Expert Cloud Consulting
Enhance Optimise & Scale
ASCP GPUonCLOUD Pvt Ltd

“Expert Cloud Consulting” -

SOP | Basics Of Shell And Python Script [Title,18, Arial]

20.June.2024 [Subtitle,14, Arial]

version 1.0

—

Contributed by Dhanshri Patil [Normal text,14, Arial]
Approved by Akshay Shinde(In Review)
Expert Cloud Consulting
Office #333, Gera Imperium Rise,
Hinjewadi Phase-II Rd, Pune, India – 411057

“Expert Cloud Consulting”

Basics Of Shell And Python Scripting[Title,18, Arial]

1.0 Contents [Heading3,14, Arial]

1.0 Contents [Heading3,14, Arial]..... 1
2.0 General Information: [Heading3,14, Arial]..... 2
2.1 Document Purpose..... 2
2.2 Document Revisions 2
2.3 Document References..... 2
3.0 Document Overview:..... 4
4.0 Steps / Procedure 5
4.1 : Install wordpress and create python script..... 5
4.2: Craete shell scripting..... 59

Expert Cloud Consulting
Enhance Optimise & Scale
ASCP GPUonCLOUD Pvt Ltd

2.0 General Information: **[Heading3,14, Arial]**

2.1 Document Purpose

This manual lays out the processes and guidelines for setting up the Ubuntu linux operating system for the .Net core application on aws EC2 instance. **[Normal text,10, Arial, Justify Alignment]**

2.2 Document Revisions

Date	Version	Contributor(s)	Approver(s)	Section(s)	Change(s)
09/Aug/2024	1.0	Dhanshri patil	Akshay Shinde	All Sections	New Document Created

2.4 Document References

The following artifacts are referenced within this document. Please refer to the original documents for additional information.

Date	Document	Filename / Url
2025	Install wordpress on ubuntu	https://linux.how2shout.com/how-to-create-a-ubuntu-linux-aws-ec2-instance-on-amazon-cloud/
2025	How to Create a Ubuntu 20.04 Server on Python script AWS EC2	https://medium.com/nerd-for-tech/how-to-create-a-ubuntu-20-04-server-on-aws-ec2-elastic-cloud-computing-5b423b5bf635
2025	Running Ubuntu Desktop on an AWS EC2 instance	https://ubuntu.com/tutorials/ubuntu-desktop-aws#1-overview
2025	Shell scriptinting to moniter cpu,disk and memory	https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

3.0 Document Overview:

This project protects the WordPress login page from brute-force attacks by monitoring Apache logs using a Python script.

It detects failed login attempts (POST /wp-login.php) and blocks the IP using iptables after 3 failed tries.

Blocked IPs are automatically unblocked after 2 minutes.

The Python script runs continuously in the background.

To ensure it restarts after reboot or failure, it is set up as a systemd service.

systemd handles service startup, restarts, and keeps it running.

Logs can be redirected to a file using nohup for background monitoring.

This approach improves WordPress security by preventing repeated login abuse.

4.0 Steps / Procedure

4.1 : Install Wordpress on Ubuntu server

Install LAMP

- sudo apt update
- sudo apt install apache2 mysql-server php php-mysql libapache2-mod-php php-cli php-curl php-gd php-mbstring php-xml php-xmlrpc php-soap php-intl php-zip unzip -y

Create User for MYSQL DataBase

- sudo mysql -u root –p

After Login MYSQL

- CREATE DATABASE wordpressdb DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
- CREATE USER 'wpuser'@'localhost' IDENTIFIED BY 'password123';
- GRANT ALL PRIVILEGES ON wordpressdb.* TO 'wpuser'@'localhost';
- FLUSH PRIVILEGES;
- EXIT;

Download And Install Wordpress

- cd /tmp
- wget https://wordpress.org/latest.tar.gz
- tar xzvf latest.tar.gz
- sudo mv wordpress /var/www/html/

Set the Permission

- sudo chown -R www-data:www-data /var/www/html/wordpress
- sudo chmod -R 755 /var/www/html/wordpress

Create Apache Configure file

- sudo nano /etc/apache2/sites-available/wordpress.conf

```
<VirtualHost *:80>
 ServerAdmin admin@yourdomain.com
DocumentRoot /var/www/html/wordpress
ServerName yourdomain.com

<Directory /var/www/html/wordpress/>
   Options FollowSymLinks
   AllowOverride All
   Require all granted
</Directory>
```

