and improve the overall performance of the server.

Key Features & Implementations:

1. User Management & Access Control:

- Created and managed user accounts with restricted privileges.
- Implemented SSH key-based authentication to enhance security.
- Configured sudo privileges for authorized users to prevent unauthorized system modifications.

2. Firewall Security & Intrusion Prevention:

- Configured **UFW (Uncomplicated Firewall)** and **iptables** to allow only necessary traffic.
- Installed and configured **Fail2Ban** to protect against brute-force attacks.
- Restricted SSH access to specific IP addresses and non-standard ports.

3. Automated Backups & Log Management:

- Scheduled automated system and data backups using **cron jobs** and **rsync**.
- Configured **Auditd** for logging security events and monitoring system changes.
- Set up log rotation to prevent disk space issues.

4. System Monitoring & Performance Optimization:

- Deployed monitoring tools such as **top**, **htop**, **iostat**, and **sar** for real-time system performance tracking.
- Configured email alerts for critical system events and failures.
- Monitored disk usage, CPU load, and memory consumption to optimize performance.

5. Automation & Maintenance:

- Automated security updates and system patches to keep the server up to date.
- Implemented scheduled tasks using **cron** to clean temporary files and logs.
- Configured scripts to auto-restart critical services upon failure.

Technologies Used:

- **AWS:** EC2
- **Linux Distributions:** Ubuntu, CentOS
- **Security Tools:** SSH, Fail2Ban, UFW, Auditd
- **Automation & Monitoring:** Bash Scripting, Cron Jobs, Systemd
- **Backup & Logging:** Rsync, Logrotate, Auditd

Outcome & Impact:

This project **significantly improved the security, efficiency, and resilience** of the Linux server by applying comprehensive security hardening and automation strategies. The combination of secure user management, firewall rules, and automated maintenance enhanced both the **security posture** and the **operational efficiency** of the system.

Key Achievements and Impact:

Enhanced Security and Threat Resilience

- Hardened the server against common attack vectors such as **brute-force attacks**, **port scanning**, and **unauthorized access**.
- Fine-tuned **firewall rules** using UFW to allow only necessary traffic, reducing exposure to external threats.
- Implemented **Fail2Ban** to detect and block malicious login attempts, improving real-time threat response.

Streamlined User Management and Access Control

- Established a structured user management framework using **least privilege principles** and secure sudo policies.
- Configured **key-based SSH authentication** and restricted SSH access to trusted IP ranges, reducing attack surfaces.
- Ensured that access to sensitive resources was controlled and monitored, improving overall system integrity.

Automation and Reduced Manual Workload

- Developed **Bash scripts** to automate system updates, log rotation, and security patching, minimizing human error and administrative effort.
- Configured **Cron Jobs** to automate backups, system health checks, and resource usage monitoring, ensuring consistent system performance.
- Reduced the time required for routine maintenance and improved overall system responsiveness.

Proactive Monitoring and Data Integrity

- Configured **Auditd** to track security-related events and user activity for forensic analysis and compliance reporting.
- Set up real-time alerts for system health, resource usage, and security incidents, enabling faster incident response.
- Automated backups ensured data integrity and minimized downtime in case of system failures.

Business and Professional Impact:

- The improved security and automation reduced downtime, increased system reliability, and enhanced data protection.
- The automated maintenance and proactive monitoring allowed for faster troubleshooting and recovery from failures.
- The project demonstrated strong expertise in **Linux system administration**, **security hardening**, and **infrastructure automation**, making the skill set highly valuable for **DevOps** and **IT security roles**.
- The project's structured approach to security and automation positioned the system to handle increased workloads and future scaling requirements without compromising security.

Summary

This project not only strengthened the Linux server's defense against cyber threats but also optimized its operational efficiency. The combination of **firewall hardening**, **secure access control**, and **automated maintenance** created a highly secure and self-sustaining server environment. The skills and methodologies applied in this project are highly transferable to **enterprise IT infrastructure**, making it a valuable asset for any DevOps or cybersecurity role.

Step 1: Set Up an AWS EC2 Instances

Inbox - gauravpawar3514@gmail.com | Instances | EC2 | ap-south-1 | EC2 Instance Connect | ap-south-1 | EC2 Instance Connect | ap-south-1 | Yum command not found | +

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#Instances:v=3;\$case=true%5C,client:false;\$regex=tags:false%5C,client:false

IAM | EC2 | S3

aws | Search [Alt+S]

EC2 | Instances

Last updated less than a minute ago

Find Instance by attribute or tag (case-sensitive)

All states

Instances (1/2) Info

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS |
|---------|---------------------|----------------|---------------|-------------------|---------------|-------------------|--|
| project | i-0c543b1dd6bae0403 | Running | t2.micro | 2/2 checks passed | View alarms + | ap-south-1b | ec2-13-232-83-249.ap-south-1.compute.amazonaws.com |
| test | i-043c345bb15d6150d | Running | t2.micro | 2/2 checks passed | View alarms + | ap-south-1b | ec2-43-204-21-135.ap-south-1.compute.amazonaws.com |

i-0c543b1dd6bae0403 (project)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary

| | | |
|------------------------------------|---|--|
| Instance ID i-0c543b1dd6bae0403 | Public IPv4 address 13.232.83.249 open address | Private IPv4 addresses 172.31.1.55 |
| IPv6 address - | Instance state Running | Public IPv4 DNS ec2-13-232-83-249.ap-south-1.compute.amazonaws.com open address |

CloudShell | Feedback

26°C | Partly cloudy

Search

Amazon Web Services | Asia Pacific (Mumbai) | bala

22:18 | 17-03-2025

Step 2: User Management and Secure SSH Access

In Step 2 of the Linux Server Hardening project, the focus is on improving the security of user access and SSH (Secure Shell) connections to protect the server from unauthorized access and brute-force attacks. This step includes setting up secure user accounts, enforcing strong password policies, and configuring SSH for secure remote access.

◆ **User Management**

1. User Creation and Privilege Assignment

- Create user accounts for administrators and regular users.
- Assign proper permissions to limit access based on user roles.
- Avoid using the root account for direct access to reduce security risks.

