Project 5

Automated Jenkins Job Triggered by Access Log Size

Batch

MCA 7 Feb 2025 &
DevOps 2 Jun 2025

Name

Suraj Molke

Name of the Mentor

Ravindra Bagle Sir
Swati Zampal Ma’am

Fortune Cloud Technologies

2nd Floor, Shirodkar House, Opposite To Amit Cafe, Congress House Road, Near Pune Municipal Corporation, Shivajinagar, Pune - 411005.

smolke9@gmail.com Suraj Molke Smolke9

Project: 5

Title: Automated Jenkins Job Triggered by Access Log Size

Objective:

The aim of this project is to create an automated log monitoring and upload system. The system continuously

checks if the access log file (/var/log/custom/access.log) exceeds 1 GB. When it does:

1. A Jenkins pipeline is automatically triggered.

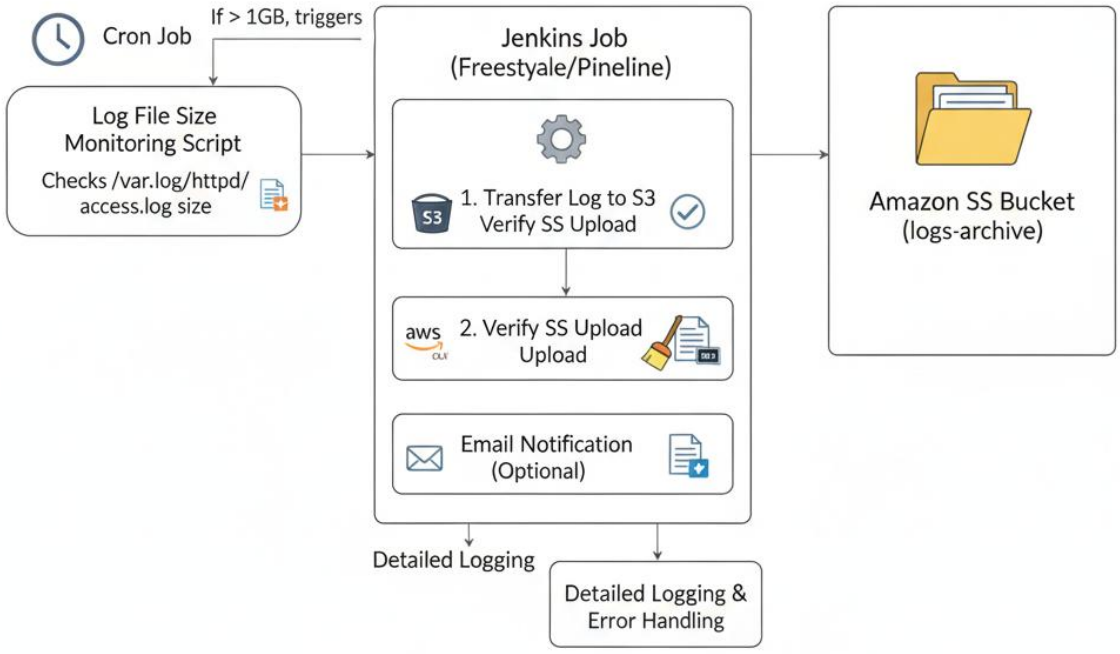
2. The log file is uploaded to an Amazon S3 bucket.

3. After successful upload, the original log file is cleared.

4. An email notification is sent.

This setup helps in automating log management and prevents excessive disk usage on the server.

Architecture diagram



STEP 1: Initial Setup

I)Update Your Ubuntu System:

sudo apt update && sudo apt upgrade –y

ii) Install Java (Required for Jenkins):

sudo apt install openjdk-17-jdk –y

iii)Install Jenkins:

**curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee **

/usr/share/keyrings/jenkins-keyring.asc > /dev/null

**echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] **

**https://pkg.jenkins.io/debian-stable binary/ | sudo tee **

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt update

sudo apt install jenkins –y

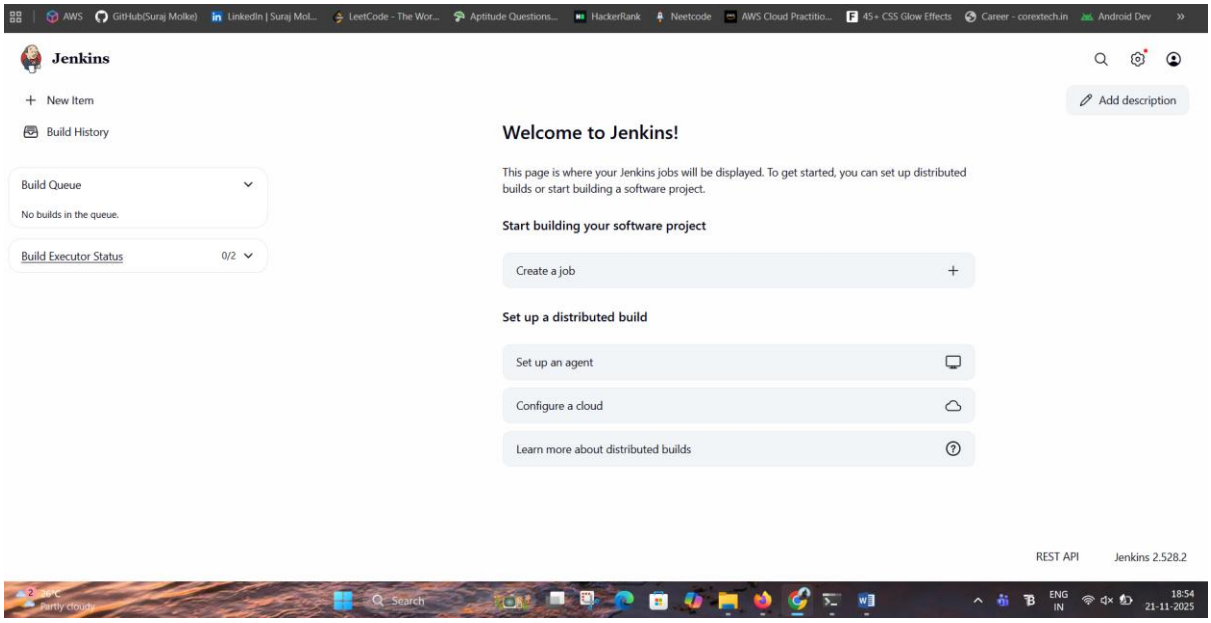
iv)Start Jenkins:

sudo systemctl start jenkins

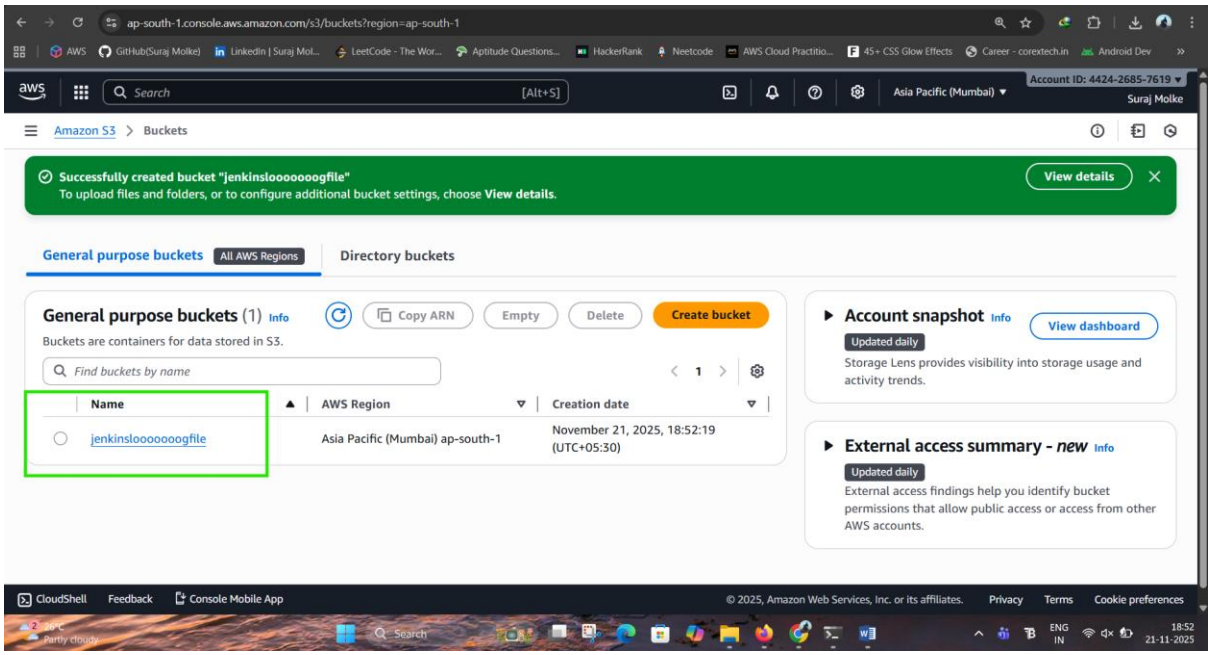
sudo systemctl enable jenkins

Unlock Jenkins using

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

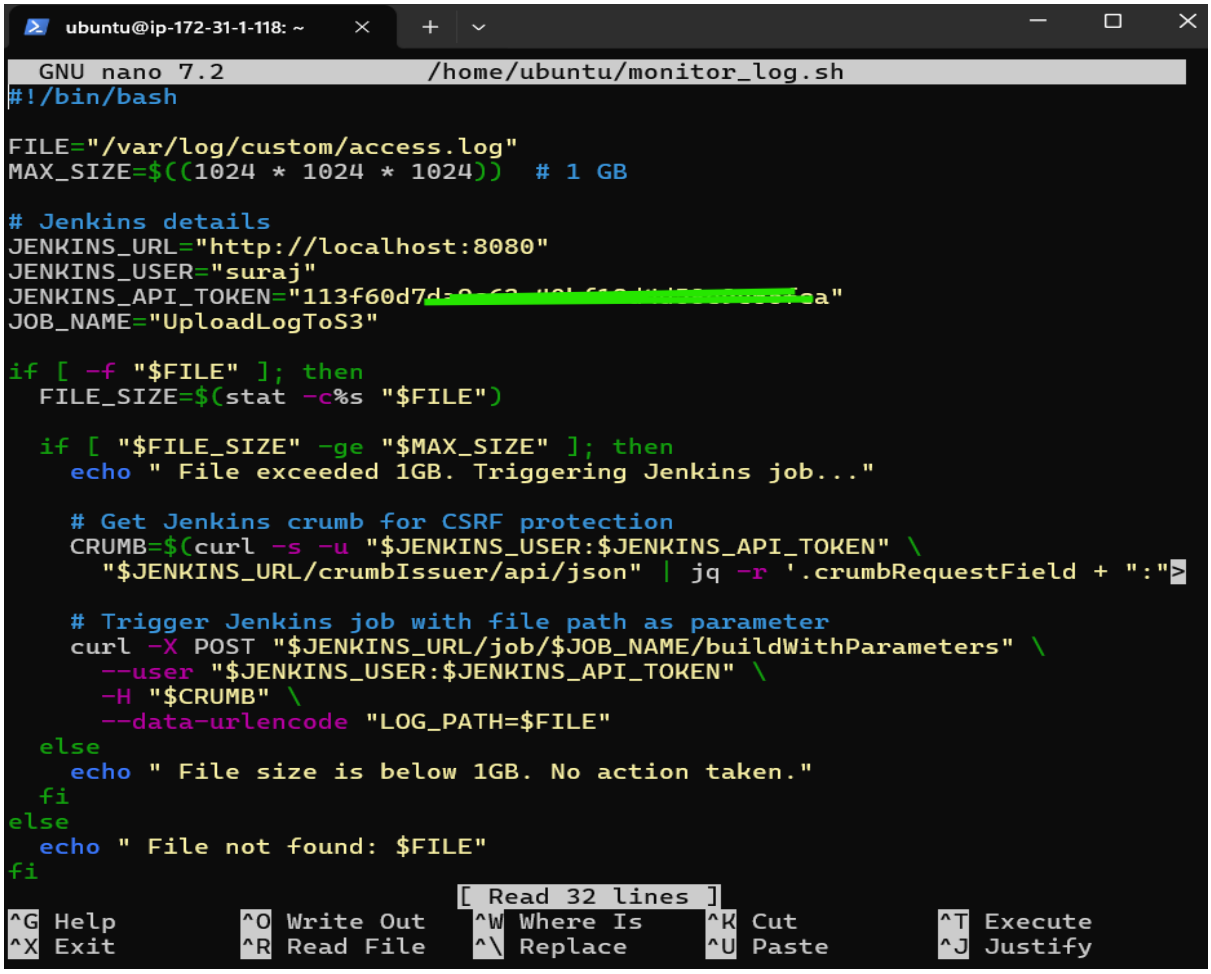


STEP 2: Create S3 Bucket:



STEP 3: Create Shell Script to Monitor Log Size:

nano ~/monitor_log.sh



Explanation of Script:

Section 1: File Path and Size Limit Definition

In this section, the path to the log file that needs to be monitored is defined. Additionally, a size

threshold is set to 1 GB (in bytes). This value is used to compare against the actual file size later.

Section 2: Jenkins Configuration

This section contains all the necessary Jenkins connection details:

• Jenkins server URL (usually localhost or a remote Jenkins server)

• Username to authenticate with Jenkins

• Jenkins API token for secure access

• Name of the Jenkins job that will be triggered if conditions are met

Section 3: Check if File Exists and Get File Size

Here, the script checks whether the specified log file actually exists on the system. If it does, the script

retrieves the file's current size in bytes using system utilities.

Section 4: Check If File Size Exceeds Threshold

This conditional section compares the actual file size with the defined 1 GB limit. If the file size is greater

than or equal to 1 GB, it proceeds to trigger a Jenkins job.

Section 6: Get CSRF Protection Crumb from Jenkins

Jenkins uses CSRF protection to prevent malicious requests. To authenticate POST requests, Jenkins

issues a special token called a crumb. In this section, the script fetches this crumb using Jenkins's API

and prepares it for use in the next step.

Section 7: Trigger Jenkins Job with Parameters

Once the crumb is obtained, the script triggers the specified Jenkins job via a POST request. It passes

the log file's path as a parameter to the Jenkins job and includes the crumb and authentication token in

the request.

Section 8: Log File Size Below Threshold

If the file exists but is smaller than 1 GB, the script prints an informational message and exits without

triggering the Jenkins job. No further action is taken.

Section 9: File Not Found Handler: If the specified log file does not exist at all, the script logs an error

message indicating that the file could not be found and terminates without performing any operation.

ii)Make it executable:

chmod +x ~/monitor_log.sh

STEP 4 : Created Jenkins Api Token

go to security ---- click on jenkins api ---give name--- create

Why: created the Jenkins API token to securely authenticate your script with Jenkins without using your

actual password. It allows the script to trigger Jenkins jobs and fetch CSRF protection crumbs via the

