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.

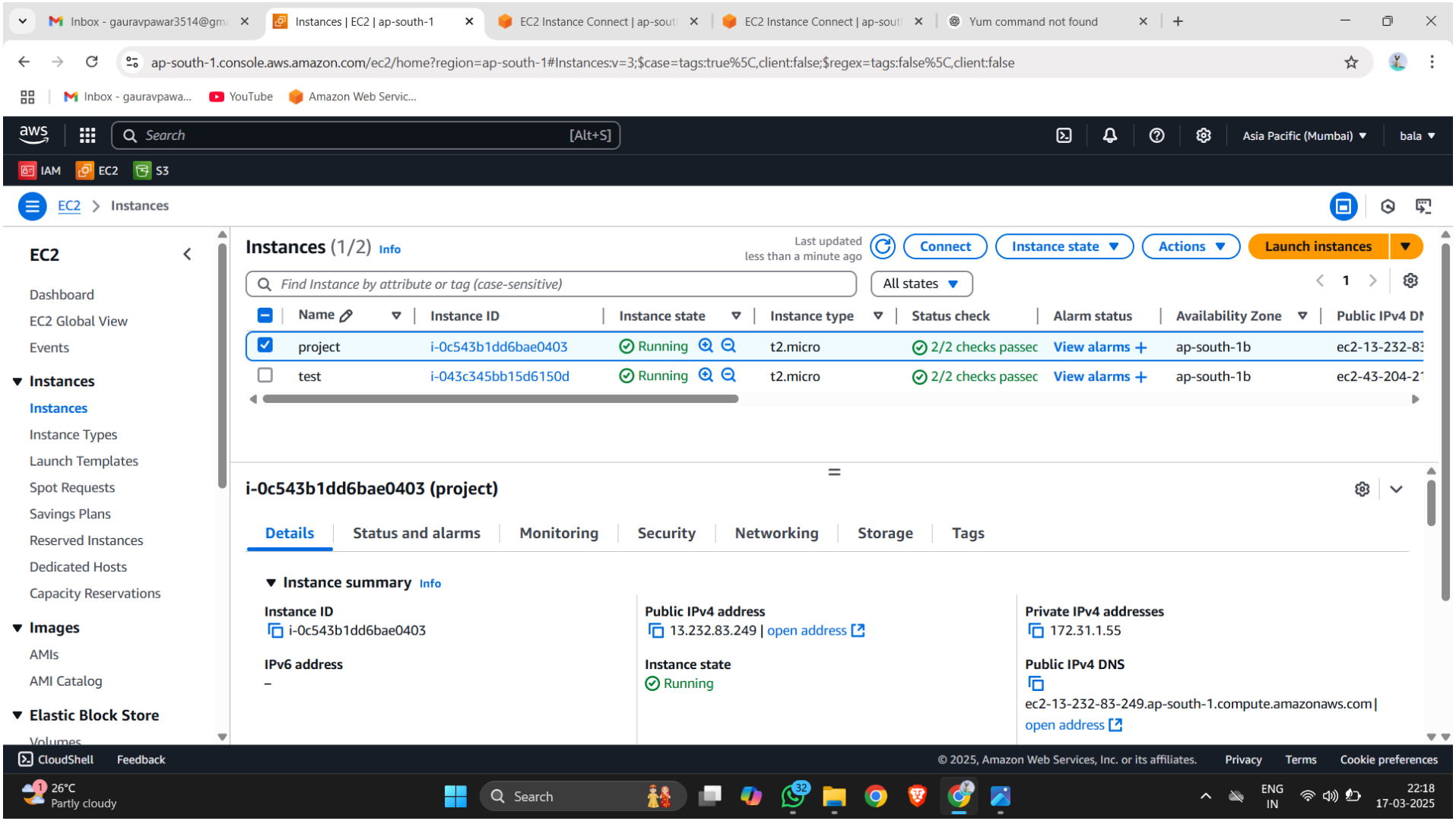
Business and Professional Impact:

- 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



Step 2: User Management and Secure SSH Access

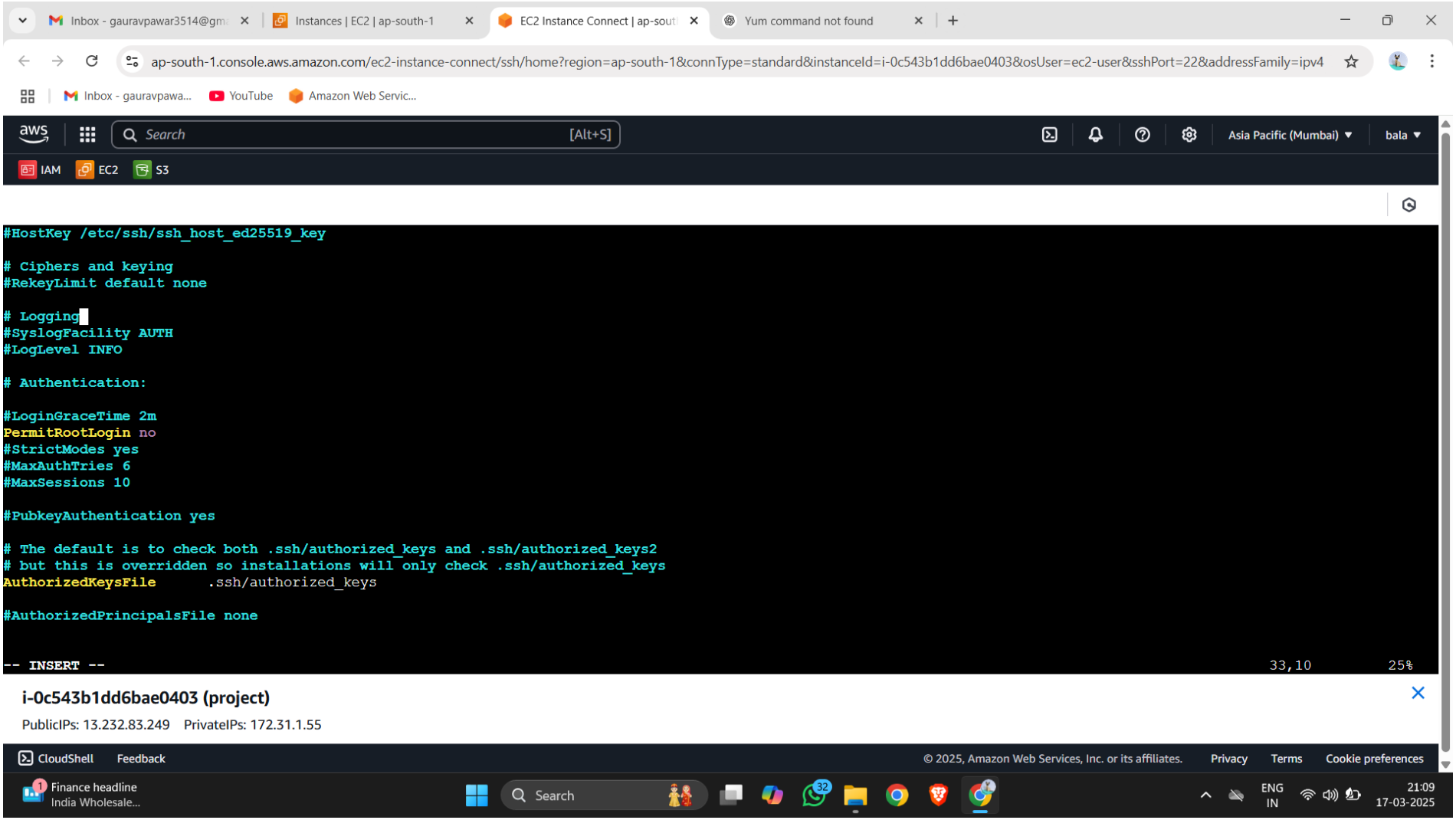
In Step 2 of the Linux Server Hardening project, the focus is on improving the security of user access and SSH (Secure Shell) connections to protect the server from unauthorized access and brute-force attacks. This step includes setting up secure user accounts, enforcing strong password policies, and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