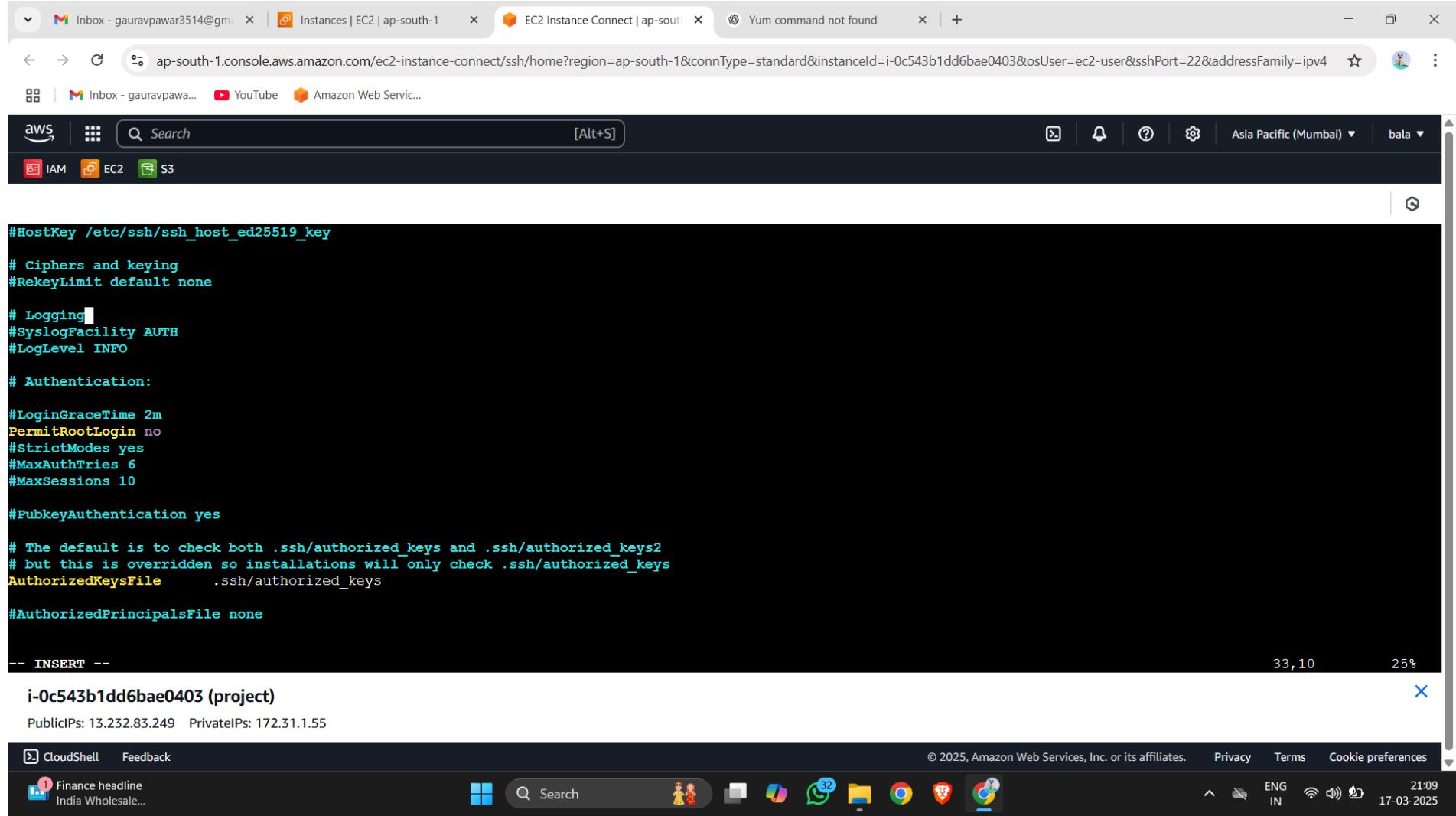
2. Password Policy Enforcement

- Define password strength requirements (e.g., minimum length, complexity).
- Set password expiration and rotation policies to prevent long-term vulnerabilities.
- Implement account lockouts and warnings for multiple failed login attempts.

Goal:

- Ensure that only authorized users have access to the server.
- Minimize the risk of unauthorized access through SSH.
- Improve overall system security by strengthening user access controls and authentication mechanisms.

Step 3: Disable Root SSH Login



The screenshot shows a terminal window with the AWS Lambda console interface at the top. The main area displays the contents of the `/etc/ssh/sshd_config` file. The configuration includes various parameters such as host keys, ciphers, logging, authentication methods, and key files. A cursor is visible near the bottom of the configuration text.

```
#HostKey /etc/ssh/ssh_host_ed25519_key
# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:
#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile      .ssh/authorized_keys

#AuthorizedPrincipalsFile none

-- INSERT --
i-0c543b1dd6bae0403 (project)
PublicIPs: 13.232.83.249 PrivateIPs: 172.31.1.55
```

At the bottom of the terminal window, there is a status bar with the following information:

- CloudShell
- Feedback
- © 2025, Amazon Web Services, Inc. or its affiliates.
- Privacy
- Terms
- Cookie preferences
- CloudWatch Metrics icon
- ENG IN
- 21:09
- 17-03-2025

The screenshot shows a web browser window with several tabs open. The tabs include:

- Inbox - gauravpawar3514@gmail.com
- Instances | EC2 | ap-south-1
- EC2 Instance Connect | ap-south-1
- Yum command not found

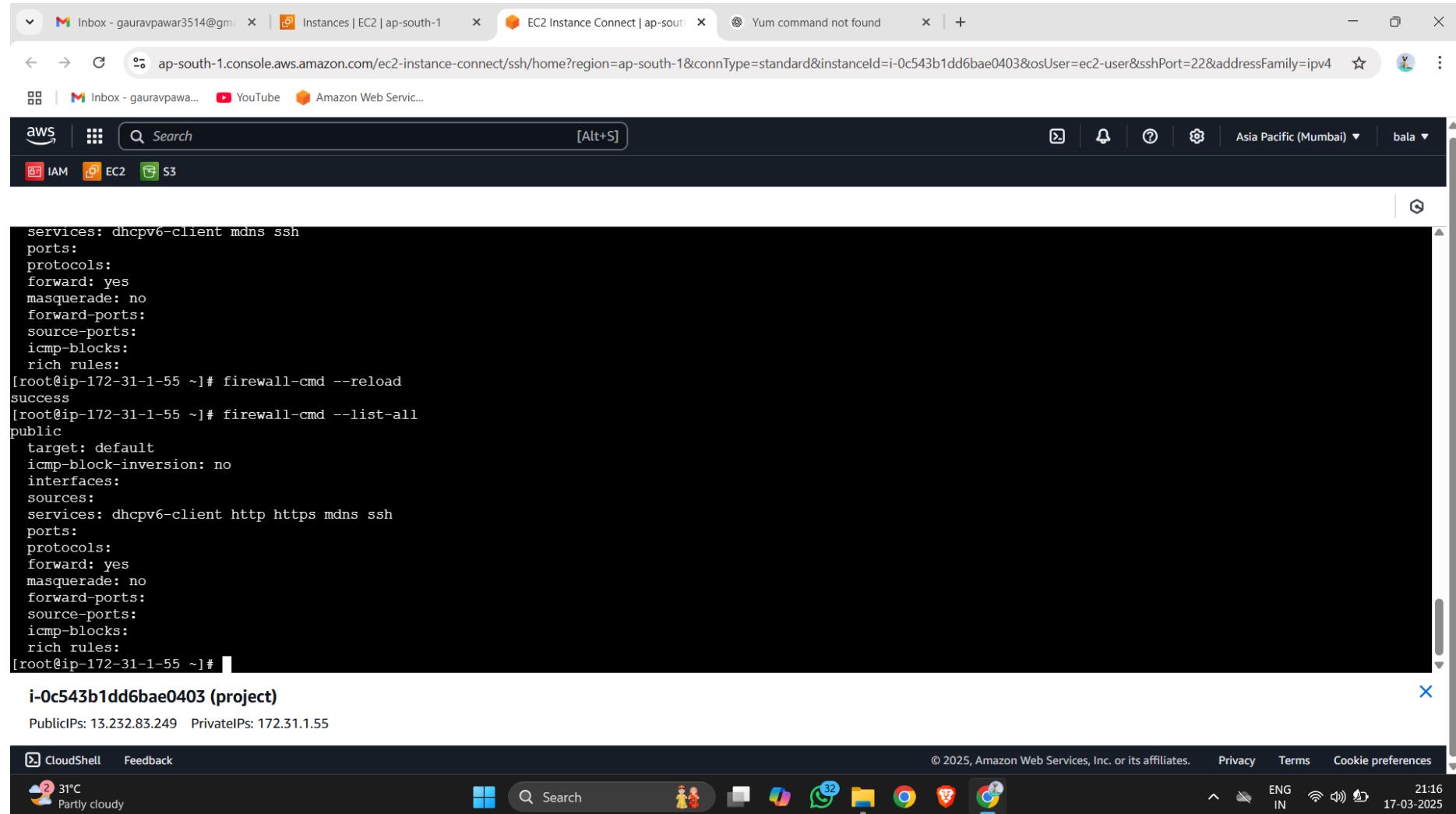
The main content area displays a terminal session in AWS CloudShell. The terminal output is as follows:

```
#PubkeyAuthentication yes
# Expect .ssh/authorized_keys2 to be disregarded by default in future.
#AuthorizedKeysFile      .ssh/authorized_keys .ssh/authorized_keys2
#AuthorizedPrincipalsFile none
#AuthorizedKeysCommand none
#AuthorizedKeysCommandUser nobody
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
#KbdInteractiveAuthentication no
-- INSERT --
```

Below the terminal output, the session identifier is shown as **i-09385a4c67c852511 (project)**. The status bar at the bottom indicates PublicIPs: 3.109.211.226 and PrivateIPs: 172.31.4.67.