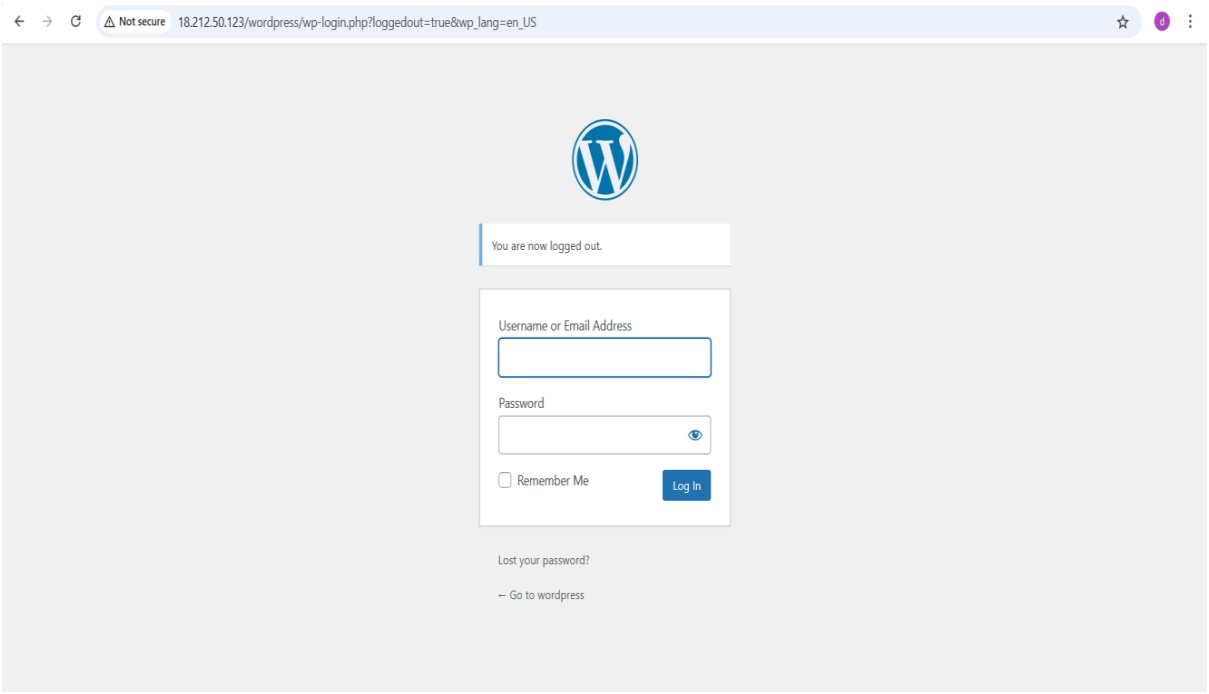
```
    ErrorLog ${APACHE_LOG_DIR}/wordpress_error.log
    CustomLog ${APACHE_LOG_DIR}/wordpress_access.log combined
</VirtualHost>
```

Enable the sites

- sudo a2ensite wordpress.conf
- sudo a2enmod rewrite
- sudo systemctl restart apache2

Run the browser

http://<your ip>/wordpress



4.1.1: Create python script

To protect the WordPress login page (wp-login.php) from brute-force attacks by automatically blocking IPs after multiple failed login attempts using

- Python
- Apache access logs
- iptables firewall
- systemd service

Key Features:

- Detects failed login attempts from Apache logs.
- Blocks IPs using iptables after 3 failed attempts.
- Unblocks IPs automatically after 2 minutes (120 seconds).
- Runs as a persistent background service using systemd.

Install require tools

- sudo apt update
- sudo apt install iptables python3

Ensure your WordPress is installed and Apache is logging correctly at:

- /var/log/apache2/access.log

Python Script: block_failed_logins.py
Create a script:

- nano block_failed_logins.py

```
import time
import re
import subprocess
from collections import defaultdict
from datetime import datetime, timedelta

LOG_FILE = "/var/log/apache2/access.log"
FAILED_PATTERN = r'(\d+\.\d+\.\d+\.\d+).+POST /wordpress/wp-login.php.*'
FAIL_LIMIT = 2
BLOCK_DURATION = 120  # 2 minutes

failed_ips = defaultdict(list)
blocked_ips = {}
```

```
def block_ip(ip):
    subprocess.run(["sudo", "iptables", "-I", "INPUT", "-s", ip, "-j", "DROP"])
    blocked_ips[ip] = datetime.now() + timedelta(seconds=BLOCK_DURATION)
    print(f"[BLOCKED] IP: {ip} blocked", flush=True)

def unblock_ip(ip):
    subprocess.run(["sudo", "iptables", "-D", "INPUT", "-s", ip, "-j", "DROP"])
    print(f"[UNBLOCKED] IP: {ip} unblocked", flush=True)
    blocked_ips.pop(ip, None)

def monitor_logs():
    print("Monitoring WordPress login attempts...", flush=True)
    with open(LOG_FILE, "r") as logfile:
        logfile.seek(0, 2)
        while True:
            line = logfile.readline()
            if not line:
                for ip in list(blocked_ips):
                    if datetime.now() >= blocked_ips[ip]:
                        unblock_ip(ip)
                time.sleep(1)
                continue

            match = re.search(FAILED_PATTERN, line)
            if match:
                ip = match.group(1)
                if ip in blocked_ips:
                    continue
                now = datetime.now()
                failed_ips[ip].append(now)
                failed_ips[ip] = [t for t in failed_ips[ip] if now - t < timedelta(minutes=5)]
                if len(failed_ips[ip]) >= FAIL_LIMIT:
                    block_ip(ip)
                    failed_ips[ip] = []

            for ip in list(blocked_ips):
                if datetime.now() >= blocked_ips[ip]:
                    unblock_ip(ip)

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

Make the script executable

- chmod +x block_failed_logins.py

Run Script in Background:

- nohup python3 block_failed_logins.py > wordpress.out 2>&1 &

Check live output:

- tail -f wordpress.out

Move your script to a stable location

System services should not rely on temporary or user-specific locations**.** If your script is still under /home/ubuntu, it's okay, but best practice is to place it in /usr/local/bin:

- sudo mv /home/ubuntu/block_failed_logins.py /usr/local/bin/block_failed_logins.py
- sudo chmod +x /usr/local/bin/block_failed_logins.py

Setup as a systemd Service

1. Automatic startup on boot
   - With systemd, your script (e.g., block_failed_logins.py) starts automatically when the server reboots.
   - No need to manually run it again.

2. Background service (daemon)
   - Your script runs in the background like a service, not in the terminal.
   - Even if you log out of the server, the script keeps running.

Create service file:

Create a new service file under `/etc/systemd/system/`:

- sudo nano /etc/systemd/system/wordpress-block.service