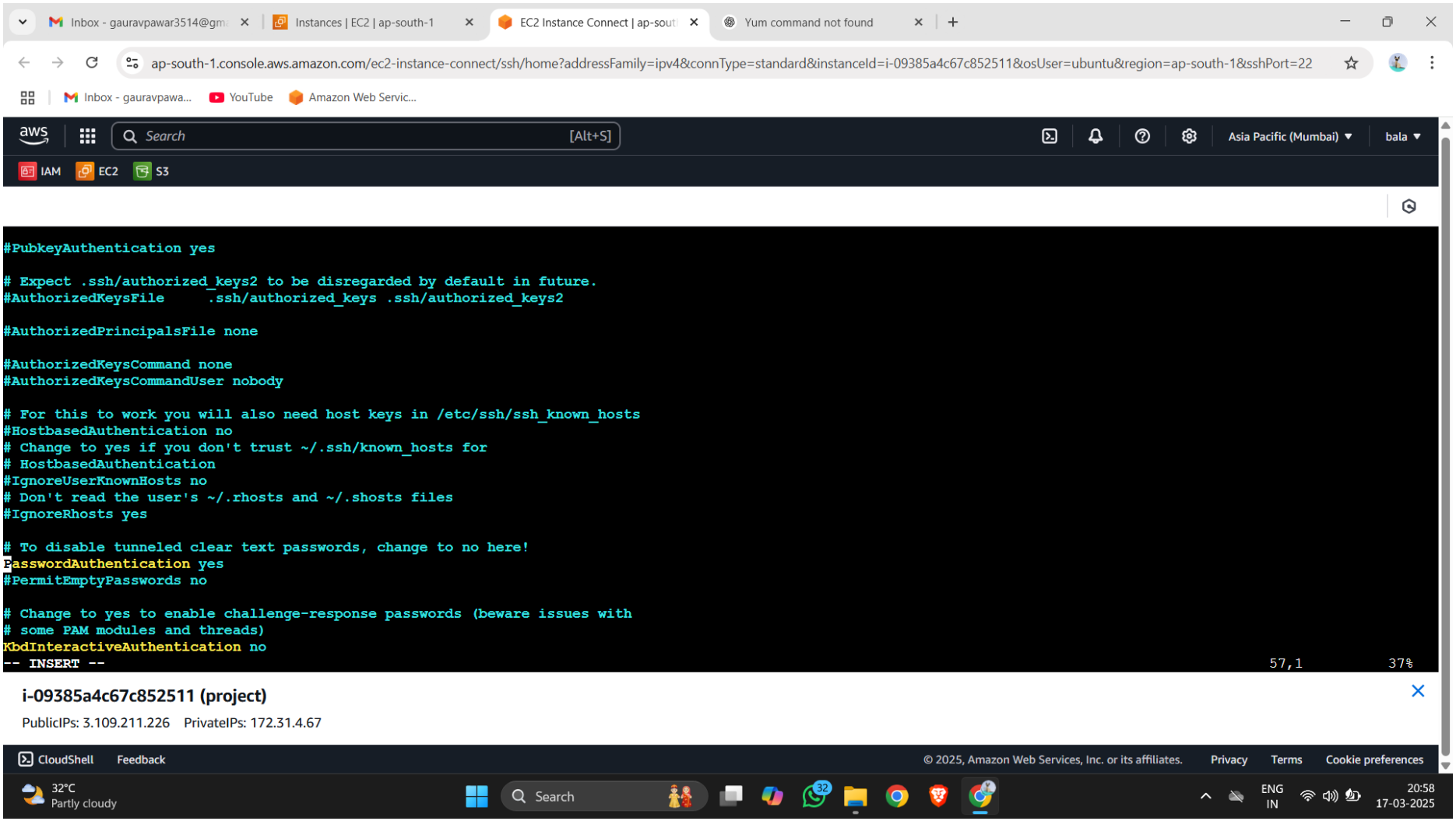
- **User Management**

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.

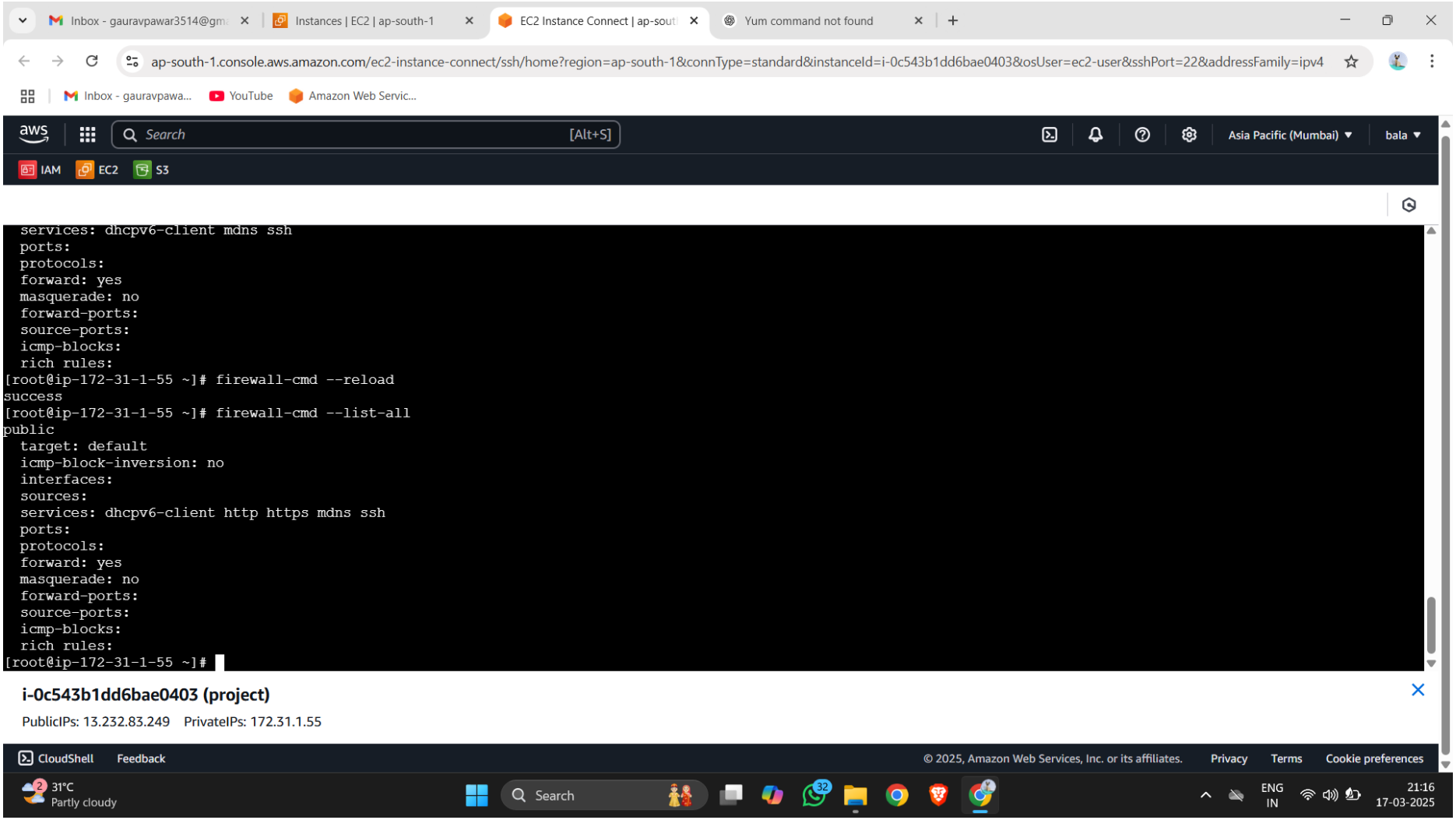
- **Goal:**
    - Ensure that only authorized users have access to the server.
    - Minimize the risk of unauthorized access through SSH.
    - Improve overall system security by strengthening user access controls and authentication mechanisms.