The AWS CloudShell interface includes a navigation bar with links to IAM, EC2, and S3, and a footer with standard AWS links like CloudShell, Feedback, Privacy, Terms, and Cookie preferences. The footer also shows the current date and time as 17-03-2025, and weather information for 32°C and Partly cloudy.

Step 4: Configure Firewall: Configuring Firewall to Allow SSH, HTTP, and HTTPS Traffic



The screenshot shows the AWS CloudShell interface with a terminal window displaying the configuration of a Linux firewall (firewall-cmd) on an Amazon EC2 instance. The terminal output is as follows:

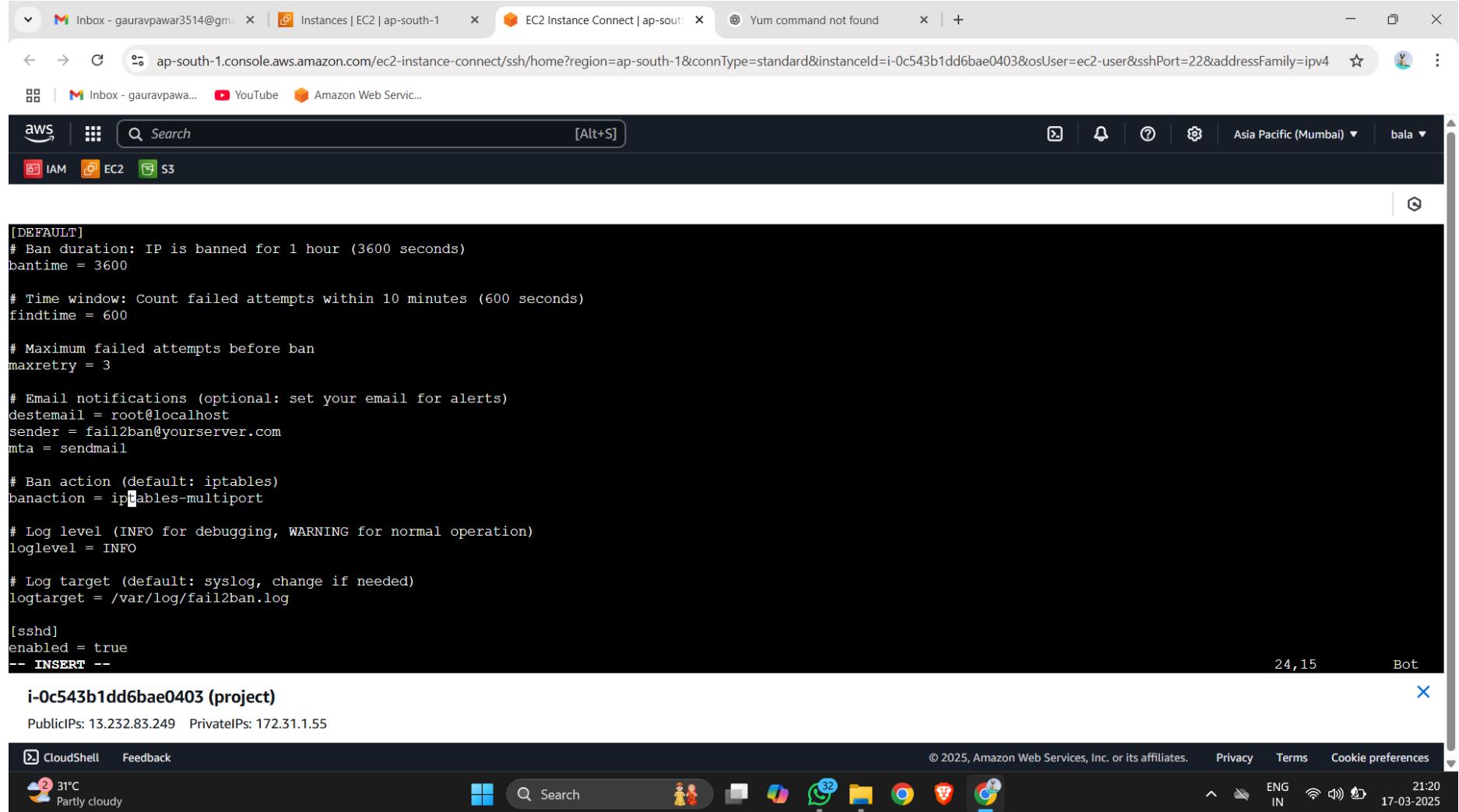
```
services: dhcpcv6-client mdns ssh
ports:
protocols:
forward: yes
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
[root@ip-172-31-1-55 ~]# firewall-cmd --reload
success
[root@ip-172-31-1-55 ~]# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpcv6-client http https mdns ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@ip-172-31-1-55 ~]#
```

Below the terminal, the instance identifier **i-0c543b1dd6bae0403 (project)** is shown along with its PublicIPs: 13.232.83.249 and PrivateIPs: 172.31.1.55.

The AWS CloudShell interface includes a header bar with tabs for Gmail, Instances, EC2 Instance Connect, and Yum command not found. The main navigation bar at the top has links for IAM, EC2, and S3. The bottom navigation bar includes CloudShell, Feedback, and various system icons like weather, search, file manager, and browser.

Step 5: Protect Against Brute Force Attacks with Fail2Ban

5.1 Configuring Fail2Ban with jail.local File



```
[DEFAULT]
# Ban duration: IP is banned for 1 hour (3600 seconds)
bantime = 3600

# Time window: Count failed attempts within 10 minutes (600 seconds)
findtime = 600

# Maximum failed attempts before ban
maxretry = 3

# Email notifications (optional: set your email for alerts)
destemail = root@localhost
sender = fail2ban@yourserver.com
mta = sendmail

# Ban action (default: iptables)
banaction = iptables-multiport

# Log level (INFO for debugging, WARNING for normal operation)
loglevel = INFO

# Log target (default: syslog, change if needed)
logtarget = /var/log/fail2ban.log

[sshd]
enabled = true
-- INSERT --
i-0c543b1dd6bae0403 (project)
PublicIPs: 13.232.83.249 PrivateIPs: 172.31.1.55
```

The screenshot shows a web browser window with multiple tabs open. The active tab is titled 'Inbox - gauravpawar3514@gmail.com'. Other tabs include 'Instances | EC2 | ap-south-1', 'EC2 Instance Connect | ap-south-1', and 'Yum command not found'. The URL in the address bar is 'ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=ap-south-1&connType=standard&instanceId=i-0c543b1dd6bae0403&osUser=ec2-user&sshPort=22&addressFamily=ipv4'. The main content area of the browser displays the 'jail.local' configuration file for Fail2Ban. The file contains several sections and parameters related to SSH protection. At the bottom of the configuration, there is a placeholder 'INSERT' preceded by a double-dash. Below this, the instance ID 'i-0c543b1dd6bae0403 (project)' and its public and private IP addresses are listed. The browser's status bar at the bottom right shows the date '17-03-2025' and time '21:20'. The overall interface is the AWS Management Console.

5.2 Verify Fail2Ban SSHD Status and Configuration Before Login Attempts

Inbox - gauravpawar3514@gmail.com Instances | EC2 | ap-south-1 EC2 Instance Connect | ap-south-1 Yum command not found

ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?region=ap-south-1&connType=standard&instanceId=i-0c543b1dd6bae0403&osUser=ec2-user&sshPort=22&addressFamily=ipv4

aws Search [Alt+S]

IAM EC2 S3

Asia Pacific (Mumbai) bala

```
|`- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
|`- Currently banned: 0
|`- Total banned: 0
`- Banned IP list:
[root@ip-172-31-1-55 ~]# systemctl restart fail2ban
[root@ip-172-31-1-55 ~]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|`- Currently failed: 1
|`- Total failed: 1
`- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
|`- Currently banned: 0
|`- Total banned: 0
`- Banned IP list:
[root@ip-172-31-1-55 ~]# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|`- Currently failed: 0
|`- Total failed: 1
`- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
|`- Currently banned: 0
|`- Total banned: 0
`- Banned IP list:
[root@ip-172-31-1-55 ~]#
```

i-0c543b1dd6bae0403 (project)

PublicIPs: 13.232.83.249 PrivateIPs: 172.31.1.55

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Finance headline India Wholesale...

