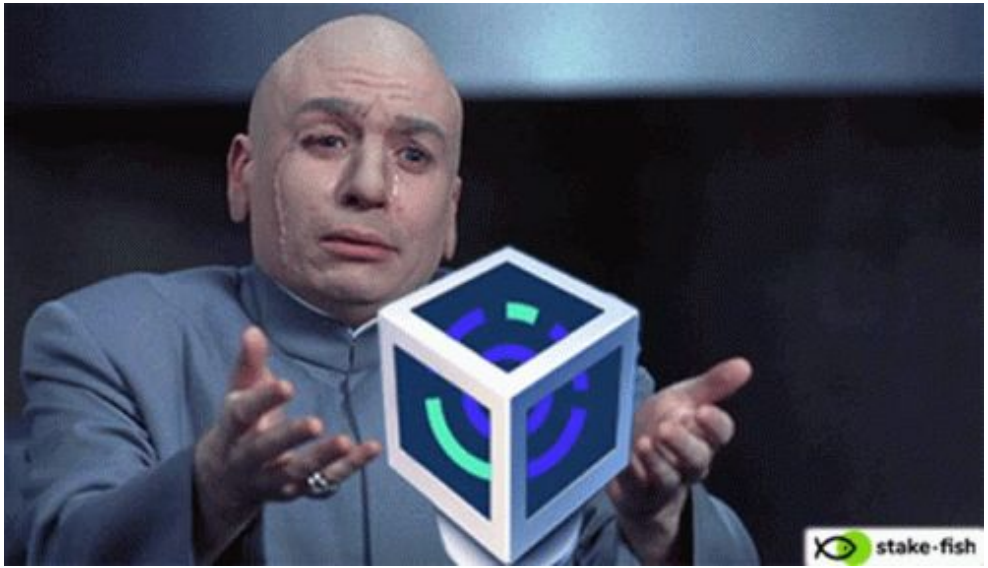
Good Afternoon CEG2350

Aiden Cox - CEG 2350 Lab Lead
Austin Kellough - CEG 2350 Lab Assistant
Quote of the week:
“It's not a bug, it's an undocumented feature.”

How was Lab07?



System Discovery? Virtual Machines?

Beginning Lab08

Lab Instructions: https://pattonsgirl.github.io/CEG2350/Labs/Lab08/Instructions.html

Lab Template: https://raw.githubusercontent.com/pattonsgirl/CEG2350/refs/heads/main/docs/Labs/Lab08/LabTemplate.md

How's life?
Anything needing called to my attention?

What do we have here? - Part 1

Part 1 - What do we have?

Your AWS instances have one block device in use - `xvda`. In this section, you will explore commands view partition and filesystem information about `xvda`. **Do not make any modification to `xvda`.**

Hint, remember that disk devices are in the `/dev/` folder

- **Useful Commands: `lsblk`, `parted`, `blkid`, `df`, `cat`**

For tasks that ask you to use a command, write the command used and include the output of the command.

1. Use `lsblk` to list only information about the `xvda` block device.
2. Use `parted` to print the partition table of the `xvda` block device.
3. For the `xvda` partition table:
   - Does it use MBR or GPT?
   - How many partitions are on the block device?
   - What is the largest partition?
4. Use `blkid` to view information of `xvda` and it's partitions. Play with `*` to get all matches that start with `\dev\xvda`
5. For the partition with the root filesystem:
   - What is the device name?
   - What is the partition label?
   - What type of filesystem is on the partition?
6. Use `df` to view file system disk space usage in human readable format (meaning it prints MB/KB/GB)
7. For the root filesystem:
   - What is the total size?
   - How much space is used?
   - Where is it mounted to?
8. View the contents of the filesystem table in `/etc/fstab`
9. Explain fields & purpose of fields in the entry that mounts the root filesystem.

```
ubuntu@ceg2350-sandbox:~$ lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0 27.6M  1 loop /snap/amazon-ssm-agent/11797
loop1      7:1    0 26.3M  1 loop /snap/amazon-ssm-agent/9881
loop2      7:2    0 73.9M  1 loop /snap/core22/2133
loop3      7:3    0 44.4M  1 loop /snap/snapd/23545
loop4      7:4    0 73.9M  1 loop /snap/core22/2111
loop5      7:5    0 50.8M  1 loop /snap/snapd/25202
xvda     202:0    0   16G  0 disk
├─xvda1  202:1    0   15G  0 part /
├─xvda14 202:14   0    4M  0 part
├─xvda15 202:15   0  106M  0 part /boot/efi
└─xvda16 259:0    0  913M  0 part /boot
xvdb     202:16   0    4G  0 disk
```

Reminder for Lab08 that we want **ONLY** the **xvda** block!

Something New - Part 2

Part 2 - Something new

You have had an unformatted disk available on your AWS instance all along. The disk is `xvdb` - you can see it, but that it has no partitions, if you run `lsblk`. Time to create a partition table and a partition on `/dev/xvdb` so that in the next Part we can create a filesystem on the partition and mount it for use.

- **Useful Commands: `df`, `lsblk`, `blkid`, `gdisk`**

1. Using the `gdisk` GPT partition table manipulator, find out what the following main menu options do:
   - `p`
   - `o`
   - `n`
   - `i`
   - `w`

2. Edit the `xvdb` block device with `gdisk`. Using the main menu, configure the disk to use the GPT partition table type, have at least 1 partition, and have that partition use the Linux filesystem type. Save your changes to the disk.
   - This will be the only partition, so it can use the recommended sizes, which is to say, start at the end of the GPT partition table, and span to the last block of the disk.

3. View the partition table of `xvdb`
4. Answer the following about `xvdb` in its current state:
   - What is the device name of the only partition?
   - What is the size of the only partition?
   - What filesystem type will be used on the only partition?

```
ubuntu@ceg2350-sandbox:~$ sudo gdisk
GPT fdisk (gdisk) version 1.0.10

Type device filename, or press <Enter> to exit: /dev/xvda
Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help):
```

```
ubuntu@ceg2350-sandbox:~$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        15G  4.0G   11G  28% /
tmpfs           479M     0  479M   0% /dev/shm
tmpfs           192M  876K  191M   1% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
/dev/xvda16     881M  151M  669M  19% /boot
/dev/xvda15     105M  6.1M   99M   6% /boot/efi
tmpfs            96M   16K   96M   1% /run/user/1000
```

Virtualize the Machine - Part 3

Part 3 - File it away

Now that you have a partition, you can create a filesystem on it in order to interact with it to store and organize files and create permissions for the files.

- **Useful Commands: `mkfs`, `mount`**

1. Make an `ext4` filesystem on the partition on `xvdb`
2. Use `blkid` to view information of the partition on `xvdb`
3. Make a directory in `/mnt/` named `expanse`
4. Mount the partition on `xvdb` to `expanse`
5. In `expanse` create some files and directories
6. `umount` the partition on `xvdb`
7. When can I interact with files on the filesystem on the partition in `xvdb`?

Take a `fstab` at this guy! - Part 4

Part 4 - Take a `fstab` at this

Right now, every time you want to access your new filesystem on `xvdb1` after a system reboot you need to mount it. It would be handy to have it auto-mount. The filesystem table file - `fstab` is a file that stores information about what to mount when the system boots. Your task in this part is to **append** a new entry to `fstab` to automount the filesystem on `xvdb1`.

1. Make a backup of the current version of `/etc/fstab` to `/etc/fstab.bak`
2. Add a line to `/etc/fstab` to mount the partition on `xvdb` to the mount point (`/mnt/expanse`)
3. Test your changes using `mount -a` to mount / remount records entries in `etc/fstab` and then check that your additional entry worked (make sure `xvdb1` is unmounted first, then test and verify)
4. **If you do not think your changes are correct** restore `/etc/fstab` from `/etc/fstab.bak`. If you think they are correct, you may leave your changes in place.

What is dead, may still be read - Part 5

Part 5 - What is dead may still be read

When you delete a file, you are used to it no longer being accessible, or to it still being temporarily available / recoverable via the Recycle Bin. But once you can't open it anymore, it should be gone, including from the disk, right? Right?!?

This part will have you acknowledge that to truly make data gone and no longer readable, there are extra steps involved. The general recommendation is to trust nothing, and take disks that have had important data on it, like tax returns, credit card info, passwords, etc, taken to a shredding center and properly ripped to computer-illegible pieces.

- **Useful Commands: `mount`, `strings`**

1. On the filesystem you created on the `xvdb` partition, create **two** files, each with a different FAKE secret about you.
2. Find out information on the `strings` command. If you referred to an internet resource, make sure you cite it by including the URL.
3. Run `strings` on the filesystem partition on `xvdb` - read through the output and make an analysis about what output you are viewing.
4. Delete **one** of the files with a secret.
5. Run `strings` on the filesystem partition on `xvdb` - read through the output and determine if the secret, while no longer accessible via the filesystem, is still readable on the partition.
6. Find out information on the `shred` command. If you referred to an internet resource, make sure you cite it by including the URL.
7. Use `shred` to overwrite the contents of your second secret file on the disk. Write a short report of steps and provide proof that the file is no longer readable on the disk or accessible in the filesystem. Include an explanation of flags used (if any).

Have a Good Weekend!

Don’t hesitate to reach out and ask questions!
Quote of the week:
“It's not a bug, it's an undocumented feature.”