Step 3: Disable Root SSH Login



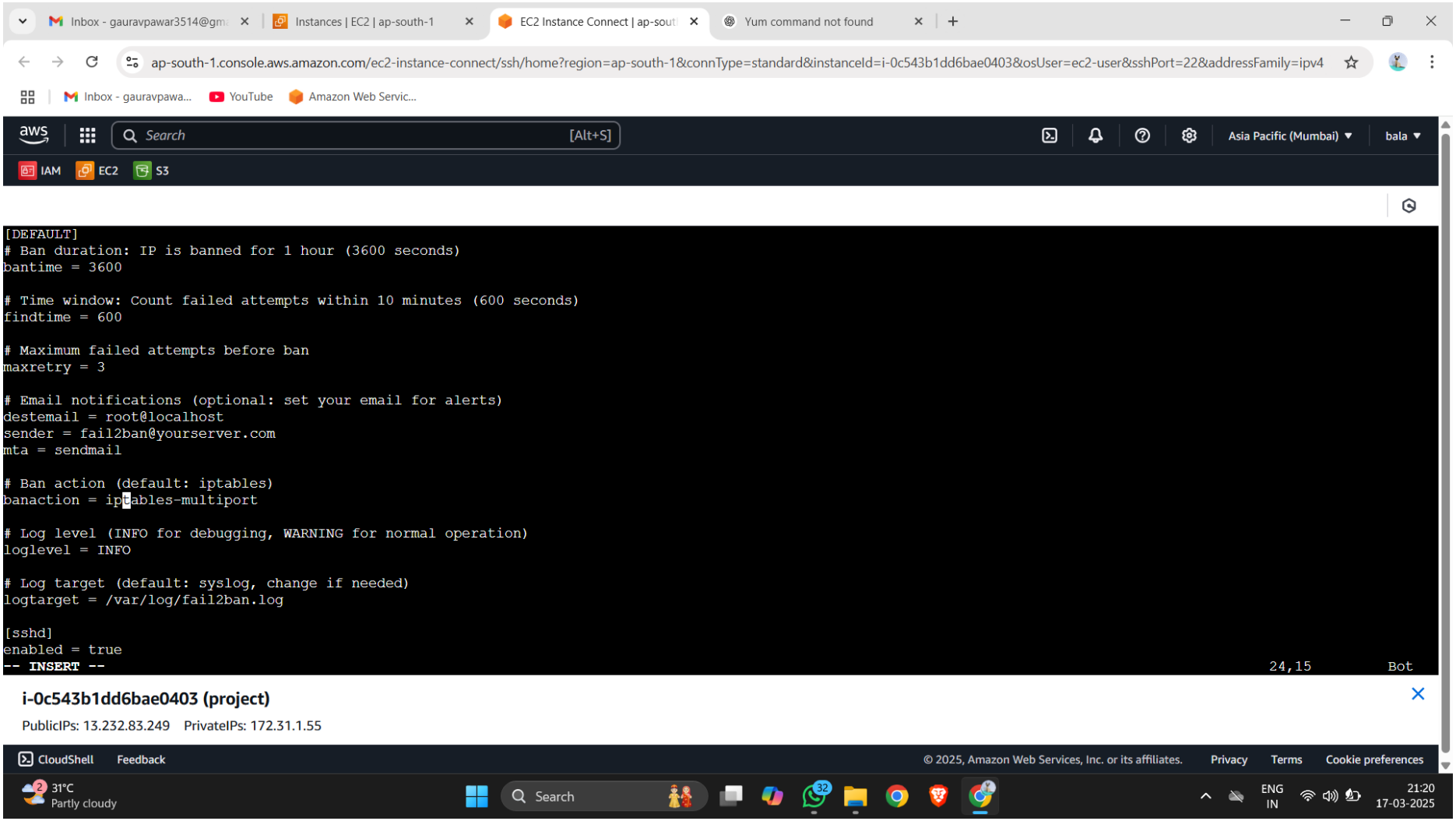

Inbox - gauravpawar3514@gm
Instances | EC2 | ap-south-1
EC2 Instance Connect | ap-sout
Yum command not found
ap-south-1.console.aws.amazon.com/ec2-instance-connect/ssh/home?addressFamily=ipv4&connType=standard&instanceId=i-09385a4c67c852511&osUser=ubuntu&region=ap-south-1&sshPort=22
Inbox - gauravpawa...
YouTube
Amazon Web Servic...
aws
Search
[Alt+S]
Asia Pacific (Mumbai)
bala
IAM
EC2
S3
#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 --
57,1
37%
i-09385a4c67c852511 (project)
PublicIPs: 3.109.211.226 PrivateIPs: 172.31.4.67
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
32°C
Partly cloudy
Search
ENG
IN
20:58
17-03-2025

