

EC2 Resource Monitoring and Alert Script Project

Goal: In this project, we can monitor the CPU and Disk usage of an EC2 instance using simple Linux commands.

If the usage crosses a specific limit (for example, CPU usage above 80%), an automatic alert will be sent via Email or AWS SNS (Simple Notification Service).

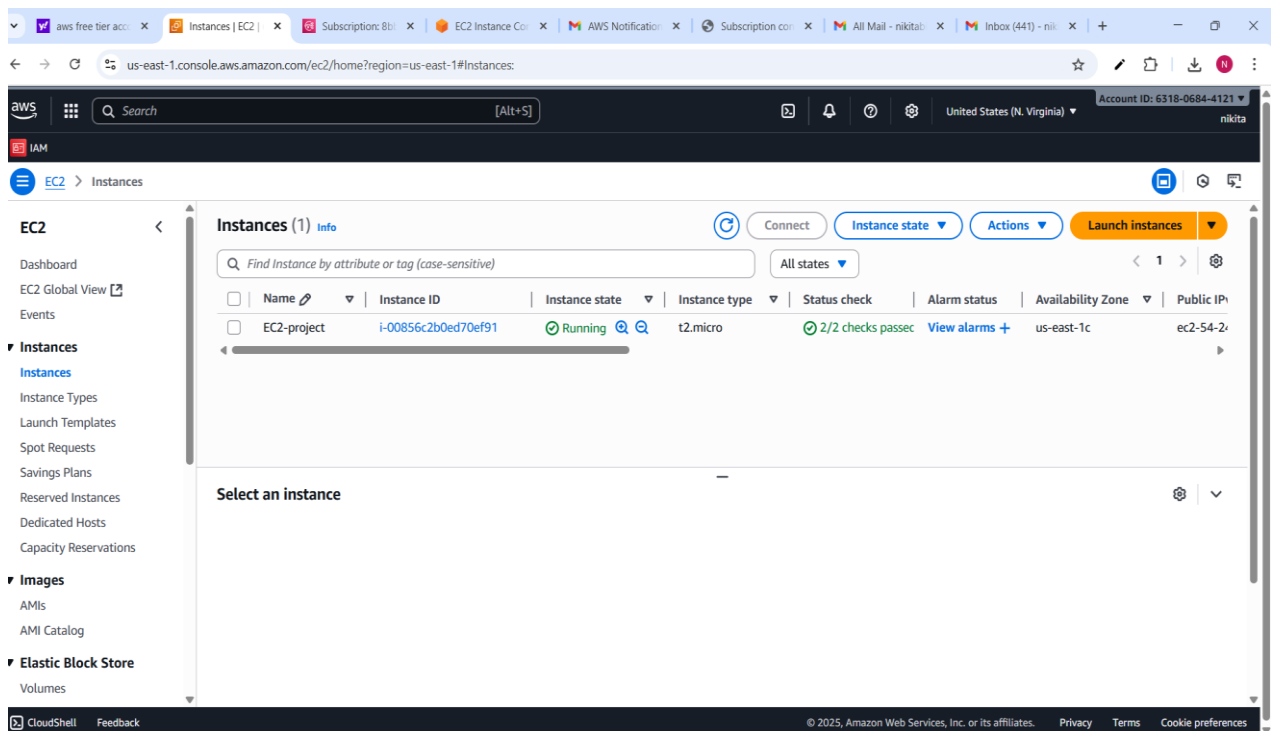
Use Case Examples:

- EC2 instance monitoring
- Bash scripting for automation
- Cron jobs (to schedule the script)
- AWS SNS integration for notifications

Key Components

Step 1. Launch an Ec2 Instance

Goto EC2 Service



Step 2: Connect to your EC2 Instance

The screenshot shows the AWS Management Console interface for connecting to an EC2 instance. The browser address bar shows the URL: `us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ConnectToInstance:instanceId=i-00856c2b0ed70ef91`. The console header includes the AWS logo, a search bar, and navigation icons. The main content area is titled "Connect to instance" and includes a sub-header "Connect to an instance using the browser-based client." Below this, there are four tabs: "EC2 Instance Connect", "Session Manager", "SSH client", and "EC2 serial console". The "EC2 Instance Connect" tab is active. It displays the "Instance ID" as `i-00856c2b0ed70ef91 (EC2-project)`. Under "Connection type", there are two options: "Connect using a Public IP" (selected) and "Connect using a Private IP". The "Public IP address" section shows the IP `54.242.122.169`. The "Username" field is set to `ec2-user`. A note at the bottom states: "In most cases, the default username, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username." The footer of the console shows "CloudShell" and "Feedback" links, along with copyright information for Amazon Web Services.