Search

21:33 17-03-2025

Step 6: Enable System Auditing with Auditd

In Step 6 of the Linux Server Hardening project, the goal is to enable and configure Auditd (Linux Auditing System) to monitor and log system events. Auditd records security-relevant information such as file access, user logins, and process activity, providing detailed insight into system activity and helping to detect security breaches and suspicious behavior.

- ◆ **Purpose of Auditd:**

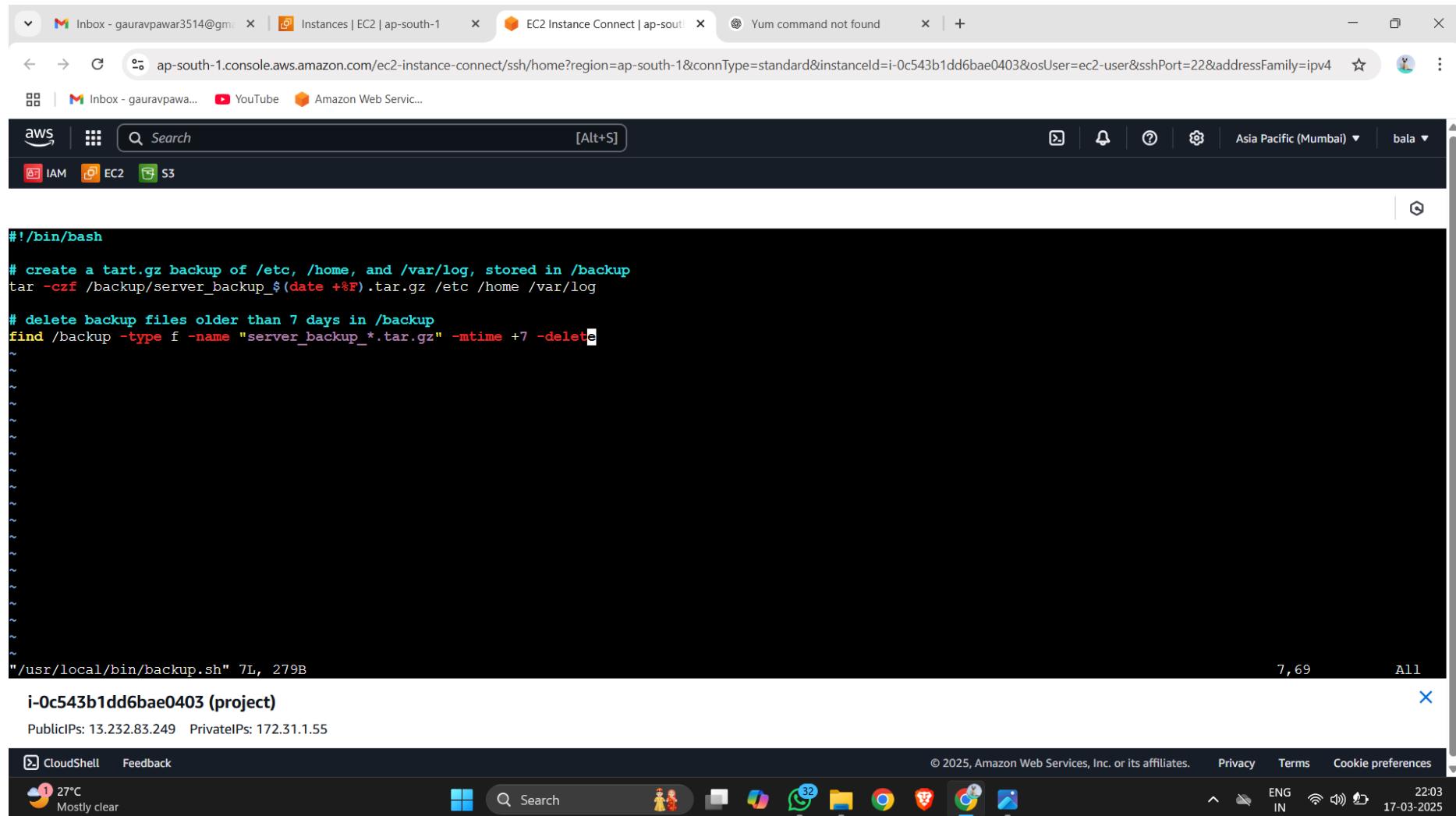
- Tracks system calls and security events.
- Logs unauthorized file modifications and user actions.
- Helps in investigating security incidents and breaches.
- Provides detailed logs for compliance and auditing purposes.

- ◆ **Key Configurations in Auditd:**

1. Install and Enable Auditd
2. Configure Audit Rules
3. Set Audit Log Rotation
4. Secure Audit Logs
5. Monitor and Analyze Logs

Step 7: Automate Backups and Maintenance with Cron Jobs

7.1 Create a Backup Script: Daily Backup Script with Auto Cleanup of Files Older Than 7 Days



The screenshot shows a web-based terminal session on an AWS EC2 instance. The terminal window displays a bash script for creating a backup and deleting old files. The script is as follows:

```
#!/bin/bash

# create a tar.gz backup of /etc, /home, and /var/log, stored in /backup
tar -czf /backup/server_backup_$(date +%F).tar.gz /etc /home /var/log

# delete backup files older than 7 days in /backup
find /backup -type f -name "server_backup_*tar.gz" -mtime +7 -delete
```

The terminal window also shows the command `"/usr/local/bin/backup.sh"` was run, resulting in 7L and 279B of output. Below the terminal, the instance ID `i-0c543b1dd6bae0403` and project information are displayed, along with public and private IP addresses. The bottom of the screen shows the AWS navigation bar and system status icons.