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

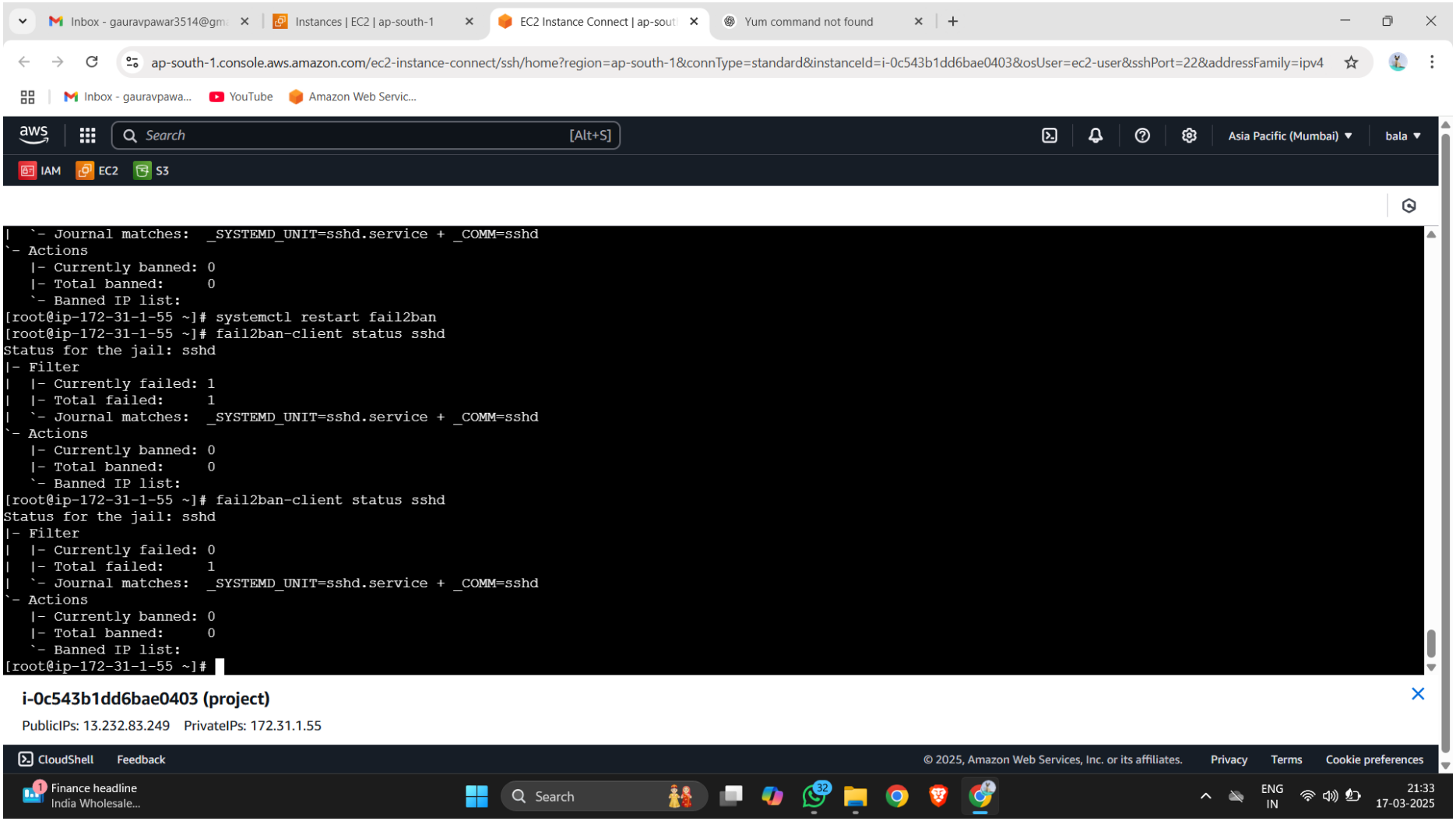


Step 5: Protect Against Brute Force Attacks with Fail2Ban

5.1 Configuring Fail2Ban with jail.local File



5.2 Verify Fail2Ban SSHD Status and Configuration Before Login Attempts



Step 6: Enable System Auditing with Auditd

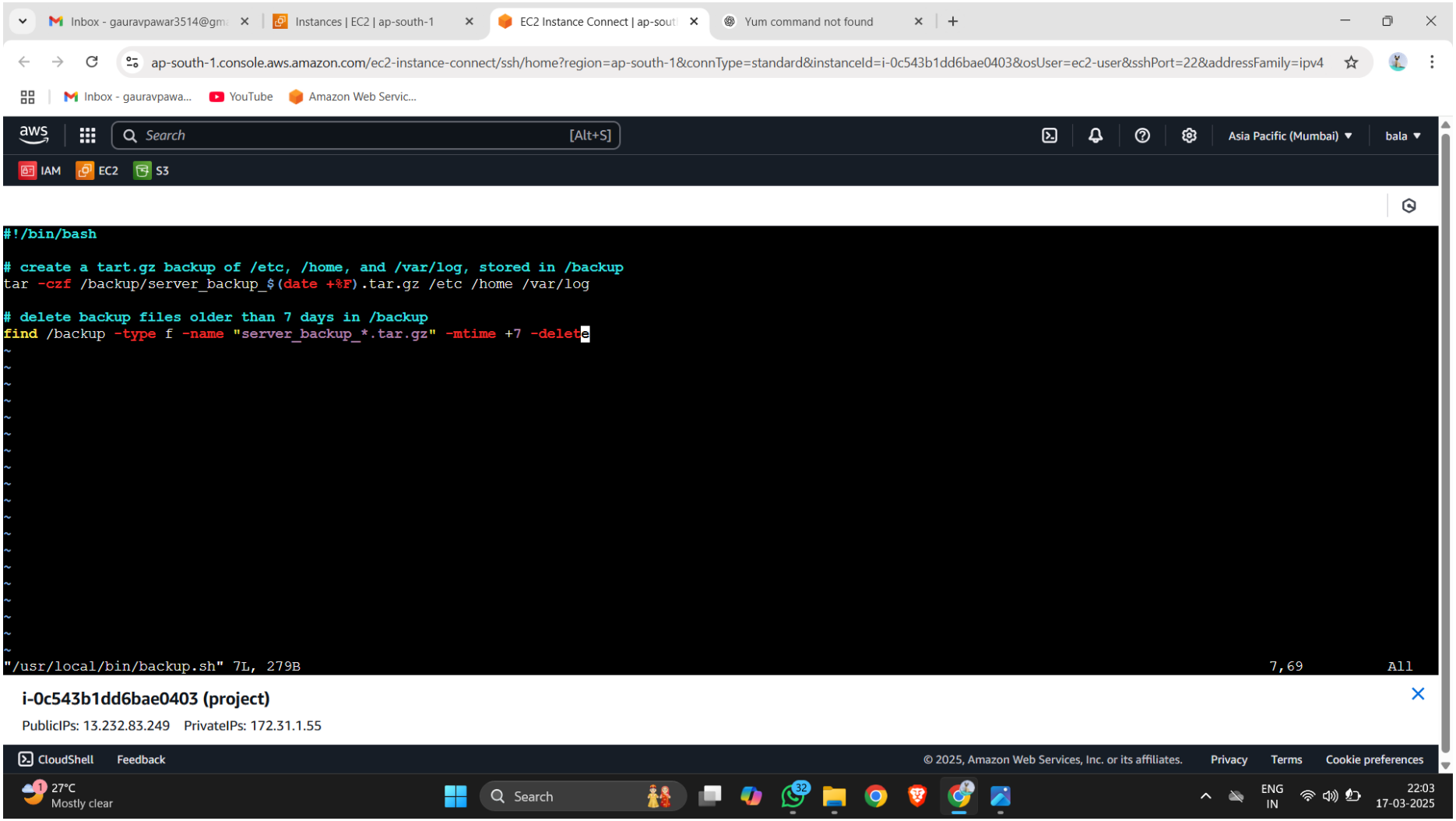
In Step 6 of the Linux Server Hardening project, the goal is to enable and configure Auditd (Linux Auditing System) to monitor and log system events. Auditd records security-relevant information such as file access, user logins, and process activity, providing detailed insight into system activity and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

