

Automated Monitoring and Maintenance for an E-commerce Website

Problem Statement:

An e-commerce company runs its website on a Linux-based web server. The company faces challenges in maintaining the server's health and ensuring the website is always up and running.

Manual monitoring and maintenance tasks are time-consuming and prone to human error. The company needs an automated solution to monitor system performance, perform regular updates, and back up critical data to ensure continuous availability and reliability of the website

Step by step Implementation:

1) Virtualized Environment Setup:

1.creating the new virtual machine : We can use the oracle

VM VirtualBox

VM Download link:

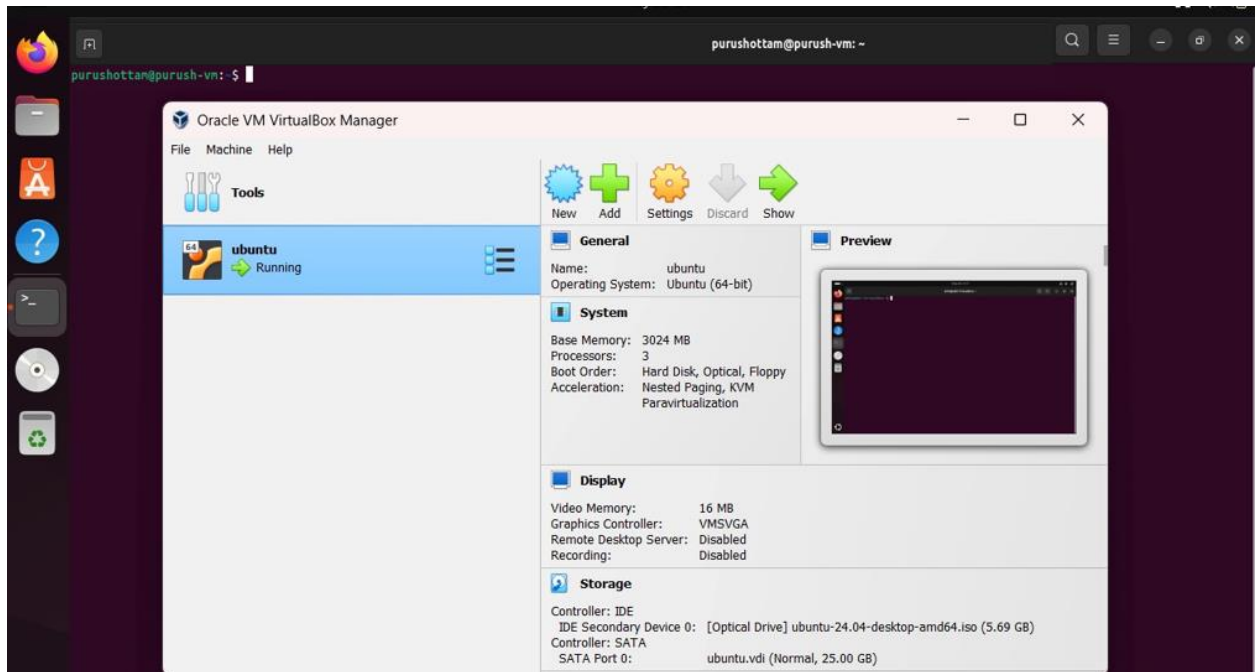
<https://www.virtualbox.org/wiki/Downloads>

2.ISO image download link:

<https://ubuntu.com/download/desktop/thank-you?version=24.04&architecture=amd64<s=true>

3. VM Configuration as:

- Type: Linux
- Version: Ubuntu 64-bit
- Memory: 3024 MB ● Virtual Hard Disk: 30 GB



3. Setup the Ubuntu Server:

- Update and upgrade the package manager

```
Sudo apt update && sudo apt upgrade -y
```

- Install apache server:

```
sudo apt install apache2 -y
```

- Start apache server:

```
sudo systemctl start apache2
```

- Enabled apache server:

```
sudo systemctl enable apache2
```

- Check the status of the apache server:

```
Sudo systemctl status apache2.service
```

```
purushottam@purush-vm:~$ sudo systemctl status apache2.service
[sudo] password for purushottam:
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: >
   Active: active (running) since Thu 2024-05-30 21:57:42 IST; 10min ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 1300 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SU>
  Main PID: 1337 (apache2)
    Tasks: 55 (limit: 9018)
   Memory: 8.2M (peak: 8.6M)
      CPU: 122ms
   CGroup: /system.slice/apache2.service
           └─1337 /usr/sbin/apache2 -k start
             1338 /usr/sbin/apache2 -k start
             1339 /usr/sbin/apache2 -k start

May 30 21:57:42 abhi-HP-Laptop-15s-fq2xxx systemd[1]: Starting apache2.service >
May 30 21:57:42 abhi-HP-Laptop-15s-fq2xxx apachectl[1332]: AH00558: apache2: Co>
May 30 21:57:42 abhi-HP-Laptop-15s-fq2xxx systemd[1]: Started apache2.service ->
lines 1-17/17 (END)
```

2) Monitoring System Performance:

1. The **top** command in Linux is a powerful tool for monitoring system activity in real-time

- **Process Information:** **top** lists all running processes along with their process ID (PID), user, CPU usage, memory usage, and more.
- **CPU Usage:** **top** displays a summary of CPU usage at the top, showing the total CPU usage, usage breakdown by user processes, system processes, and idle CPU time.
- **Memory Usage:** Along with CPU usage, **top** also provides information about memory usage, including total memory, used memory, free memory, and memory usage by individual processes

2. Check disk usage: It use to check the memory uses of the system

```
df -h
```

3. Monitor network activity

ip a

-----> Get the network info

```
purushottam@purush-vm:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:37:c1:87 brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.9.10/0 brd 255.255.255.255 scope global noprefixroute ens33
        valid_lft forever preferred_lft forever
    inet6 2401:4900:54db:f17b:e3e2:b8b1:90a:375c/64 scope global temporary dynamic
        valid_lft 6951sec preferred_lft 6951sec
    inet6 2401:4900:54db:f17b:54fb:3e5d:ba7f:7857/64 scope global dynamic mngtmpaddr noprefixroute
        valid_lft 6951sec preferred_lft 6951sec
    inet6 fe80::d3aa:78cb:202f:6286/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
purushottam@purush-vm:~$
```

netstat -tuln-----> Displays information about active network connections and listening ports.

3) Automate System Updates:

1. Create an Update Script:

```
#!/bin/bash

#this use to update the package manager
sudo apt update && sudo apt upgrade -y

echo "System update completed."
```

In this script we use to update the package manager

```
Sudo apt update && sudo apt upgrade -y
```

2. Make the script executable: Using this command we can give the execute permission to the all user to execute the script

```
chmod +x update_system.sh
```

3. Schedule the Script: Create the cron job to run this script on every 2AM

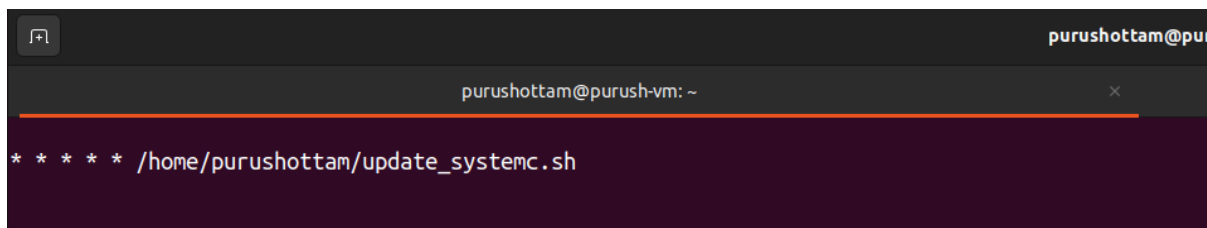
```
Crontab -e
```

```
0 2 * * * /home/purushottam/update_system.sh
```

0: This field specifies the minute when the command will run. In this case, 0 means the command will run at the start of the hour (2:00 AM).

2: This field specifies the hour when the command will run. In this case, 2 means the command will run at 2:00 AM.

*: The asterisks in the next three fields represent wildcard characters, meaning "every" or "any". So * * * specifies "every day of the month", "every month", and "every day of the week".

A screenshot of a terminal window with a dark background. The window title is "purushottam@purush-vm: ~". The terminal shows the command "crontab -e" being executed, and the output is a crontab configuration line: "* * * * * /home/purushottam/update_systemc.sh". The cursor is at the end of the line.

```
purushottam@purush-vm: ~  
crontab -e  
* * * * * /home/purushottam/update_systemc.sh
```

4) Automate Data Backup:

1. Create a Backup Script:

```
#!/bin/bash

tar -czvf /var/backups/web_backup_$(date +%F).tar.gz /var/www/html

echo "Backup completed"
```

In this script we can backup the `/var/www/html` directory content , it contains our website file.

```
tar -czvf /var/backups/web_backup_$(date +%F).tar.gz /var/www/html
```

tar: This is the command-line utility used for archiving files. It can create, view, and manipulate tar archives.

-czvf: These are options passed to **tar** with specific meanings:

- **-c:** Create a new archive.
- **-z:** Compress the archive using gzip.
- **-v:** Verbose mode, which displays progress and filenames as the archive is created.
- **-f:** Specifies the filename of the archive.

/var/backups/web_backup_\$(date +%F).tar.gz: This is the path and filename of the resulting archive. It's constructed dynamically using the **date** command:

- **\$(date +%F):** This part of the command substitutes the output of the **date** command, which generates the current date in the format **YYYY-MM-DD (%F)**.

2. Make the script executable: using **chmod** we can provide the execution permission.

```
chmod +x backup.sh
```

3. Schedule the Script: schedule the backup script weekly on Sundays at 3 AM

```
crontab -e
```

```
0 3 * * 0 /home/purushottam/backup.sh
```

0: This field specifies the minute when the command will run.

Here, 0 means the command will run at the beginning of the hour. 3: This field specifies the hour when the command will run. In this case, 3 means the command will run at 3:00 AM.

*: The asterisks in the next three fields represent wildcard characters, meaning "every" or "any". So * * * specifies "every day of the month", "every month", and "every day of the week". 0: This field specifies the day of the week when the command will run. In cron notation, Sunday is represented by 0

```
0 2 * * * /home/purushottam/update_systemc.sh
0 3 * * 0 /home/purushottam/backup.sh
```

Website that host on the apache server:

