**EBS (Elastic Block Store)**

1. AWS storage service that provides block-level storage for EC2 instances.
2. Offers persistent storage, meaning data remains even after the instance is stopped.
3. Supports snapshots for backup and recovery.
4. Provides different volume types like SSD (gp3, io2) and HDD (st1, sc1).
5. Can be resized without downtime.
6. Used for databases, applications, and system drives.

**Horizontal Scaling**

1. Increases capacity by adding more machines/instances.
2. Improves fault tolerance and load distribution.
3. Commonly used in cloud environments with auto-scaling.
4. Requires load balancers to distribute traffic efficiently.
5. More cost-effective for handling high traffic.
6. Example: Adding multiple servers to handle web traffic.

**Vertical Scaling**

1. Increases capacity by upgrading an existing machine (CPU, RAM, storage).
2. Easier to implement but has hardware limitations.
3. Suitable for databases and monolithic applications.
4. No need for a load balancer since it's a single system.
5. Downtime may be required during upgrades.
6. Example: Upgrading an EC2 instance from t2.micro to m5.large.

**Steps to Create an EC2 Instance with Necessary Settings**

**. Launch an EC2 Instance**

* Click **"Launch Instance"**
* Enter **instance name** (e.g., "MyEC2")

**. Choose an Amazon Machine Image (AMI)**

* Select an OS (e.g., **Amazon Linux, Ubuntu, Windows**)

**. Choose an Instance Type**

* Select instance type (e.g., **t2.micro** for free-tier)

**. Configure Instance Settings**

* **Number of Instances:** Default = 1
* **IAM Role:** Select if required
* **Shutdown Behavior:** Choose **Stop**

**. Configure Storage**

* Default: **8GB (EBS)** (Increase if needed)

**. Add Security Group (Firewall Rules)**

* Create or select a **security group**
* **Allow SSH (port 22)** for Linux
* **Allow RDP (port 3389)** for Windows
* Add **HTTP (80), HTTPS (443)** if hosting a website

**. Choose Key Pair (For SSH Access)**

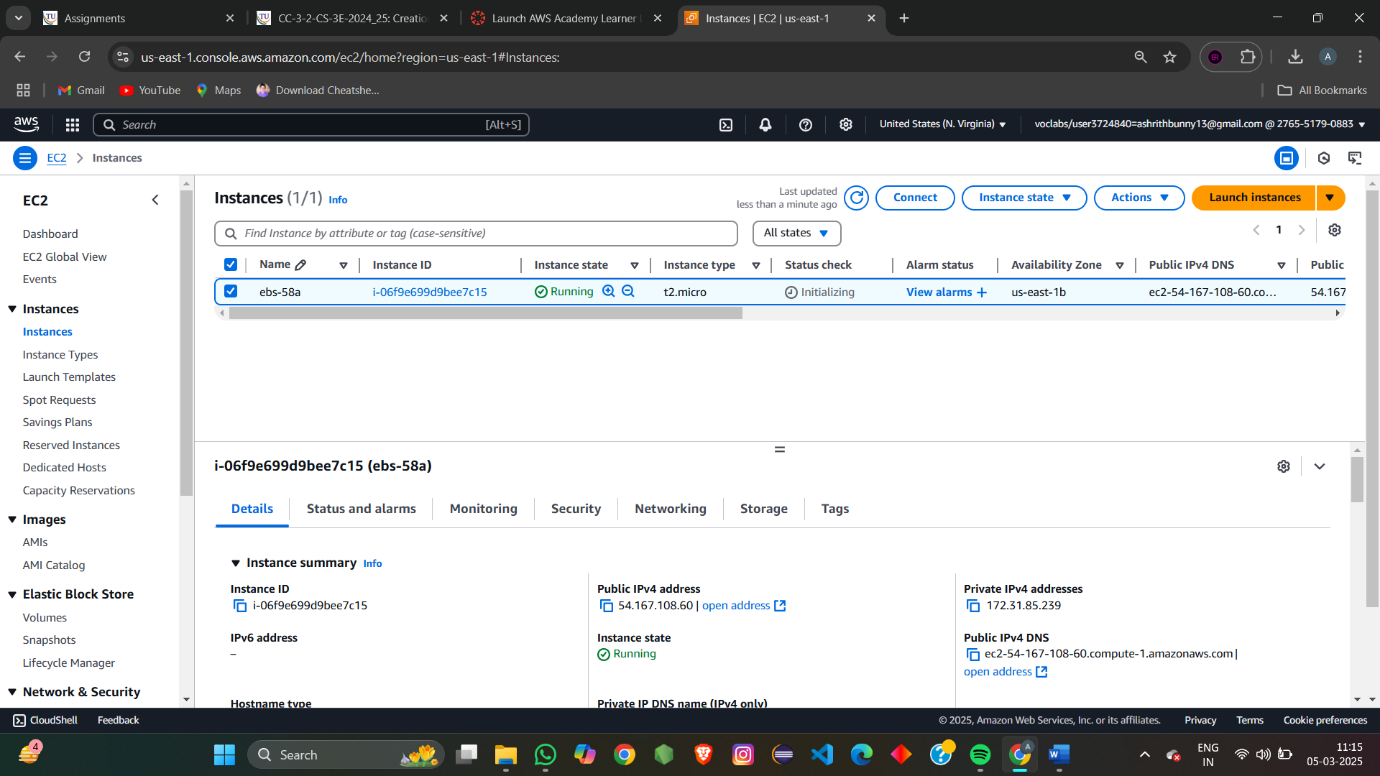
* Select **existing key pair** or **create new key pair**
* Download **.pem** file (Important for SSH login)

**. Launch the Instance**

* Click **Launch Instance**

Experiment: EBS service

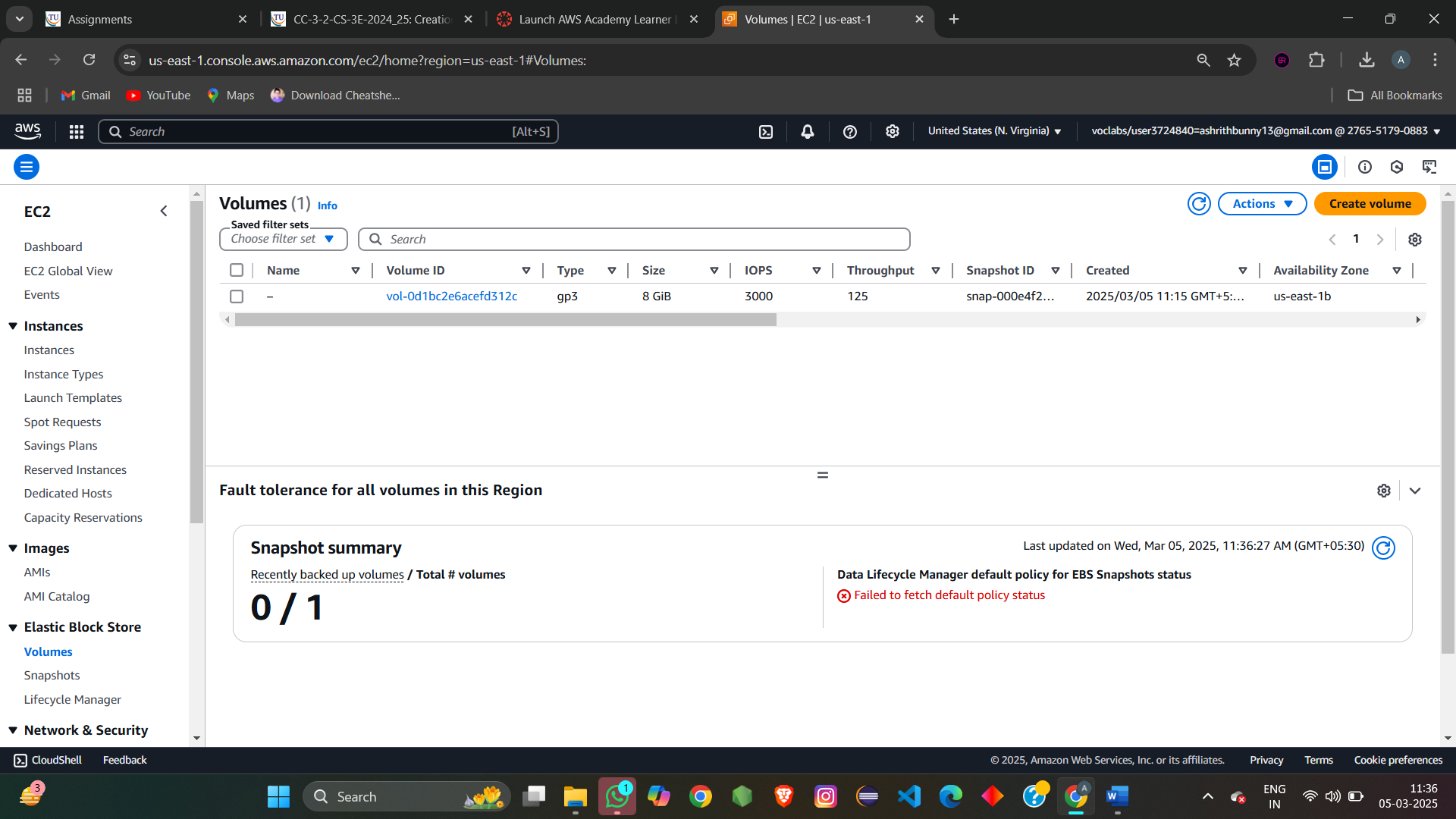
Create an ec2 instance:



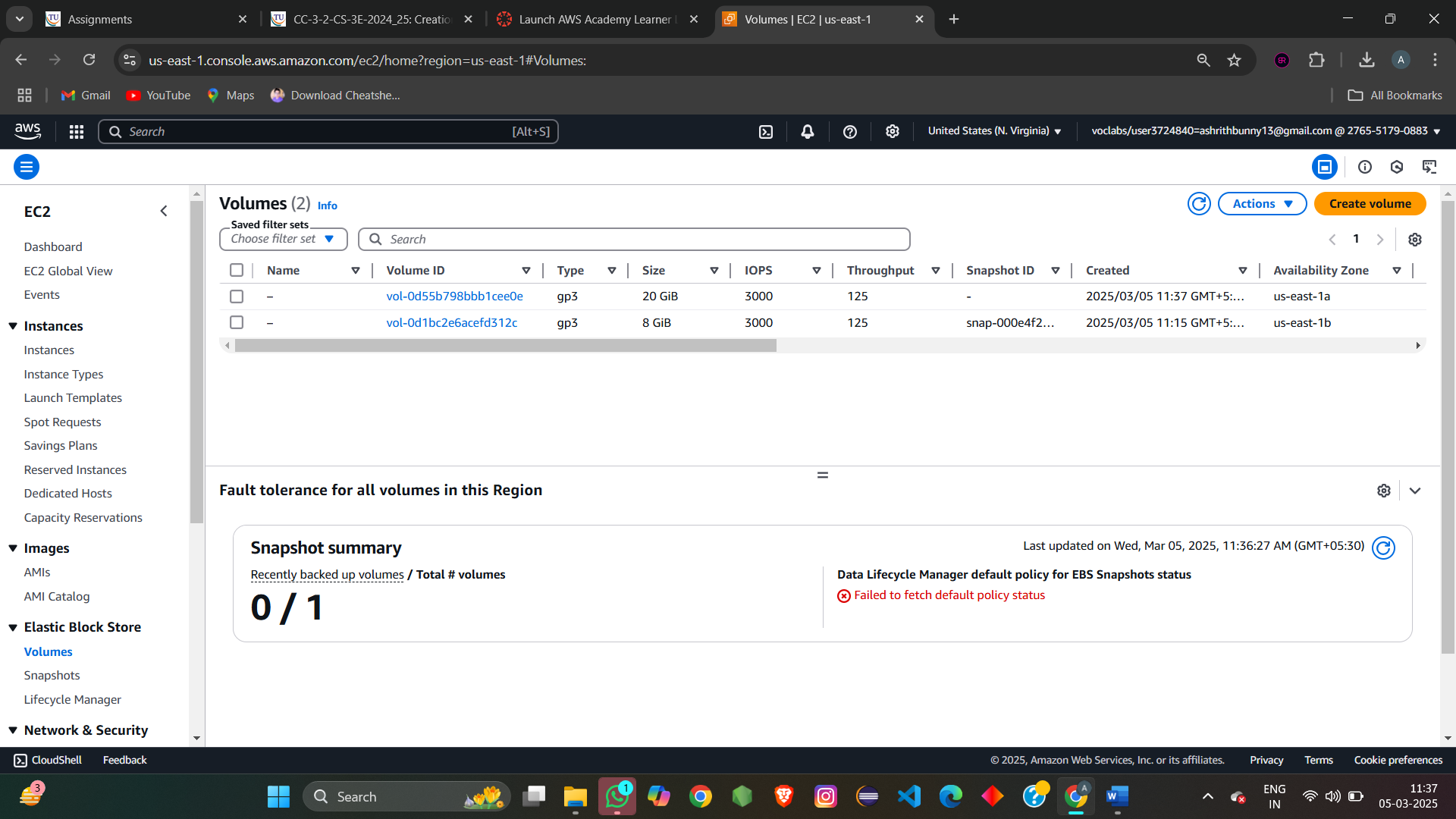
We can add ebs volume while creating ec2 instance before launching the instance.

Or else we can add ebs volume after creating ec2 instance .

Elastic block store 🡪volumes



By clicking on creating volume create the required volume 🡪click on create volume