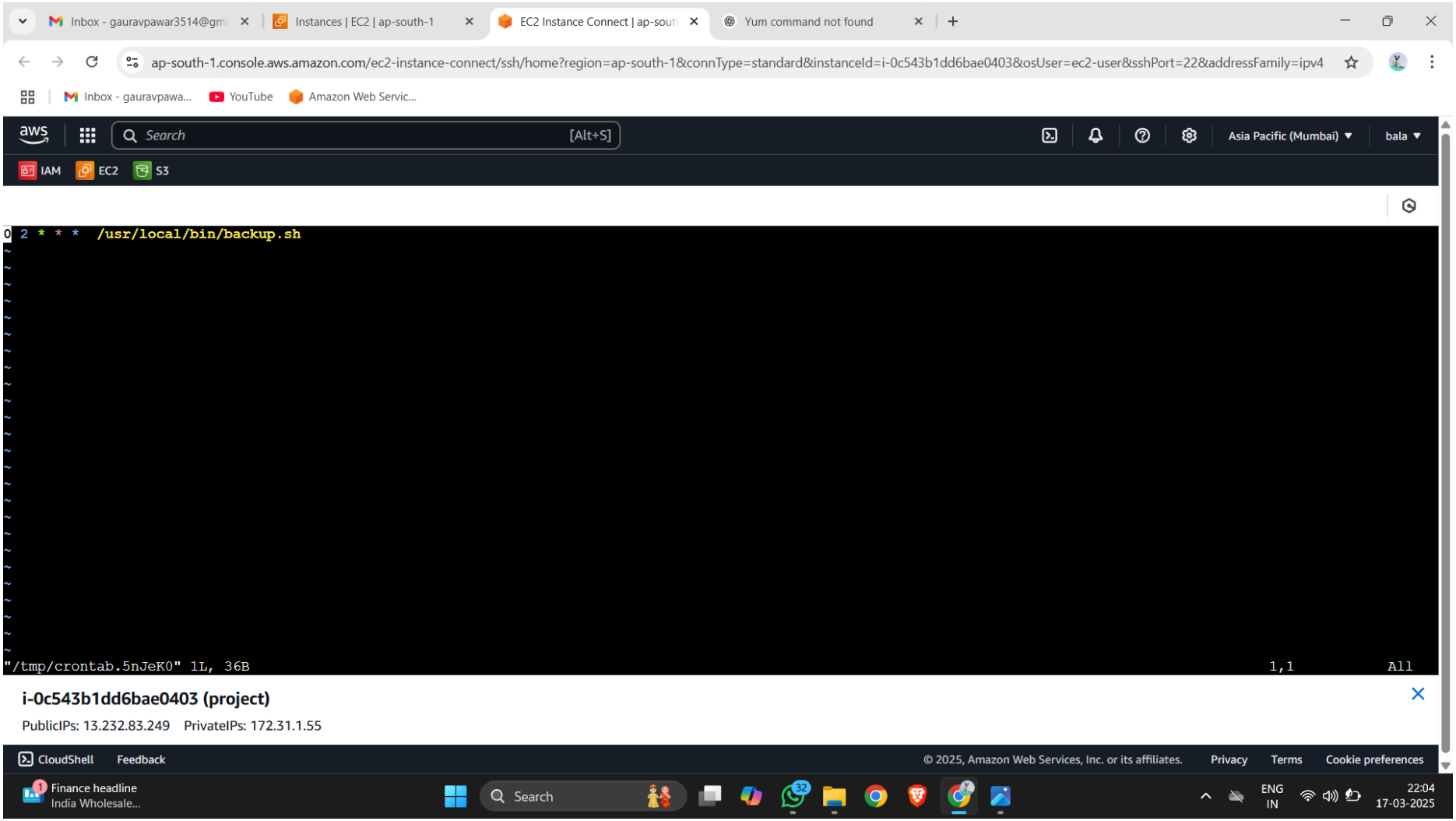
- **Key Configurations in Auditd:**
  1. Install and Enable Auditd
  2. Configure Audit Rules
  3. Set Audit Log Rotation
  4. Secure Audit Logs
  5. Monitor and Analyze Logs

Step 7: Automate Backups and Maintenance with Cron Jobs

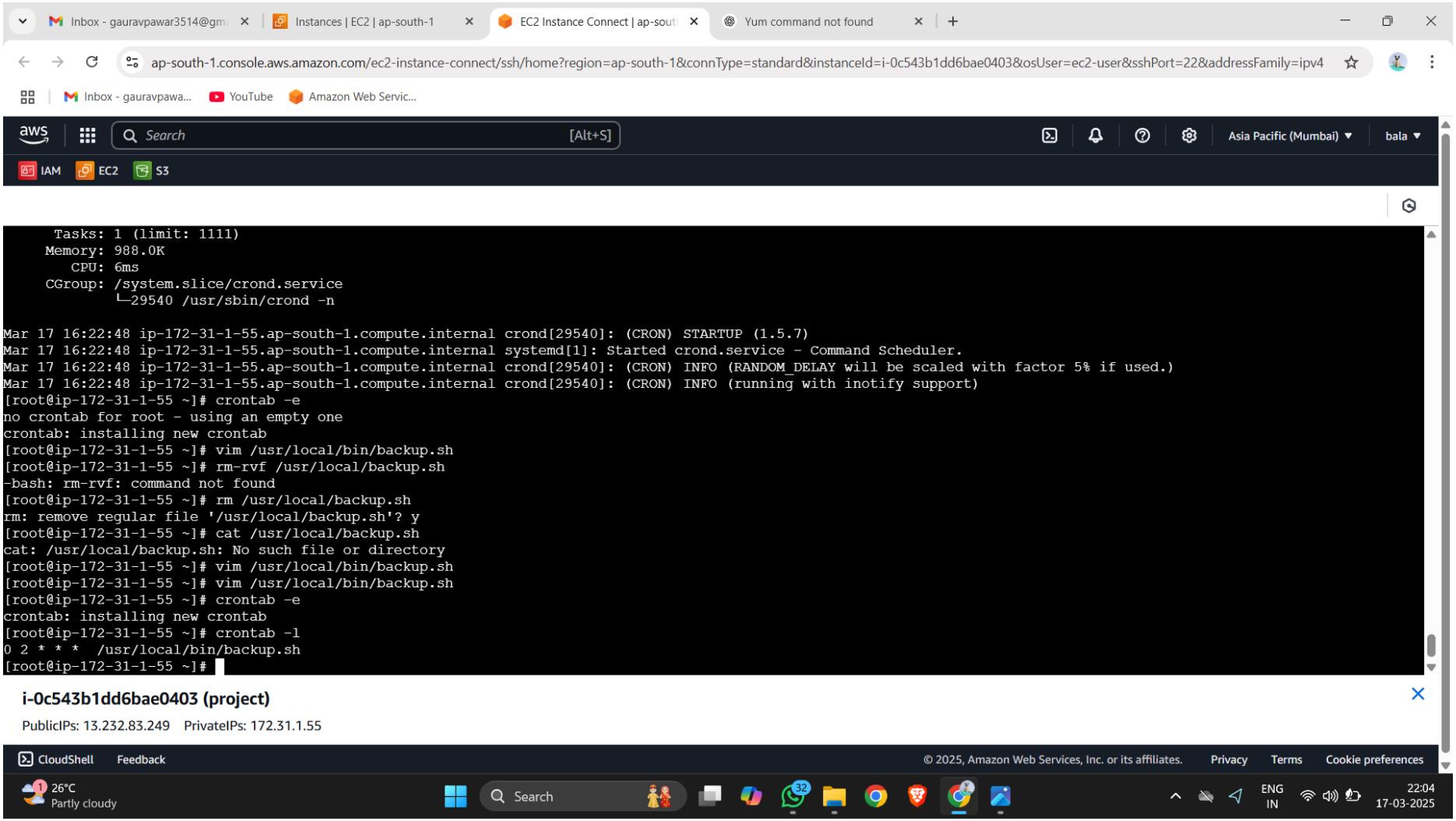
7.1 Create a Backup Script: Daily Backup Script with Auto Cleanup of Files Older Than 7 Days