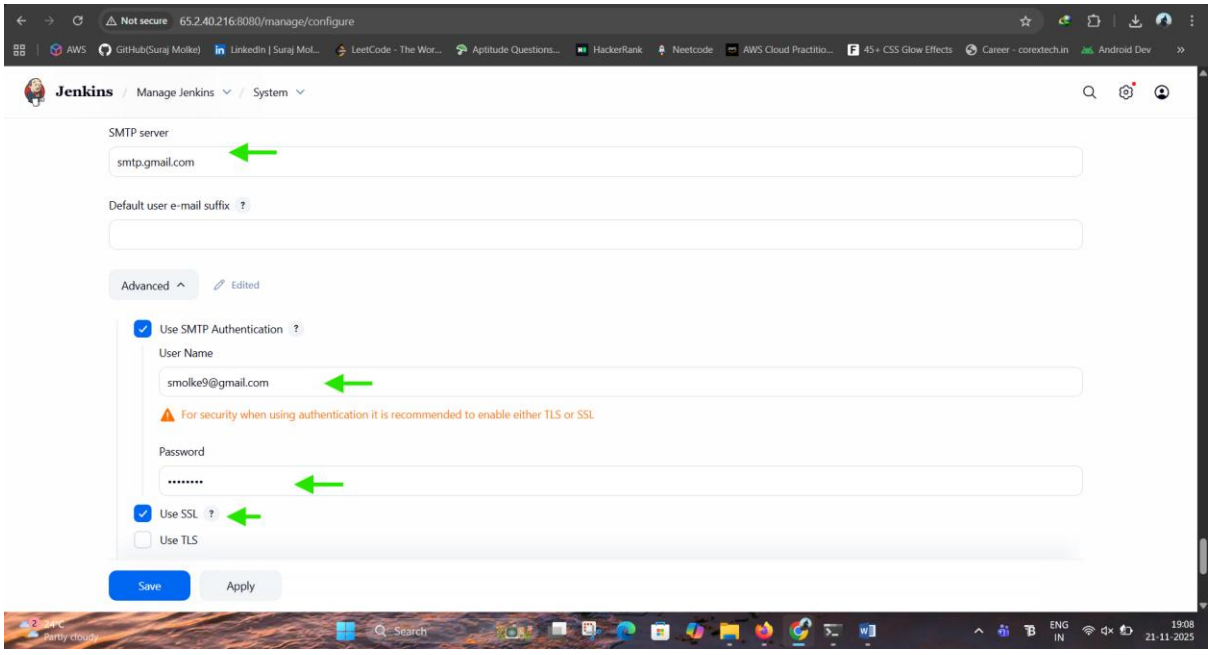
REST API.

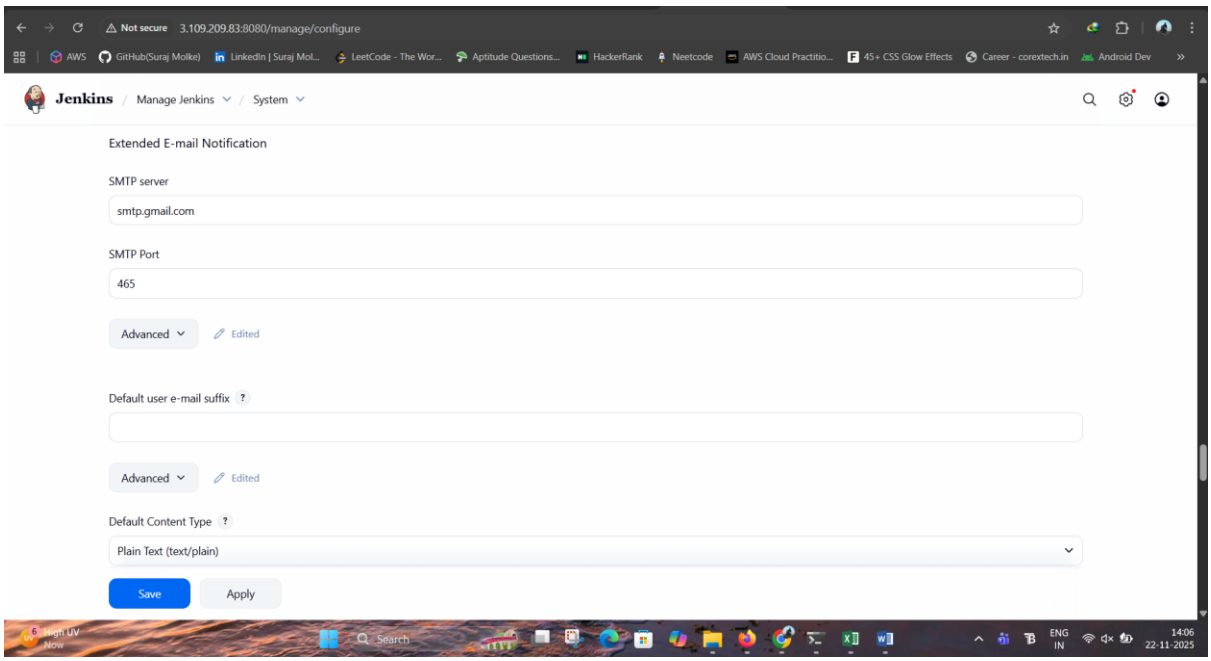
STEP 6 : Setting up email notification:

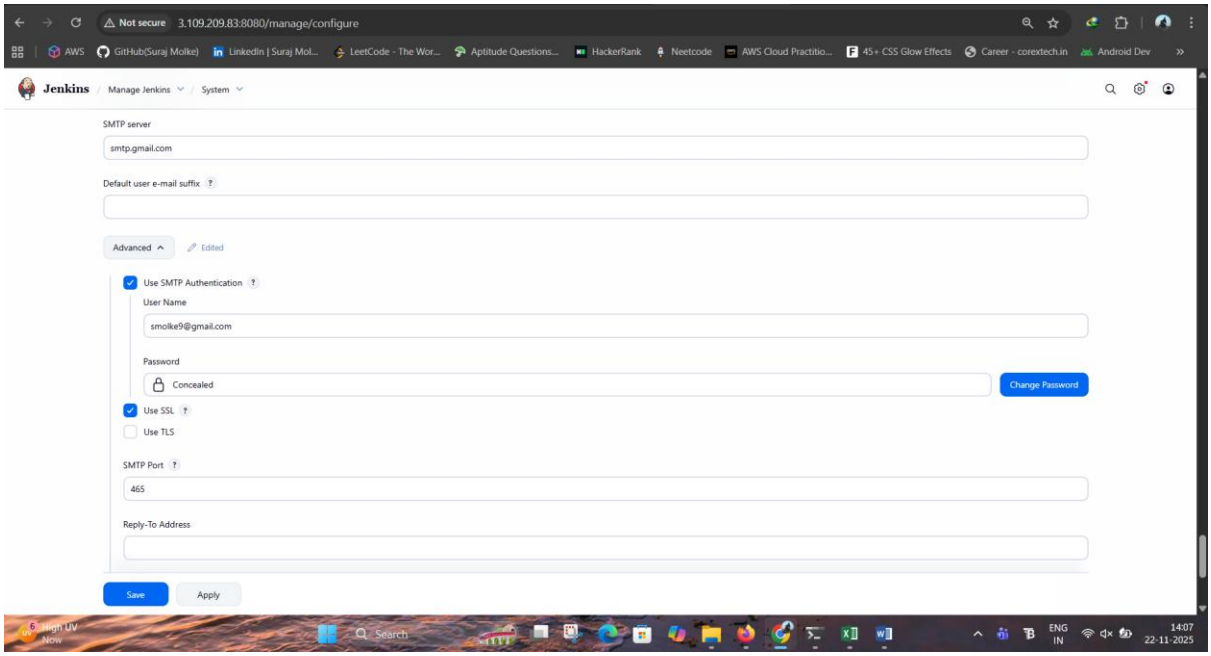
To set up email notifications in Jenkins,

go to Manage Jenkins → Configure System, and configure the E-mail Notification section with your

SMTP server details. Then save and test the configuration.







STEP 6 : Create New Job

• Go to Jenkins → New Item → "UploadLogToS3"

• Choose Pipeline

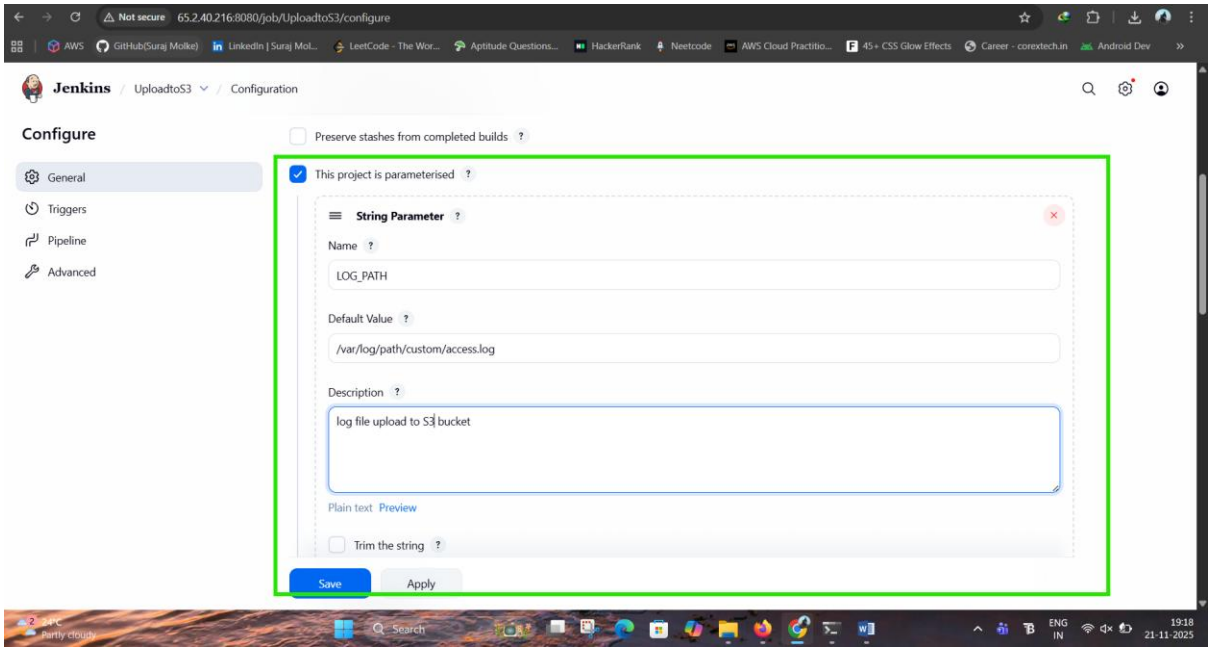
• Add a string parameter: LOG_PATH

Why:

create a string parameter LOG_PATH in the Jenkins job so the script can pass the log file path to the

pipeline during the build trigger. This allows the pipeline to know which specific log file to process or upload

.



ii)Added security credentials in Jenkins:

Gmail does NOT allow normal password login from Jenkins.