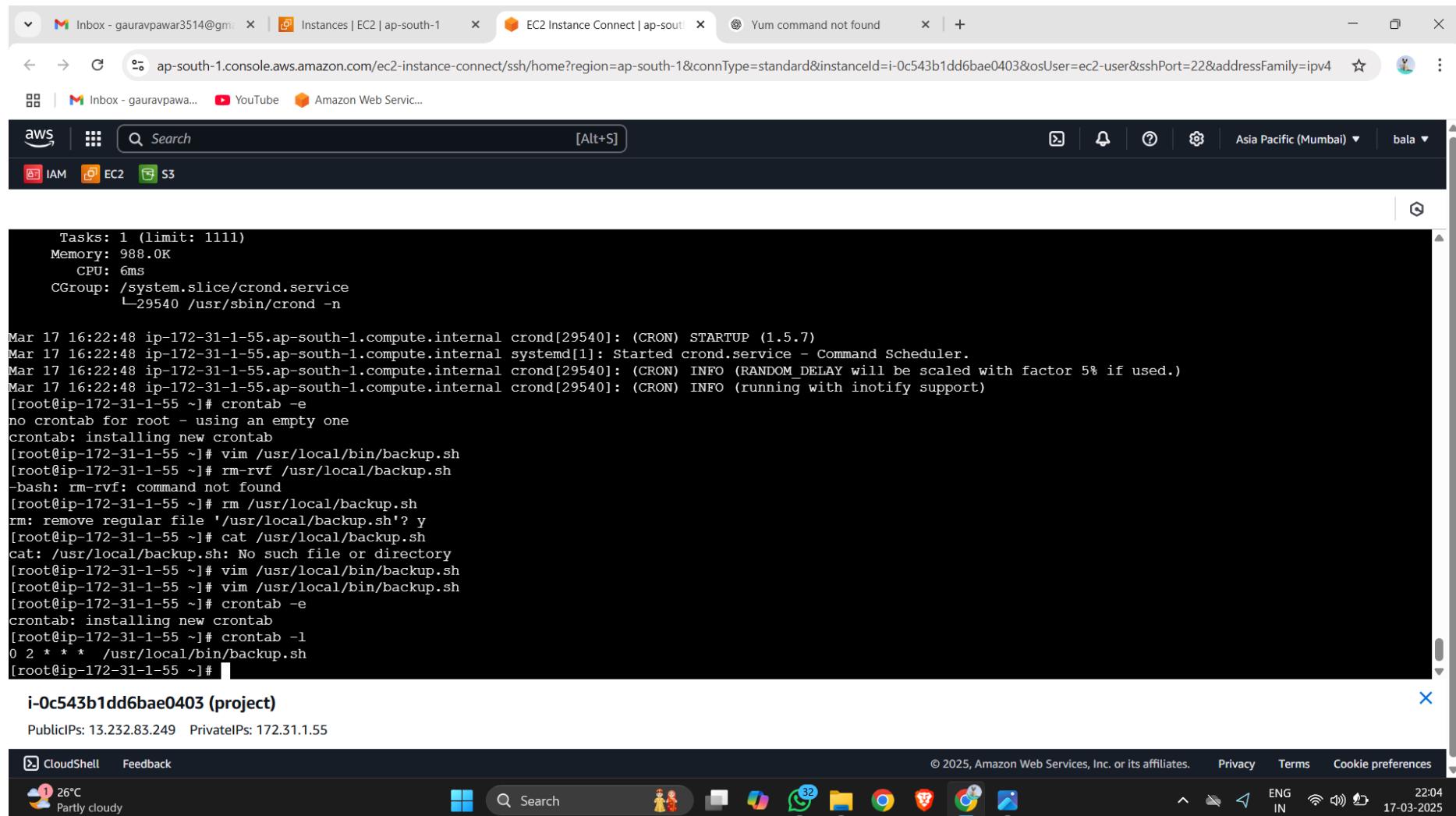
7.2 Schedule with Cron: Runs every day at exactly 2:00 AM server time.

The screenshot shows a CloudShell terminal window with the following content:

```
0 2 * * * /usr/local/bin/backup.sh
```

At the bottom of the terminal, it shows the command was run in `/tmp/crontab.5nJeK0` and consumed 36B of memory. The terminal interface includes a status bar with the instance ID `i-0c543b1dd6bae0403 (project)`, public and private IP addresses, and a timestamp of `17-03-2025 22:04`.

7.3 Reviewing Scheduled Cron Jobs: To ensure proper execution, timing, and system resource management.



The screenshot shows a browser window with multiple tabs open, including 'Inbox - gauravpawar3514@gmail.com', 'Instances | EC2 | ap-south-1', 'EC2 Instance Connect | ap-south-1', and 'Yum command not found'. The main content area displays a terminal session on an AWS EC2 instance. The terminal output shows system resource usage (Tasks: 1, Memory: 988.0K, CPU: 6ms, CGroup: /system.slice/crond.service) and a log of cron daemon startup and configuration steps. The user runs 'crontab -e' to edit the crontab file, removes a backup script, and adds a new cron entry for daily backups. The bottom of the terminal shows the project ID 'i-0c543b1dd6bae0403 (project)', public and private IP addresses, and a weather widget indicating 26°C and partly cloudy conditions. The browser interface includes standard navigation buttons, a search bar, and a header with AWS branding and account information.

```
Tasks: 1 (limit: 1111)
Memory: 988.0K
CPU: 6ms
CGroup: /system.slice/crond.service
└─29540 /usr/sbin/crond -n

Mar 17 16:22:48 ip-172-31-1-55.ap-south-1.compute.internal crond[29540]: (CRON) STARTUP (1.5.7)
Mar 17 16:22:48 ip-172-31-1-55.ap-south-1.compute.internal systemd[1]: Started crond.service - Command Scheduler.
Mar 17 16:22:48 ip-172-31-1-55.ap-south-1.compute.internal crond[29540]: (CRON) INFO (RANDOM_DELAY will be scaled with factor 5% if used.)
Mar 17 16:22:48 ip-172-31-1-55.ap-south-1.compute.internal crond[29540]: (CRON) INFO (running with inotify support)
[root@ip-172-31-1-55 ~]# crontab -e
no crontab for root - using an empty one
crontab: installing new crontab
[root@ip-172-31-1-55 ~]# vim /usr/local/bin/backup.sh
[root@ip-172-31-1-55 ~]# rm-rvf /usr/local/backup.sh
-bash: rm-rvf: command not found
[root@ip-172-31-1-55 ~]# rm /usr/local/backup.sh
rm: remove regular file '/usr/local/backup.sh'? y
[root@ip-172-31-1-55 ~]# cat /usr/local/backup.sh
cat: /usr/local/backup.sh: No such file or directory
[root@ip-172-31-1-55 ~]# vim /usr/local/bin/backup.sh
[root@ip-172-31-1-55 ~]# vim /usr/local/bin/backup.sh
[root@ip-172-31-1-55 ~]# crontab -e
crontab: installing new crontab
[root@ip-172-31-1-55 ~]# crontab -l
0 2 * * * /usr/local/bin/backup.sh
[root@ip-172-31-1-55 ~]#
```

i-0c543b1dd6bae0403 (project)

