Task on EC2

1. **Launch one EC2 using Amazon Linux 2 image and add a script in user data to install Apache.**

**Project Parameters:**

* Create a t2.micro EC2 instance with the OS of your choice (Just make sure it is free tier).
* In the user-data field, use a script that updates all packages, installs Apache, and starts the Apache service.
* Verify that the instance has the Apache web server downloaded and installed through the public IP.

We’ll need to have a personal AWS account and know your way around Linux and Bash.

**Step 1: Log in to AWS**

1. Go to **AWS Management Console** → Search for **EC2** → Click **EC2 Dashboard**.
2. Click **Launch Instance**.

Step Add User Data (Apache Install Script)

Scroll down to **Advanced details → User data**.  
Paste this script:

#!/bin/bash

# Update system packages

yum update -y

# Install Apache (httpd)

yum install -y httpd

# Start Apache service

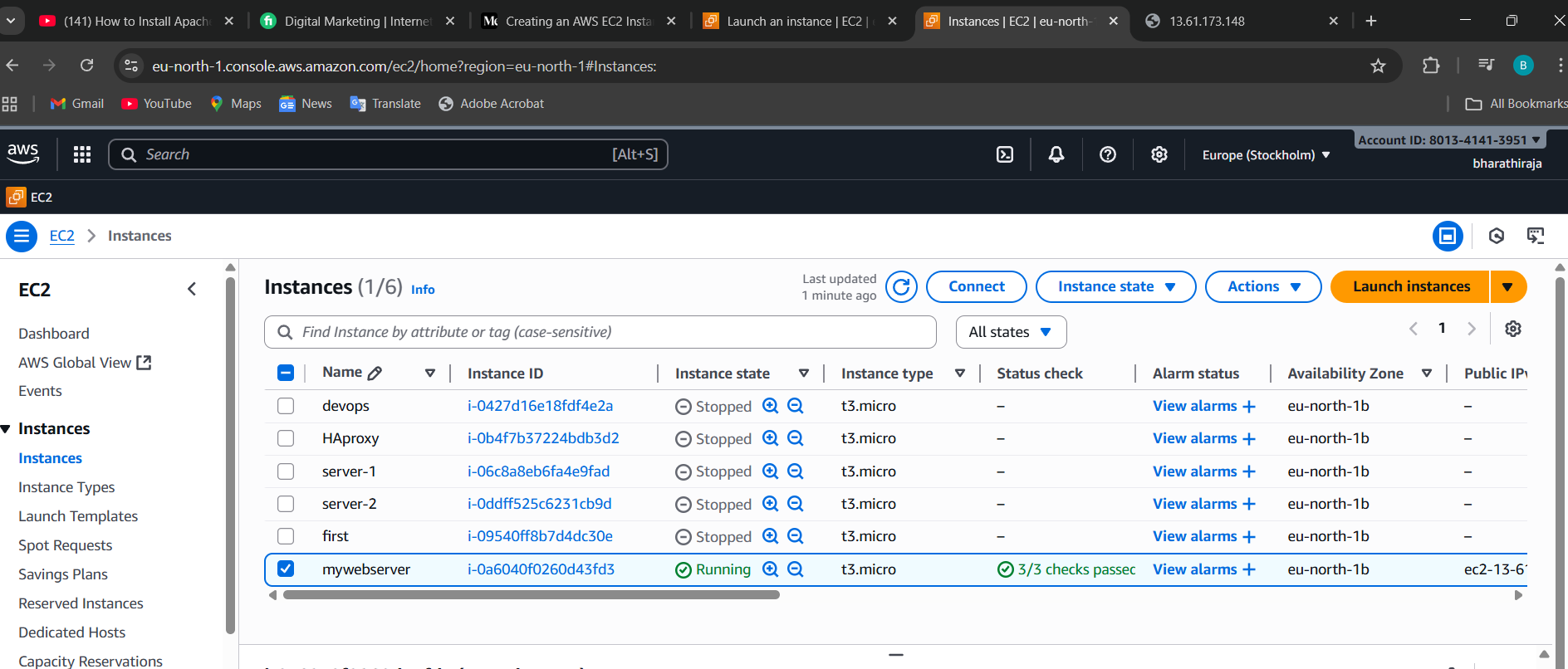
systemctl start httpd

systemctl enable httpd

# Create a sample index.html page

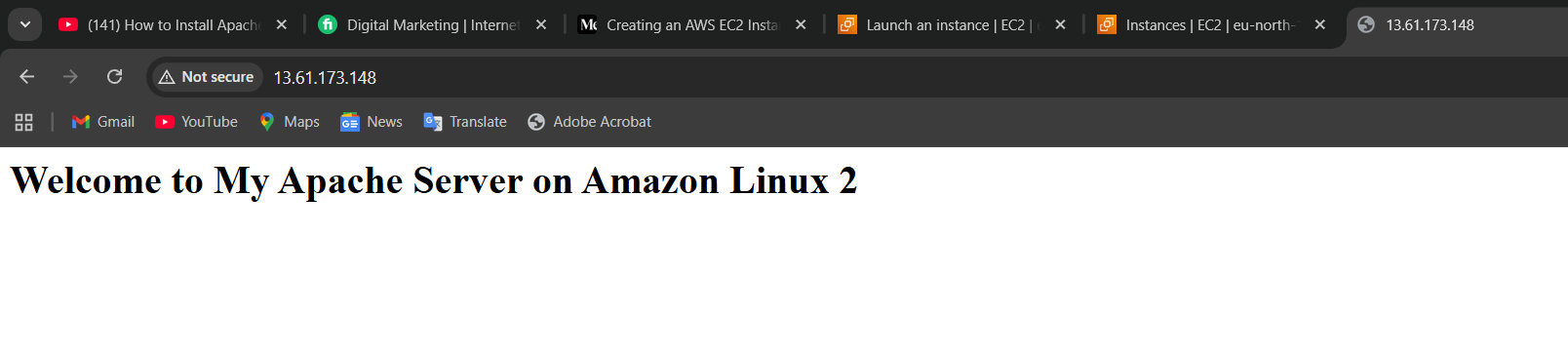
echo "<h1>Welcome to My Apache Server on Amazon Linux 2</h1>" > /var/www/html/index.html

 Verify that the instance has the Apache webserver downloaded and installed through the public IP.



**Step 5: Test Apache Server**

1. Copy the **Public IPv4 address** of your instance.
2. Open browser → http://<your-public-ip>  
   Example: http://3.110.23.45
3. You should see:  
   **“Welcome to My Apache Server on Amazon Linux 2”** 🎉