Follow this:

1. Open Google Account → Security

https://myaccount.google.com/security

2. Enable 2-Step Verification

(Required for App Password)

3. Create App Password

Go to:
Security → App Passwords → Select App → Other → Type "Jenkins"

Google will generate a 16-digit password

4— Configure Jenkins Email (SMTP) Settings

Go to:

Jenkins Dashboard → Manage Jenkins → Configure System

Scroll to **Extended E-mail Notification**

Fill in:

✔ **SMTP Server**

smtp.gmail.com

✔ **SMTP Port**

465

✔ **Use SSL**

✔ Check this

✔ **Authentication**

Username: your Gmail (example: smolke9@gmail.com)
Password: <your Gmail App Password>

✔ **Default Content Type**

text/html

5— **Also Configure "E-mail Notification" section**

Scroll down:

SMTP Server

smtp.gmail.com

Click **Advanced**, set:

Port: 465
Use SSL: ✔
Username: your Gmail
Password: <App Password>
Reply-To: your Gmail
Charset: UTF-8

6— Test Email

Click **"Test configuration"**

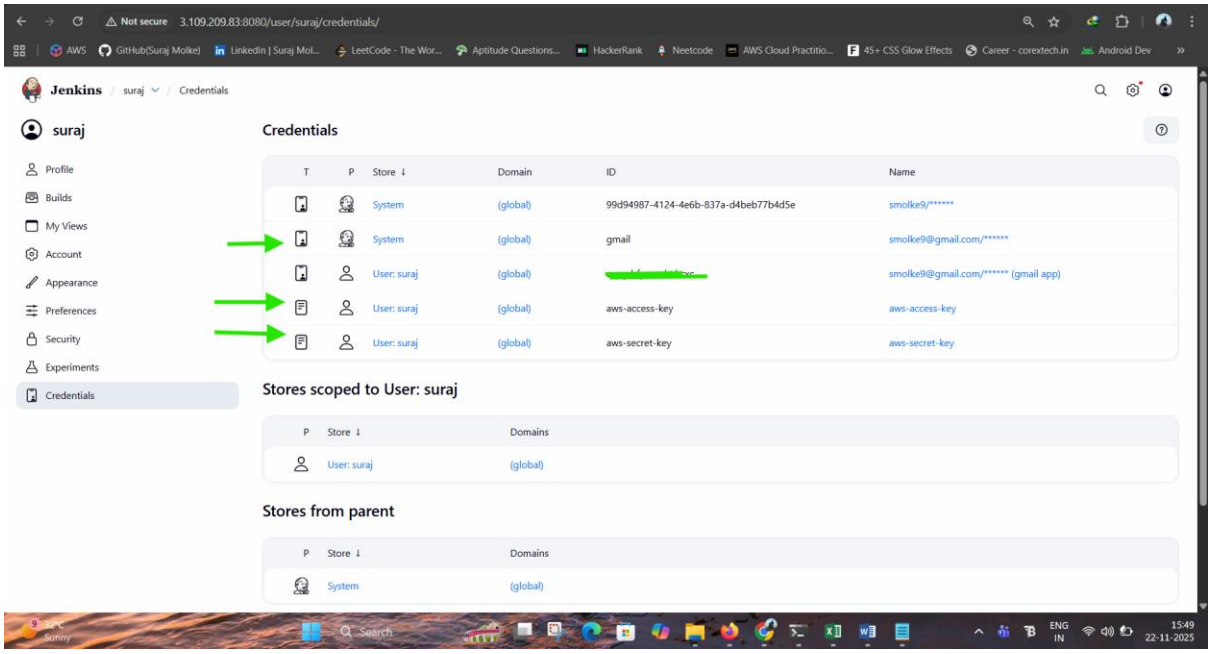
Enter your email:

smolke9@gmail.com

Click **Send Test Email**

You should see: Email sent successfully!