PublicIPs: 13.232.83.249 PrivateIPs: 172.31.1.55

CloudShell Feedback

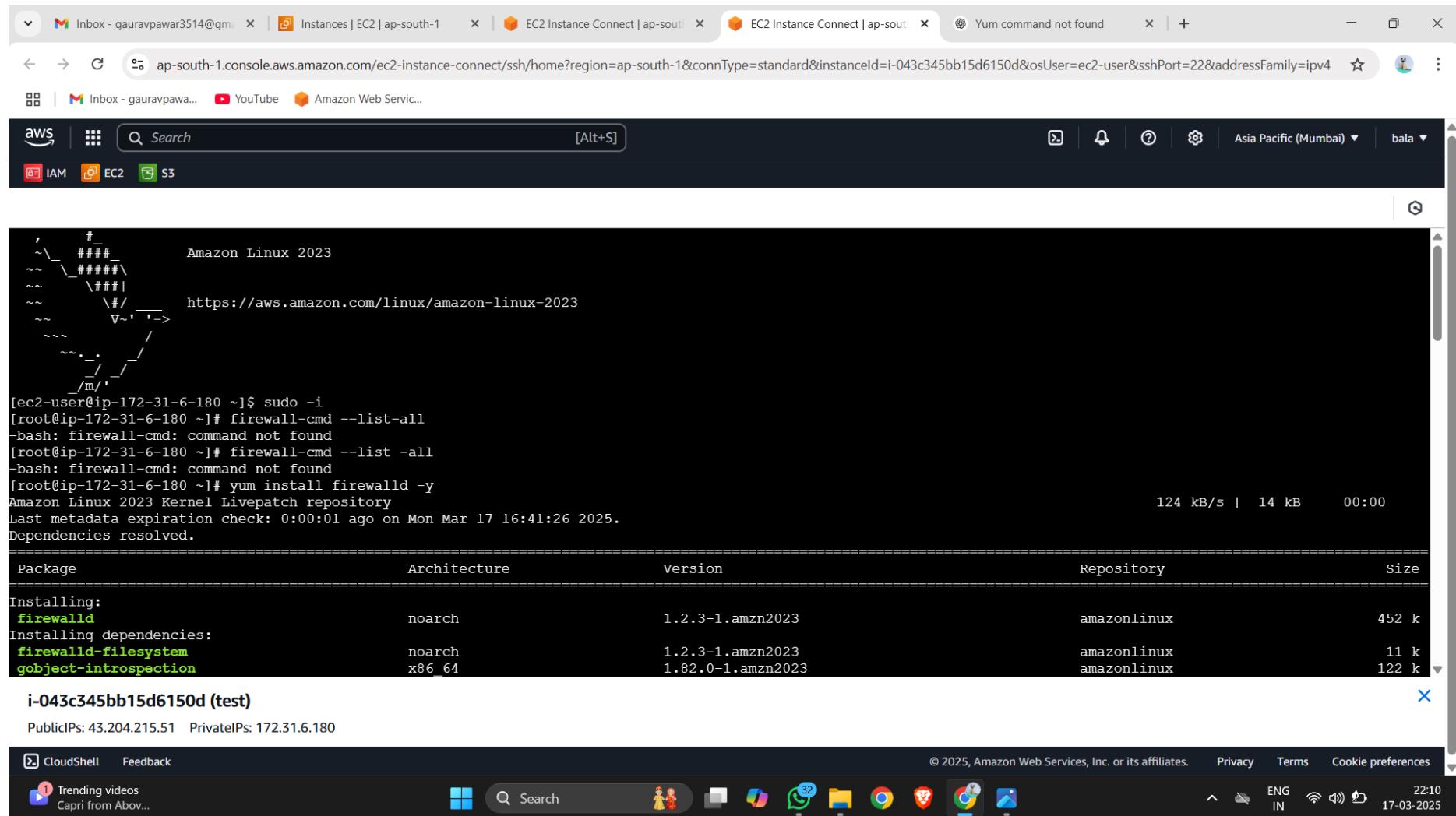
26°C Partly cloudy

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

ENG IN 22:04 17-03-2025

Step 8: Testing SSH Access with a Different User on the Target Machine

8.1 Configure Firewall: Allowing SSH on the Testing Machine

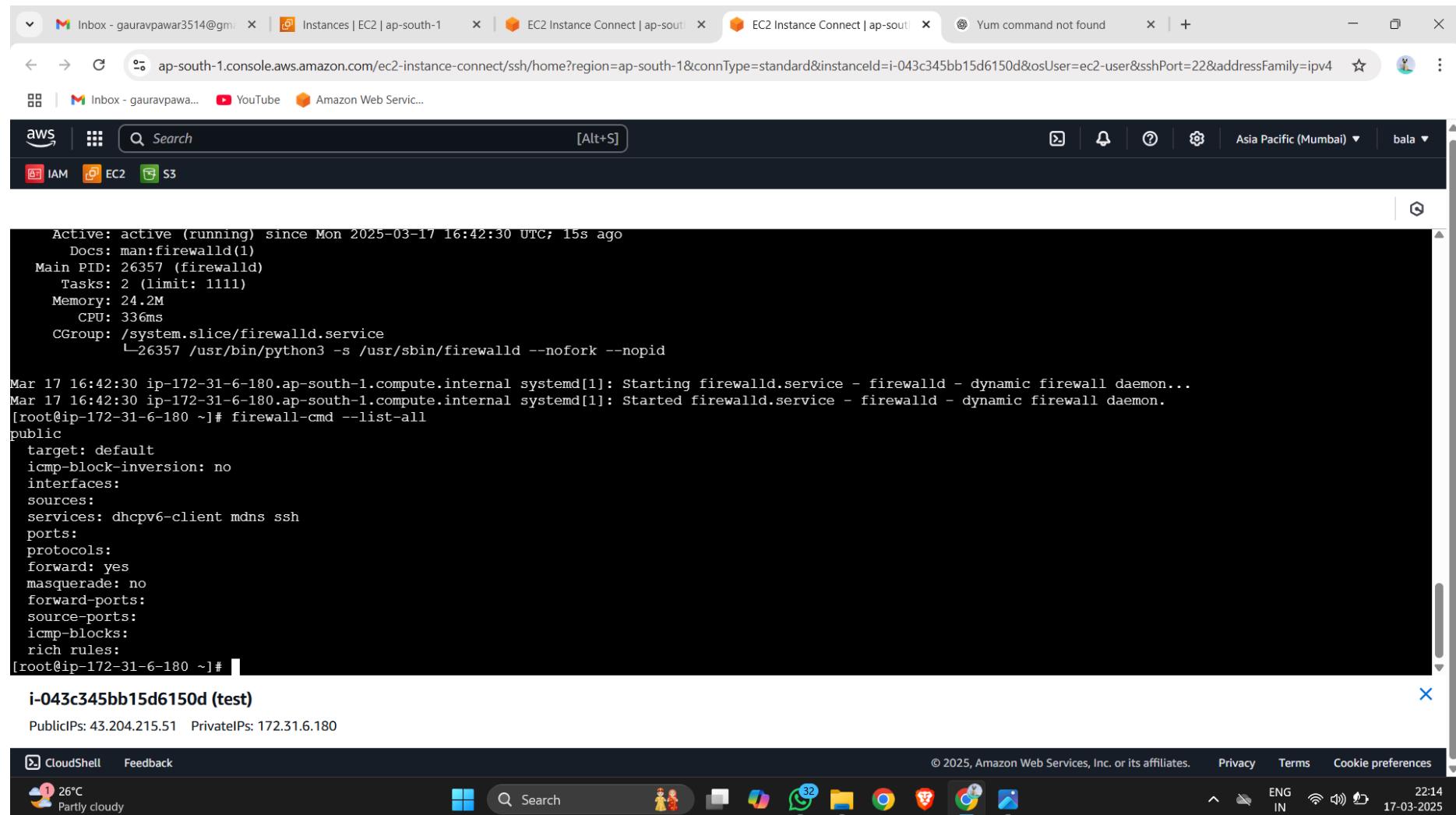


The screenshot shows a web browser with multiple tabs open. The main content area displays a terminal session on an Amazon Linux 2023 instance. The terminal shows the user attempting to install the firewalld package via yum, but encountering errors due to missing firewall-cmd commands. The user then installs firewalld directly, which completes successfully.

```
# Amazon Linux 2023
# https://aws.amazon.com/linux/amazon-linux-2023
[ec2-user@ip-172-31-6-180 ~]$ sudo -i
[root@ip-172-31-6-180 ~]# firewall-cmd --list-all
bash: firewall-cmd: command not found
[root@ip-172-31-6-180 ~]# firewall-cmd --list-all
bash: firewall-cmd: command not found
[root@ip-172-31-6-180 ~]# yum install firewalld -y
Amazon Linux 2023 Kernel Livepatch repository
Last metadata expiration check: 0:00:01 ago on Mon Mar 17 16:41:26 2025.
Dependencies resolved.
=====
Package           Architecture      Version       Repository      Size
=====
Installing:
firewalld        noarch          1.2.3-1.amzn2023   amazonlinux    452 k
Installing dependencies:
firewalld-filesystem  noarch          1.2.3-1.amzn2023   amazonlinux    11 k
gobject-introspection x86_64         1.82.0-1.amzn2023   amazonlinux    122 k
=====
i-043c345bb15d6150d (test)
PublicIPs: 43.204.215.51 PrivateIPs: 172.31.6.180
```

The terminal also shows the user navigating to the AWS CloudShell interface, where they run the command `firewall-cmd --list-all`. This command fails because it is not installed. The user then installs firewalld using `yum`, which completes successfully. Finally, the user runs `firewall-cmd --list-all` again, which now succeeds.