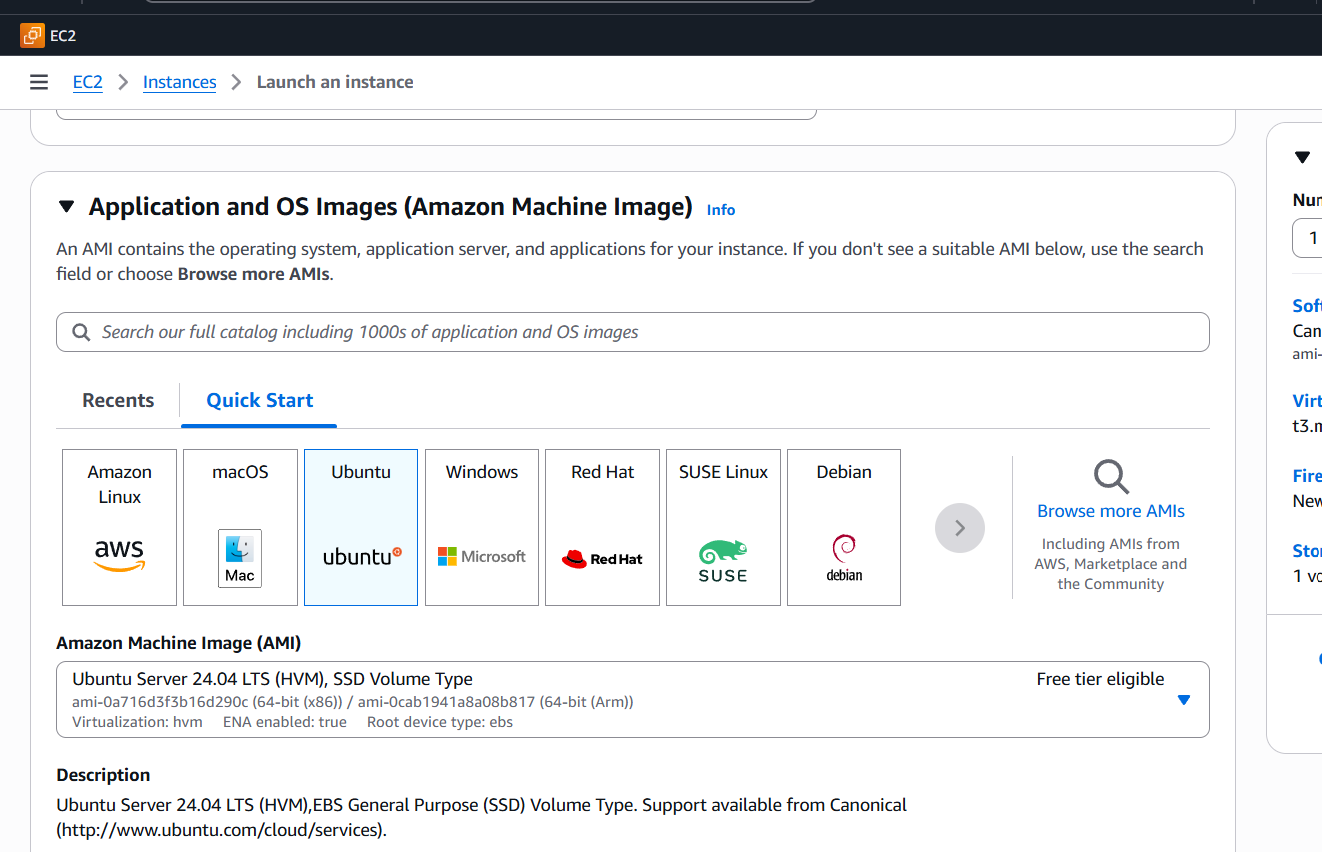


1. Launch one EC2 using the Ubuntu image and add a script in user data to install Nginx.

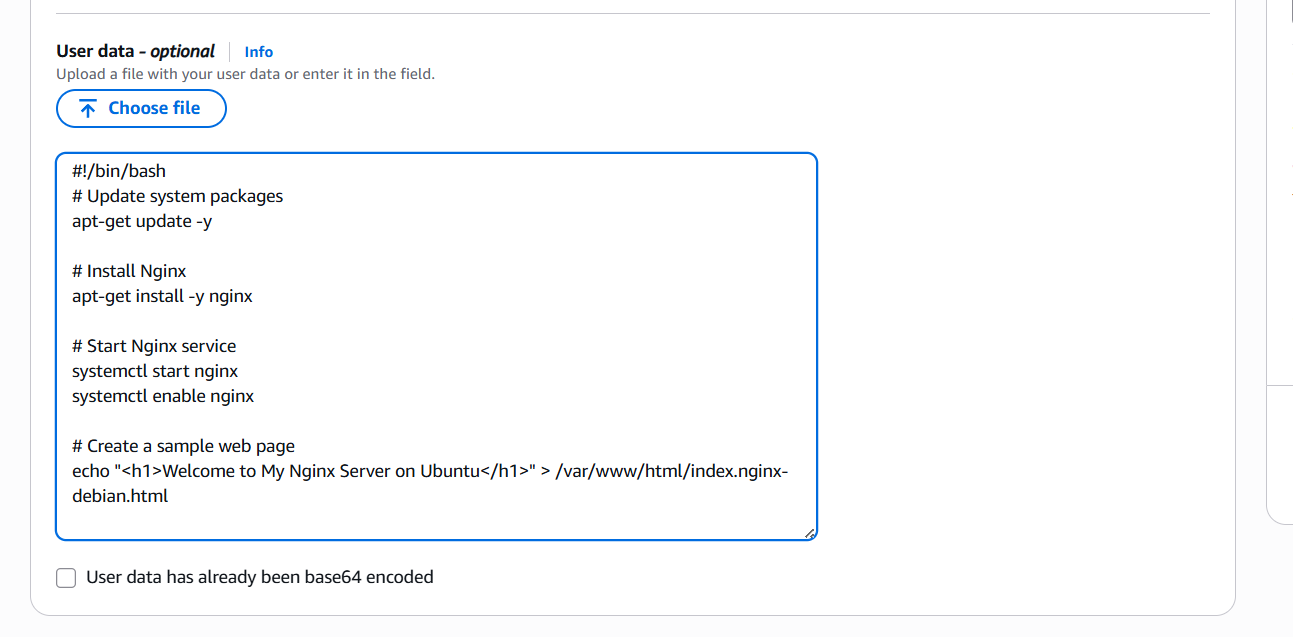
**Step 1: Log in & Launch EC2**

1. Go to **AWS Management Console** → **EC2 Dashboard** → **Launch instance**.

Fill details:



Scroll down → **Advanced details → User data**.  
Paste this script:



 Update Ubuntu package lists

 Install **Nginx**

 Start and enable Nginx (so it runs on boot)

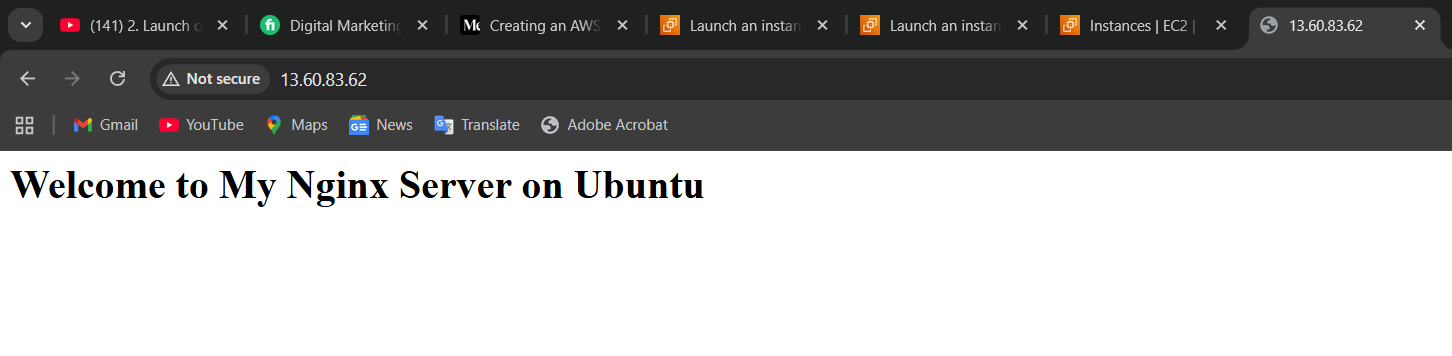
 Replace the default Nginx page with a custom welcome page

**Launch Instance**

* Click **Launch Instance**.
* Wait until status = **Running**.

**Test Nginx Web Server**

* Copy your instance’s **Public IPv4 address**.
* Open browser → http://<your-public-ip>  
  Example: http://54.210.12.34
* You should see:  
  **“Welcome to My Nginx Server on Ubuntu”**