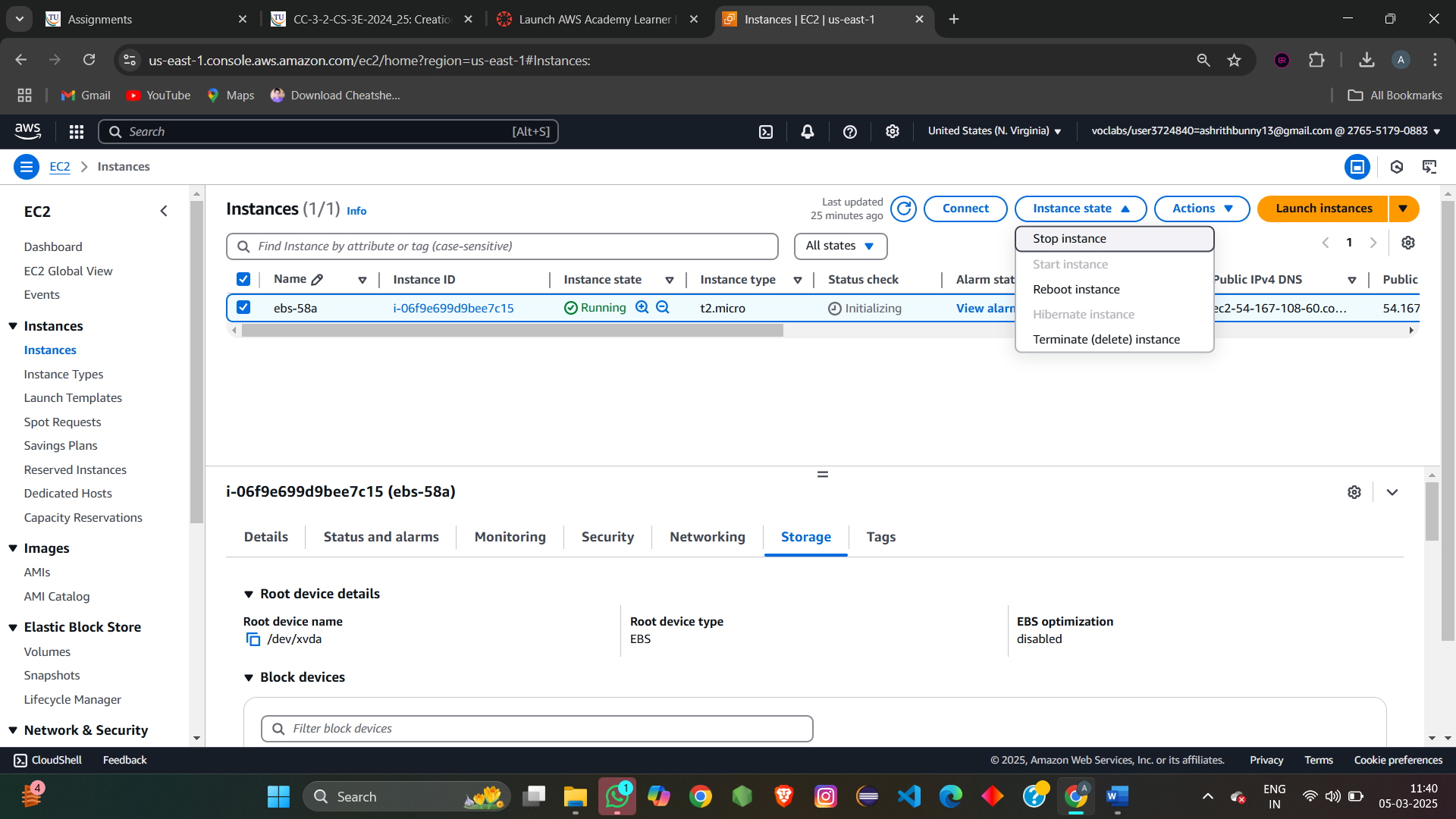


**Vertical Scaling**

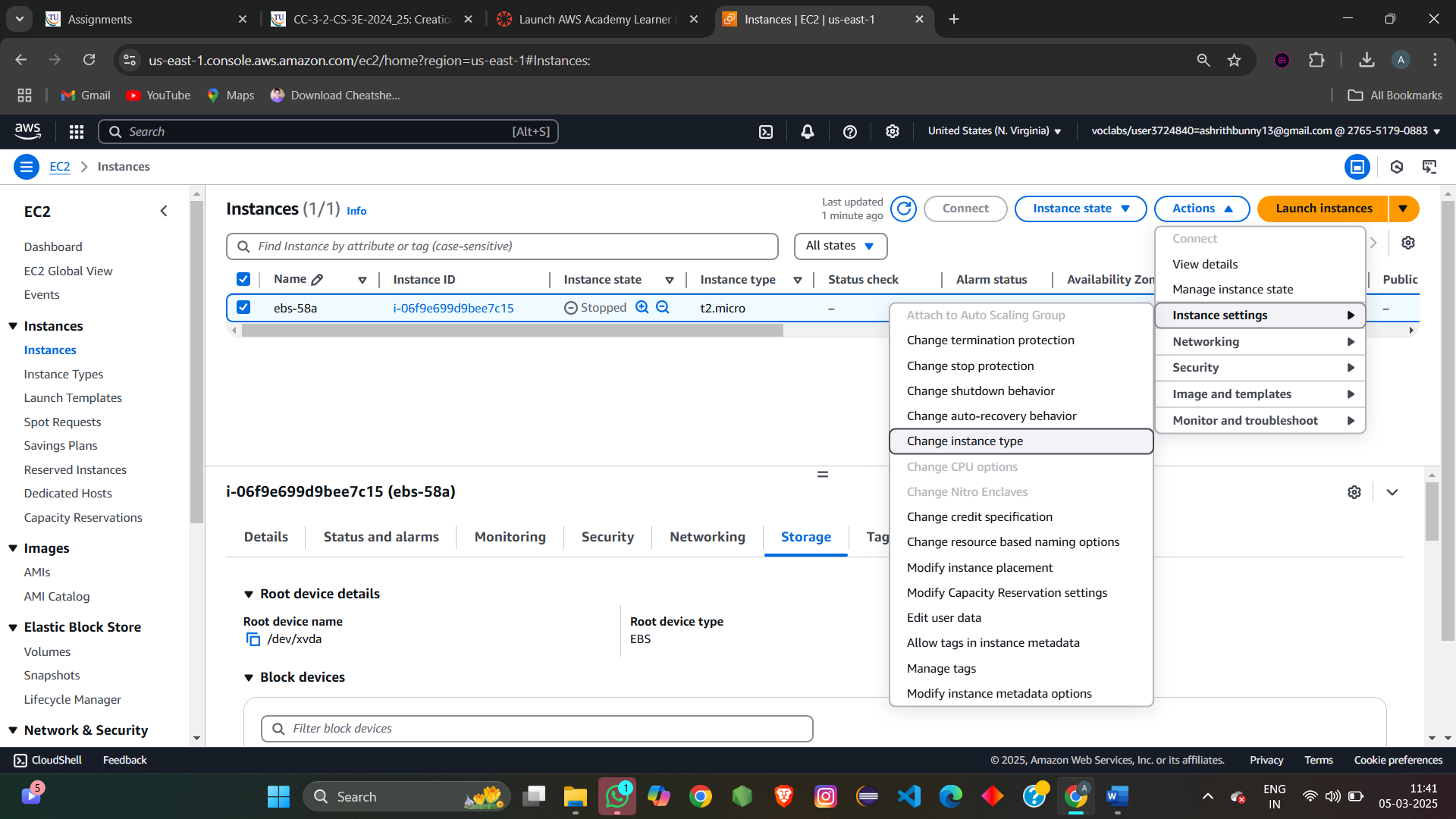
1. Example: Upgrading an EC2 instance from t2.micro to m5.large.

That is changing instance type

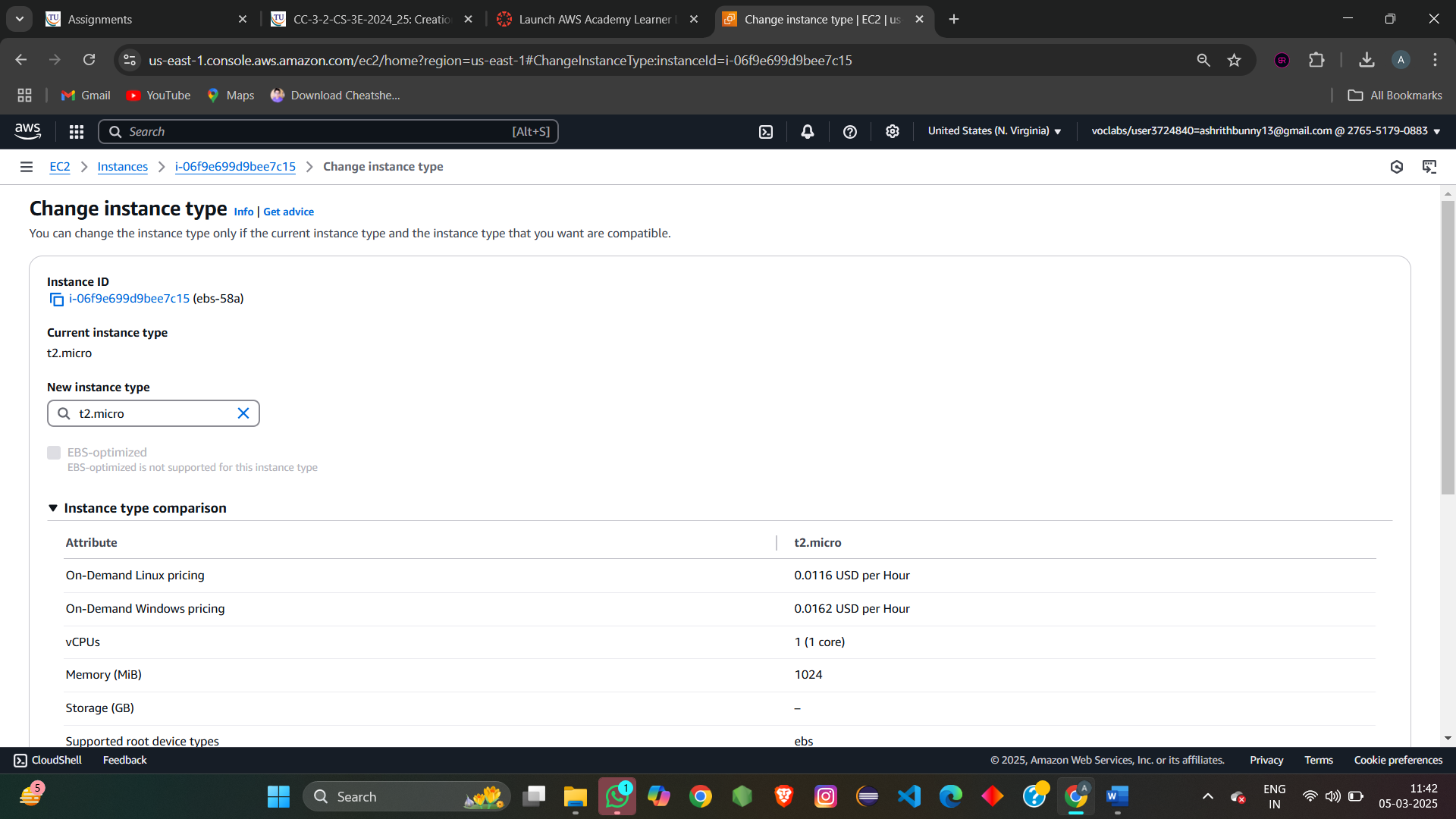
We have to first stop the instance to change the instance type