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

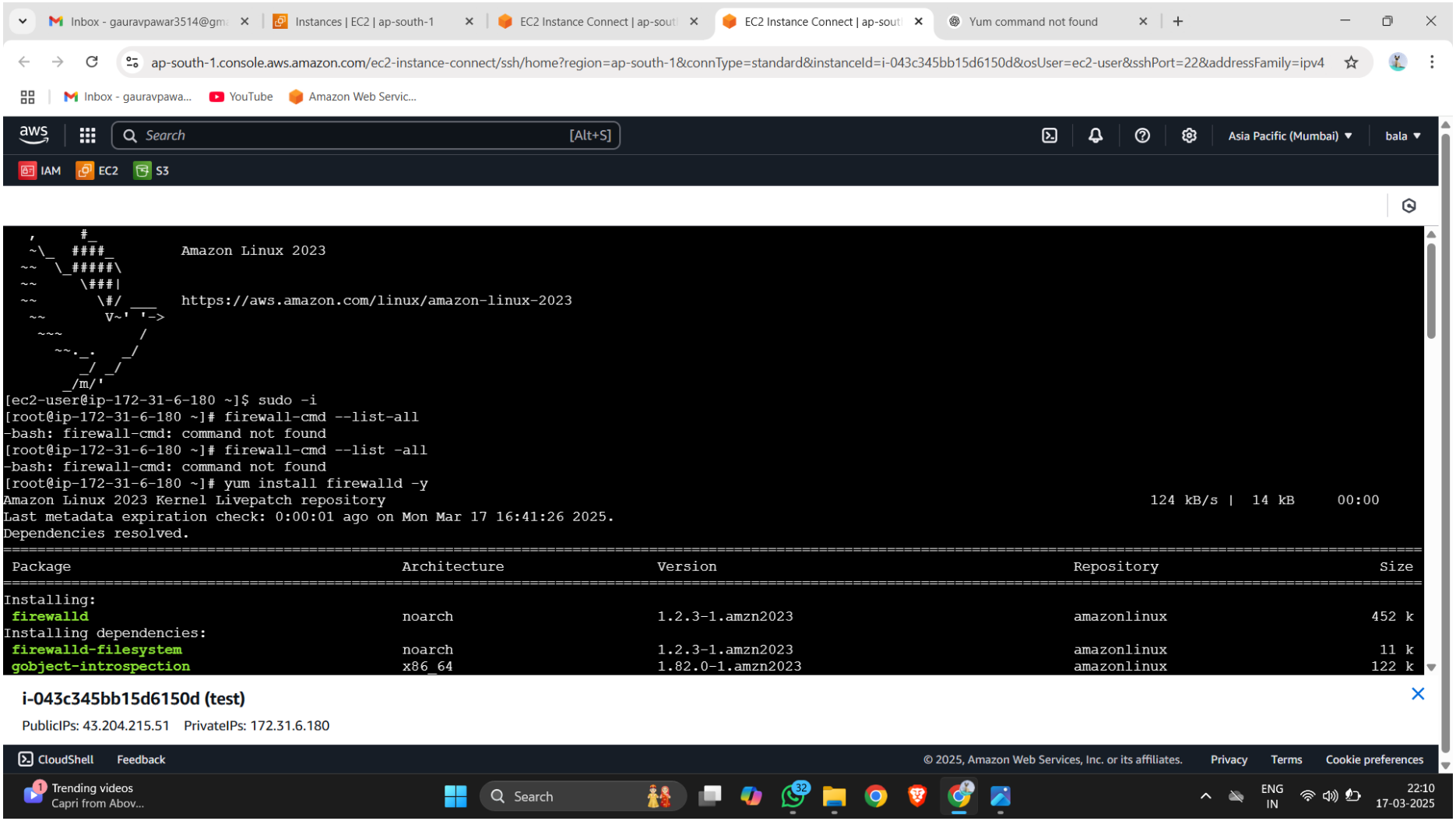


7.3 Reviewing Scheduled Cron Jobs: To ensure proper execution, timing, and system resource management.

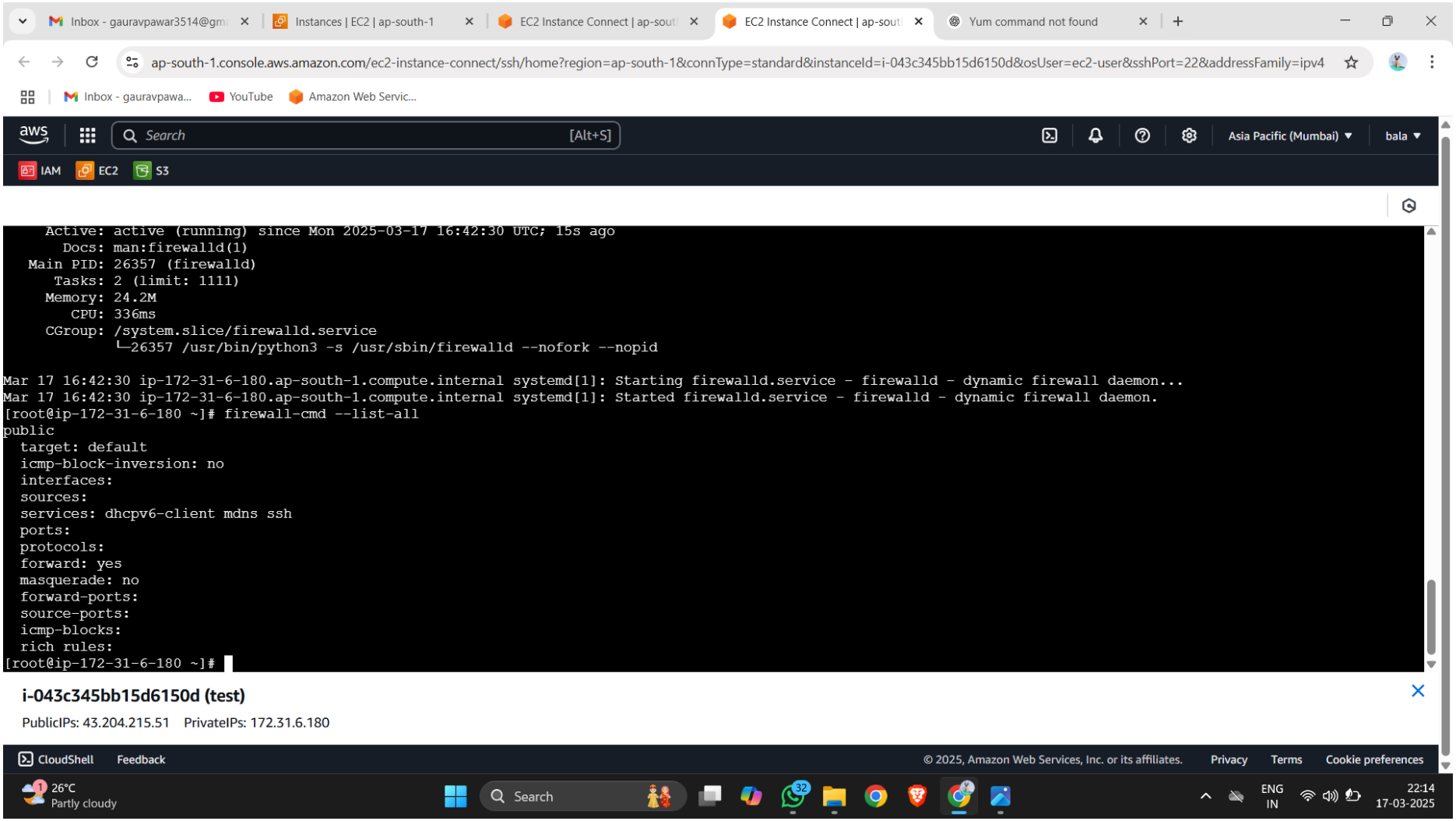


Step 8: Testing SSH Access with a Different User on the Target Machine