8.2 Verifying and Analyzing Firewall Rules: Including allowing SSH access for secure remote connections.



```
Active: active (running) since Mon 2025-03-17 16:42:30 UTC; 15s ago
  Docs: man:firewalld(1)
Main PID: 26357 (firewalld)
  Tasks: 2 (limit: 1111)
 Memory: 24.2M
   CPU: 336ms
 CGroup: /system.slice/firewalld.service
         └─26357 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

Mar 17 16:42:30 ip-172-31-6-180.ap-south-1.compute.internal systemd[1]: Starting firewalld.service - firewalld - dynamic firewall daemon...
Mar 17 16:42:30 ip-172-31-6-180.ap-south-1.compute.internal systemd[1]: Started firewalld.service - firewalld - dynamic firewall daemon.

[root@ip-172-31-6-180 ~]# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: dhcpcv6-client mdns ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
[root@ip-172-31-6-180 ~]#
```

i-043c345bb15d6150d (test)

Public IPs: 43.204.215.51 Private IPs: 172.31.6.180

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

26°C Partly cloudy Search 32 22:14 ENG IN 17-03-2025

8.3 Verifying SSH Connectivity: Access with 3 Incorrect Password Attempts Within 10 Minutes

```
public
target: default
icmp-block-inversion: no
interfaces:
sources:
services: dhcpcv6-client mdns ssh
ports:
protocols:
forward: yes
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
[root@ip-172-31-6-180 ~]# ssh root@13.232.83.249
The authenticity of host '13.232.83.249 (13.232.83.249)' can't be established.
ED25519 key fingerprint is SHA256:cDipyJgb9k3Tdnb/xkA9XFgDeNT+xMpiuJC+mTl+3/I.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.232.83.249' (ED25519) to the list of known hosts.
root@13.232.83.249's password:
Permission denied, please try again.
root@13.232.83.249's password:
Permission denied, please try again.
root@13.232.83.249's password:
root@13.232.83.249: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[root@ip-172-31-6-180 ~]#
```

i-043c345bb15d6150d (test)

Public IPs: 43.204.215.51 Private IPs: 172.31.6.180

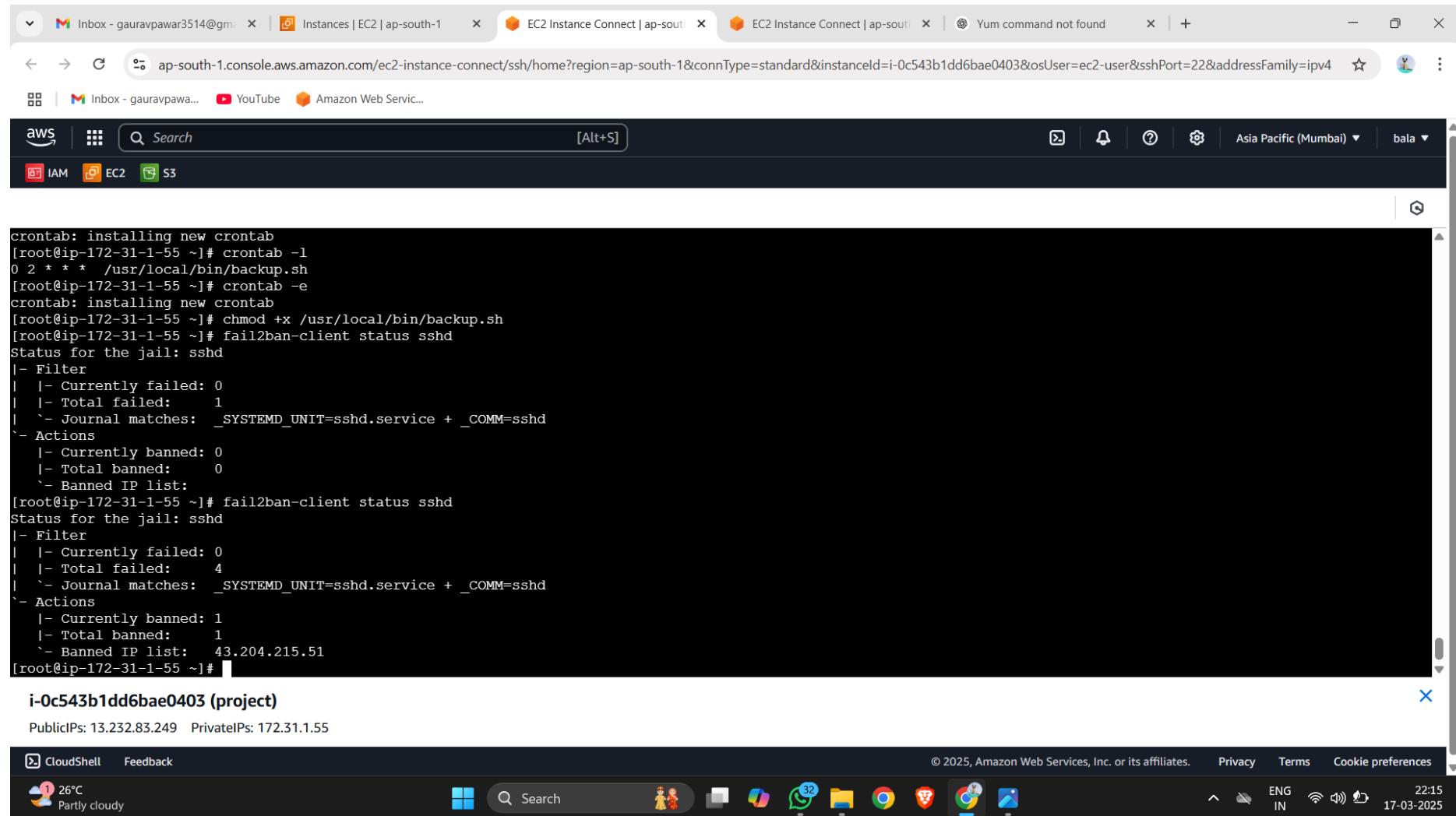
CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

26°C Partly cloudy Search 32 22:15 ENG IN 17-03-2025

8.4 Fail2Ban IP Blocking: After 3 Failed Attempts Within 10 Minutes the IP will remain blocked for 10 minutes

```
sources:
services: dhcpcv6-client mdns ssh
ports:
protocols:
forward: yes
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
[root@ip-172-31-6-180 ~]# ssh root@13.232.83.249
The authenticity of host '13.232.83.249 (13.232.83.249)' can't be established.
ED25519 key fingerprint is SHA256:cDipyJgb9k3TdnB/xkA9XFgDeNT+xMpiuJC+mT1+3/I.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.232.83.249' (ED25519) to the list of known hosts.
root@13.232.83.249's password:
Permission denied, please try again.
root@13.232.83.249's password:
Permission denied, please try again.
root@13.232.83.249's password:
root@13.232.83.249: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
[root@ip-172-31-6-180 ~]# ssh root@13.232.83.249
ssh: connect to host 13.232.83.249 port 22: Connection refused
[root@ip-172-31-6-180 ~]# ssh root@13.232.83.249
ssh: connect to host 13.232.83.249 port 22: Connection refused
[root@ip-172-31-6-180 ~]# 
```

Step 9: Checking Fail2Ban Status and Banned IPs on the Target Machine



```
crontab: installing new crontab
[root@ip-172-31-1-55 ~]# crontab -l
0 2 * * * /usr/local/bin/backup.sh
[root@ip-172-31-1-55 ~]# crontab -e
crontab: installing new crontab
[root@ip-172-31-1-55 ~]# chmod +x /usr/local/bin/backup.sh
[root@ip-172-31-1-55 ~]# fail2ban-client status sshd
Status for the jail: sshd
`- Filter
|  |- Currently failed: 0
|  |- Total failed: 1
|  `- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 0
   |- Total banned: 0
   `- Banned IP list:
[root@ip-172-31-1-55 ~]# fail2ban-client status sshd
Status for the jail: sshd
`- Filter
|  |- Currently failed: 0
|  |- Total failed: 4
|  `- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
   |- Currently banned: 1
   |- Total banned: 1
   `- Banned IP list: 43.204.215.51
[root@ip-172-31-1-55 ~]#
```