Launch one Windows server and install Tomcat on Windows

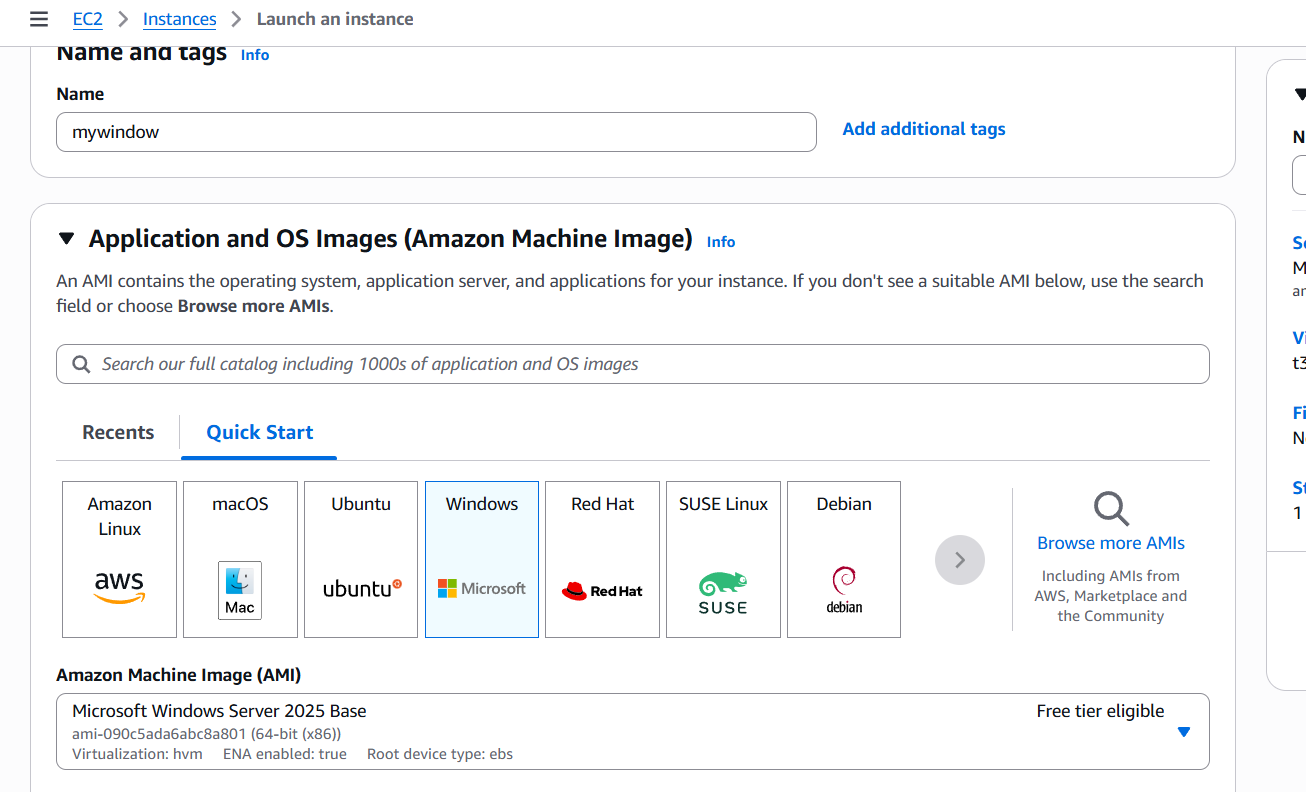
 Log in to **AWS Console** → go to **EC2 Dashboard** → click **Launch instance**.