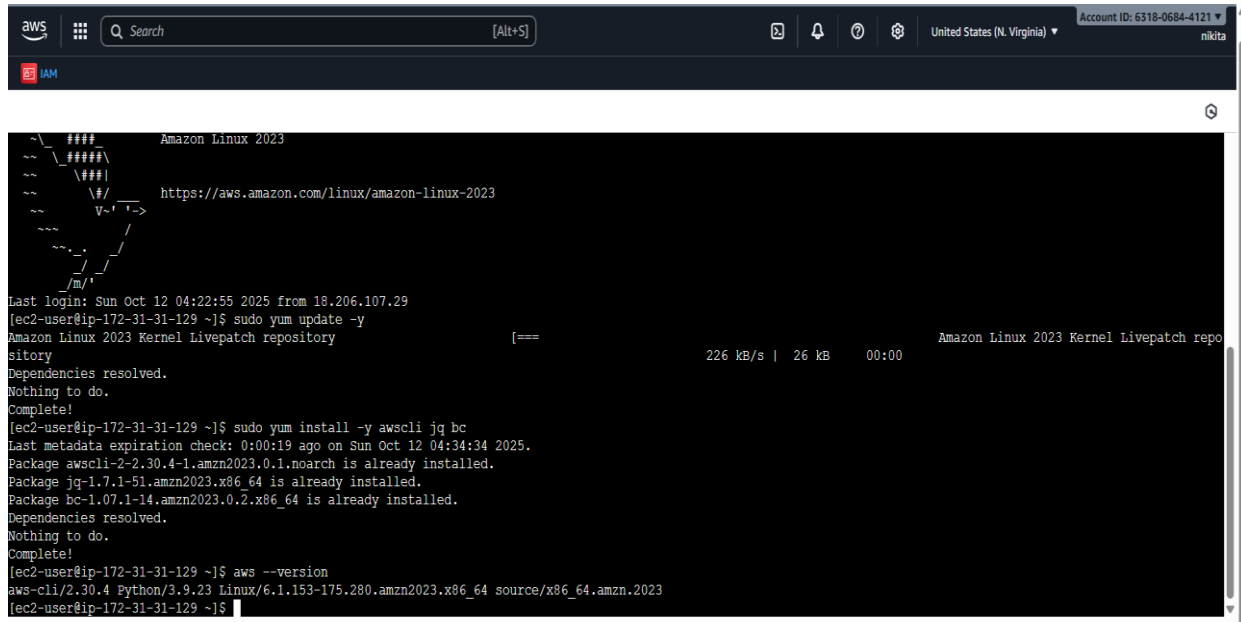
```

Last login: Sun Oct 12 04:22:26 2025 from 18.206.107.29
[ec2-user@ip-172-31-31-129 ~]$ sudo ssh -i "project.key.pem" ec2-user@54.242.122.169
Warning: Identity file project.key.pem not accessible: No such file or directory.
ec2-user@54.242.122.169: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[ec2-user@ip-172-31-31-129 ~]$ vi project.key.pem
[ec2-user@ip-172-31-31-129 ~]$ chmod 400 project.key.pem
[ec2-user@ip-172-31-31-129 ~]$ ssh -i "project.key.pem" ec2-user@172.31.31.129
The authenticity of host '172.31.31.129 (172.31.31.129)' can't be established.
ED25519 key fingerprint is SHA256:wb8nXu5p5HTnUEJDj05PyxYa6rSeueuvyPf+giXXZM4.
This host key is known by the following other names/addresses:
~/.ssh/known_hosts:1: 54.242.122.169
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '172.31.31.129' (ED25519) to the list of known hosts.

#
~\  #####      Amazon Linux 2023
~~ \#####\
~~  \###\
~~   \#/\_      https://aws.amazon.com/linux/amazon-linux-2023
~~    V-!  ->
~~
~~
Last login: Sun Oct 12 04:22:55 2025 from 18.206.107.29
[ec2-user@ip-172-31-31-129 ~]$
```