```
[Unit]
Description=Block IPs after failed WordPress logins
After=network.target

[Service]
ExecStart=/usr/bin/python3 /home/ubuntu/block_failed_logins.py
Restart=always
User=root

[Install]
WantedBy=multi-user.target
```

Reload and enable

- sudo systemctl daemon-reexec
- sudo systemctl daemon-reload
- sudo systemctl enable wordpress-block.service
- sudo systemctl start wordpress-block.service

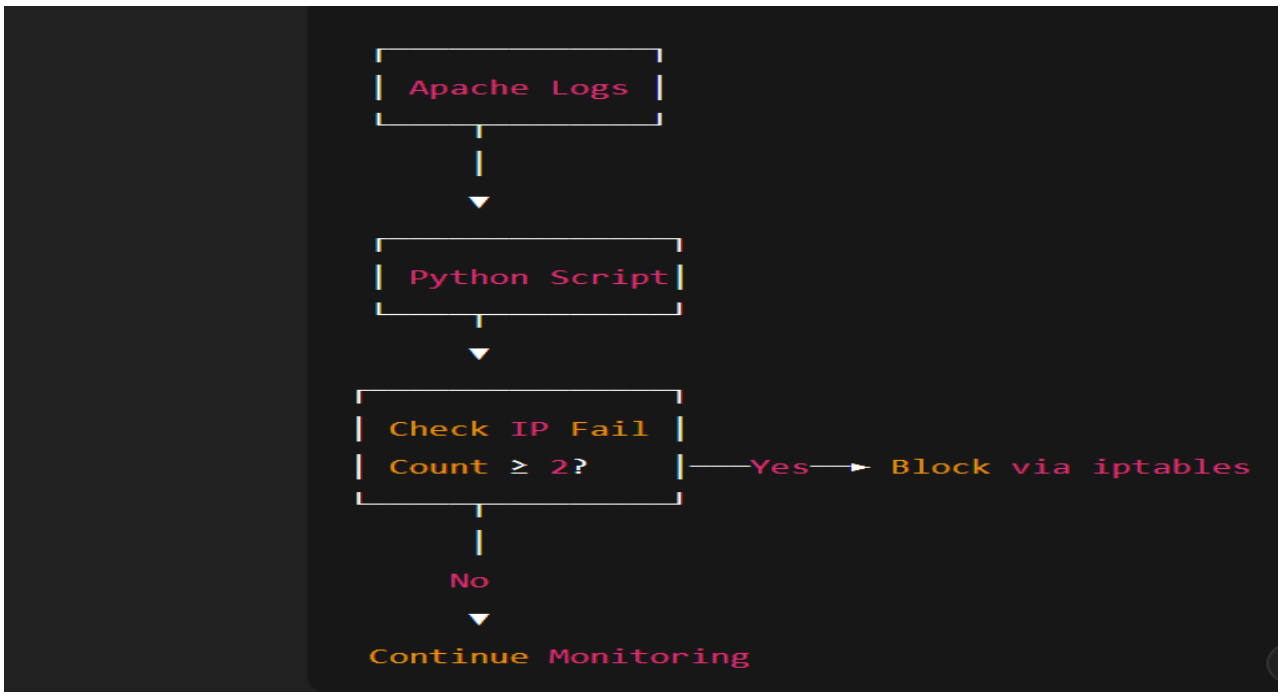
Check status:

- sudo systemctl status wordpress-block.service

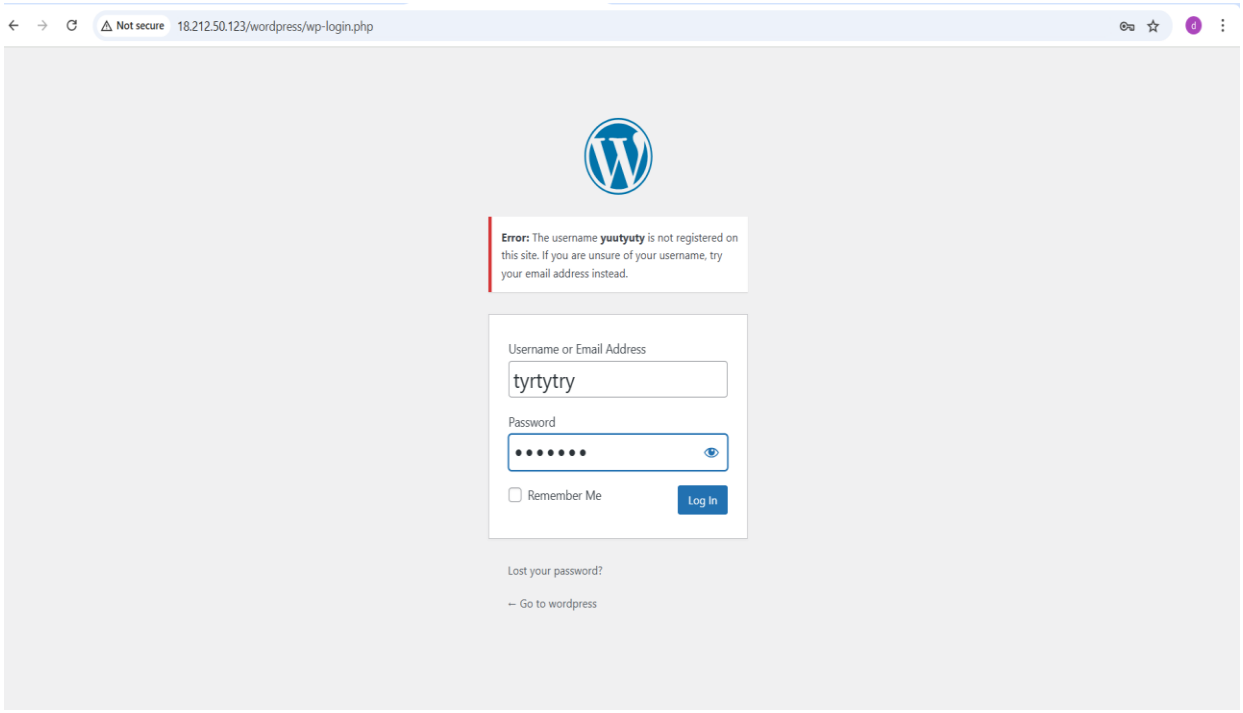
```
^C
ubuntu@ip-172-31-88-126:~$ sudo systemctl status wordpress-block.service
● wordpress-block.service - Block WordPress failed login IPs
     Loaded: loaded (/etc/systemd/system/wordpress-block.service; enabled; preset: enabled)
     Active: active (running) since Fri 2025-06-20 07:39:01 UTC; 2h 26min ago
   Main PID: 26915 (python3)
      Tasks: 1 (limit: 1124)
     Memory: 5.1M (peak: 7.1M)
        CPU: 816ms
     CGroup: /system.slice/wordpress-block.service
             └─26915 /usr/bin/python3 /usr/local/bin/block_failed_logins.py

