Project Title: Automated Jenkins Job Triggered by Access Log Size

---

Objective:

Create an automated system using Jenkins that monitors an access log file (e.g., `/var/log/nginx/access.log`). If the log file exceeds 1GB in size, a Jenkins job will:

1. Upload the log file to an Amazon S3 bucket.
2. Clear the original log file to make room for new entries.

Tools and Technologies Used:

- **Ubuntu EC2 Instance** (for Jenkins and scripting)
- **Jenkins** (Automation)
- **Amazon S3** (Storage for log backups)
- **AWS CLI** (For interacting with S3)
- **Cron Job** (Automation and scheduling)
- **Shell Scripting** (Monitoring and automation logic)

Step-by-Step Implementation:

Step 1: Launch an Ubuntu EC2 Instance

- Choose Ubuntu Server 22.04 LTS
- Create a key pair and open ports: 22, 80, 8080
- SSH into your instance

Step 2: Install Jenkins

```
sudo apt update
sudo apt install openjdk-11-jdk -y
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo apt update
sudo apt install jenkins -y
sudo systemctl start jenkins
sudo systemctl enable jenkins
```

Step 3: Install & Configure AWS CLI

```
sudo apt update
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Then configure:

```
aws configure
# Enter Access Key, Secret Key, Region, Output Format
```

Step 4: Create S3 Bucket

- Go to AWS S3 Console

- Create a bucket: `access-log-backup-prashant`
- Disable ACLs (recommended)
- Enable “Block all public access”

Step 5: Create Jenkins Job

- Open Jenkins: `http://<public-ip>:8080`
- Create Freestyle Project: `upload-log-to-s3`
- In “Build” section:

```
#!/bin/bash
TS=$(date +"%Y%m%d%H%M%S")
TMPFILE="/tmp/access-$TS.log"

# Copy the current access log
sudo cp /var/log/nginx/access.log $TMPFILE

# Upload to S3
aws s3 cp $TMPFILE s3://access-log-backup-prashant/
if [ $? -eq 0 ]; then
    echo "✅Upload successful, clearing original log file..."
    sudo sh -c '> /var/log/nginx/access.log'
else
    echo "❌Upload failed!"
fi
```

Step 6: Create Monitoring Script

Create file: `check-log-size.sh`

```
#!/bin/bash
FILE="/var/log/nginx/access.log"
MAXSIZE=$((1024 * 1024 * 1024)) # 1 GB

ACTUALSIZE=$(stat -c%s "$FILE")
if [ "$ACTUALSIZE" -gt "$MAXSIZE" ]; then
    echo "Log file exceeded 1GB. Triggering Jenkins Job."
    curl -X POST http://localhost:8080/job/upload-log-to-s3/build --user
admin:<API_TOKEN>
else
    echo "Log size under limit. Nothing to do."
fi
```

Make it executable:

```
chmod +x check-log-size.sh
```

Step 7: Automate with Cron

Run every 5 mins:

```
crontab -e
```

Add:

```
*/5 * * * * /home/ubuntu/check-log-size.sh >> /tmp/log-monitor.log 2>&1
```

How We Created a 1GB Log File for Testing:

```
base64 /dev/urandom | head -c 1G > /var/log/nginx/access.log
```

Architecture Flow:

1. **Shell Script** checks access log file size.
2. If size > 1GB, **Jenkins job** is triggered.
3. Jenkins job:
    - Copies log to S3
    - Clears the original log
4. **Cron job** ensures script runs every 5 minutes.

Final Deliverables:

- Fully working Jenkins Job
- Automated backup to S3
- Shell script & cron integration
- Logs rotated and cleared after backup
- Tested with large file uploads

GitHub Project Repository:

Link: https://github.com/prashantgharate/Automated-Jenkins-Job-Triggered-by-Access-Log-Size.git

This project is a powerful demonstration of DevOps automation using scripting, scheduling, cloud storage, and CI/CD tools like Jenkins. It's scalable, simple, and production-ready.