Step 3: Install AWS CLI and Tools

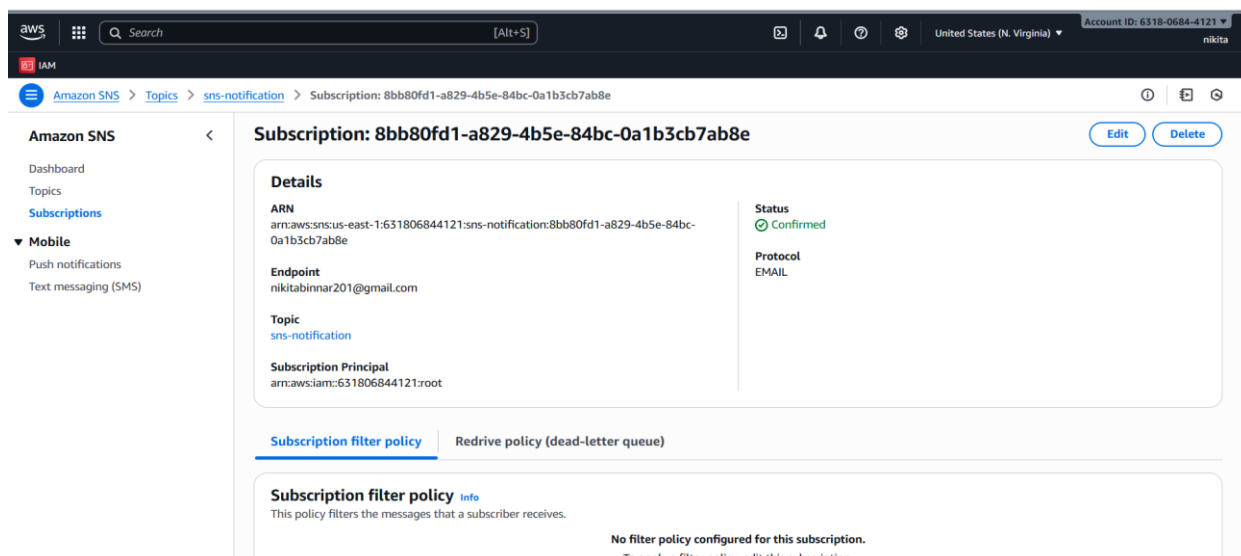
```
sudo yum update -y
sudo yum install -y awscli jq bc
```



```
~\
#####
~\
#####\
~\
#####|
~\
\#/\
~\
V~'  '→ https://aws.amazon.com/linux/amazon-linux-2023

Last login: Sun Oct 12 04:22:55 2025 from 18.206.107.29
[ec2-user@ip-172-31-31-129 ~]$ sudo yum update -y
Amazon Linux 2023 Kernel Livepatch repository                               [==
                                                                    226 kB/s | 26 kB    00:00  Amazon Linux 2023 Kernel Livepatch repo
sitory
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-31-129 ~]$ sudo yum install -y awscli jq bc
Last metadata expiration check: 0:00:19 ago on Sun Oct 12 04:34:34 2025.
Package awscli-2-2.30.4-1.amzn2023.0.1.noarch is already installed.
Package jq-1.7.1-51.amzn2023.x86_64 is already installed.
Package bc-1.07.1-14.amzn2023.0.2.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[ec2-user@ip-172-31-31-129 ~]$ aws --version
aws-cli/2.30.4 Python/3.9.23 Linux/6.1.153-175.280.amzn2023.x86_64 source/x86_64.amzn.2023
[ec2-user@ip-172-31-31-129 ~]$
```

Step 4: Create SNS Topic (for email alerts)



The screenshot shows the AWS Management Console for Amazon SNS. The left sidebar contains navigation links for Dashboard, Topics, Subscriptions, and Mobile. The main content area displays the details for a specific subscription: **Subscription: 8bb80fd1-a829-4b5e-84bc-0a1b3cb7ab8e**. The details include the ARN, Endpoint (nikitabinnar201@gmail.com), Topic (sns-notification), and Subscription Principal. The status is **Confirmed** and the protocol is **EMAIL**. Below the details, there are tabs for **Subscription filter policy** and **Redrive policy (dead-letter queue)**. The filter policy section indicates that no filter policy is currently configured for this subscription.

