2

Perform the following two tasks:

1. Create a **bash script** that recursively copies the `/var/www/` directory into the `/opt/www-backup/` directory

   Save your script at `/opt/script.sh`. Remember, the script file you create also has to be **executable**.
2. Make sure that your script `/opt/script.sh` automatically runs every day at **4AM**. More specifically, create a cron job that runs that script every day at 4AM. Put this in the **system-wide cron table** (not root's local cron table) and make sure the script executes under the root user.

Solution

Create a script `/opt/script.sh` to backup of `/var/www/`

```
#!/bin/bash

cp -a /var/www/. /opt/www-backup/
```

Provide **executable** permission to the `/opt/script.sh`

```
sudo chmod +x /opt/script.sh
```

Run below command to add cron job in system-wide cron table

```
sudo vi /etc/crontab

0 4 * * *      root     /opt/script.sh
```

3

Task

Enforce some **limits** on two users:

1. Set a limit on the user called **john** so that he can open no more than **30 processes**. This should be a **hard limit**.
2. For the user called **jane** make sure she can create **files not larger than 1024 kilobytes**. Make this a **soft limit**.

Solution

To enforce limits on user **jane** and **john** edit the `/etc/security/limits.conf` file as below:

```
john hard nproc 30
jane soft fsize 1024
```

4

Task

Create a new user on this system called `mary`

- Set her password to `1234`.
- Leave the full name and other personal details empty.
- Set her default shell to `/bin/dash`.
- Make sure she can **execute sudo commands** by adding her to the secondary group called `sudo`.
- At this point Mary's primary group is **mary**. And her secondary group is **sudo**. Change her primary group to `developers`. Without affecting her secondary group.

Solution

1. Create the user with a home directory and set the default shell to **/bin/zsh**

```
sudo useradd -m -s /bin/dash mary
```

2. Set the Password

```
echo "mary:1234" | sudo chpasswd
```

3. Modify the User’s Groups

```
sudo usermod -aG sudo mary
```

4. Change Primary Group to **developers**

```
sudo usermod -g developers mary
```

5

Task

Modify the following **kernel runtime parameter**:

1. `vm.swappiness` set it to a value of `10`. This should be a persistent change, added to a file so that `vm.swappiness` is set to `10` every time the **system boots up**. However, after you create the proper file, **also set this runtime parameter to 10 for this session as well**. Otherwise said, the file will set the parameter to 10 the next time the system boots up, but we want to set it to 10 even for this current, active session, instead of waiting until the next boot until that takes effect.

Solution

Setting `vm.swappiness` for the Current Session

```
sudo sysctl vm.swappiness=10
```

Making the Change Persistent

```
echo "vm.swappiness=10" | sudo tee /etc/sysctl.d/99-swappiness.conf
```

6

Task

You have an **xfs filesystem** on **/dev/vdb1**. Also, there's an **ext4 filesystem** on **/dev/vdb2**. Finally, there's an empty partition **/dev/vdb3** that you'll need to format.

1. Edit the correct file in /etc/ so that **/dev/vdb1** is automatically mounted into the **/backups** directory every time the system boots up. **Default** mount options should be used.

2. **/dev/vdb2** is already mounted in **/mnt/**. But there is a problem. Sensitive data exists on this **ext4** filesystem and you want to make sure that it's not accidentally modified. To solve this problem, **remount /dev/vdb2 into the /mnt directory**, but this time, with the **read-only mount** option. It does not matter what the other mount options are. Just make sure this mount point is read-only so that users cannot change contents on this filesystem.

3. **Format /dev/vdb3** with the **xfs filesystem**. To make this easier to spot in the future among the other filesystems, set the filesystem label to **ExamFS** when you format it. Make sure that the label is exactly **ExamFS** and not **examfs** or anything that has different letters in UPPERCASE or lowercase

Solution

1. To have the mount permanent, edit the `/etc/fstab` file

```
/dev/vdb1    /backups    xfs    defaults    0    2
```

2. To Remount the `/dev/vdb2` Filesystem as Read-Only

```
sudo mount -o remount,ro /dev/vdb2 /mnt
```

3. Format the `/dev/vdb3` partition to xfs filesystem with `ExamFS` label

```
sudo mkfs.xfs -f -L ExamFS /dev/vdb3
```

7

Task

Use the Logical Volume Manager to perform the following tasks:

1. Add `/dev/vdc` and `/dev/vdd` as **Physical Volumes** to LVM.
2. Create a **Volume Group** on these two physical volumes. Call the volume `volume1`.
3. On the Volume Group called `volume1` create a new **Logical Volume**. Call this Logical Volume `website_files`. Set the size of the Logical Volume to `3GB`.