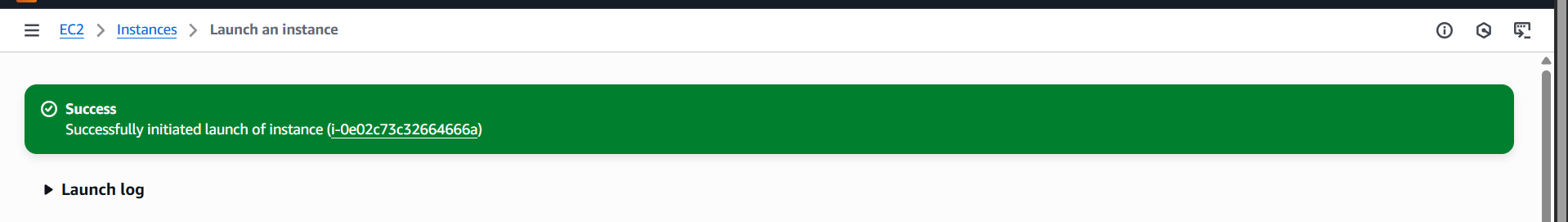
 **Name**: WindowsTomcatServer

 **AMI**: Select **Microsoft Windows Server 2019 Base** (or 2022).

 **Instance type**: Choose t2.micro (free tier eligible).

 **Key pair**: Create or select an existing one (needed to RDP into Windows).



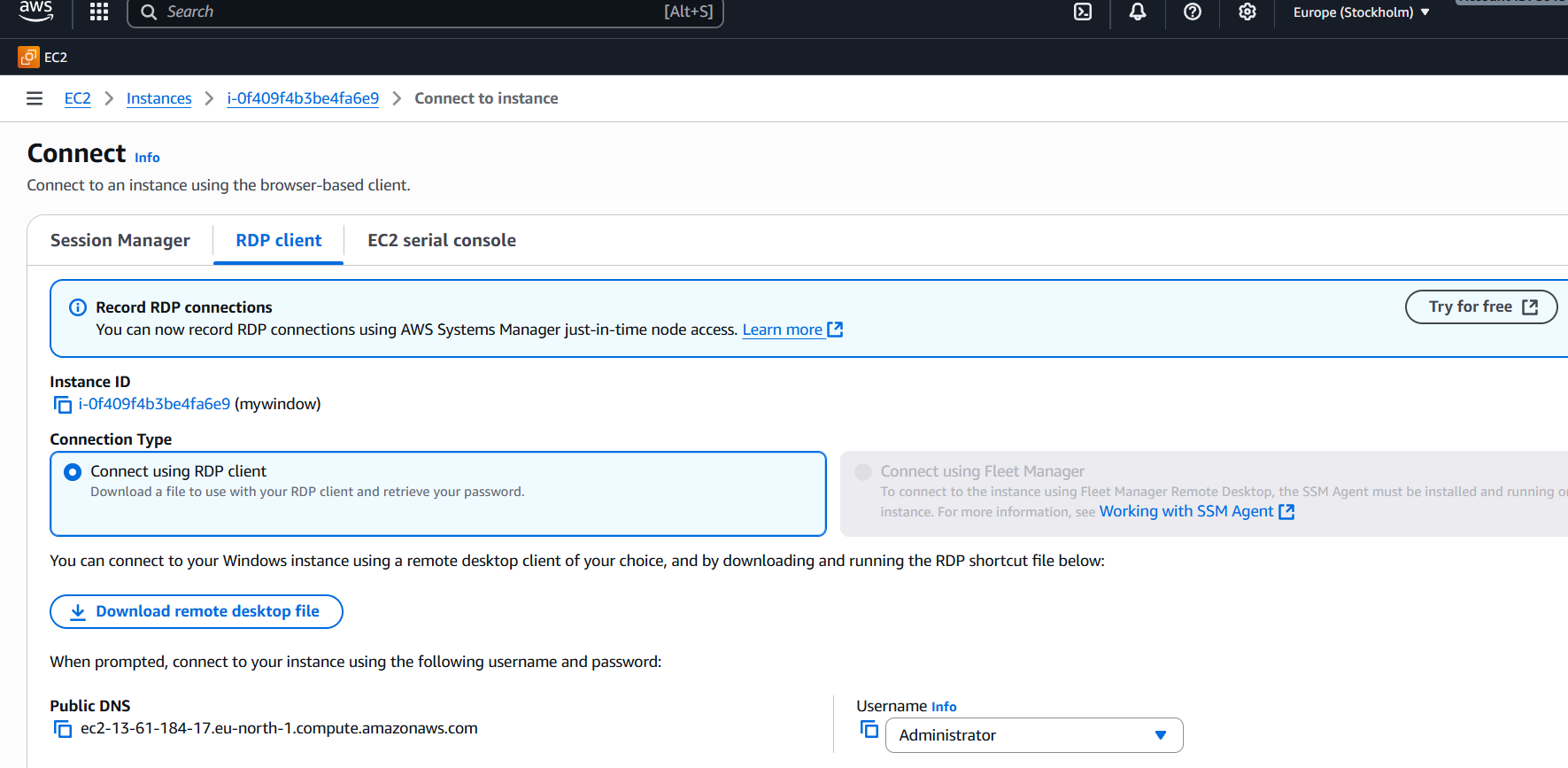


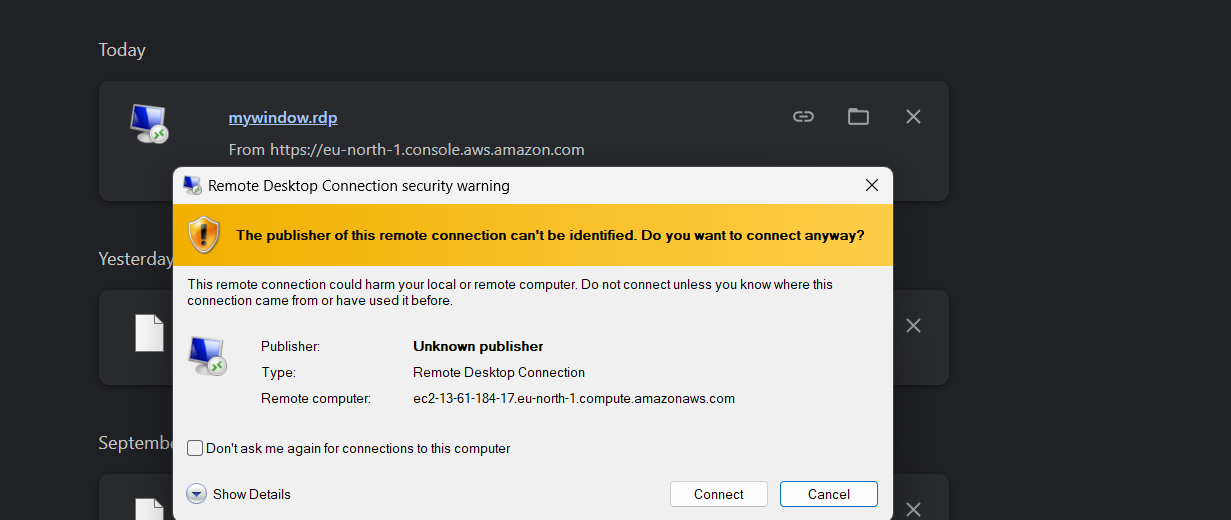
 Go to **EC2 Dashboard → Instances → Select your Windows instance**.

 Click **Connect → RDP client**.

 Download the RDP file & get **Administrator password** (decrypt with your key pair).

 Open the RDP file → log in.





Open browser inside your PC (not inside Windows VM) →

**Step 7: Connect to Windows Instance**

1. Select your instance → click **Connect** (top-right).
2. Choose **RDP Client** tab.
3. Click **Download Remote Desktop File**.
4. Click **Get Password** (you need the key pair .pem file you downloaded earlier).
   * Upload the key → AWS will decrypt and show **Administrator password**:
   * **Username**: Administrator
   * **Password**: (the one you just decrypted)

**Verify Windows Server**

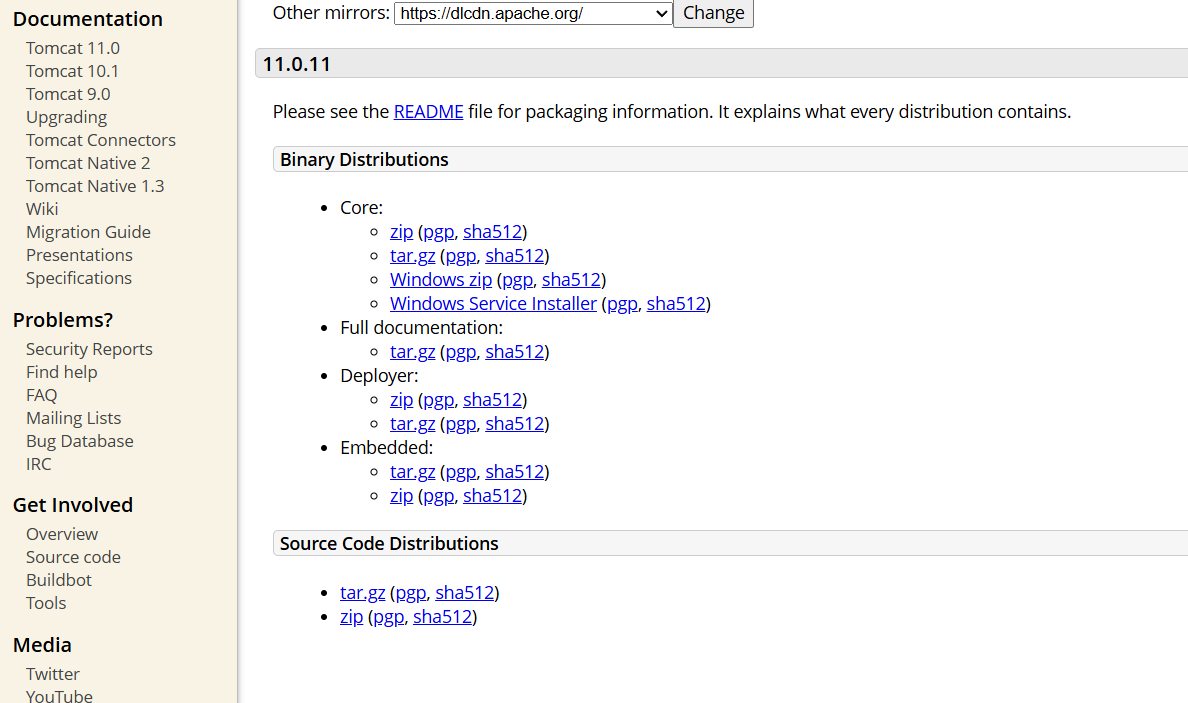
Once logged in, you’ll see a normal Windows desktop inside your EC2 instance 🎉.  
From here you can:

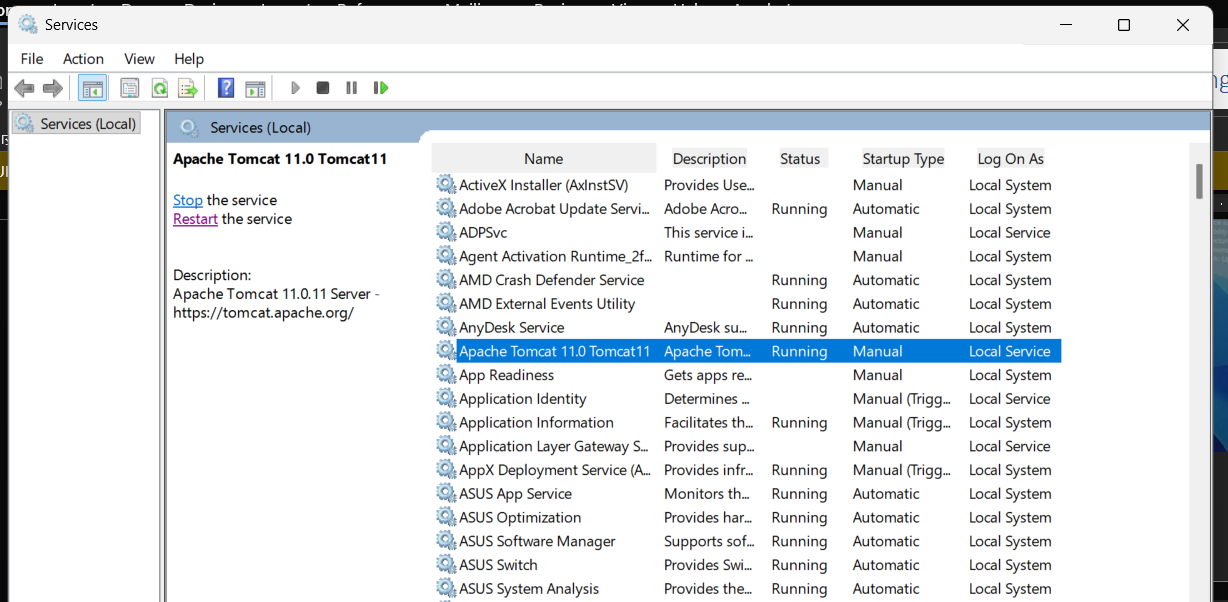
Open **Server Manager**

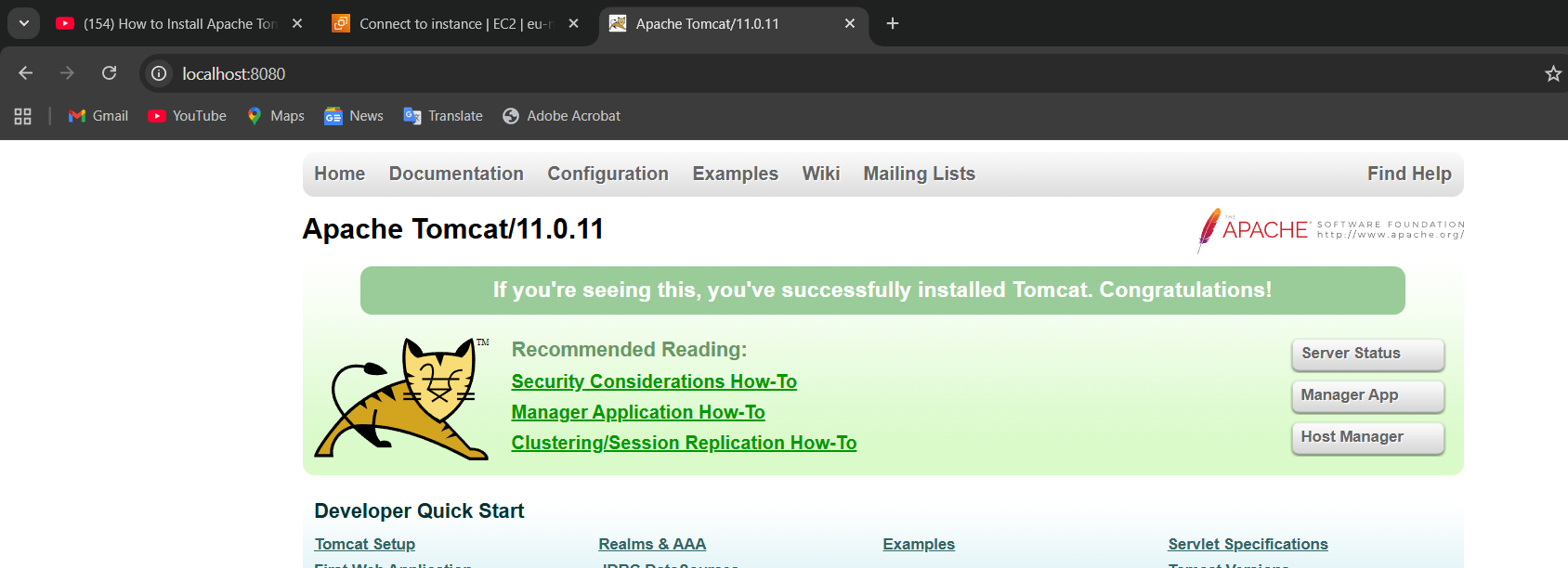
Install software (like Tomcat, Java, etc.)

Configure your server as needed

* + Install java and then tomcat also



* + First, go to the service setting there, Tomcat, click restart, then
  + Go to browser (local:8080)it will work
  + 



4.Take a snapshot of the instance created in Task 1.

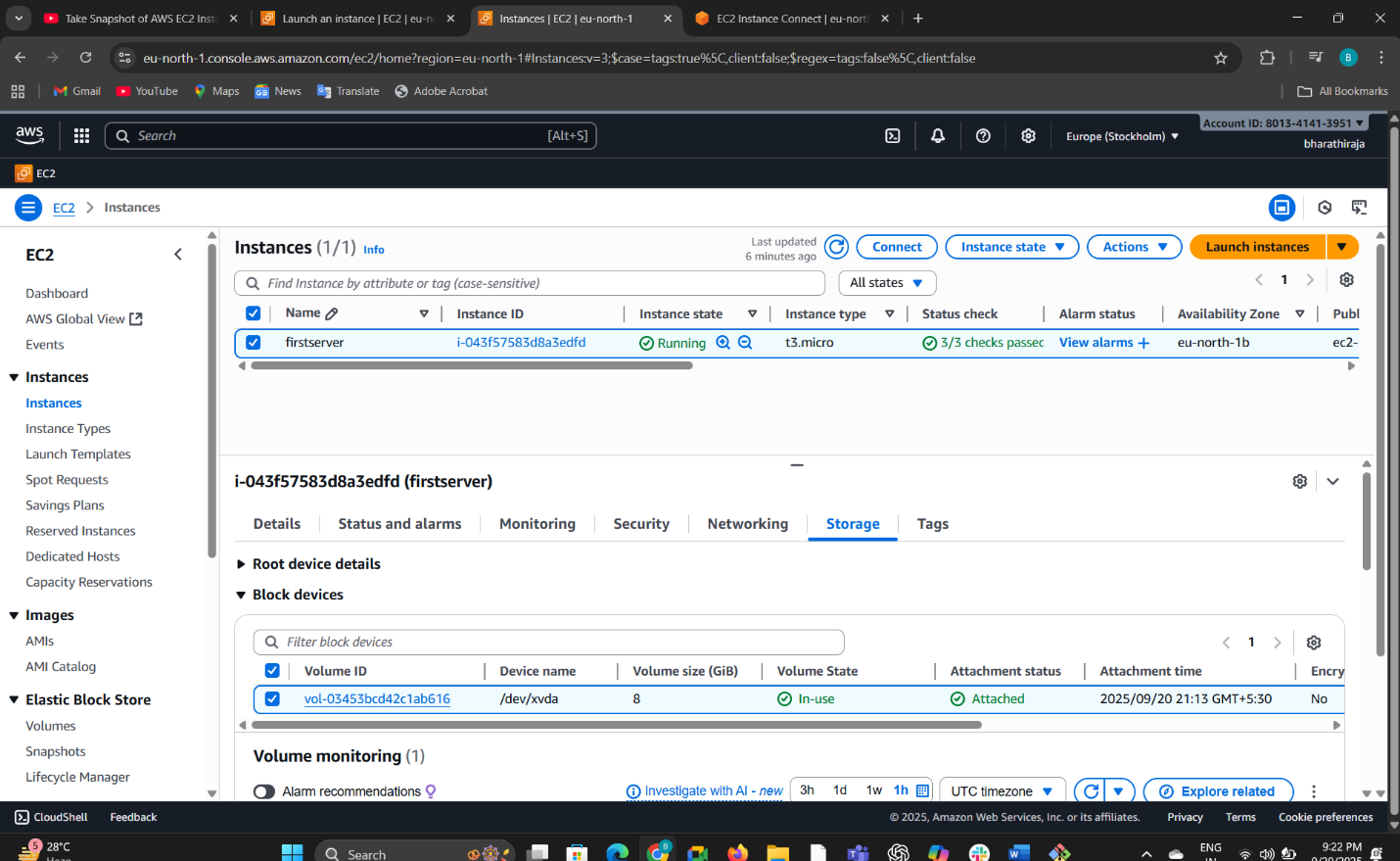
First => Sign in to the AWS Management Console → **EC2**.

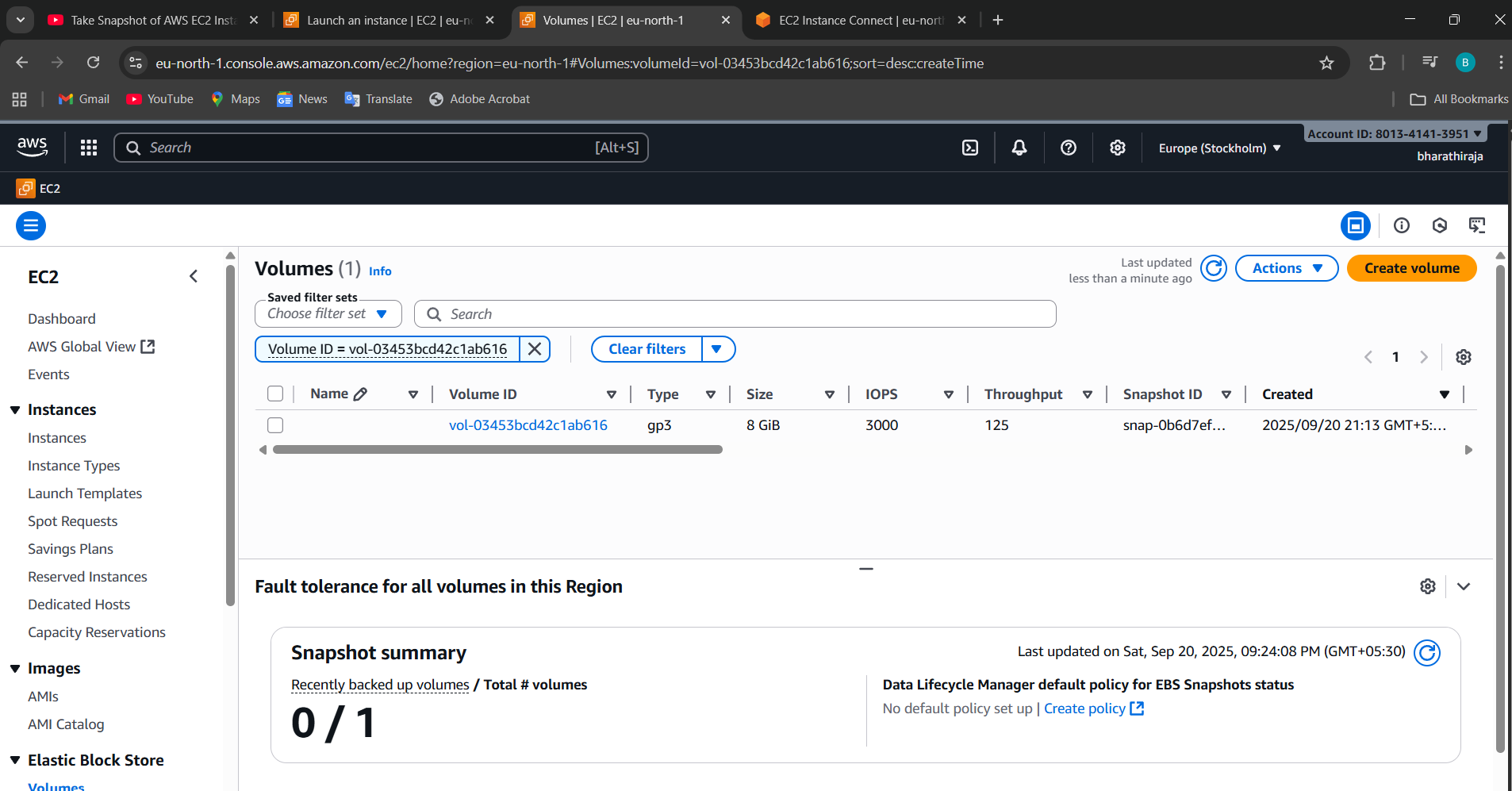
 A **snapshot** in AWS is a **backup of an EBS volume** (Elastic Block Store volume).