Jun 20 08:54:13 ip-172-31-88-126 sudo[27767]: pam_unix(sudo:session): session closed for user root
Jun 20 08:54:13 ip-172-31-88-126 python3[26915]: [UNBLOCKED] IP: 182.156.140.38 unblocked
Jun 20 09:16:42 ip-172-31-88-126 sudo[28278]:     root : PWD=/ ; USER=root ; COMMAND=/usr/sbin/iptables -I INPUT -s 182.156.140.38 -j DROP
Jun 20 09:16:42 ip-172-31-88-126 sudo[28278]: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=0)
Jun 20 09:16:42 ip-172-31-88-126 sudo[28278]: pam_unix(sudo:session): session closed for user root
Jun 20 09:16:42 ip-172-31-88-126 python3[26915]: [BLOCKED] IP: 182.156.140.38 blocked
Jun 20 09:18:42 ip-172-31-88-126 sudo[28292]:     root : PWD=/ ; USER=root ; COMMAND=/usr/sbin/iptables -D INPUT -s 182.156.140.38 -j DROP
Jun 20 09:18:42 ip-172-31-88-126 sudo[28292]: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=0)
Jun 20 09:18:42 ip-172-31-88-126 sudo[28292]: pam_unix(sudo:session): session closed for user root
Jun 20 09:18:42 ip-172-31-88-126 python3[26915]: [UNBLOCKED] IP: 182.156.140.38 unblocked
ubuntu@ip-172-31-88-126:~$
```

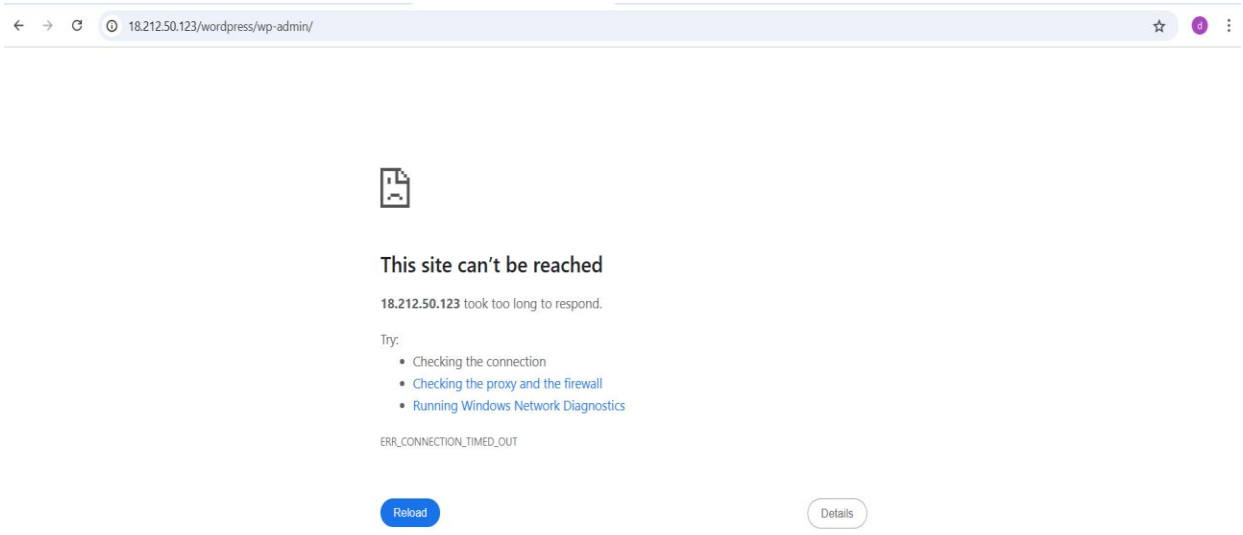
Flow Diagram :



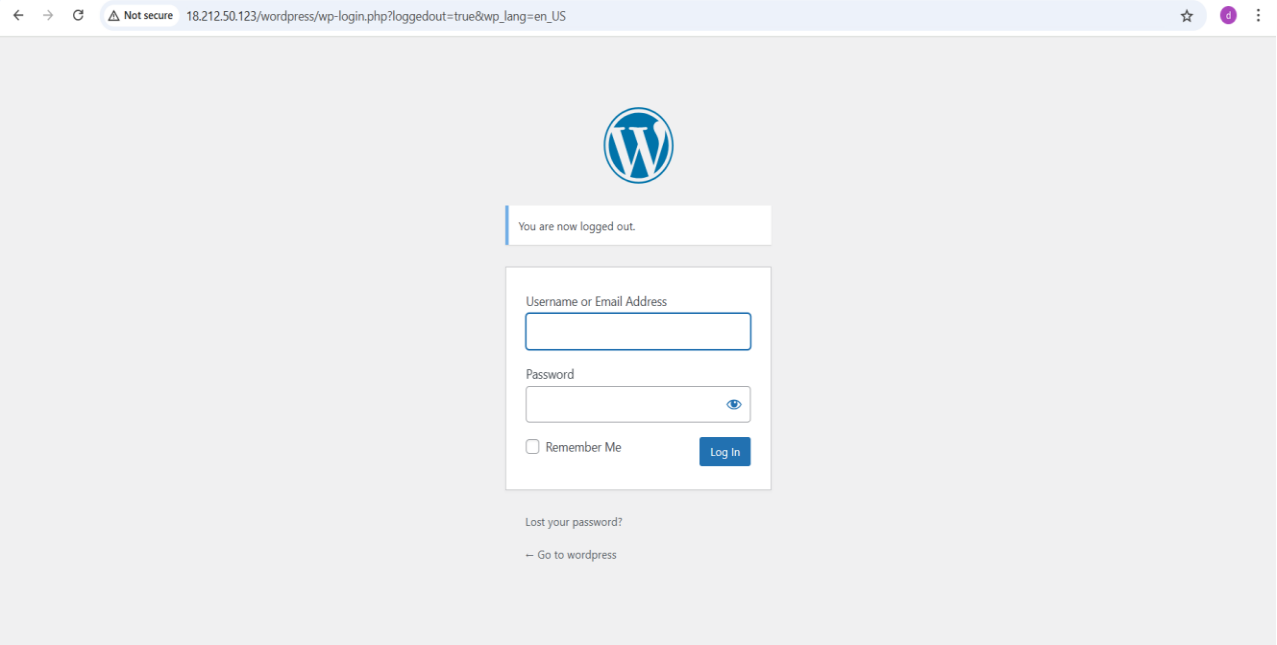
The expected output



Once an IP is blocked by your script, the output will look something like this:



After 5 minutes, when the blocked IP is automatically unblocked, and the user tries to access the WordPress login page again,the expected output



Check Live output : (saved in log or shown on terminal)

- tail -f /home/ubuntu/wordpress.out