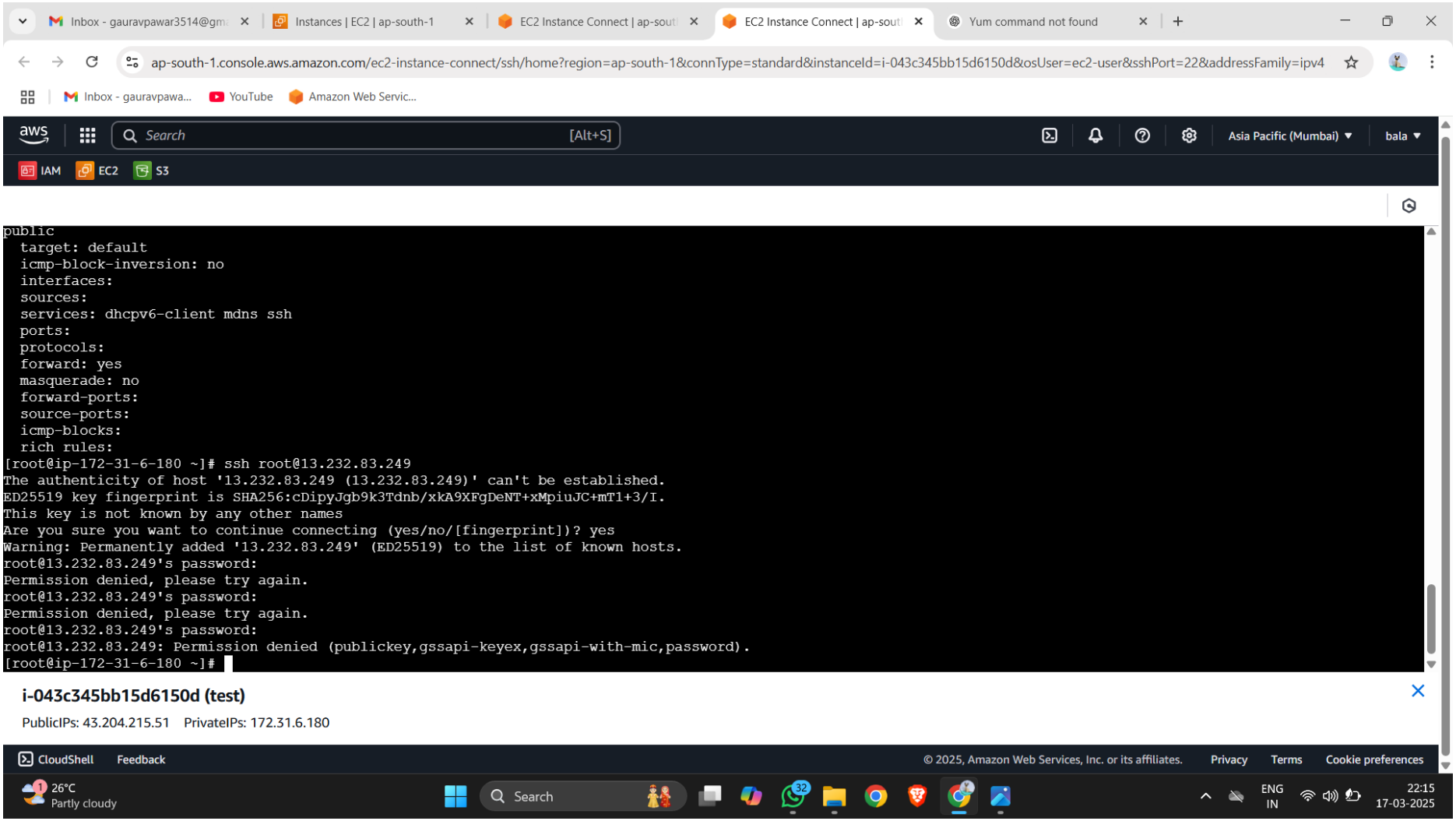
8.1 Configure Firewall: Allowing SSH on the Testing Machine



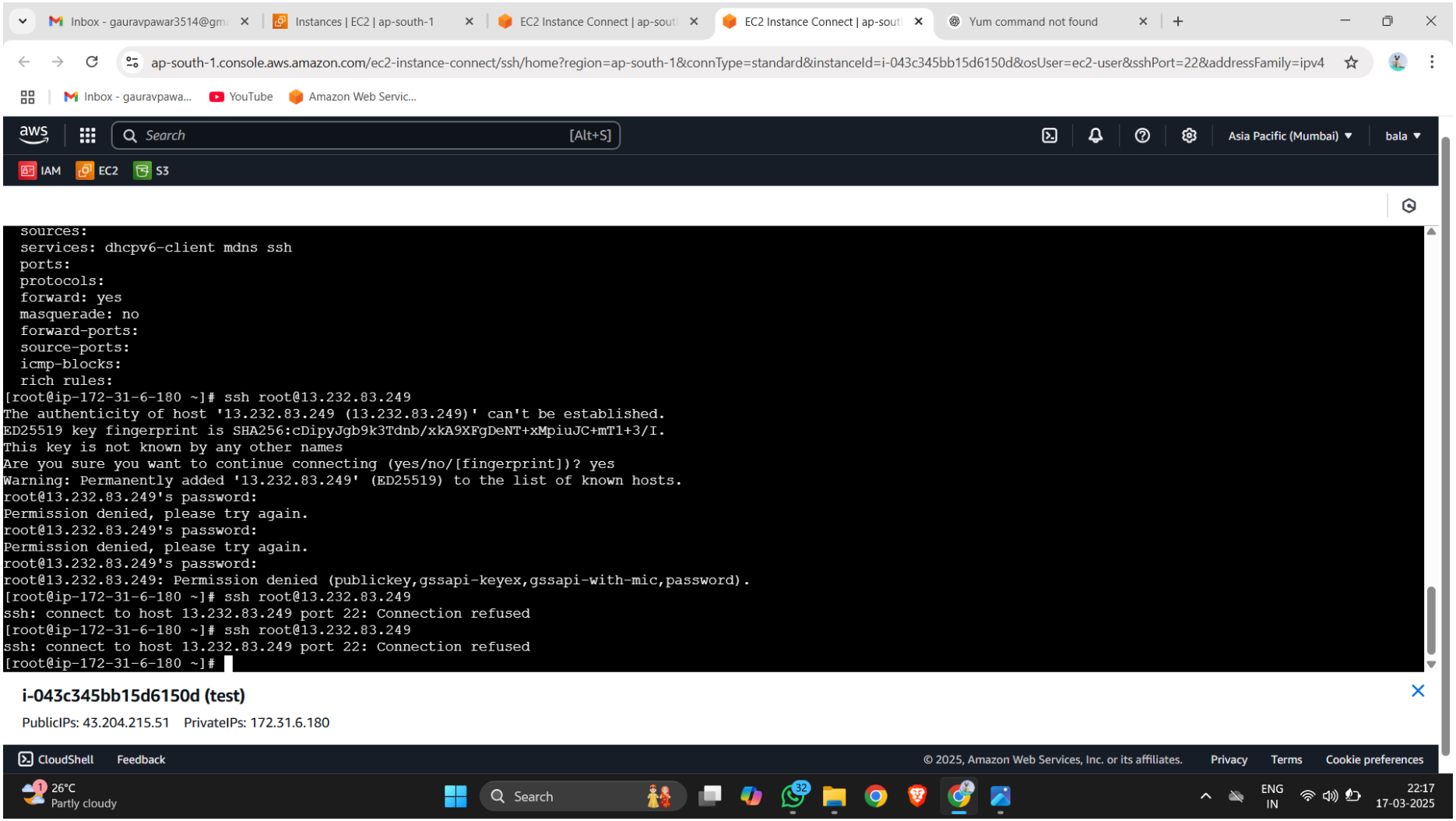
8.2 Verifying and Analyzing Firewall Rules: Including allowing SSH access for secure remote connections.