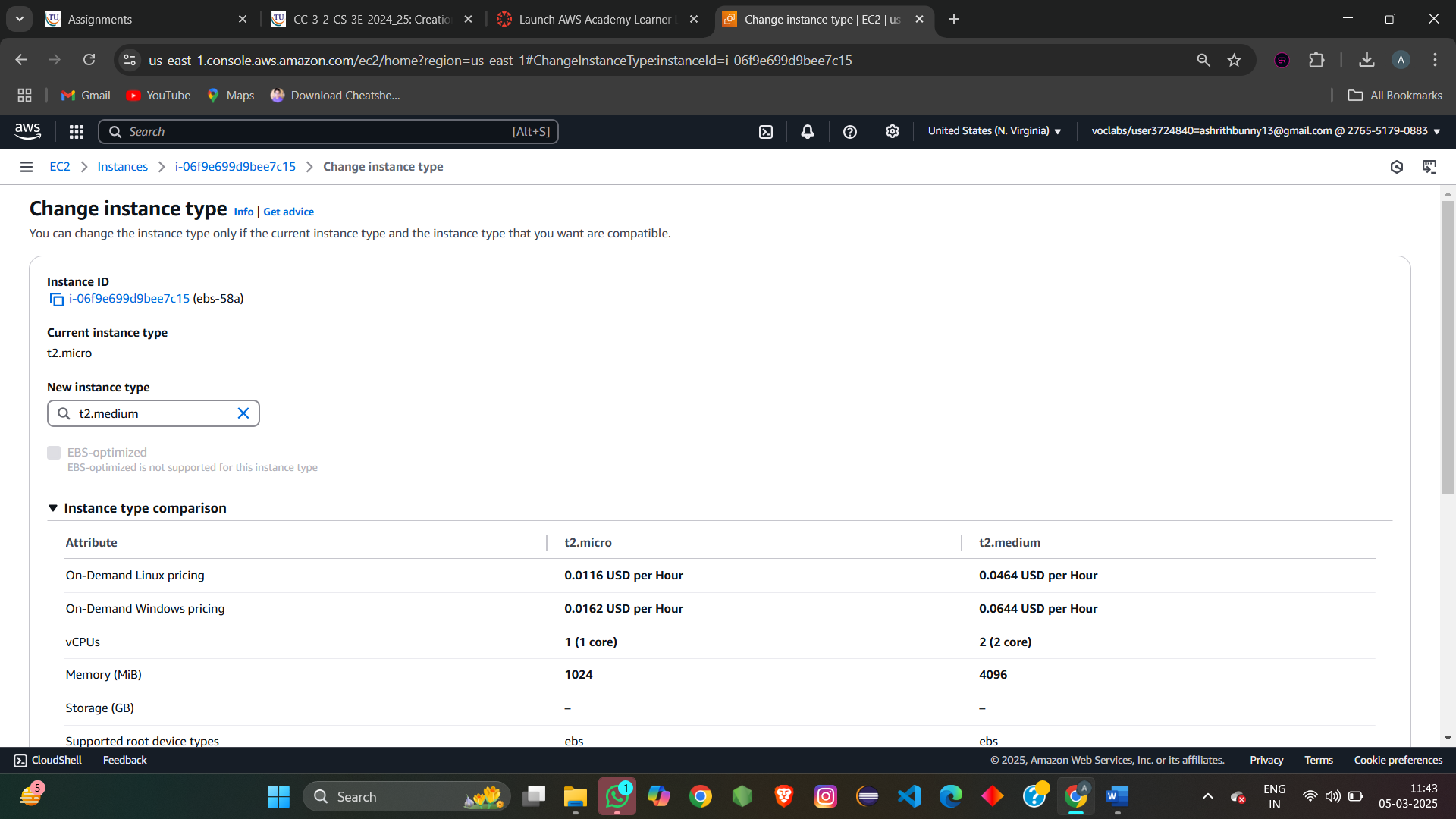
Change the instance type



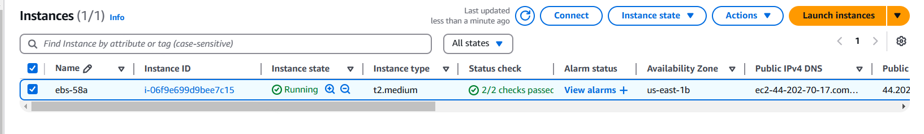
change instance menu open



Change the instance type to t2.medium

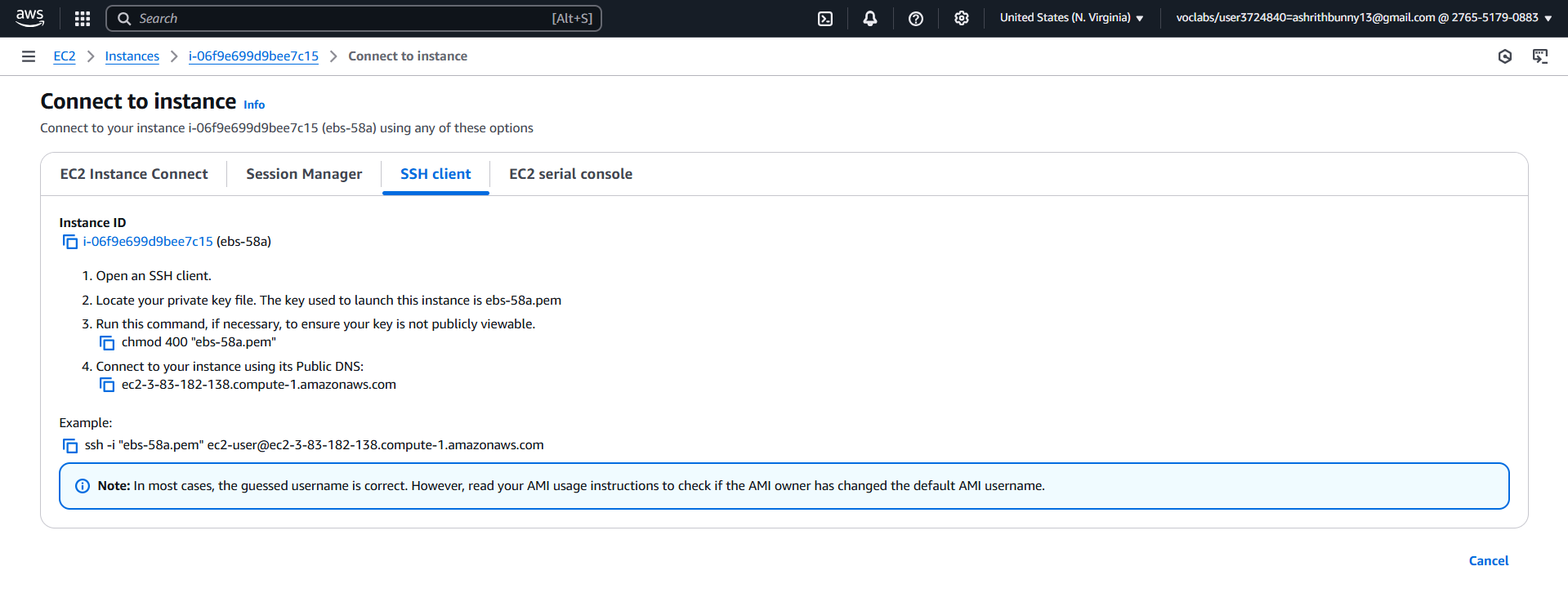


Instance type get changed to t2.medium

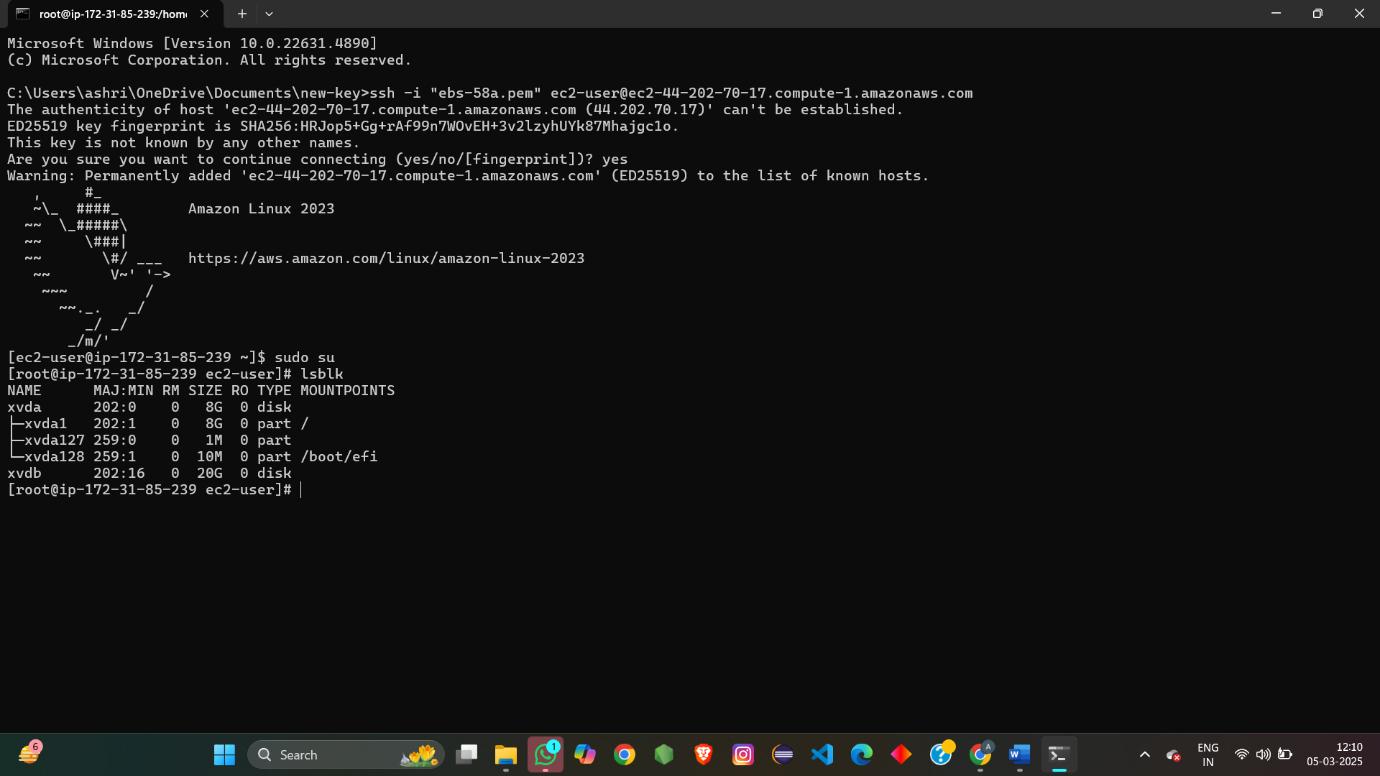


Then start the instance after the changes.

Now connect to the instance via EC2 instance connect



Than terimanl opens



Type the commands to create the partion in the volume

**1. Gain Root Access**

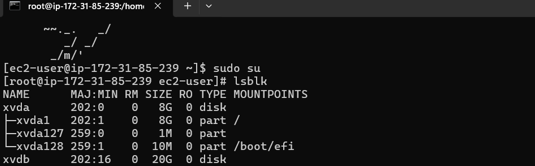
sudo su

* Switches to the **superuser (root)** mode to get administrative privileges.
* The prompt may change from $ to #, indicating root access.