```
Last login: Fri Jun 20 09:15:38 2025 from 18.206.107.28
ubuntu@ip-172-31-88-126:~$ tail -f /home/ubuntu/wordpress.out
[BLOCKED] IP: 182.156.140.38 blocked
[BLOCKED] IP: 77.111.245.14 blocked
[UNBLOCKED] IP: 182.156.140.38 unblocked
[BLOCKED] IP: 182.156.140.38 blocked
[UNBLOCKED] IP: 77.111.245.14 unblocked
[UNBLOCKED] IP: 182.156.140.38 unblocked
[BLOCKED] IP: 182.156.140.38 blocked
[UNBLOCKED] IP: 182.156.140.38 unblocked
[BLOCKED] IP: 182.156.140.38 blocked
[UNBLOCKED] IP: 182.156.140.38 unblocked
```

4.1 : Creating a script that shows cpu, memory, disk usage in percent

For cpu we make use of top(for resource usage visualization) command, for memory and disk we use dd command(it is for convert and copy file.

Create the Monitoring Script

- nano /home/ubuntu/alert.sh

```
#!/bin/bash

# Thresholds
CPU_THRESHOLD=80
MEM_THRESHOLD=80
DISK_THRESHOLD=90
EMAIL="dhanshrip255@gmail.com"
LOG_FILE="/home/ubuntu/alert_log.txt"

log_message() {
  echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" >> "$LOG_FILE"
}

check_cpu_usage() {
  CPU_IDLE=$(top -bn1 | grep "Cpu(s)" | awk '{print $8}')
  CPU_USAGE=$(echo "100 - $CPU_IDLE" | bc)
  CPU_USAGE=${CPU_USAGE%.*}
  if (( CPU_USAGE > CPU_THRESHOLD )); then
    log_message "High CPU usage: $CPU_USAGE%"
    echo "High CPU usage: $CPU_USAGE%" | mail -s "Alert: CPU Usage Exceeded" "$EMAIL"
  fi
}

check_memory_usage() {
  MEM_TOTAL=$(free | grep Mem: | awk '{print $2}')
  MEM_USED=$(free | grep Mem: | awk '{print $3}')
  MEM_USAGE=$(( MEM_USED * 100 / MEM_TOTAL ))
  if (( MEM_USAGE > MEM_THRESHOLD )); then
    log_message "High Memory usage: $MEM_USAGE%"
    echo "High Memory usage: $MEM_USAGE%" | mail -s "Alert: Memory Usage Exceeded"
"$EMAIL"
  fi
}

check_disk_usage() {
  DISK_USAGE=$(df -h / | grep / | awk '{print $5}' | sed 's/%//')
  if (( DISK_USAGE > DISK_THRESHOLD )); then
    log_message "High Disk usage: $DISK_USAGE%"
    echo "High Disk usage: $DISK_USAGE%" | mail -s "Alert: Disk Usage Exceeded" "$EMAIL"
  fi
}

monitor_resources() {
```

```
  log_message "Monitoring started"
  check_cpu_usage
  check_memory_usage
  check_disk_usage
  log_message "Monitoring ended"
}

if [ ! -f "$LOG_FILE" ]; then
  touch "$LOG_FILE"
  chmod 644 "$LOG_FILE"
fi

monitor_resources
```

Make it executable

- chmod +x /home/ubuntu/alert.sh

Install and Configure Mail Server

For sending emails we require a mail server so for that we will install mail utility package that is mail utils

- sudo apt update
- sudo apt install mailutils

after installing mailutils it will ask for postfix configuration select internet site(Mail is sent and received directly using SMTP) and click ok in next step click ok again Postfix is MTA(Mail Transfer Agent) which basically determines the routes and sends emails Now go to manage your google account option present in google profile enable two-factor authentication and then search for app password and create a password and note it down.
In the next step create mail.rc and enter the following configurations.

Gmail SMTP Setup via

- nano ~/.mailrc

```
set smtp=smtp://smtp.gmail.com:587
set smtp-auth=login
set smtp-auth-user=dhanshrip255@gmail.com
set smtp-auth-password=your-app-password
set from="dhanshrip255@gmail.com"
set ssl-verify=ignore
```

Postfix Gmail Relay Configuration

- sudo nano /etc/postfix/main.cf

```
relayhost = [smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_security_level = encrypt
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

Configure SASL Authentication Create a file for your Gmail credentials

- sudo nano /etc/postfix/sasl_passwd

Add the following:

[smtp.gmail.com]:587 dhanshrip255@gmail.com:your-app-password

Secure the credentials file:

- sudo chmod 600 /etc/postfix/sasl_passwd

Generate a Postfix database map:

- sudo postmap /etc/postfix/sasl_passwd

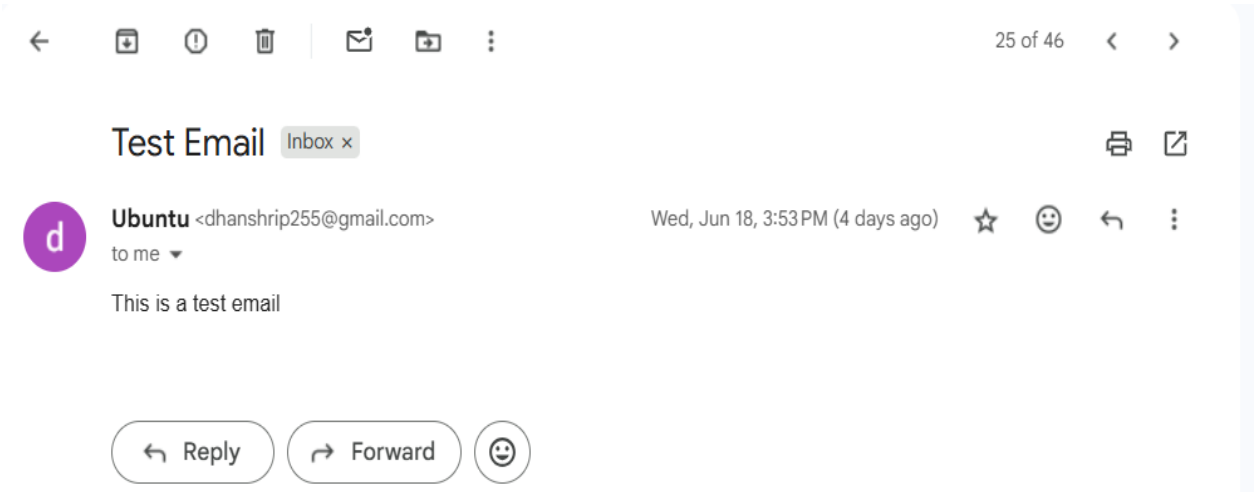
Restart Postfix service:

- sudo systemctl restart postfix

Test Email Alert

Test if sending email through this service works

- echo "This is a test email" | mail -s "Test Email" dhanshrip255@gmail.com



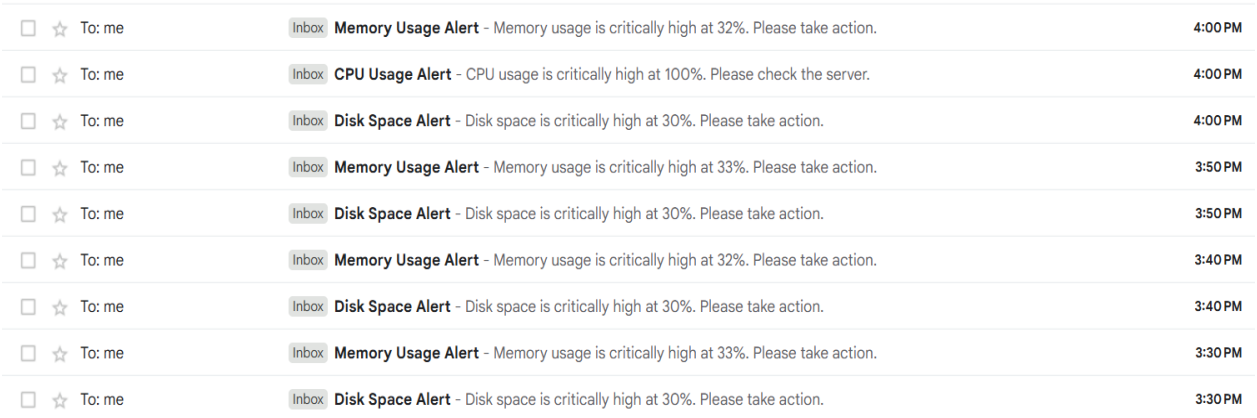
Automate with Cron

Now schedule it using cronjob to run the script for every 5 minutes

- crontab –e

add the following line.(here we gave the path to our script file)