Step 5: Create Monitoring Script

```
#!/bin/bash

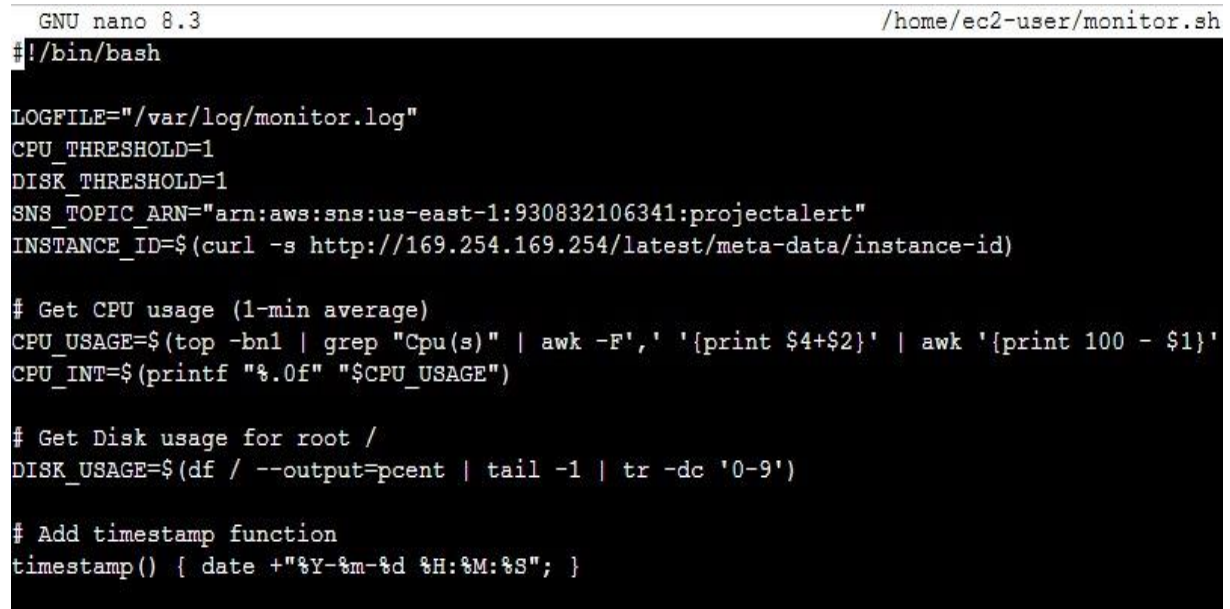
CPU_THRESHOLD=1
DISK_THRESHOLD=1
SNS_TOPIC_ARN="<arn:aws:sns:us-east-1:930832106341:projectalert>"
INSTANCE_ID=$(curl -s http:// 169.254.169.254/latest/meta-data/instance-id)

CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | awk '{print 100 - $8}')
CPU_USAGE=${CPU_USAGE%.*}

DISK_USAGE=$(df -h / | grep / | awk '{print $5}' | sed 's/%//g')

if [ "$CPU_USAGE" -gt "$CPU_THRESHOLD" ]; then
    aws sns publish --topic-arn "$SNS_TOPIC_ARN" --message "CPU HIGH ALERT: $CPU_USAGE% on $INSTANCE_ID"
fi

if [ "$DISK_USAGE" -gt "$DISK_THRESHOLD" ]; then
    aws sns publish --topic-arn "$SNS_TOPIC_ARN" --message "DISK HIGH ALERT: $DISK_USAGE% on $INSTANCE_ID"
fi
```



```
GNU nano 8.3 /home/ec2-user/monitor.sh
#!/bin/bash

LOGFILE="/var/log/monitor.log"
CPU_THRESHOLD=1
DISK_THRESHOLD=1
SNS_TOPIC_ARN="arn:aws:sns:us-east-1:930832106341:projectalert"
INSTANCE_ID=$(curl -s http://169.254.169.254/latest/meta-data/instance-id)

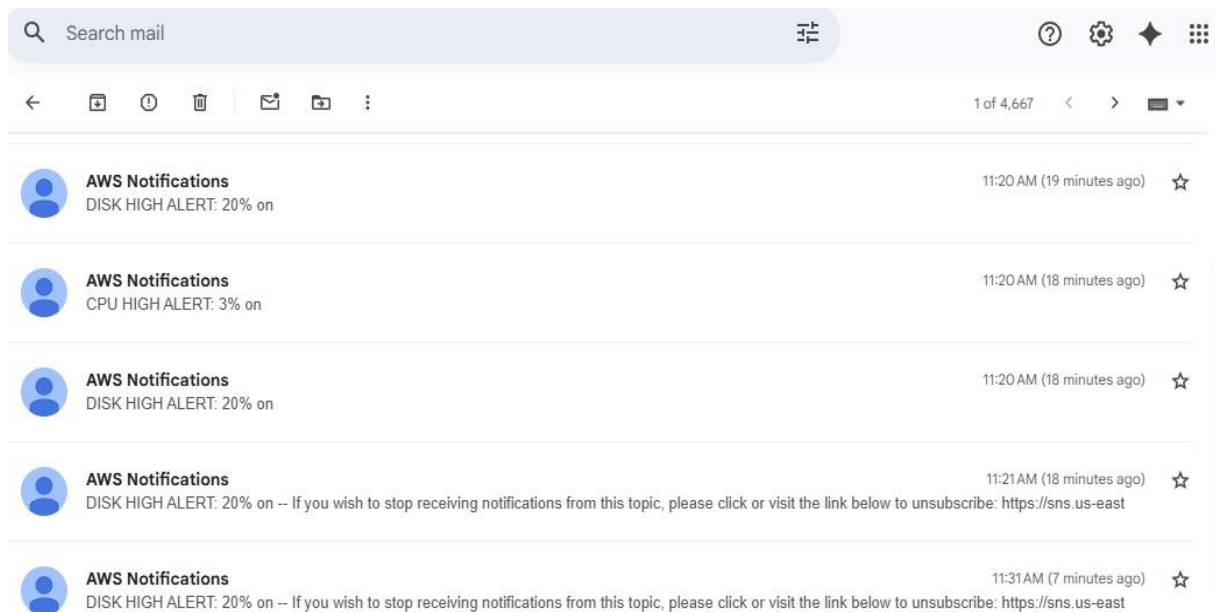
# Get CPU usage (1-min average)
CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | awk -F',' '{print $4+$2}' | awk '{print 100 - $1}')
CPU_INT=$(printf "%.0f" "$CPU_USAGE")

# Get Disk usage for root /
DISK_USAGE=$(df / --output=pcent | tail -1 | tr -dc '0-9')

# Add timestamp function
timestamp() { date +"%Y-%m-%d %H:%M:%S"; }
```