 Think of it like taking a photo of your disk at a particular moment — you can later **restore it** or **make a new volume** from it.

 Snapshots are **incremental** → after the first full backup, only changes are saved.

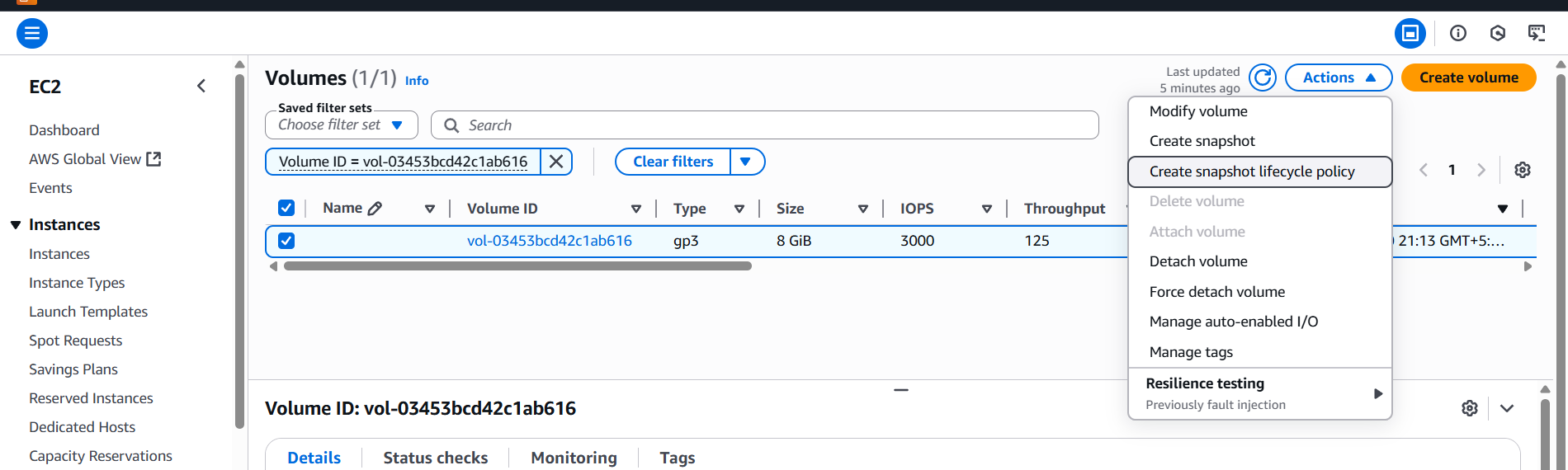
 Stored in **Amazon S3 internally** (you don’t see it in S3, but AWS manages it).  
click the storage, you will get the volume ID. Click that one