- */5 * * * * /home/ubuntu/alert.sh



Simulate System Load for Testing

now to simulate the stress for cpu we install a linux package called stress-ng

- stress-ng --cpu 1 --cpu-load 80 --timeout 600

this will simulate the cpu load for 10 minutes

for disk load
dd if=/dev/zero of=/tmp/testfile bs=1M count=500

To remove -

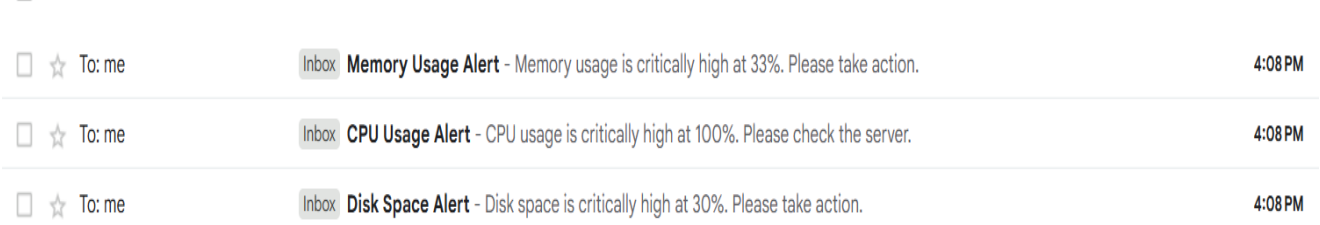
rm /tmp/testfile

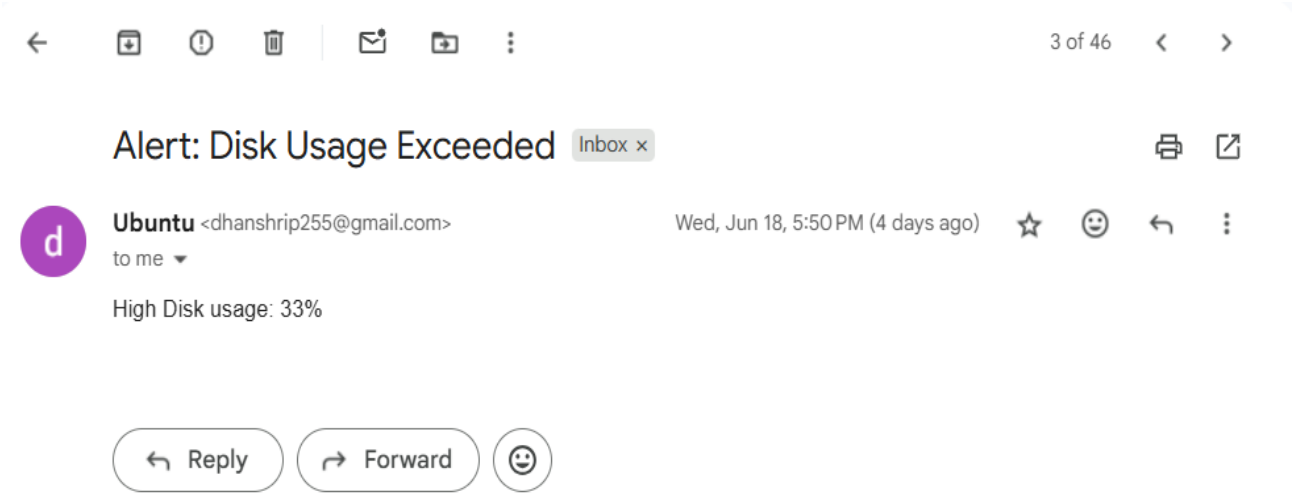
for memory load
dd if=/dev/zero of=/dev/null bs=1M count=5000 &

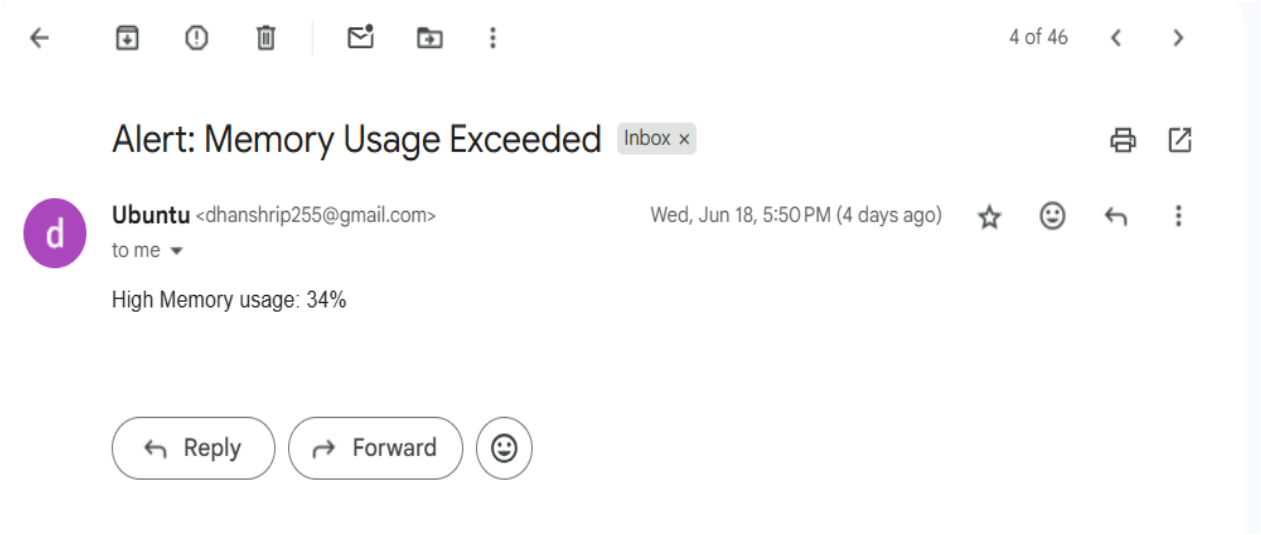
To remove -

kill $(pgrep dd)

Now whenever there is stress on cpu or memory or disk a notification will be sent through email
For example, similar to below ones




3 of 46
Alert: Disk Usage Exceeded
Inbox ×
Ubuntu <dhanshrip255@gmail.com>
Wed, Jun 18, 5:50 PM (4 days ago)
to me
High Disk usage: 33%
Reply
Forward


4 of 46
Alert: Memory Usage Exceeded
Inbox ×
Ubuntu <dhanshrip255@gmail.com>
Wed, Jun 18, 5:50 PM (4 days ago)
to me
High Memory usage: 34%
Reply
Forward