Step 6: Test Alerts :

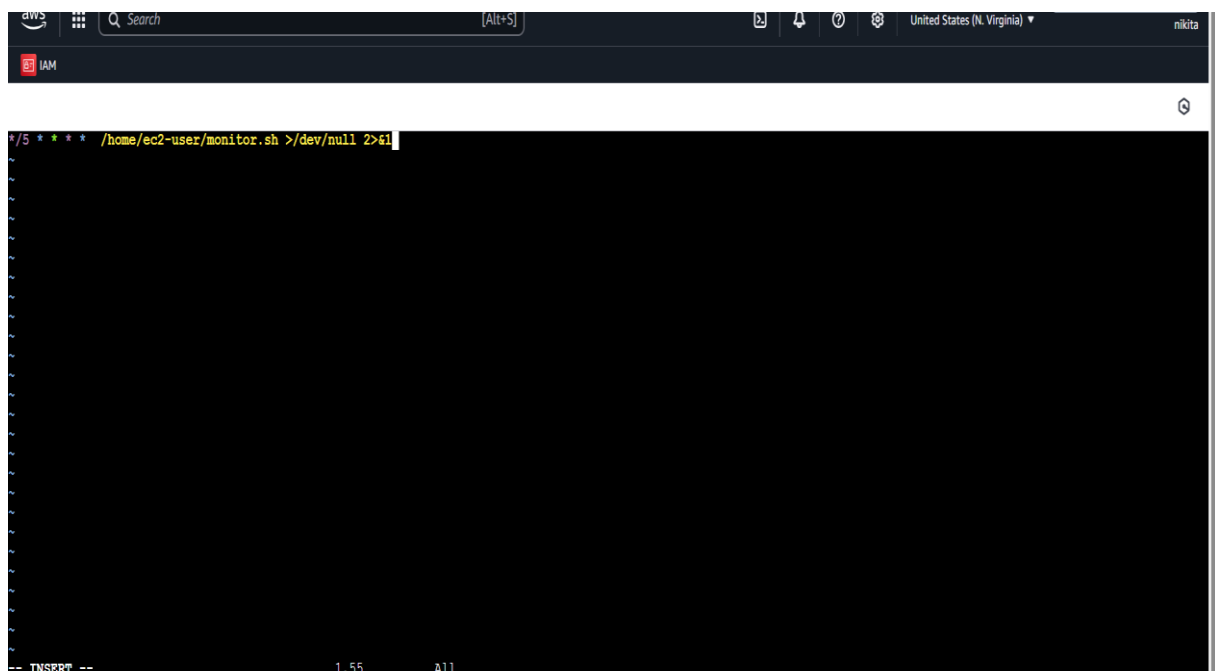
```
CPU_THRESHOLD=1
DISK_THRESHOLD=1
~/monitor.sh
```