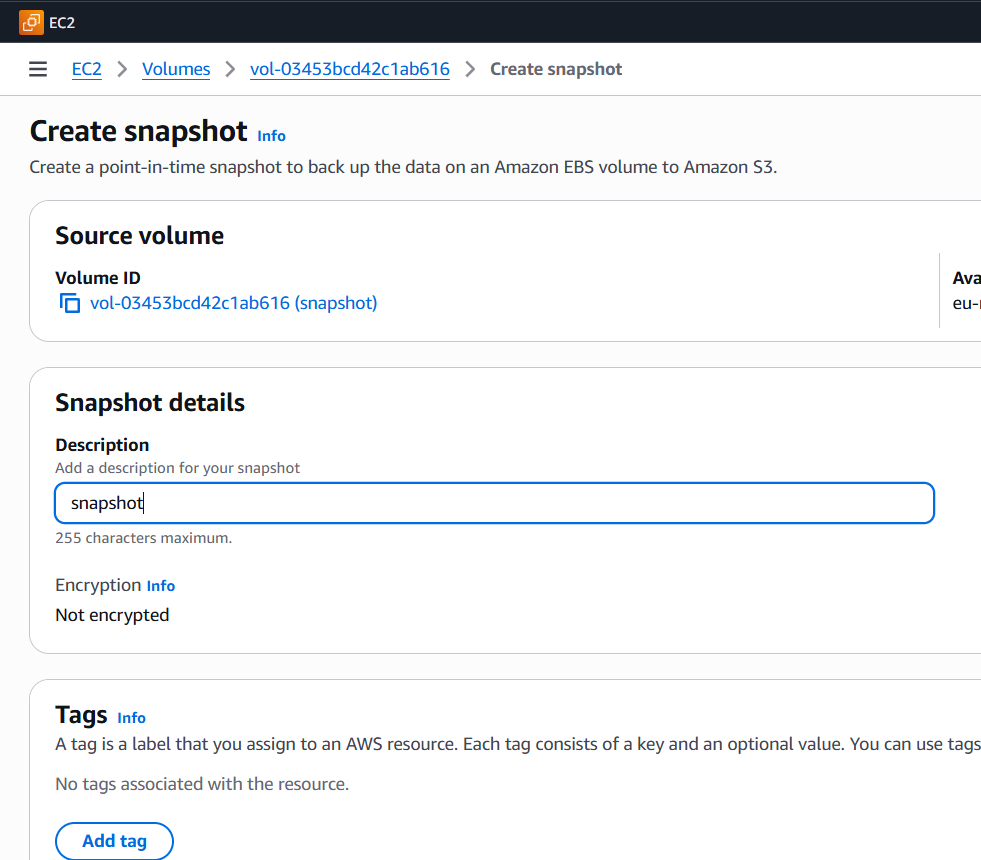
First, open the EC2 instance

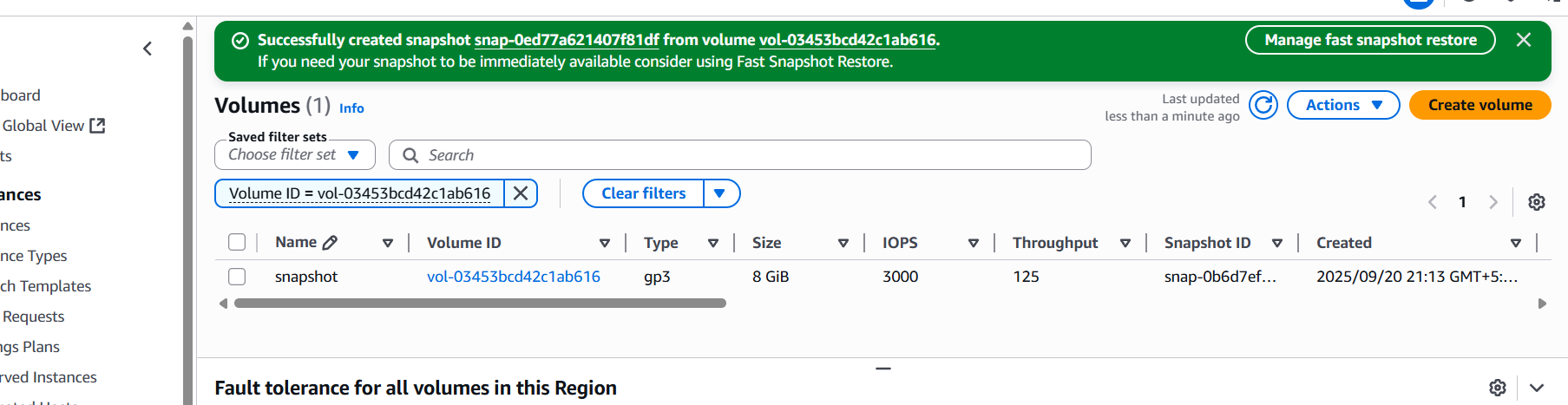


You see if you click the volume ID

Click the under or the box and go to action, is create snapshot





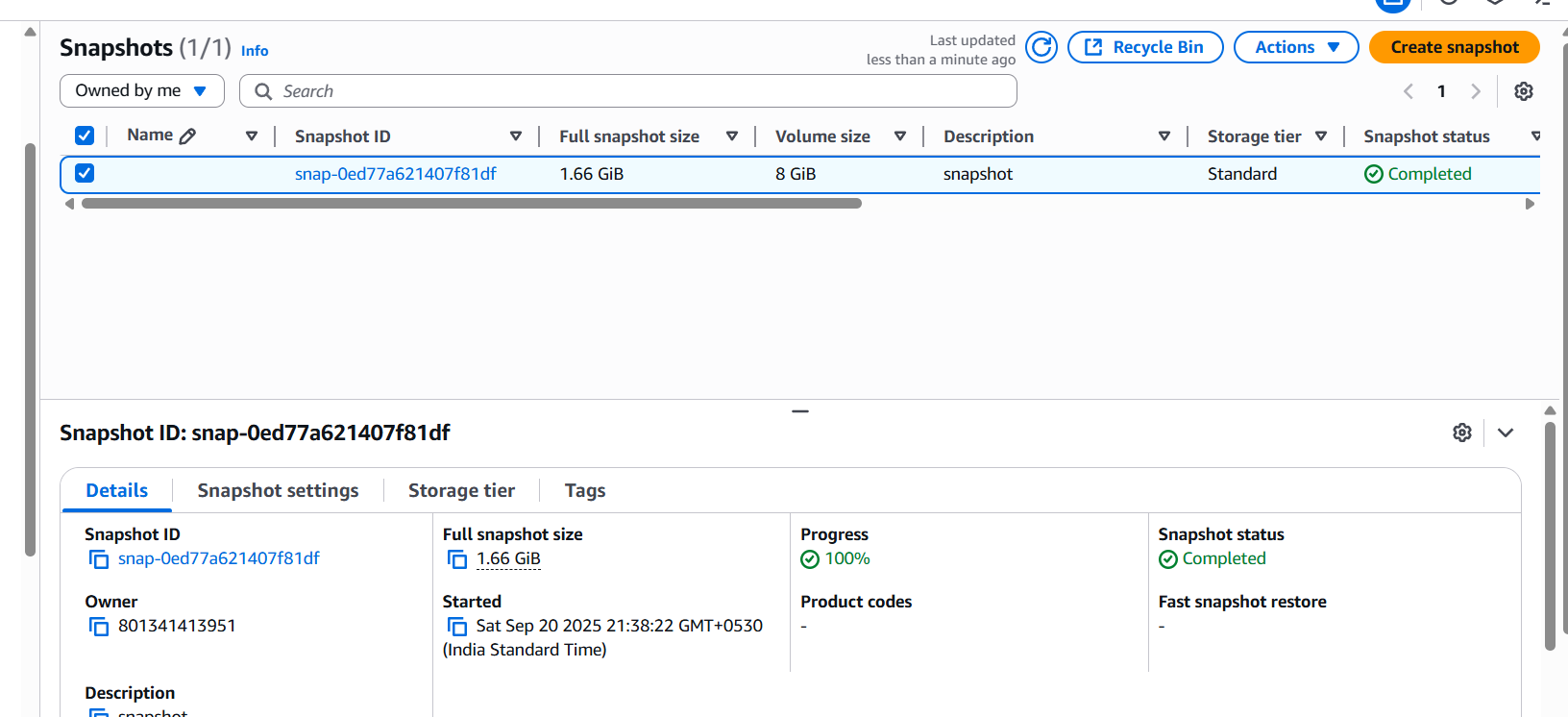


**Why do we use Snapshots?**

**Backup & Recovery**

1. If your instance crashes or data is lost, you can restore the volume from a snapshot

This is what I created a snapshot

Now snapshot is ready whatever we can get ec2

 **Create new volumes in same/different AZ**

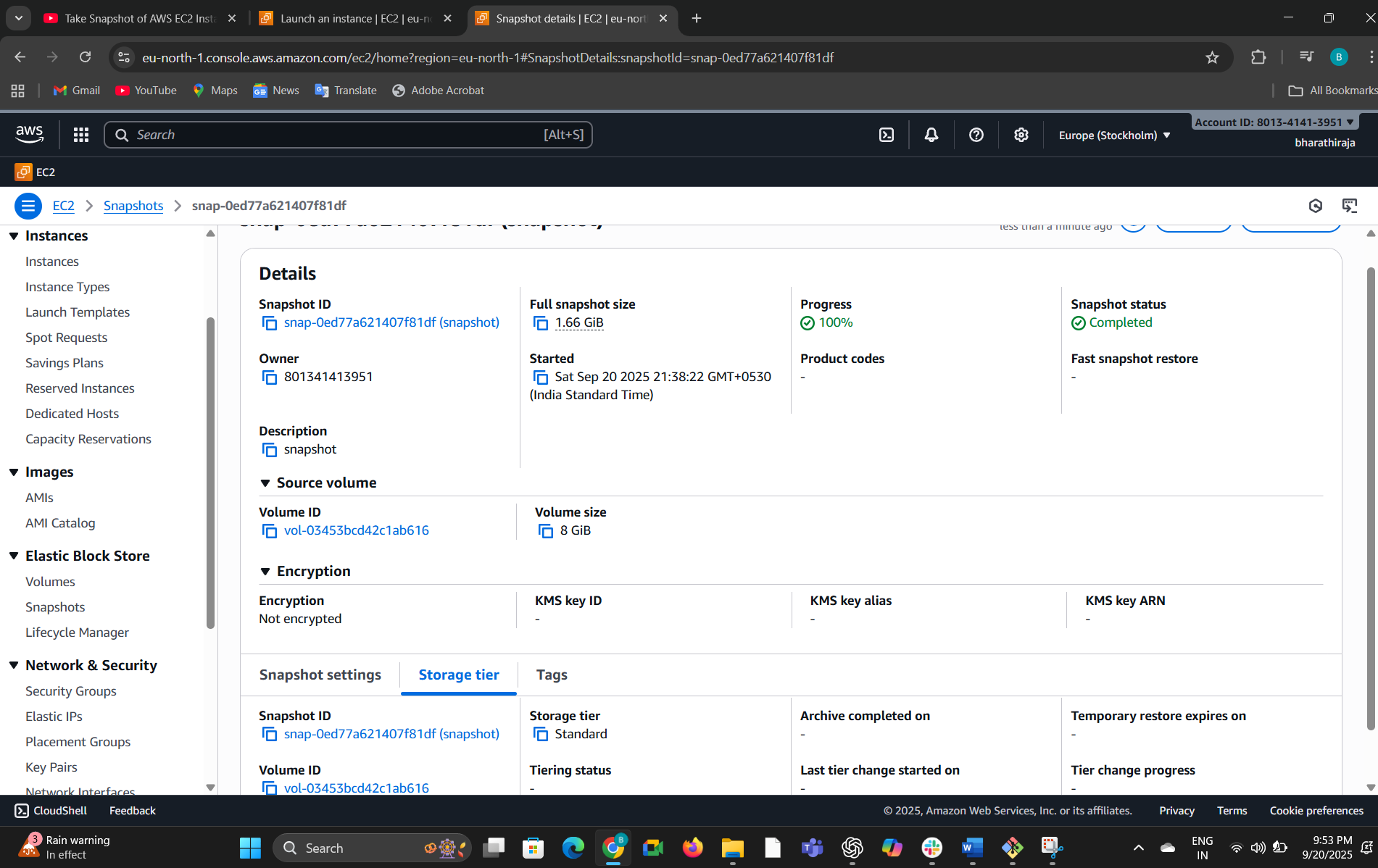
* You can copy snapshots across regions for **disaster recovery**.

