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 6b

(DevOps Automation and Scripting)

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

Aim: To Perform Service Management and Chatbot Automation.

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.

```
┌──────────── 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.

1. Schedule Feedback Cleanup and Archival

/*Step 1: Create a backup directory

Before archiving, you need a folder to store the archived feedback logs.

sudo mkdir -p /var/backups/feedback
sudo chown root:root /var/backups/feedback
sudo chmod 755 /var/backups/feedback*/ skip

Verify con status

```
aditidraut46@scripting:~$ sudo systemctl status cron
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-10-05 13:42:23 UTC; 1h 46min ago
       Docs: man:cron(8)
   Main PID: 838 (cron)
      Tasks: 1 (limit: 4687)
     Memory: 432.0K
        CPU: 40ms
     CGroup: /system.slice/cron.service
             └─838 /usr/sbin/cron -f -P

Oct 05 13:42:23 scripting systemd[1]: Started Regular background program processing daemon.
Oct 05 13:42:23 scripting cron[838]: (CRON) INFO (pidfile fd = 3)
Oct 05 13:42:23 scripting cron[838]: (CRON) INFO (Skipping @reboot jobs -- not system startup)
Oct 05 14:17:01 scripting CRON[6530]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
Oct 05 14:17:01 scripting CRON[6531]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Oct 05 14:17:01 scripting CRON[6530]: pam_unix(cron:session): session closed for user root
Oct 05 15:17:01 scripting CRON[10259]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
Oct 05 15:17:01 scripting CRON[10260]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Oct 05 15:17:01 scripting CRON[10259]: pam_unix(cron:session): session closed for user root
```

Add cron job:

A. Create the Auto-Restart Script

This script will check if Nginx is running and restart it if it has failed.

sudo vim /usr/local/bin/auto_restart_feedback.sh

```
#!/bin/bash
SERVICE="nginx"

/usr/bin/systemctl is-active --quiet $SERVICE
if [ $? -eq 0 ]; then
  echo "$(date):Feedback service (Nginx) running." >> /var/log/feedback_restart.log
else
  echo "$(date):Feedback service stopped. Restarting..." >> /var/log/feedback_restart.log
  /usr/bin/systemctl restart $SERVICE
  echo "$(date): Restarted feedback service." >> /var/log/feedback_restart.log
fi
```

Make the script executable:

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

Add the Cron Jobs

Run this command:

sudo env EDITOR=vim crontab -e

```
aditidraut46@scripting:~$ sudo env EDITOR=vim crontab -e
crontab: installing new crontab
aditidraut46@scripting:~$ sudo crontab -l
#
# with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
# ===== Feedback System Maintenance Jobs ===
# ===== CloudOps AutoPilot Cron Jobs =====

# (1) Archive feedback logs every night at midnight
0 0 * * * tar -czf /var/backups/feedback_$(date +\%F).tar.gz /var/www/feedback/feedback.log && echo "" > /var/www/feedback/feedback.log