Step 7: Automate with Cron

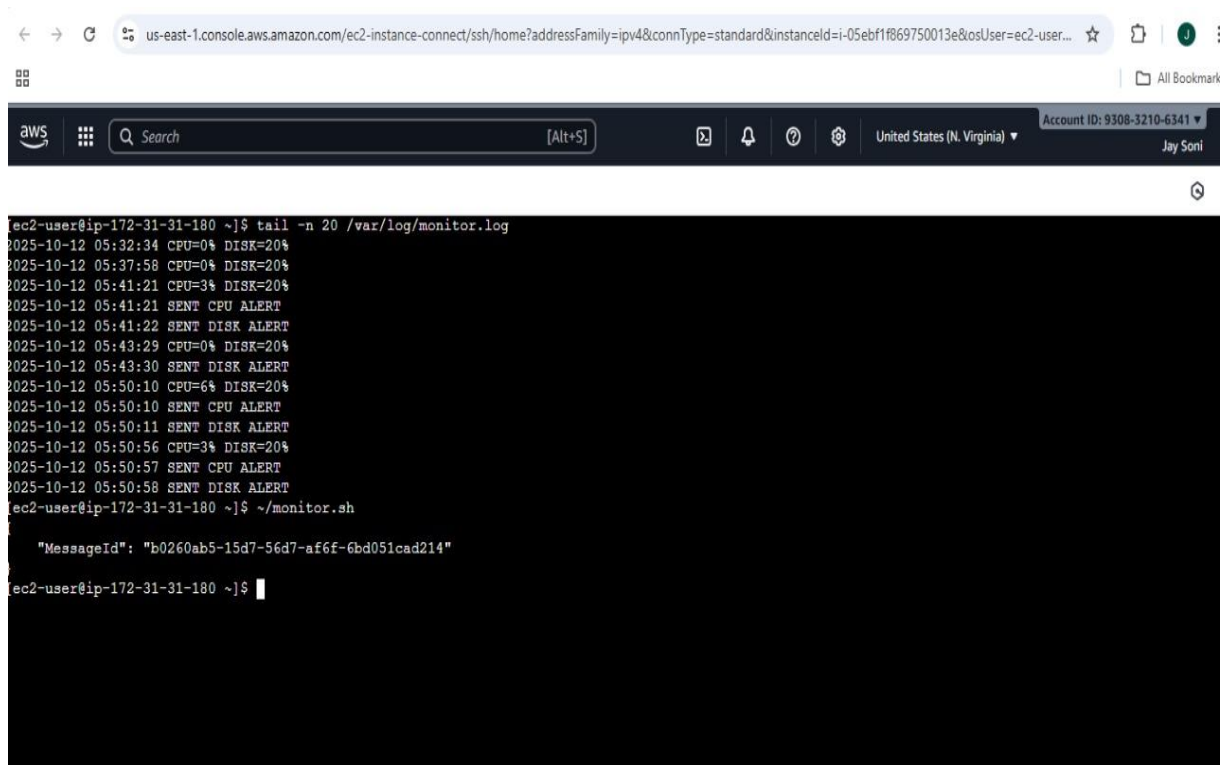
Cron will run the script **automatically every 5 minutes**.