**2. List Available Block Devices**

lsblk

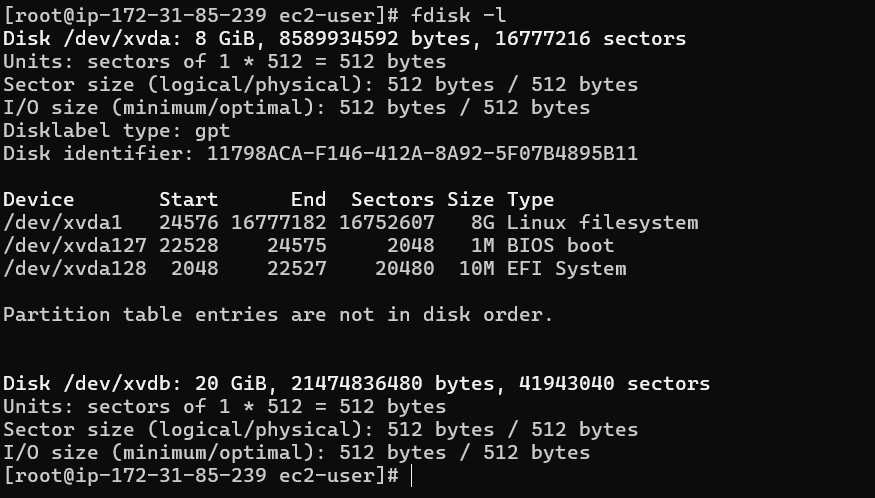
* Displays a **list of available storage devices** and their partitions.
* It shows **disk names, mount points, and file system types** (if any).



**3. View Disk Information**

fdisk -l

* Lists all available **disks and their partitions**.
* Shows disk **size, sector information, and partition table format** (MBR or GPT).



**4. Select and Modify a Disk**

fdisk /dev/xvdb

* Opens the partition table editor (fdisk) for the **unmounted disk /dev/xvdb**.

**5. Inside fdisk Utility**

command: m

* Shows a **help menu** listing available options in fdisk.