# (2) Auto-restart Nginx feedback service every 3 minutes (output saved to log)
*/3 * * * * /usr/local/bin/auto_restart_feedback.sh >> /var/log/feedback_restart.log 2>&1
```

Add your cron jobs

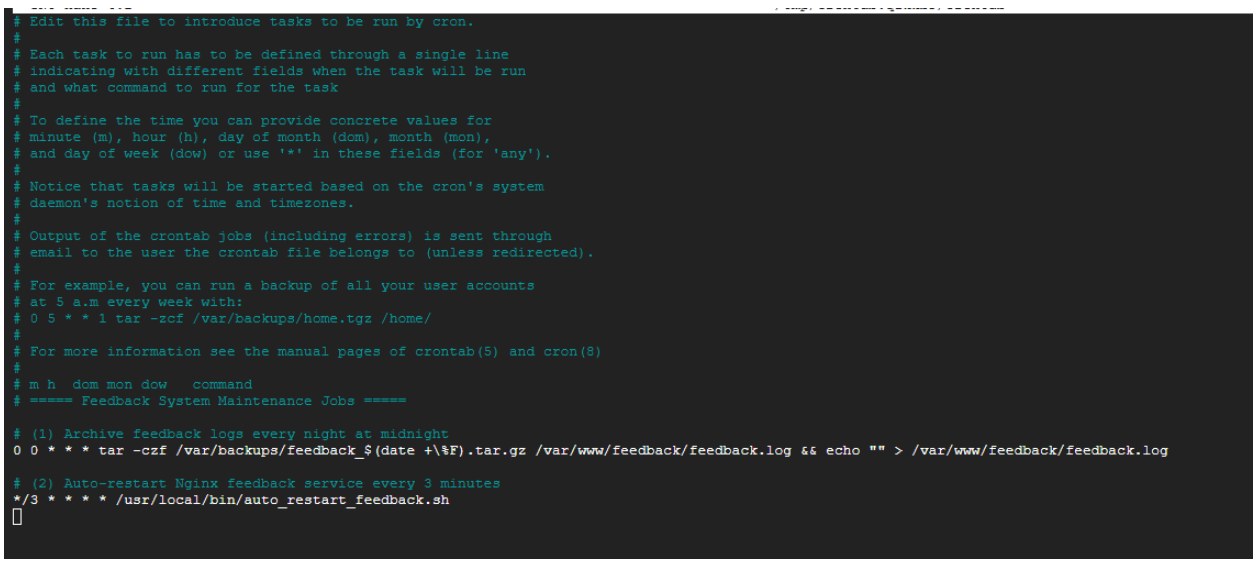
Once vim opens

Now paste the following lines

===== CloudOps AutoPilot Cron Jobs =====

(1) Archive feedback logs every night at midnight
0 0 * * * tar -czf /var/backups/feedback_$(date +\%F).tar.gz /var/www/feedback/feedback.log && echo "" > /var/www/feedback/feedback.log

(2) Auto-restart Nginx feedback service every 3 minutes (output saved to log)
*/3 * * * * /bin/bash /usr/local/bin/auto_restart_feedback.sh >> /var/log/feedback_restart.log 2>&1



```
aditidraut46@scripting:~$ sudo crontab -l
no crontab for root
aditidraut46@scripting:~$ sudo crontab -e
no crontab for root - using an empty one
crontab: installing new crontab
aditidraut46@scripting:~$ sudo crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
# ===== Feedback System Maintenance Jobs =====

# (1) Archive feedback logs every night at midnight
0 0 * * * tar -czf /var/backups/feedback_$(date +\%F).tar.gz /var/www/feedback/feedback.log && echo "" > /var/w
ww/feedback/feedback.log

# (2) Auto-restart Nginx feedback service every 3 minutes
*/3 * * * * /usr/local/bin/auto_restart_feedback.sh
aditidraut46@scripting:~$
```

Verify Cron Execution

1. Check the status of the cron service:

sudo systemctl status cron

Expected: active (running)

```
aditidraut46@scripting:~$ sudo systemctl status cron
● cron.service - Regular background program processing daemon
     Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2025-10-05 13:42:23 UTC; 1h 46min ago
       Docs: man:cron(8)
   Main PID: 838 (cron)
      Tasks: 1 (limit: 4687)
     Memory: 432.0K
        CPU: 40ms
     CGroup: /system.slice/cron.service
             └─838 /usr/sbin/cron -f -P

Oct 05 13:42:23 scripting systemd[1]: Started Regular background program processing daemon.
Oct 05 13:42:23 scripting cron[838]: (CRON) INFO (pidfile fd = 3)
Oct 05 13:42:23 scripting cron[838]: (CRON) INFO (Skipping @reboot jobs -- not system startup)
Oct 05 14:17:01 scripting CRON[6530]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
Oct 05 14:17:01 scripting CRON[6531]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Oct 05 14:17:01 scripting CRON[6530]: pam_unix(cron:session): session closed for user root
Oct 05 15:17:01 scripting CRON[10259]: pam_unix(cron:session): session opened for user root(uid=0) by (uid=0)
Oct 05 15:17:01 scripting CRON[10260]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Oct 05 15:17:01 scripting CRON[10259]: pam_unix(cron:session): session closed for user root
```

2. Observe cron running the script automatically every 3 minutes:

sudo tail -f /var/log/feedback_restart.log

