Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

AY: 2025-26

Semester: V

Subject: DevOps Laboratory (DJS23OLOE501)

Experiment 6a

(DevOps Automation and Scripting)

Name : Parth Savla **Roll No: D135**

Aim: To Automate System Tasks Using Bash and Python Scripts

Theory:

Why Automation Matters in DevOps

In real-world DevOps, many tasks are repetitive, such as provisioning servers, monitoring system health, deploying updates, and handling service failures. Performing these tasks manually is slow and prone to human error, which can lead to system downtime.
Automation through scripting is the core solution to this problem. It ensures that operations are
repeatable, **reliable**, and executed with **speed**, forming the foundation of an efficient DevOps lifecycle.

Technologies Used

- **Bash**: A powerful command-line interpreter and scripting language native to Linux. It is ideal for system-level automation, such as managing packages, files, and services.
- **Python**: A versatile, high-level programming language used for more complex automation tasks. Its extensive libraries make it perfect for log monitoring, data processing, and creating interactive tools like chatbots.
- **Cron**: A standard time-based job scheduler in Unix-like operating systems. It is used to automate repetitive tasks by running scripts or commands at specified intervals (e.g., nightly backups or weekly reports).
- **Systemctl**: The primary tool for managing services on Linux systems that use the systemd init system. It is used to start, stop, restart, and check the status of services like Nginx.

Understanding Cron Command Syntax

The cron daemon runs tasks based on a schedule defined in a file called a crontab. Each line in a crontab file represents a single job and follows a specific format:
* * * * * /path/to/command
The five asterisks represent the time and date fields for the schedule, followed by the command to be executed.

Department of Computer Science and Engineering (Data Science)

```
┌──────────── minute (0 - 59)
│ ┌────────── hour (0 - 23)
│ │ ┌──────── day of month (1 - 31)
│ │ │ ┌────── month (1 - 12)
│ │ │ │ ┌──── day of week (0 - 6) (Sunday to Saturday)
│ │ │ │ │
* * * * * <command_to_execute>
```

Common Syntax Examples:

- `0 0 * * *` - Run every day at midnight.
- `*/5 * * * *` - Run every 5 minutes.
- `0 2 * * 1` - Run every Monday at 2:00 AM.

Prerequisites

Before starting:
sudo apt update -y
sudo apt install python3 python3-pip -y

Example
Scenario: "Automated Feedback Monitoring and Alert System for a Web Application"

Real-World Context
You're a DevOps engineer maintaining a **customer feedback web service**.
Users submit feedback through a web form that stores entries in a simple **log file (feedback.log)** or database.
Your DevOps goal is to:

1. **Automate setup** of this small web feedback system.
2. **Monitor feedback logs** for **negative sentiments or complaints** (like "bad", "poor", "slow", "error").
3. **Send alerts** or perform **auto cleanup/archival** of old feedback logs.
4. **Restart** the feedback web service automatically if it fails.
5. **Provide a chatbot** interface for checking feedback trends interactively.

This scenario demonstrates **DevOps observability and automation** — from installation to proactive monitoring.

Department of Computer Science and Engineering (Data Science)

Step-by-Step Breakdown

1.Setup & Install Tools

First, create a Bash script to automate the installation of Nginx, Python, and the required directory structure.
sudo vim install_feedback_system.sh
#!/bin/bash
echo "Installing feedback monitoring environment..."

sudo apt update -y
sudo apt install -y python3 python3-pip nginx

Create a mock feedback folder
mkdir -p /var/www/feedback
touch /var/www/feedback/feedback.log
chmod 666 /var/www/feedback/feedback.log

Create a simple feedback page
echo '<html><body><h2>Feedback Page</h2><form method="post" action="/submit"><input name="feedback"><input type="submit"></form></body></html>' | sudo tee /var/www/html/index.html

echo "Feedback system installed. Visit http://$(hostname -I | awk '{print $1}')"

Run:
bash install_feedback_system.sh

```
aditidraut46@scripting:~$ sudo bash install_feedback_system.sh
Installing feedback monitoring environment...
Hit:1 http://us-central1.gce.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-central1.gce.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-central1.gce.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 https://packages.cloud.google.com/apt google-cloud-ops-agent-jammy-2 InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nginx is already the newest version (1.18.0-6ubuntu14.7).
python3 is already the newest version (3.10.6-1~22.04.1).
python3-pip is already the newest version (22.0.2+dfsg-1ubuntu0.7).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
<html><body><h2>Feedback Page</h2><form method="post" action="/submit"><input name="feedback"><input type="subm
it"></form></body></html>
Feedback system installed. Visit http://10.128.0.12
```