Add Gmail or aws access key and secret key

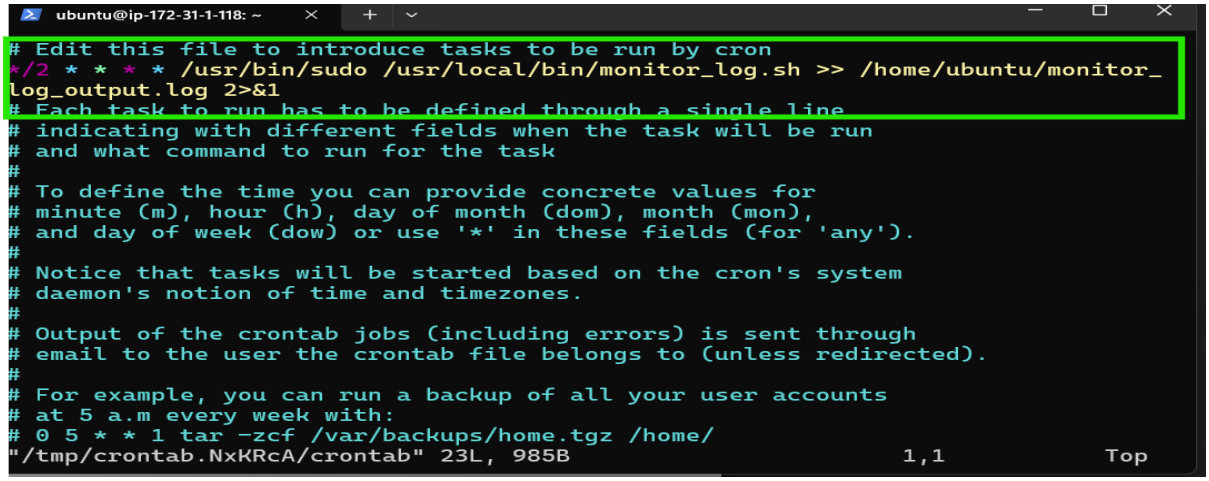


STEP 7 : Edit Crontab

sudo cp /home/ubuntu/monitor_log.sh /usr/local/bin/monitor_log.sh

sudo chmod +x /usr/local/bin/monitor_log.sh

crontab -e



*/2 * * * * /usr/bin/sudo /usr/local/bin/monitor_log.sh >> /home/ubuntu/monitor_log_output.log 2>&1

If you want, I can also make a **systemd service** that runs every minute — more reliable than cron.

STEP 9 : Testing

i) manually increased the size of log file for testing: Using:

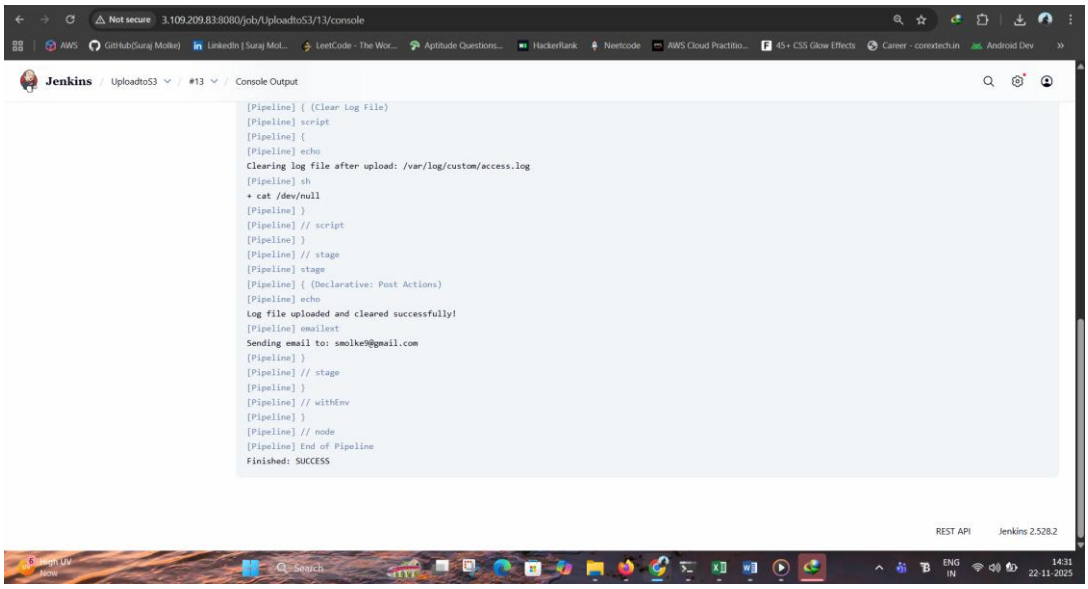
truncate -s 1G /var/log/custom/access.log

ii)Run the monitor_log.sh

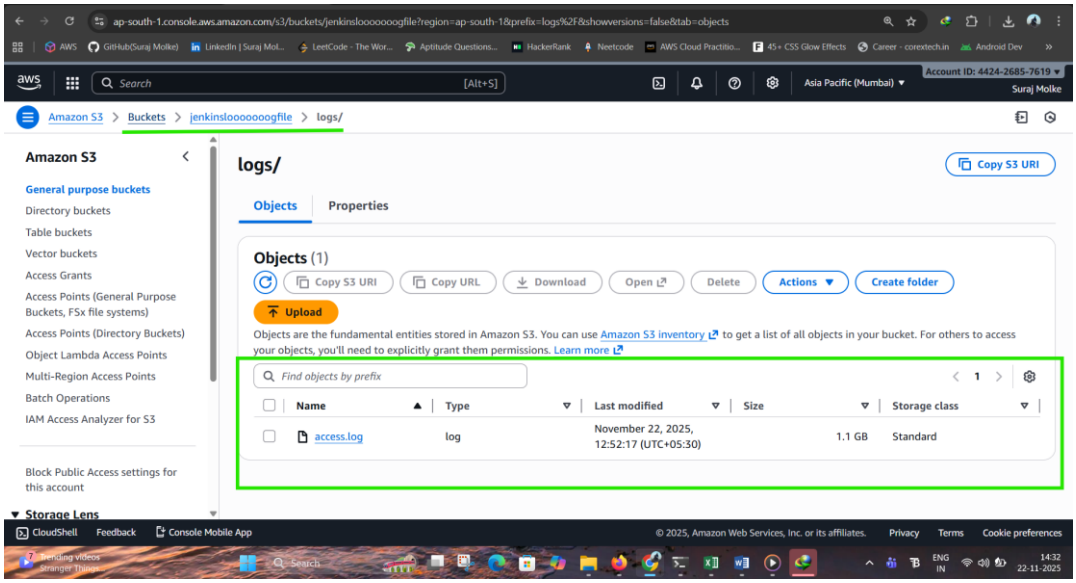
sudo ./monitor_log.sh

```
ubuntu@ip-172-31-1-118:~$ sudo truncate -s 1100M /var/log/custom/access.log
ubuntu@ip-172-31-1-118:~$ sudo ./monitor_log.sh
Current file size: 1153433600 bytes
[INFO] File exceeded 1GB. Triggering Jenkins job...
[INFO] Triggering Jenkins job: UploadtoS3
[SUCCESS] Jenkins job triggered successfully!
```