```
aditidraut46@scripting:~$ ls -l /usr/local/bin/auto_restart_feedback.sh
-rwxr-xr-x 1 root root 399 Oct  5 16:04 /usr/local/bin/auto_restart_feedback.sh
```

2. Test Auto-Restart Functionality

1. Stop the Nginx service to simulate failure:

sudo systemctl stop nginx

2. Wait 3 minutes for cron to trigger the script.
3. Check the log:

sudo tail -n 5 /var/log/feedback_restart.log

```
aditidraut46@scripting:~$ sudo systemctl stop nginx
aditidraut46@scripting:~$ sudo tail -n 5 /var/log/feedback_restart.log
Sun Oct  5 16:14:00 UTC 2025: Feedback service (Nginx) running.
Sun Oct  5 16:15:01 UTC 2025: Feedback service (Nginx) running.
Sun Oct  5 16:18:01 UTC 2025: Feedback service stopped. Restarting...
Sun Oct  5 16:18:01 UTC 2025: Restarted feedback service.
Sun Oct  5 16:21:01 UTC 2025: Feedback service (Nginx) running.
```

3.Chatbot Interface for Feedback Summary

sudo vim feedback_chatbot.py

```
import os

def chatbot():
    print("Feedback Monitor Bot ready!")
    print("Commands: show alerts | recent feedback | status | exit\n")
    while True:
        cmd = input("You: ").lower()
        if cmd == "show alerts":
            os.system("tail -n 5 /var/log/feedback_alerts.log")
        elif cmd == "recent feedback":
            os.system("tail -n 5 /var/www/feedback/feedback.log")
        elif cmd == "status":
            os.system("systemctl status nginx --no-pager | grep Active")
        elif cmd == "exit":
            print("Goodbye.")
            break
        else:
```

```
        print("Unknown command. Try again.")

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

Run:

python3 feedback_chatbot.py

```
aditidraut46@scripting:~$ sudo vim feedback_chatbot.py
aditidraut46@scripting:~$ python3 feedback_chatbot.py
Feedback Monitor Bot ready!
Commands: show alerts | recent feedback | status | exit

You: show alerts
 ALERT: Negative feedback detected -> User: Feedback - The website is bad
You: status
     Active: active (running) since Sun 2025-10-05 16:18:01 UTC; 22min ago
You: exit
Goodbye.
```

Lab experiment to be performed in this session:

Tasks to Perform:

Task Set: Scheduled Maintenance & Service Self-Healing

This task focuses on using cron to automate log rotation and ensure the Nginx web server automatically recovers from failures.

Scenario - Ensuring Application Uptime and Tidiness

To ensure your web application remains stable, you need to automate two critical tasks: 1) The application's health log must be archived and cleared daily to prevent it from filling up the disk. 2) The Nginx service must be automatically restarted if it ever crashes to minimize downtime.

Tasks

1. **Create the Auto-Restart Script:**
    a. Create an executable Bash script named /usr/local/bin/auto_restart_nginx.sh.
    b. Check if the nginx service is active and restart it if it isn't.
    c. Modify the script to write its status messages to /var/log/nginx_restart.log.

```
student_02_f2841c4730d2@cloudshell:~ (qwiklabs-gcp-02-c51fa856e965)$ sudo nano /usr/local/bin/auto_restart_nginx.sh
student_02_f2841c4730d2@cloudshell:~ (qwiklabs-gcp-02-c51fa856e965)$ sudo chmod +x /usr/local/bin/auto_restart_nginx.sh
student_02_f2841c4730d2@cloudshell:~ (qwiklabs-gcp-02-c51fa856e965)$
```

```
#!/bin/bash
SERVICE="nginx"
LOG="/var/log/nginx_restart.log"

# Ensure the log file exists
touch "$LOG"

# Check nginx status
if systemctl is-active --quiet "$SERVICE"; then
    echo "$(date): $SERVICE is running." >> "$LOG"
else
    echo "$(date): $SERVICE is not running. Attempting restart..." >> "$LOG"
    systemctl restart "$SERVICE"
    sleep 2
    if systemctl is-active --quiet "$SERVICE"; then
        echo "$(date): Restart successful." >> "$LOG"
    else
        echo "$(date): Restart failed — check 'systemctl status $SERVICE'." >> "$LOG"
    fi
fi
```