n

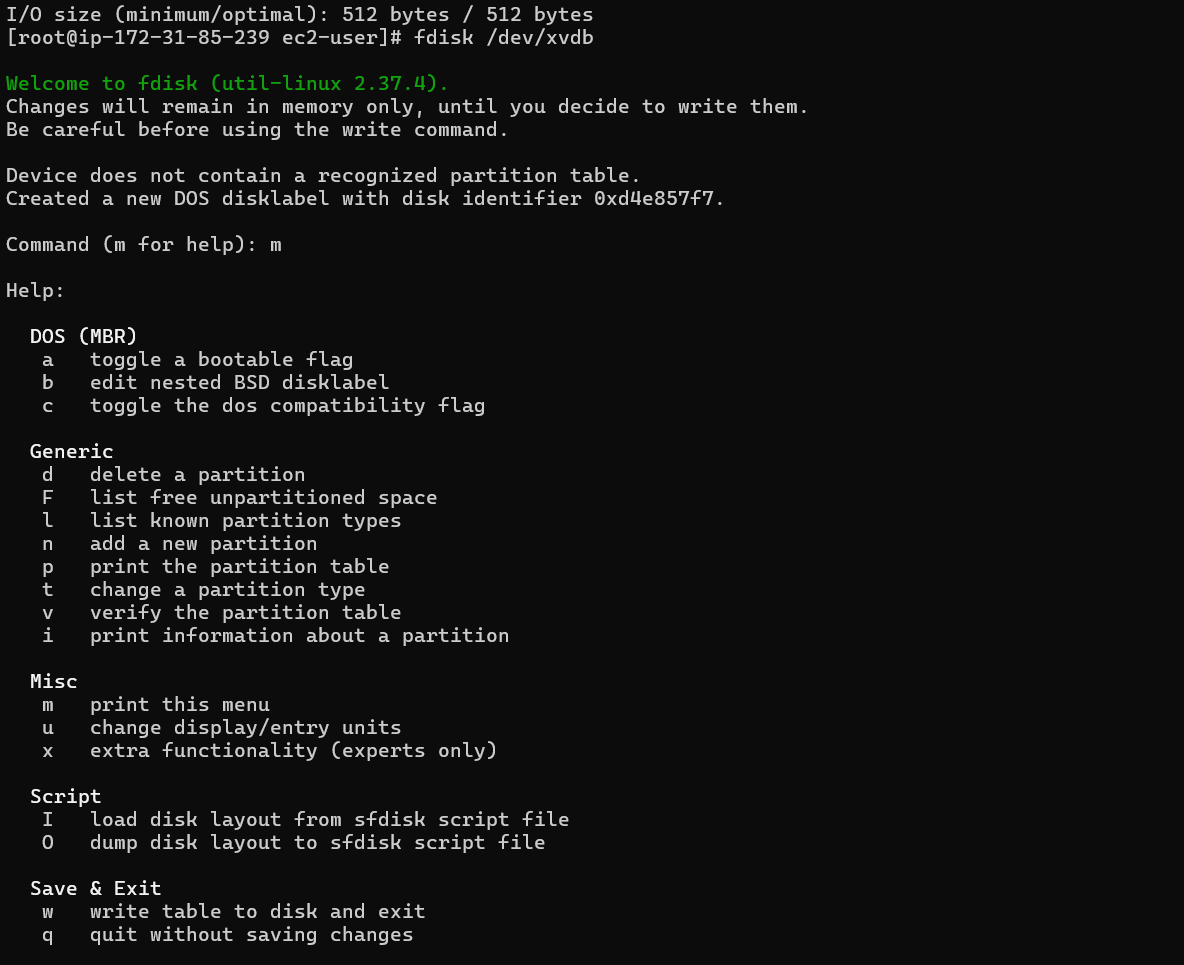
* Creates a **new partition** on /dev/xvdb.
* It asks for partition type (**primary or extended**)—you choose primary.

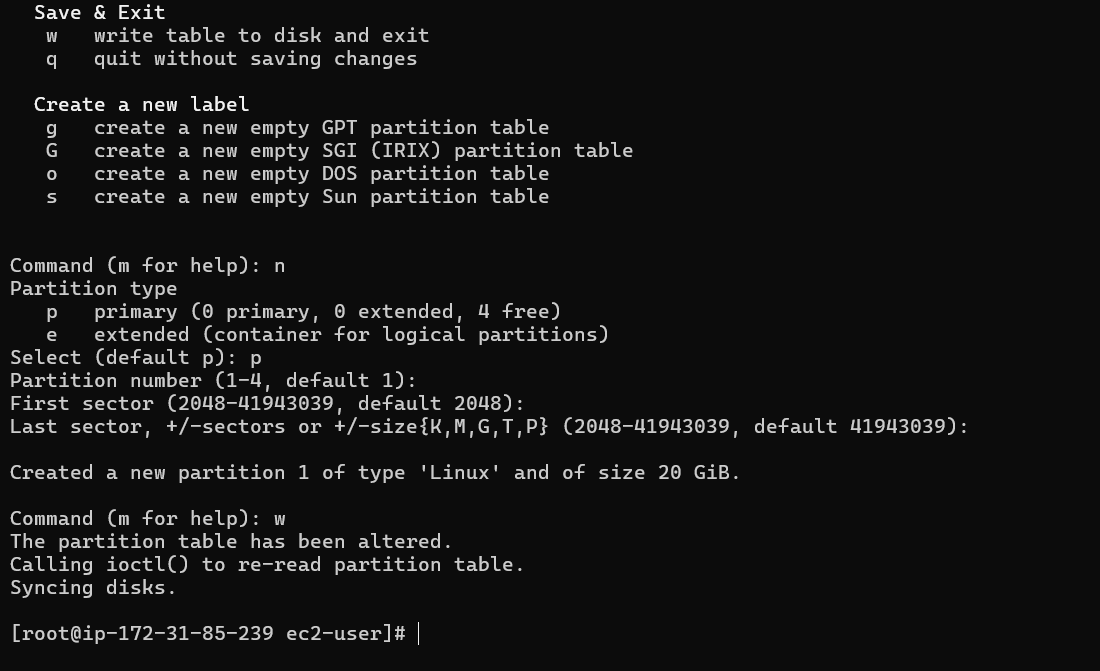
p

* Selects a **primary partition** type (since it's a new disk, it’s the first partition: /dev/xvdb1).

w

* Writes changes to disk and **exits** fdisk.
* The new partition /dev/xvdb1 is created but **not yet formatted**.





**6. Refresh Kernel Partition Table**

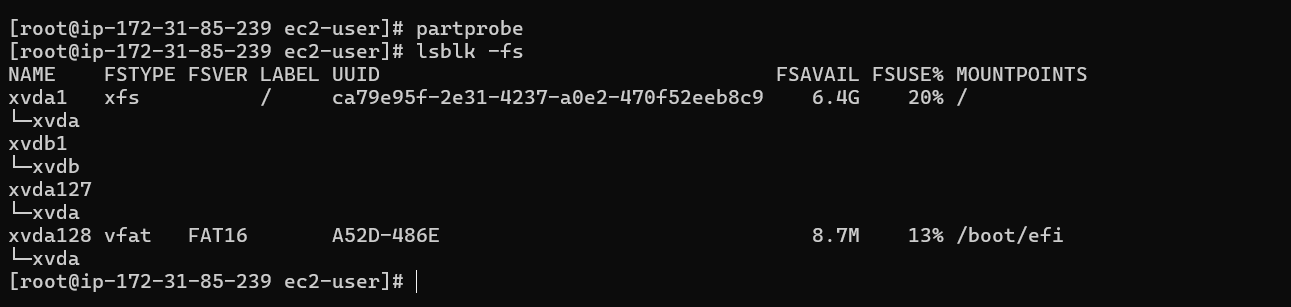
partprobe

* Informs the **kernel** about partition table changes **without requiring a reboot**.

**7. Verify Partition Details**

lsblk -fs

* Lists block devices with **file system information**.
* It confirms that /dev/xvdb1 exists, but it’s still **unformatted**.



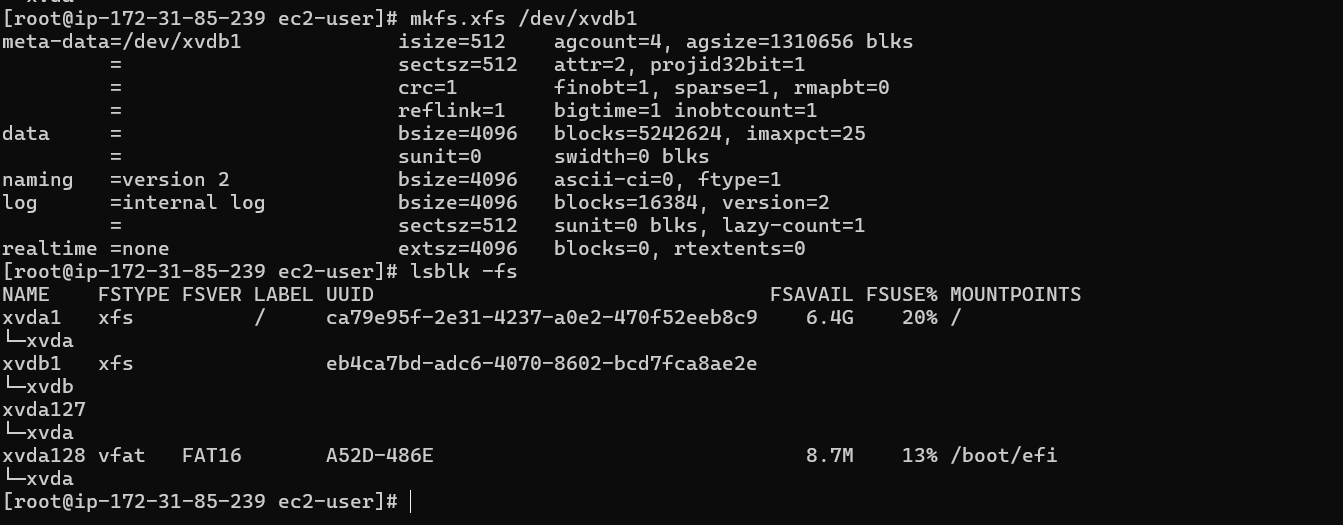
**8. Format the Partition with XFS**

mkfs.xfs /dev/xvdb1

* Creates an **XFS file system** on /dev/xvdb1.
* XFS is a high-performance file system, commonly used in **AWS EC2** instances.