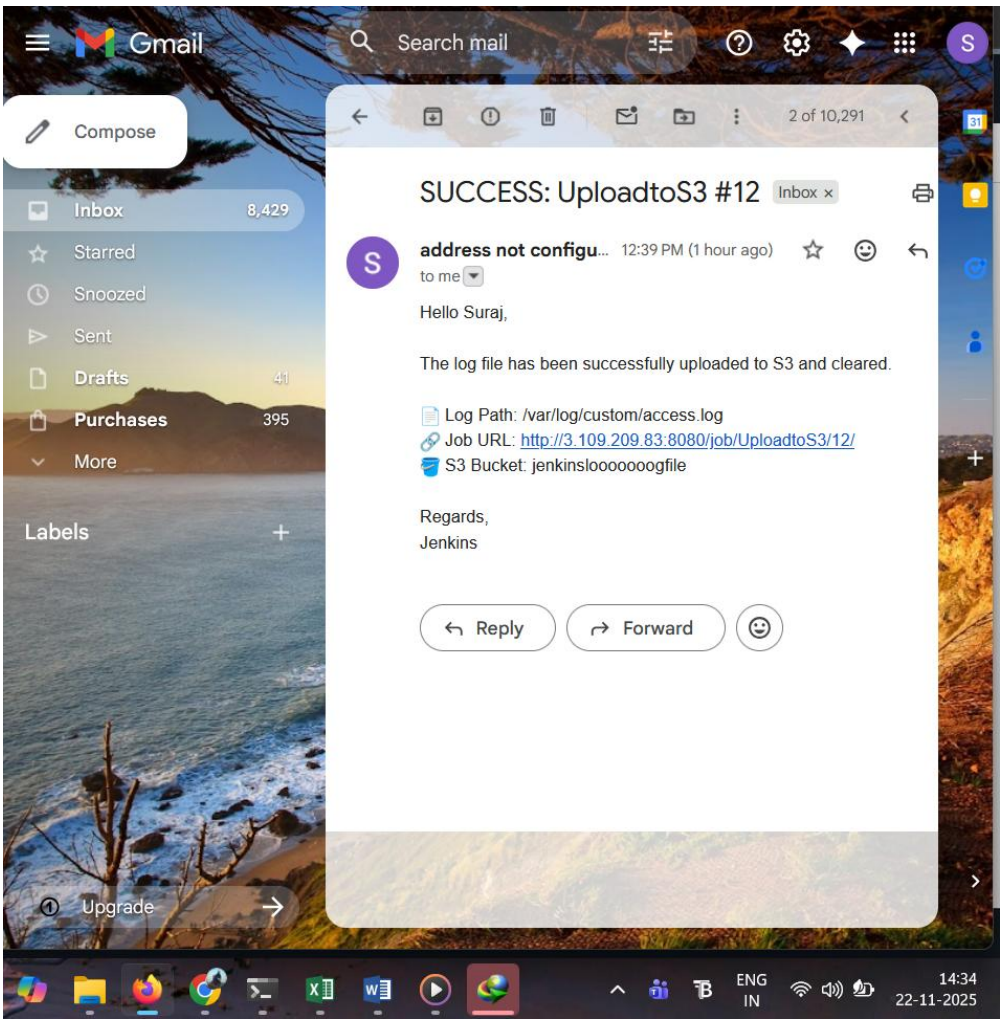
iii)Automatically builds the pipeline:



iv) Log File uploaded to s3:



v) Email notification Received:



Conclusion

This project delivers a fully automated and intelligent log management workflow powered by Bash scripting, Jenkins CI/CD, AWS S3, and secure credential handling. A lightweight Bash monitoring script continuously checks system log size, and whenever the file exceeds 1 GB, it automatically triggers a Jenkins job with the log path passed parameter.

Within Jenkins, the pipeline validates the file, securely uploads it to an Amazon S3 bucket using AWS CLI, and clears the local logs—ensuring optimal disk usage and preventing system overload. The pipeline also integrates robust error handling with email notifications, ensuring real-time alerts on both success and failure.

Security is reinforced through Jenkins crumbs for CSRF protection and encrypted AWS credentials, making the automated process reliable, secure, and scalable. This setup eliminates manual intervention and provides a seamless, automated cloud-based log archiving system ideal for production environments.