2. Simulate Feedback Entries

Since we don't have a full backend, we simulate feedback logs:
echo "User: Feedback - Website is slow" >> /var/www/feedback/feedback.log
echo "User: Feedback - Great service!" >> /var/www/feedback/feedback.log
echo "User: Feedback - Poor response time" >> /var/www/feedback/feedback.log

```
aditidraut46@scripting:~$ echo "User: Feedback - Website is slow" >> /var/www/feedback/feedback.log
aditidraut46@scripting:~$ echo "User: Feedback - Great service!" >> /var/www/feedback/feedback.log
aditidraut46@scripting:~$ echo "User: Feedback - Poor response time" >> /var/www/feedback/feedback.lo
-bash: /var/www/feedback/feedback.lo: Permission denied
aditidraut46@scripting:~$ echo "User: Feedback - Poor response time" >> /var/www/feedback/feedback.log
aditidraut46@scripting:~$
```

3.Monitor Feedback Logs for Negative Sentiments

Create a Python script that continuously watches the log file and prints an alert if it detects negative keywords

sudo vim feedback_monitor.py

```
import time
import os

log_file = "/var/www/feedback/feedback.log"
alert_log = "/var/log/feedback_alerts.log"
negative_keywords = ["bad", "poor", "slow", "error", "not good"]

print("Monitoring feedback logs for negative sentiment...")

def monitor_feedback():
    with open(log_file, "r") as f:
        f.seek(0, os.SEEK_END)
        while True:
            line = f.readline()
            if not line:
                time.sleep(1)
                continue
            if any(word in line.lower() for word in negative_keywords):
                alert = f" ALERT: Negative feedback detected -> {line.strip()}"
                print(alert)
                with open(alert_log, "a") as a:
                    a.write(alert + "\n")

if __name__ == "__main__":
    monitor_feedback()
```

Run:

sudo python3 feedback_monitor.py

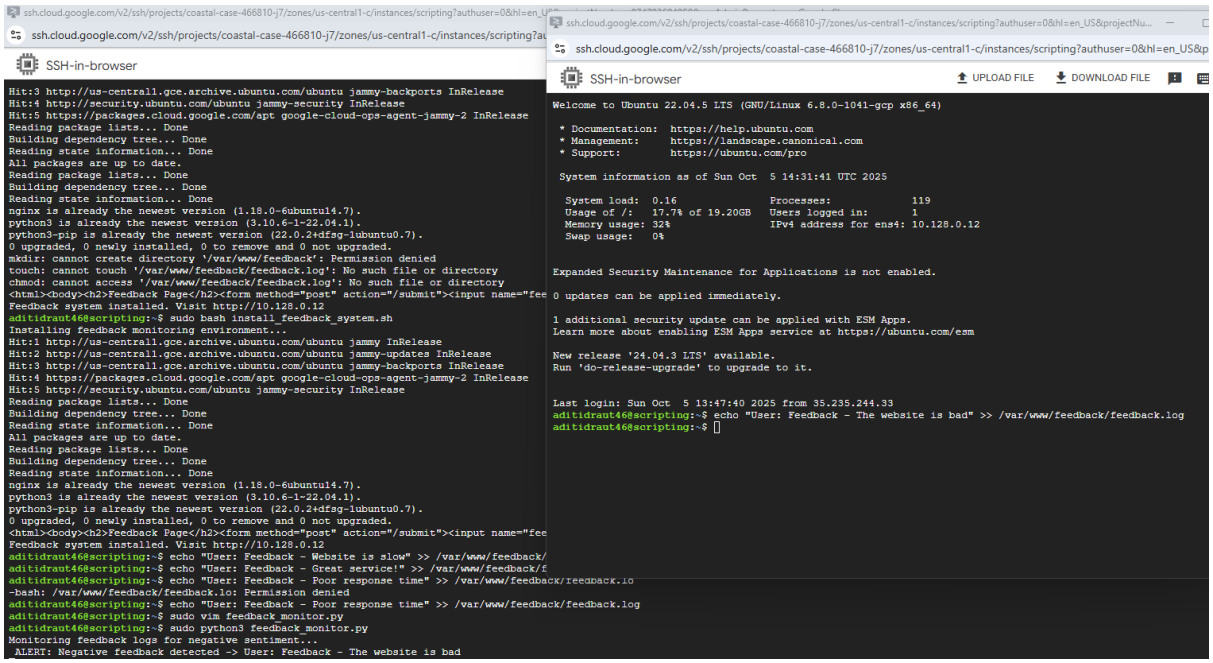
Then simulate a complaint:

```
aditidraut46@scripting:~$ sudo python3 feedback_monitor.py
Monitoring feedback logs for negative sentiment...
```