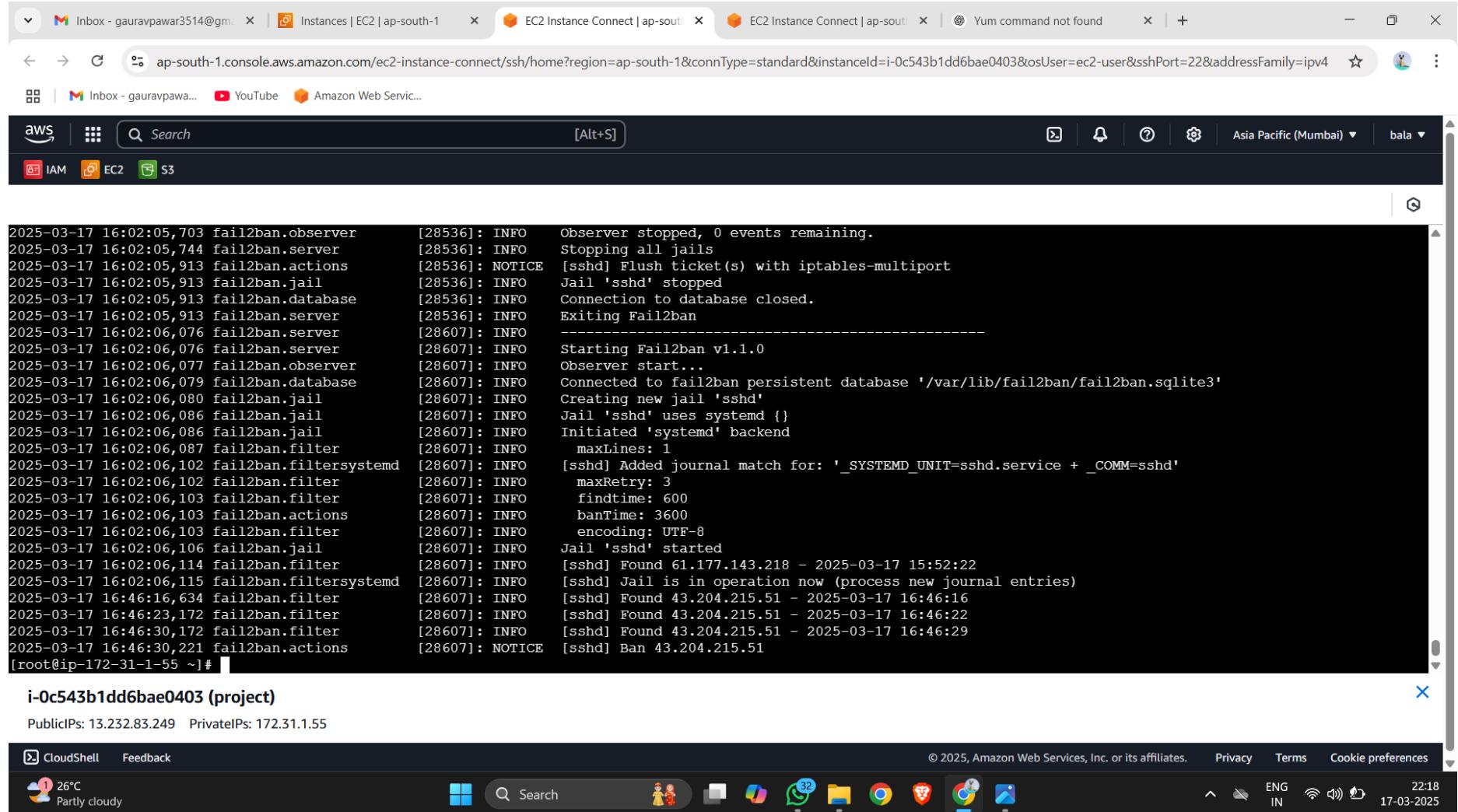
i-0c543b1dd6bae0403 (project)

PublicIPs: 13.232.83.249 PrivateIPs: 172.31.1.55

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

26°C Partly cloudy Search 32 22:15 ENG IN 17-03-2025

Step 10: Reviewing Logs for Failed SSH Login Attempts on the Target Machine



The screenshot shows a terminal window on an AWS EC2 instance. The window title is "EC2 Instance Connect | ap-south-1". The terminal displays log entries from the fail2ban service. The logs show the initialization of the fail2ban system, creation of jails, and monitoring of journal entries for new SSH login attempts. A specific entry at the end indicates a ban was applied to an IP address.

```
2025-03-17 16:02:05,703 fail2ban.observer [28536]: INFO Observer stopped, 0 events remaining.
2025-03-17 16:02:05,744 fail2ban.server [28536]: INFO Stopping all jails
2025-03-17 16:02:05,913 fail2ban.actions [28536]: NOTICE [sshd] Flush ticket(s) with iptables-multiport
2025-03-17 16:02:05,913 fail2ban.jail [28536]: INFO Jail 'sshd' stopped
2025-03-17 16:02:05,913 fail2ban.database [28536]: INFO Connection to database closed.
2025-03-17 16:02:05,913 fail2ban.server [28536]: INFO Exiting Fail2ban
2025-03-17 16:02:06,076 fail2ban.server [28607]: INFO -----
2025-03-17 16:02:06,076 fail2ban.server [28607]: INFO Starting Fail2ban v1.1.0
2025-03-17 16:02:06,077 fail2ban.observer [28607]: INFO Observer start...
2025-03-17 16:02:06,079 fail2ban.database [28607]: INFO Connected to fail2ban persistent database '/var/lib/fail2ban/fail2ban.sqlite3'
2025-03-17 16:02:06,080 fail2ban.jail [28607]: INFO Creating new jail 'sshd'
2025-03-17 16:02:06,086 fail2ban.jail [28607]: INFO Jail 'sshd' uses systemd {}
2025-03-17 16:02:06,086 fail2ban.jail [28607]: INFO Initiated 'systemd' backend
2025-03-17 16:02:06,087 fail2ban.filter [28607]: INFO maxLines: 1
2025-03-17 16:02:06,102 fail2ban.filtersystemd [28607]: INFO [sshd] Added journal match for: '_SYSTEMD_UNIT=sshd.service + _COMM=sshd'
2025-03-17 16:02:06,102 fail2ban.filter [28607]: INFO maxRetry: 3
2025-03-17 16:02:06,103 fail2ban.filter [28607]: INFO findtime: 600
2025-03-17 16:02:06,103 fail2ban.actions [28607]: INFO banTime: 3600
2025-03-17 16:02:06,103 fail2ban.filter [28607]: INFO encoding: UTF-8
2025-03-17 16:02:06,106 fail2ban.jail [28607]: INFO Jail 'sshd' started
2025-03-17 16:02:06,114 fail2ban.filter [28607]: INFO [sshd] Found 61.177.143.218 - 2025-03-17 15:52:22
2025-03-17 16:02:06,115 fail2ban.filtersystemd [28607]: INFO [sshd] Jail is in operation now (process new journal entries)
2025-03-17 16:46:16,634 fail2ban.filter [28607]: INFO [sshd] Found 43.204.215.51 - 2025-03-17 16:46:16
2025-03-17 16:46:23,172 fail2ban.filter [28607]: INFO [sshd] Found 43.204.215.51 - 2025-03-17 16:46:22
2025-03-17 16:46:30,172 fail2ban.filter [28607]: INFO [sshd] Found 43.204.215.51 - 2025-03-17 16:46:29
2025-03-17 16:46:30,221 fail2ban.actions [28607]: NOTICE [sshd] Ban 43.204.215.51
[root@ip-172-31-1-55 ~]#
```

i-0c543b1dd6bae0403 (project)

Public IPs: 13.232.83.249 Private IPs: 172.31.1.55

CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

26°C Partly cloudy 32 22:18 ENG IN 17-03-2025

Final Checklist:

- ✓ Secure SSH login with password authentication
- ✓ Hardened firewall rules using UFW
- ✓ Protected against brute-force attacks with Fail2Ban
- ✓ Set up automated backups and system updates with Cron
- ✓ Configured system auditing with Auditd

Final Outcome:

Successfully configured and secured a Linux server by implementing robust user management, firewall security, and secure SSH access. Automated backups and system monitoring were set up using Bash scripting and Cron jobs, ensuring consistent system health and quick recovery in case of failures. Security measures, including Fail2Ban and UFW, were applied to protect against unauthorized access and brute-force attacks. Comprehensive auditing with Auditd was configured to monitor and log system activity for improved security and compliance.