Solution

1. To initialize both **/dev/vdc** and **/dev/vdd** as LVM **physical volumes** (PVs):

```
sudo pvcreate /dev/vdc /dev/vdd
```

2. Create a volume group (VG) named `volume1` using the two physical volumes:

```
sudo vgcreate volume1 /dev/vdc /dev/vdd
```

3. Create a logical volume (LV) named `website_files` in the `volume1` volume group:

```
sudo lvcreate -n website_files -L 3G volume1
```

8

Task

In your home directory you will find a subdirectory called **kode**. Git tools are pre-installed. Switch to the **kode** subdirectory and perform the following tasks:

1. **Initialize** this subdirectory as an **empty Git repository**.
2. Associate this local Git repository with the remote repository found at `https://github.com/kodekloudhub/git-for-beginners-course.git`. Add this as a remote repository and call it (alias it as) `origin`.
3. **Download** all the latest changes from the `master branch` from that remote repository into your local repository.

Solution

1. Initialize the Subdirectory as a Git Repository

```
cd ~/kode

git init
```

2. Add a Remote Repository

```
git remote add origin https://github.com/kodekloudhub/git-for-beginners-course.git
```

3. Pull the Latest Changes from the Master Branch

```
git pull origin master
```

9

Task

A Docker container is running on `node01`. Perform the following tasks:

1. Stop and **remove the container** that is currently running, since it's not configured correctly.
2. In your home directory you will find a subdirectory called **kode_web**. It contains all the necessary build instructions for Docker. Use that directory to build a new Docker image. Call this image `kodekloudwebserv`.

3. Finally, **launch a container** based on the `kodekloudwebserv` image. In your command, make sure that all connections incoming to port `8081` on the host are redirected to port `80` of the container. Call this container `webserver2`.

Credentials to access node01:
Name: `bob`
Password: `caleston123`

Solution

First log in to the node01 using below command:

```
ssh node01
```

Stop and remove the container named `webserver1` using below command:

```
docker rm webserver1 --force
```

Build a new image named `kodekloudwebserv` using the Dockerfile present in `/home/bob/kode_web`:

```
bob@node01:~/kode_web$ docker build -t kodekloudwebserv .
```

Launch container named `webserver2` using `kodekloudwebserv` image:

```
docker run --detach --publish 8081:80 --name webserver2
kodekloudwebserv
```

10

Task

NFS server and client tools are installed on `caleston-lp10` system. Instruct the NFS server to share the **/home** directory in **read-only** mode with IP addresses in the **10.0.0.0/24** CIDR range.

Solution

To share the `/home` directory with the IP addresses in the 10.0.0.0/24 CIDR range. Edit the `/etc/exports` as below:

```
/home 10.0.0.0/24(ro)
```

After editing the `/etc/exports` file, apply the changes by exporting the shared directories:

```
sudo exportfs -ra
```

11

Task

Find the application that is accepting incoming connections on **port 80**. Make note of the exact **name** of that **application**. You will need it later on.

As you investigated what application is accepting incoming connections to **port 80**, you might have noticed that two or more PIDs are associated with that. Basically, the application has forked multiple processes to do its job. Figure out which PID is associated with the **master process**.

With both of these things noted, create the following file: **/opt/process.txt**.

- On the **first line** add the **name** of the application associated with port 80, in all **lowercase letters**.
- On the **second line** add the **PID number** associated with the **master process**.

Solution

To find which application is listening on port 80, run below command:

```
sudo lsof -i :80
```

To fins the master process of the application, run below command:

```
pgrep -a nginx
```

12

Task

Explore your network settings and perform the following tasks:

1. There is currently one network interface which does not have any IPv4 address associated with it. **Temporarily** assign it the following IPv4 address: **10.5.0.1/24**. This should not be a permanent change (no need to edit configuration files).
2. What is the **default route for this system**? Create a file, and add a single line where you save the IP address for the gateway used by this default route (i.e., requests are routed to what IP address?). Save the address in this file: `/opt/gateway.txt`.
3. For the final task, find out what is the IP address of the main **DNS resolver configured for this system**. Create the file `/opt/dns.txt` and add a single line to it with that IP address.

Solution

1) Run `ip addr` to determine which network interface lacks an IP address.
2) To assign an IP address to eth1, execute the following command:

```
sudo ip addr add 10.5.0.1/24 dev eth1
```

3) To identify the default route for this system, use the command below:

```
ip route show default
```

4) To discover the DNS resolver configured for this system, use the command below:

```
cat /etc/resolv.conf
```

13

Task

A **new disk** was added to the system. Determine its device name. You can identify the disk because it is currently **unpartitioned**, not mounted anywhere, and is **4 GB** in size.

- Create **two partitions** of equal size on this disk: **2GB** each.
- Create an **ext4** filesystem on the **first partition**.
- Create an **xfs** filesystem on the **second partition**.
- Create two directories: **/part1** and **/part2**.
- **Manually mount** the **ext4 filesystem** in the **/part1** directory. Manually mount the **xfs** filesystem in the **/part2** directory.
- Also, **configure the system to automatically mount** these the same way, every time the operating system boots up.