`crontab -e`



Final Output:

- Cron job is scheduled.
- Check logs (after 5–10 min):

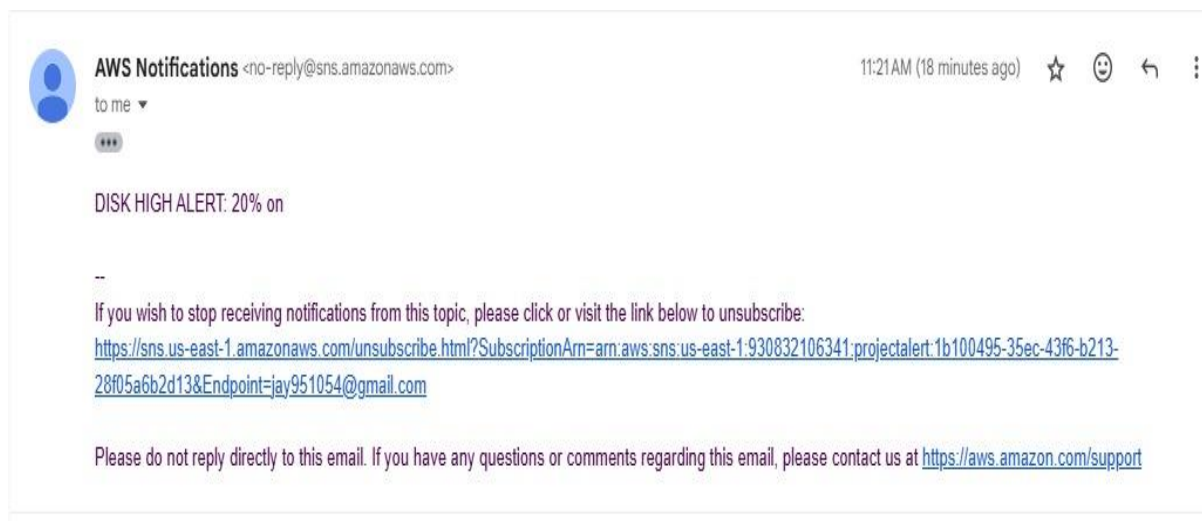


The screenshot shows the AWS Management Console interface for an EC2 instance. The terminal window displays the output of a `tail -n 20 /var/log/monitor.log` command, showing a series of log entries with timestamps, CPU usage, disk usage, and alerts. The user then runs `~/monitor.sh`, which outputs a JSON message ID.

```
ec2-user@ip-172-31-31-180 ~]$ tail -n 20 /var/log/monitor.log
2025-10-12 05:32:34 CPU=0% DISK=20%
2025-10-12 05:37:58 CPU=0% DISK=20%
2025-10-12 05:41:21 CPU=3% DISK=20%
2025-10-12 05:41:21 SENT CPU ALERT
2025-10-12 05:41:22 SENT DISK ALERT
2025-10-12 05:43:29 CPU=0% DISK=20%
2025-10-12 05:43:30 SENT DISK ALERT
2025-10-12 05:50:10 CPU=6% DISK=20%
2025-10-12 05:50:10 SENT CPU ALERT
2025-10-12 05:50:11 SENT DISK ALERT
2025-10-12 05:50:56 CPU=3% DISK=20%
2025-10-12 05:50:57 SENT CPU ALERT
2025-10-12 05:50:58 SENT DISK ALERT
ec2-user@ip-172-31-31-180 ~]$ ~/monitor.sh

{"MessageId": "b0260ab5-15d7-56d7-af6f-6bd051cad214"}
ec2-user@ip-172-31-31-180 ~]$
```

Email Alerts:



Testing & Validation

1. CPU and Disk usage logged in `/var/log/monitor.log`.
2. Email alert from SNS when thresholds are exceeded.
3. Cron automatically runs every 5 minutes.
4. Optional: manually test SNS → receive alert instantly.

Contribution:

Jay Soni

Umesh Chimankar

Sakshi Jain

Nikita Binnar