Followed by the command “ lsblk -fs “

Ensures that /dev/xvdb1 now has an **XFS file system**.



**9. Create a Mount Directory**

mkdir /mnt/csee58a

* Creates a **mount point (/mnt/csee58a)** where the disk will be accessed.

**10. Mount the New Partition**

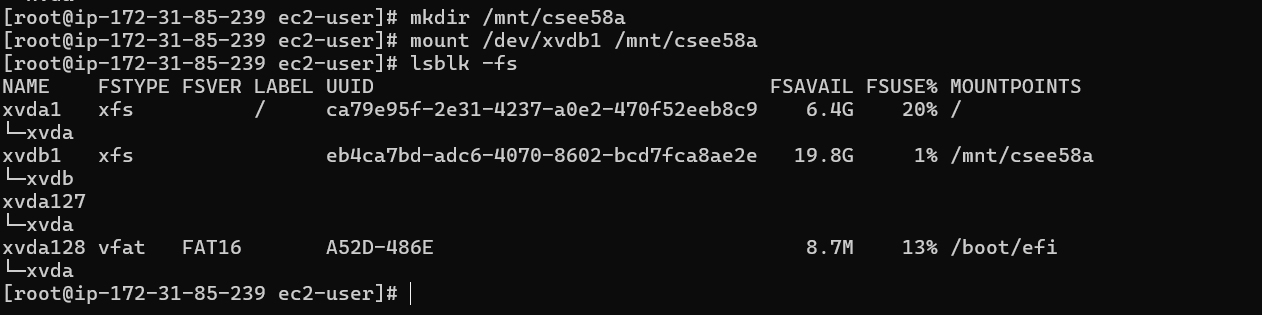
mount /dev/xvdb1 /mnt/csee58a

* Attaches (mounts) /dev/xvdb1 to /mnt/csee58a.
* The partition can now be accessed through /mnt/csee58a.

**11. Verify Mount Details**

lsblk -fs

* Confirms that /dev/xvdb1 is **mounted** at /mnt/csee58a.



**12. Navigate to the Mounted Directory**

cd /mnt/csee58a

* Moves into the newly mounted **storage directory**.

**13. Create 10 Test Files**

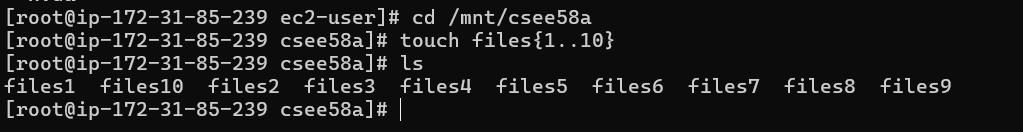
touch files{1..10}

* Creates **10 empty files** (files1, files2, ..., files10) in /mnt/csee58a.

**14. List the Created Files**

ls

* Displays the **10 empty files** in /mnt/csee58a.



**15. Navigate to Root Directory**

cd /

* Moves back to the **root (/) directory**.

**16. Edit /etc/fstab for Persistent Mounting**

vi /etc/fstab

* Opens the fstab file in vi editor to make the **mounting persistent** across reboots.
* You add an entry like: to insert enter “I” and click on end key to go to last

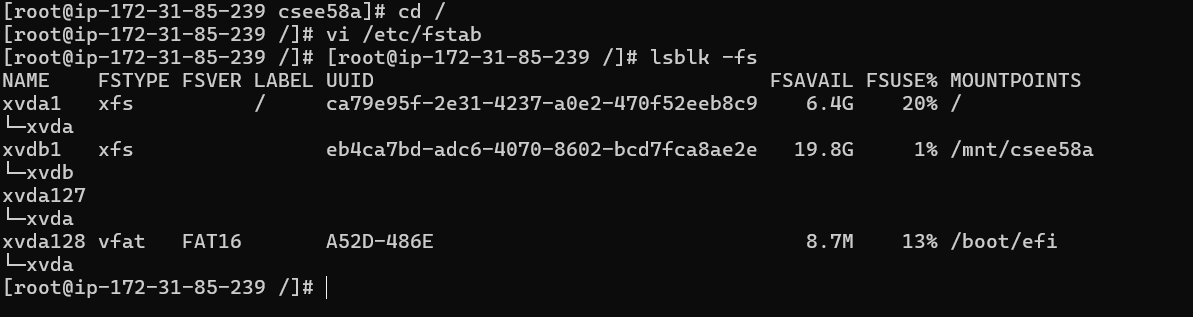
/dev/xvdb1 /mnt/csee58a xfs defaults 0 0

* Saves and exits (ESC → :wq).

**17. Verify Again**

lsblk -fs

* Ensures /dev/xvdb1 is **still mounted** at /mnt/csee58a.



**18. Navigate to Mounted Directory Again**

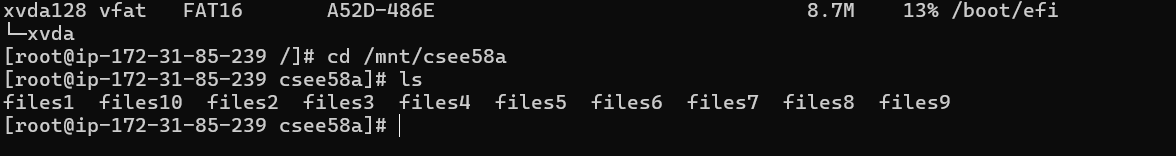
cd /mnt/csee58a

* Moves back to /mnt/csee58a.

**19. Check Files Again**

ls

* Confirms that **files1 to files10** still exist.



**21. Move Back to Root Directory**

cd /

* Moves to the root directory before **unmounting**.

**22. Unmount the Partition**

umount /mnt/csee58a

* Unmounts the partition /dev/xvdb1.
* /mnt/csee58a is no longer accessible until remounted.

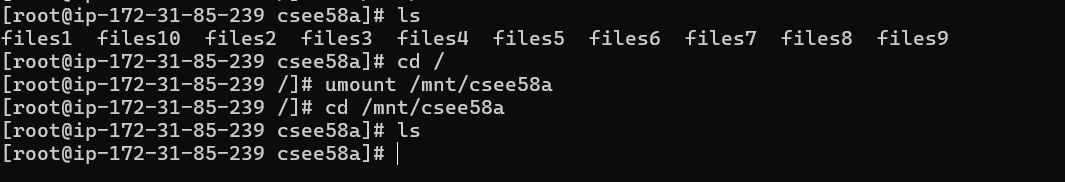
**23. Verify if Mount Point is Empty**

cd /mnt/csee58a

* Tries to access the directory, but since it's unmounted, it may return **an error** or show an empty folder.