Solution

Run `lsblk` command to list the block devices and find the block disk without any partitions.
To create 2 partition of 2Gb each, run the below command:

```
echo -e "g\nn\n\n\n+2G\nn\n\n\n+2G\nn\n\n\n\nw" | sudo fdisk /dev/vdb
```

Create an **ext4** filesystem on the first partition and an **xfs** filesystem on the second partition.

```
sudo mkfs.ext4 /dev/vdb1
```

```
sudo mkfs.xfs /dev/vdb2
```

Create two directories for mounting the partitions.

```
sudo mkdir /part1

sudo mkdir /part2
```

Mount the **ext4** filesystem on **/part1** and the **xfs** filesystem on **/part2**

```
sudo mount /dev/vdb1 /part1

sudo mount /dev/vdb2 /part2
```

To ensure these partitions mount automatically at boot, you'll need to add them to the `/etc/fstab` file.

```
/dev/vdb1    /part1    ext4    defaults    0    2
/dev/vdb2    /part2    xfs    defaults    0    2
```

14

Task

Configure the system to use `/swfile` as a swap file on **node01**.

1. First, create `/swfile`. Make the size of this file **exactly 1024 MB**.
2. Then take all the necessary steps to **temporarily mount this as swap**. (So that it's immediately used as swap for this current boot session).
3. But also make sure to **configure the system to also use this as swap every time it will boot** up in the future.

Credentials to access node01:
Name: `bob`
Password: `caleston123`

Solution

1) Login to `node01` server
2) Execute the following command to create a swap file of exactly **1024 MB**

```
sudo fallocate -l 1024M /swfile
```

3) Change the permission of the /swfile as below:

```
sudo chmod 600 /swfile
```

4) Use `mkswap` to set up the file as Linux swap area:

```
sudo mkswap /swfile
```

5) Activate the Swap File

```
sudo swapon /swfile
```

6) To ensure that the swap file is used on every boot, you need to add it to the `/etc/fstab` file.

```
/swfile none swap sw 0 0
```

15

Task

On `node01` two processes are overusing our storage device. One is executing a lot of I/O operations per second (small data transfers, but a very large number of such transfers). Otherwise said, the process has a high **tps/IOPS**. The other process is **reading** very **large volumes of data**.

1. Identify the process with the **high tps**. What **partition** is it using? Create the file `/opt/devname.txt` and write the device name of that partition inside that file. For example, if it's using **/dev/vde5**, you would simply write **/dev/vde5** on a single line in that file. Note that there might be some abstractions behind this, and we're not interested in **device mapper** names, but rather, the **real device** the mapper is using.
2. Identify the process with the **high read transfer rate/second**. Create the file `/opt/highread.pid` and write the **PID number** of that process in that file. For example, if the PID is 3886 you just write 3886 in that file (only the number, on a single line).

Credentials to access node01:
Name: `bob`
Password: `caleston123`

Solution

To identify the process with **high TPS** and the **partition** it is using, follow the steps below:

- Run the `sudo dstat --top-io --top-bio` command to get the process name with I/O activity.
- Run the `pgrep python3` command to get the PID of the process.
- Run `sudo lsof -p <PID>` to list the open files by the process.
- Run `sudo lsof -p <PID> | awk '{print $9}' | while read file; do df $file; done` to get the device details.
- Find the actual partition used by running the `pvs` command and store the actual device name in `/opt/devname.txt`.

Run the command below to get the **PID** of the process with **high kB_read/s**:

```
   sudo pidstat -d 1
```

16

Task

On `node01` list all filesystems to check out how much free space they have remaining. You'll find one which is almost full (should be around **98% full**). To confirm it is the correct filesystem, see where it is mounted, and you should find many directories on it in the form of numbers from 1 to 999. Find the directory which has the **largest file** and **delete that file (only that file, nothing else)**.

Solution

Run the below command to get the largest file:

```
bob@node01:~$ sudo find /data -type f -exec du -h {} + | sort -rh | head -n 1

196M    /data/683/lf
```

Delete only the largest file:

```
sudo rm -rf /data/683/lf
```

17

Task

On `caleston-lp10` change the configuration for the SSH daemon. **Disable** X11 forwarding **globally**. Then, make an exception for just one user called **bob**. For that user alone **enable** X11 forwarding.

Do not restart the SSH service after making the changes.

Solution

1) To disable X11 forwarding globally, find the line that contains `X11Forwarding` in `/etc/ssh/sshd_config`. It may be commented out by default with a #. Change it to:

```
X11Forwarding no
```

2) To enable X11 Forwarding for User **bob**, add a conditional block at the end of the sshd_config file to enable X11 forwarding specifically for the user **bob**:

```
Match User bob
    X11Forwarding yes
```