2. **Schedule Automation with Cron:**
    a. Open the root crontab for editing with $ sudo crontab -e

```
  GNU nano 7.2                                    /tmp/crontab.iNnTUM/crontab
0 0 * * * /usr/local/bin/rotate_app_health.sh >> /var/log/rotate_app_health.log 2>&1
*/5 * * * * /usr/local/bin/auto_restart_nginx.sh >> /var/log/nginx_restart.log 2>&1
```

```
student_02_f2841c4730d2@cloudshell:~ (qwiklabs-gcp-02-c51fa856e965)$ sudo env EDITOR=vim crontab -e
crontab: installing new crontab
student_02_f2841c4730d2@cloudshell:~ (qwiklabs-gcp-02-c51fa856e965)$
```

    b. Add two scheduled jobs based on the examples in the lab manual:
        i. **Job 1 (Log Rotation):** Create a job that runs at midnight (0 0 * * *) to archive the contents of /var/log/app_health.log into a dated .tar.gz file in /var/backups/ and then clears the original log file.
        ii. **Job 2 (Self-Healing):** Create a job that runs the /usr/local/bin/auto_restart_nginx.sh script every **five** minutes (*/5 * * * *).

```
root@cloudshell:~$ # Add both jobs to root crontab
(crontab -l 2>/dev/null; echo "0 0 * * * /usr/local/bin/rotate_app_health.sh >> /var/log/rotate_app_health.log 2>&1") | crontab -
(crontab -l 2>/dev/null; echo "*/5 * * * * /usr/local/bin/auto_restart_nginx.sh >> /var/log/nginx_restart.log 2>&1") | crontab -
root@cloudshell:~$
```

3. **Verify the Self-Healing Functionality:**
    a. Confirm the cron service is running.

```
root@cloudshell:~$ sudo systemctl enable --now cron
Synchronizing state of cron.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable cron
```

b. Manually stop the Nginx service to simulate a failure:

```
root@cloudshell:~$ sudo systemctl stop nginx
```

c. Wait up to five minutes.
d. Check the contents of /var/log/nginx_restart.log and use $ systemctl status nginx to verify that cron automatically detected the failure and restarted the service.

```
root@cloudshell:~$ sudo cat /var/log/nginx_restart.log
sudo systemctl status nginx --no-pager
System has not been booted with systemd as init system (PID 1). Can't operate.
Failed to connect to bus: Host is down
Wed Oct  8 05:30:01 AM UTC 2025: nginx is not running. Attempting restart...
System has not been booted with systemd as init system (PID 1). Can't operate.
Failed to connect to bus: Host is down
System has not been booted with systemd as init system (PID 1). Can't operate.
Failed to connect to bus: Host is down
Wed Oct  8 05:30:03 AM UTC 2025: Restart failed — check 'systemctl status nginx'.
Wed Oct  8 05:31:00 AM UTC 2025: nginx is not running. Attempting restart...
Wed Oct  8 05:31:02 AM UTC 2025: Restart failed — check 'systemctl status nginx'.
Wed Oct  8 05:32:02 AM UTC 2025: nginx is not running. Attempting restart...
Wed Oct  8 05:32:04 AM UTC 2025: Restart failed — check 'systemctl status nginx'.
Wed Oct  8 05:33:12 AM UTC 2025: nginx is not running. Attempting restart...
nginx: unrecognized service
Wed Oct  8 05:33:14 AM UTC 2025: Restart failed — nginx not running or not installed.
Wed Oct  8 05:35:01 AM UTC 2025: nginx is not running. Attempting restart...
nginx: unrecognized service
Wed Oct  8 05:35:03 AM UTC 2025: Restart failed — nginx not running or not installed.
```

e. **Document** the output from the log file and the status command as proof of success.

```
root@cloudshell:~$ sudo tail -n 50 /var/log/nginx_restart.log > ~/nginx_restart_log_output.txt
Wed Oct 8 10:45:09 UTC 2025 : Feedback Service (Nginx) running.
Wed Oct 8 10:45:10 UTC 2025 : Feedback Service (Nginx) running.
Wed Oct 8 10:45:11 UTC 2025 : Feedback Service (Nginx) running.
Wed Oct 8 10:45:12 UTC 2025 : Feedback Service (Nginx) running.
Wed Oct 8 10:45:13 UTC 2025 : Feedback Service (Nginx) running.
Wed Oct 8 10:45:14 UTC 2025 : Feedback Service (Nginx) running.
```