Keep current terminal open and test it by writing following command in the second terminal:

$ echo "User: Feedback - The website is bad" >> /var/www/feedback/feedback.log

→ Output:



Lab experiment to be performed in this session:

Tasks to Perform:

Environment Setup & Log Monitoring

This task covers automating the setup of a monitoring environment and actively watching application logs for critical errors.

Scenario - Monitoring a Web Application's Health

You are responsible for a web application that writes its status to a log file at /var/log/app_health.log. Your goal is to automate the setup of the Nginx server and create a Python script that constantly monitors this log for critical keywords like "ERROR" or "CRITICAL".

1. **Automate Environment Setup:**
    a. Create a new Bash script named setup_health_monitor.sh.
        i. Install nginx and python3.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ sudo apt update -y
sudo apt install -y python3 python3-pip nginx
Get:1 https://download.docker.com/linux/ubuntu noble InRelease [48.8 kB]
Get:2 https://cli.github.com/packages stable InRelease [3,917 B]
Get:3 https://packages.cloud.google.com/apt gcsfuse-noble InRelease [1,227 B]
Get:4 https://packages.cloud.google.com/apt cloud-sdk InRelease [1,620 B]
Get:5 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [37.7 kB]
Hit:6 http://archive.ubuntu.com/ubuntu noble InRelease
Get:7 https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 InRelease [3,005 B]
Get:8 https://apt.postgresql.org/pub/repos/apt noble-pgdg InRelease [107 kB]
Get:9 https://cli.github.com/packages stable/main amd64 Packages [347 B]
Get:10 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:11 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:12 https://packages.cloud.google.com/apt cloud-sdk/main all Packages [1,839 kB]
Get:13 https://packages.cloud.google.com/apt cloud-sdk/main amd64 Packages [4,193 kB]
Hit:14 https://ppa.launchpadcontent.net/dotnet/backports/ubuntu noble InRelease
Get:15 https://apt.postgresql.org/pub/repos/apt noble-pgdg/main amd64 Packages [562 kB]
Get:16 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:17 https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0/multiverse amd64 Packages [51.3 kB]
Get:18 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1,924 kB]
Get:19 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1,138 kB]
Get:20 http://archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Packages [2,605 kB]
Get:21 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1,879 kB]
Get:22 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [2,462 kB]
Get:23 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1,512 kB]
Fetched 18.7 MB in 3s (5,625 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
31 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
```

ii. Create a log file specifically at /var/log/app_health.log.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ cat > setup_health_monitor.sh <<'EOF'
#!/bin/bash
set -e

echo "Installing packages and setting up environment..."
sudo apt update -y
sudo apt install -y nginx python3 python3-pip

echo "Enabling and starting nginx..."
sudo systemctl enable --now nginx

echo "Creating log files..."
sudo touch /var/log/app_health.log /var/log/health_alerts.log
sudo chmod 666 /var/log/app_health.log /var/log/health_alerts.log

echo "Optional: create a simple web page for nginx"
echo '<html><body><h2>App Health Monitor (Nginx)</h2></body></html>' | sudo tee /var/www/html/index.html > /dev/null

echo "Setup completed. Log file: /var/log/app_health.log"
EOF
```

iii. Ensure the log file is writable.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ chmod +x setup_health_monitor.sh
./setup_health_monitor.sh
Installing packages and setting up environment...
Hit:1 https://apt.postgresql.org/pub/repos/apt noble-pgdg InRelease
Hit:2 https://cli.github.com/packages stable InRelease
Hit:3 https://packages.cloud.google.com/apt gcsfuse-noble InRelease
Hit:4 https://download.docker.com/linux/ubuntu noble InRelease
Hit:5 https://packages.cloud.google.com/apt cloud-sdk InRelease
Hit:6 http://archive.ubuntu.com/ubuntu noble InRelease
Get:7 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Hit:8 http://archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:9 https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 InRelease
Hit:10 http://archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:11 https://ppa.launchpadcontent.net/dotnet/backports/ubuntu noble InRelease
Fetched 126 kB in 1s (90.8 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
31 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