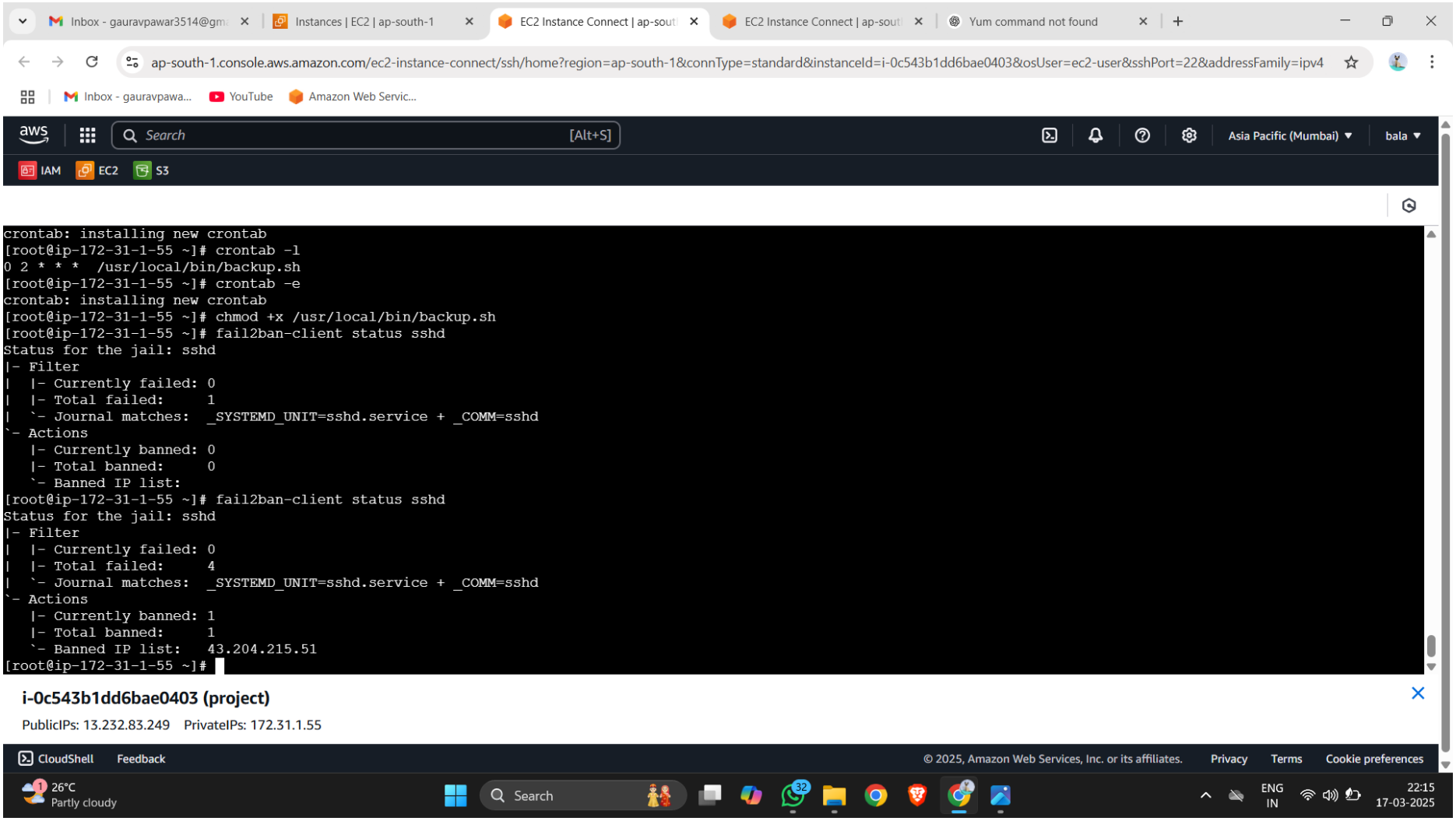
8.3 Verifying SSH Connectivity: Access with 3 Incorrect Password Attempts Within 10 Minutes



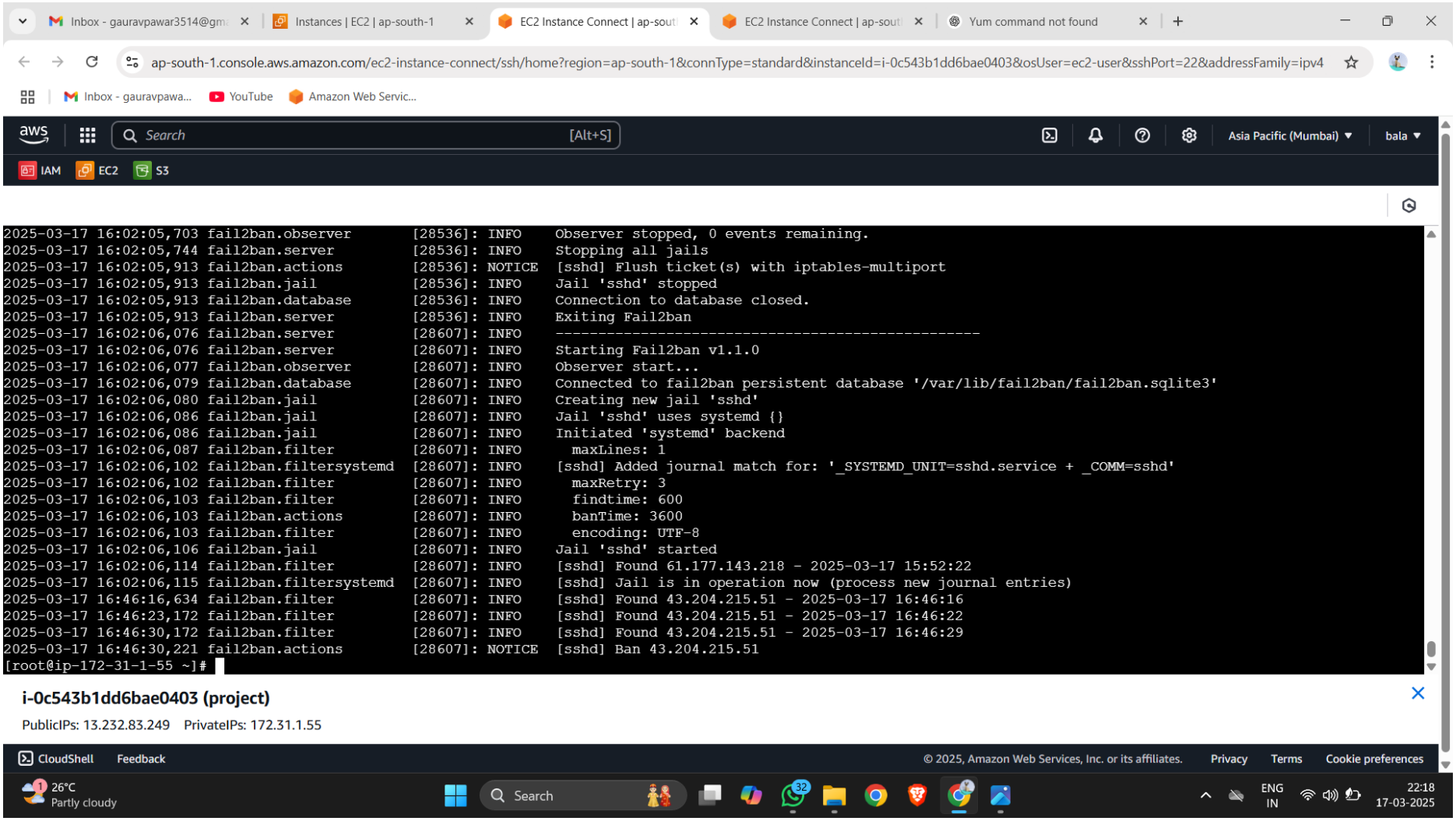
8.4 Fail2Ban IP Blocking: After 3 Failed Attempts Within 10 Minutes the IP will remain blocked for 10 minutes



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



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



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.