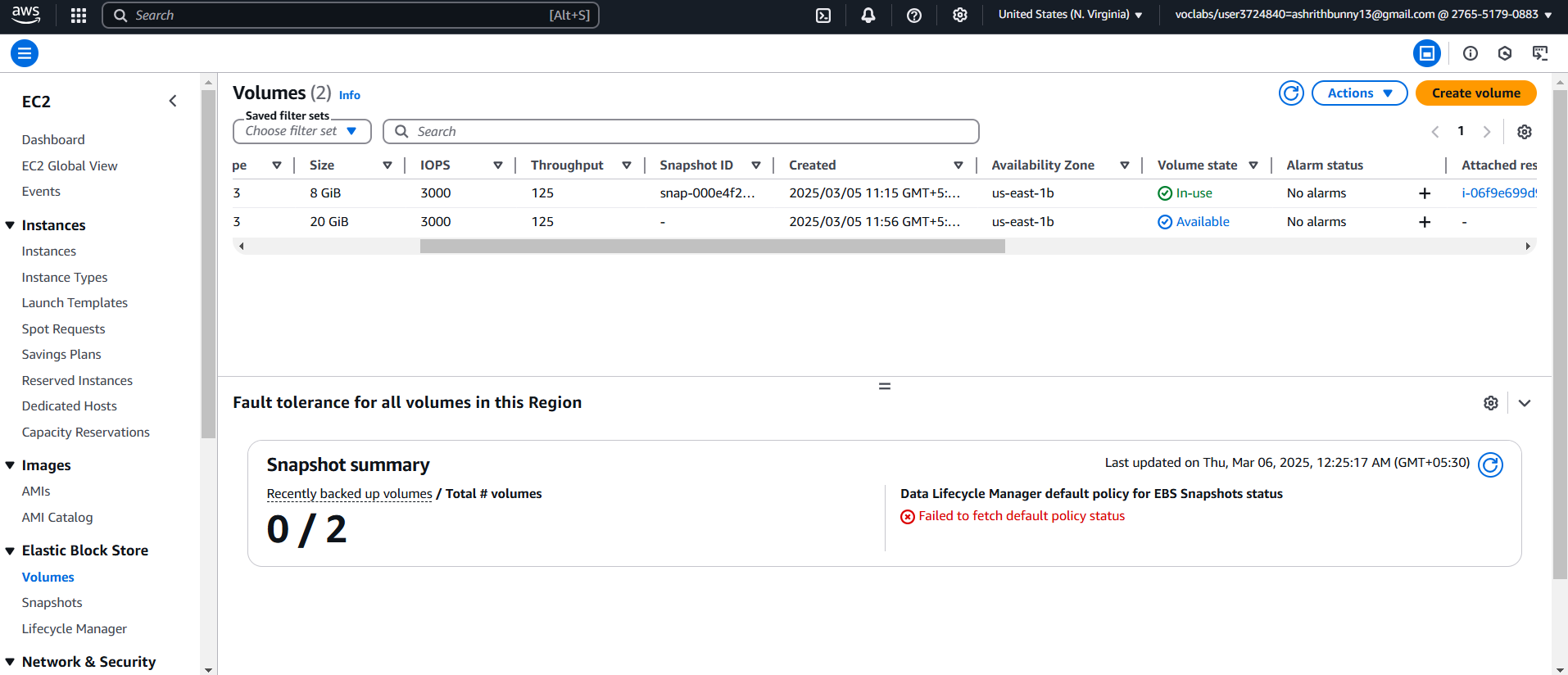
ls

* If the partition was successfully unmounted, **no files will be listed**.



This is how we are able to partion the volume , mount the volume and unmount the volume

Now detach the volume , from volume sec in left side window



If you detach the volume will be available in available state.

As you know the volume that you attached to the new instance has the 10 text files created before

Now mount the path in the same way you have done for before instance.

Now that you've attached the volume to your new instance, follow these steps to list the text files.

**1️ Verify the Attached Volume**

Check if the volume is recognized by the new instance:

lsblk

You should see an entry like /dev/xvdb1 (or similar).

**2️ Find the Filesystem Type**

Check if the volume has a filesystem:

lsblk -fs

If it shows **xfs** or **ext4**, then it is formatted. If not, you may need to format it.

**3️ Mount the Volume**

If the volume is unmounted, mount it to a directory:

sudo mkdir -p /mnt/mydata

sudo mount /dev/xvdb1 /mnt/mydata

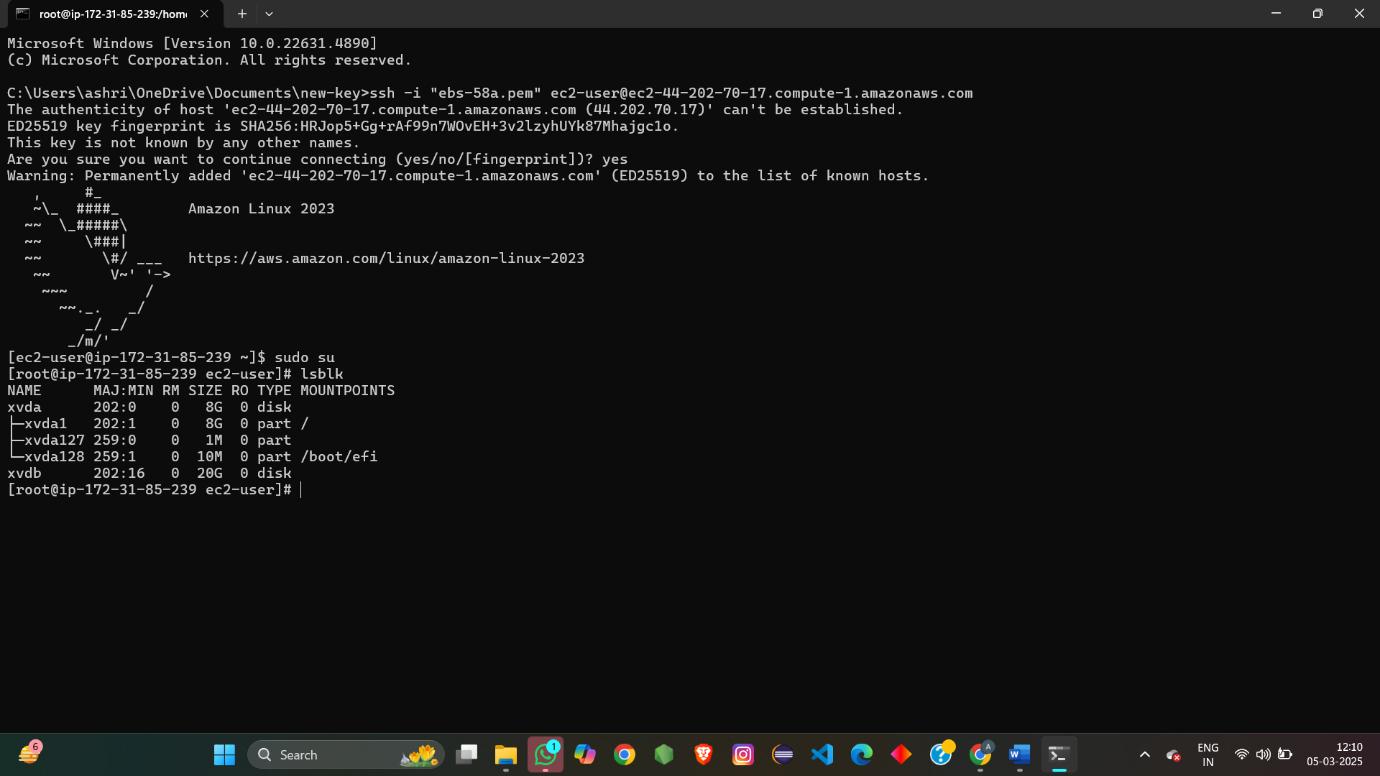
**4️ List the Text Files**

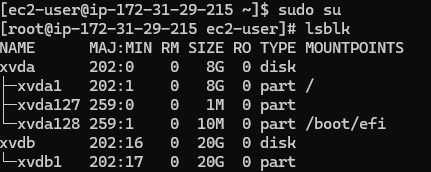
Go to the mounted directory and list the files:

cd /mnt/mydata

ls -l

If you created files like file1.txt, file2.txt, etc., they should be listed.





A screenshot of a computer

Description automatically generated