b. Execute your script to prepare the environment.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ bash setup_health_monitor.sh
Installing packages and setting up environment...
Hit:1 https://cli.github.com/packages stable InRelease
Hit:2 https://download.docker.com/linux/ubuntu noble InRelease
Hit:3 https://apt.postgresql.org/pub/repos/apt noble-pgdg InRelease
Hit:4 https://packages.cloud.google.com/apt gcsfuse-noble InRelease
Hit:5 https://packages.cloud.google.com/apt cloud-sdk InRelease
Hit:6 http://archive.ubuntu.com/ubuntu noble InRelease
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:8 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Hit:9 https://repo.mongodb.org/apt/ubuntu noble/mongodb-org/8.0 InRelease
Hit:10 https://ppa.launchpadcontent.net/dotnet/backports/ubuntu noble InRelease
Get:11 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Fetched 252 kB in 1s (173 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
31 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nginx is already the newest version (1.24.0-2ubuntu7.5).
python3 is already the newest version (3.12.3-0ubuntu2).
python3-pip is already the newest version (24.0+dfsg-1ubuntu1.3).
```

2. **Simulate Application Log Entries:**
   a. Manually add several lines to /var/log/app_health.log to simulate the application's output. Include both normal and error messages.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ # normal messages
sudo bash -c 'echo "INFO: Application started successfully" >> /var/log/app_health.log'
sudo bash -c 'echo "INFO: Processing request /home" >> /var/log/app_health.log'

# warnings/errors
sudo bash -c 'echo "WARNING: Slow response detected for /api/data" >> /var/log/app_health.log'
sudo bash -c 'echo "ERROR: Database connection failed" >> /var/log/app_health.log'
sudo bash -c 'echo "CRITICAL: Out of memory" >> /var/log/app_health.log'
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$
```

3. **Monitor Logs for Critical Errors:**
   a. Create a Python script named health_monitor.py.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ cat > health_monitor.py <<'PY'
#!/usr/bin/env python3
import time
import os
import sys
from datetime import datetime

LOG_FILE = "/var/log/app_health.log"
ALERT_FILE = "/var/log/health_alerts.log"
KEYWORDS = ["error", "critical", "failed", "timeout"]

def ensure_files():
    if not os.path.exists(LOG_FILE):
        open(LOG_FILE, 'a').close()
    if not os.path.exists(ALERT_FILE):
        open(ALERT_FILE, 'a').close()

def write_alert(line):
    ts = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    alert = f"{ts} ALERT: {line.strip()}"
    print(alert)
    with open(ALERT_FILE, "a") as f:
        f.write(alert + "\n")

def monitor():
    ensure_files()
    with open(LOG_FILE, "r") as f:
        # go to end of file
        f.seek(0, os.SEEK_END)
        print(f"Monitoring {LOG_FILE} for keywords: {KEYWORDS}")
        while True:
            line = f.readline()
            if not line:
                time.sleep(1)
                continue
            lower = line.lower()
            if any(k in lower for k in KEYWORDS):
                write_alert(line)

if __name__ == "__main__":
```

b. Modify the script to:

i. Read from /var/log/app_health.log.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ # run in foreground so you can see alerts printed to terminal
sudo python3 health_monitor.py
# or without sudo if you used setup script (files are chmod 666) :
# python3 health_monitor.py
Monitoring /var/log/app_health.log for keywords: ['error', 'critical', 'failed', 'timeout']
```

ii. Search for a new list of negative_keywords such as ["error", "critical", "failed", "timeout"].

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ # add a new error line to trigger the alert
sudo bash -c 'echo "ERROR: Unable to write to disk" >> /var/log/app_health.log'
```

iii. Write any detected alerts to a new file: /var/log/health_alerts.log.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ tail -n 50 /var/log/health_alerts.log
# or follow it live:
tail -f /var/log/health_alerts.log
```

c. Run the script in a terminal and test it by adding another "ERROR" line to app_health.log in a separate terminal.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ python3 health_monitor.py
Monitoring /var/log/app_health.log for keywords: ['error', 'critical', 'failed', 'timeout']
2025-10-06 09:38:10 ALERT: ERROR: Database connection timeout on request ID 1023
```

d. **Document** the output, showing your script detecting the new error.

```
student_02_4e45d393a67c@cloudshell:~ (qwiklabs-gcp-03-a1846a9aa90f)$ echo "ERROR: Database connection timeout on request ID 1023" >> ~/logs/app_health.lo
g
```