 **Create AMI (Amazon Machine Image)**

* When you create an AMI, AWS automatically makes snapshots of the root volume (and other attached volumes).

 **Migration & Cloning**

* You can launch another instance with the same data by restoring from a snapshot.

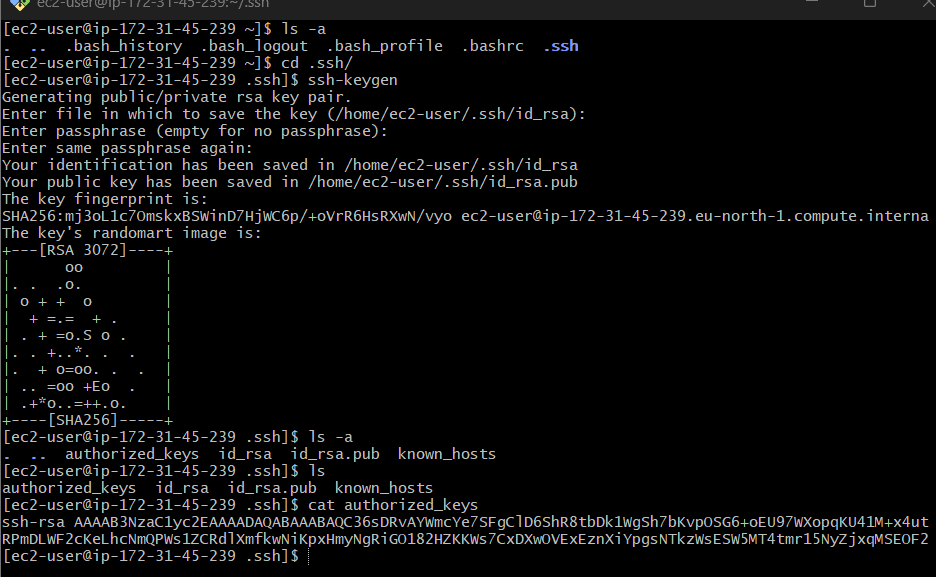


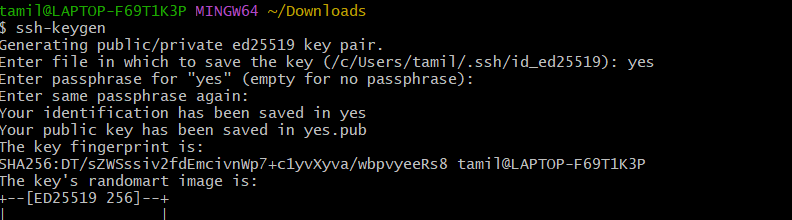
 **Snapshot = Backup of EBS volume.**

 **Why = Safety, recovery, migration, cloning.**

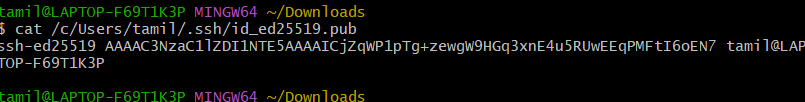
 **How = Console (easy) or CLI (automation).**

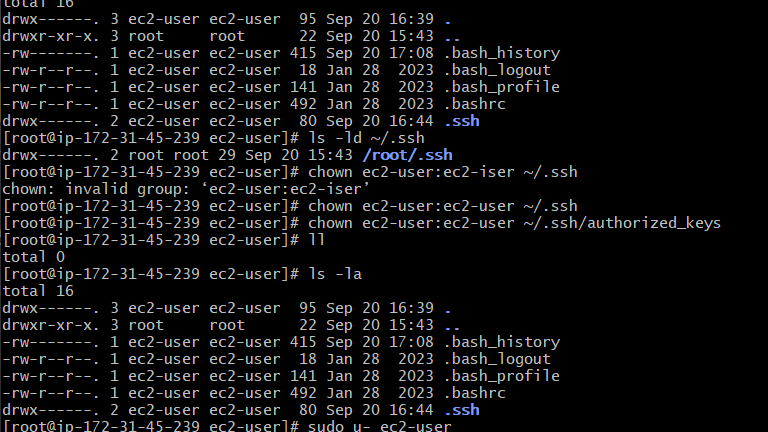
1. Assign passwordless authentication for the EC2 created in Task 2.





And copy the public key:



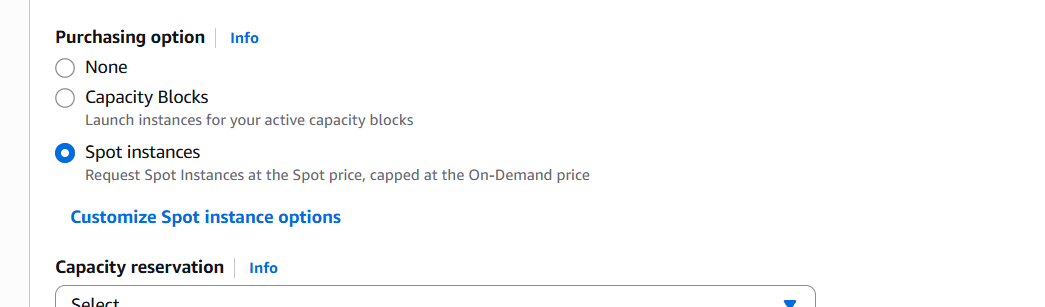


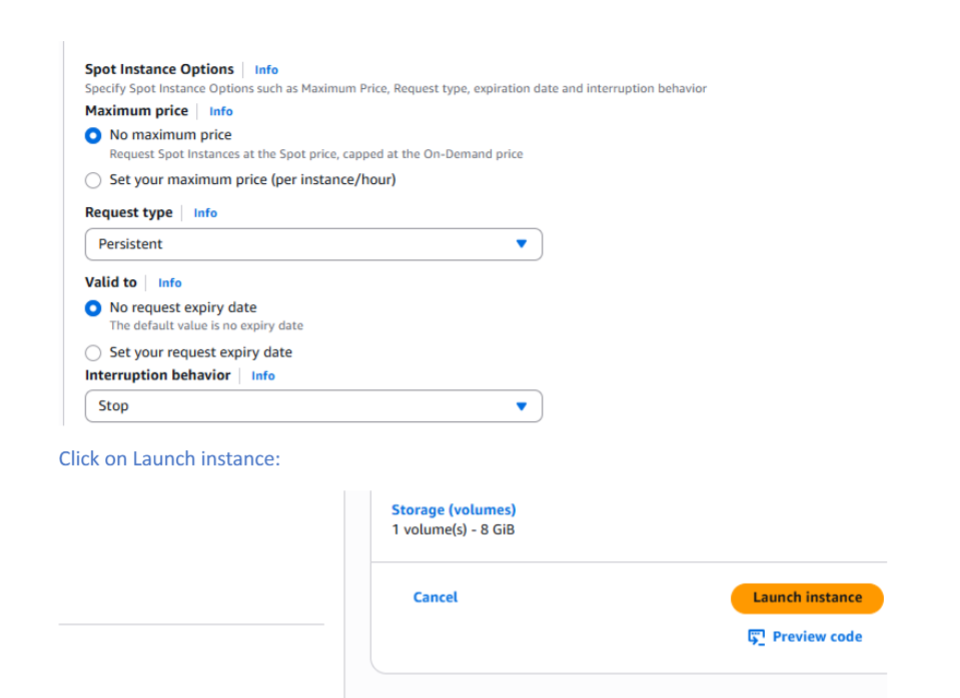


1. Launch any EC2 using the spot purchasing option.

**What is a Spot Instance in AWS?**

* **Spot Instance** = An **unused EC2 capacity** in AWS offered at a **much lower price** (up to 90% cheaper than On-Demand).
* But: AWS can **interrupt** (stop/terminate) the instance when they need the capacity back → you get a **2-minute warning**.
* **Cost Saving** – Best for workloads where cost is more important than guaranteed uptime.





the created spot instance we can see in spot Request:



 **Purpose:** Save money (up to 90%).**Why use:** Best for non-critical, fault-tolerant workloads

1. Enable termination policy on the EC2 created in Task 2.

**What is Termination Protection?**

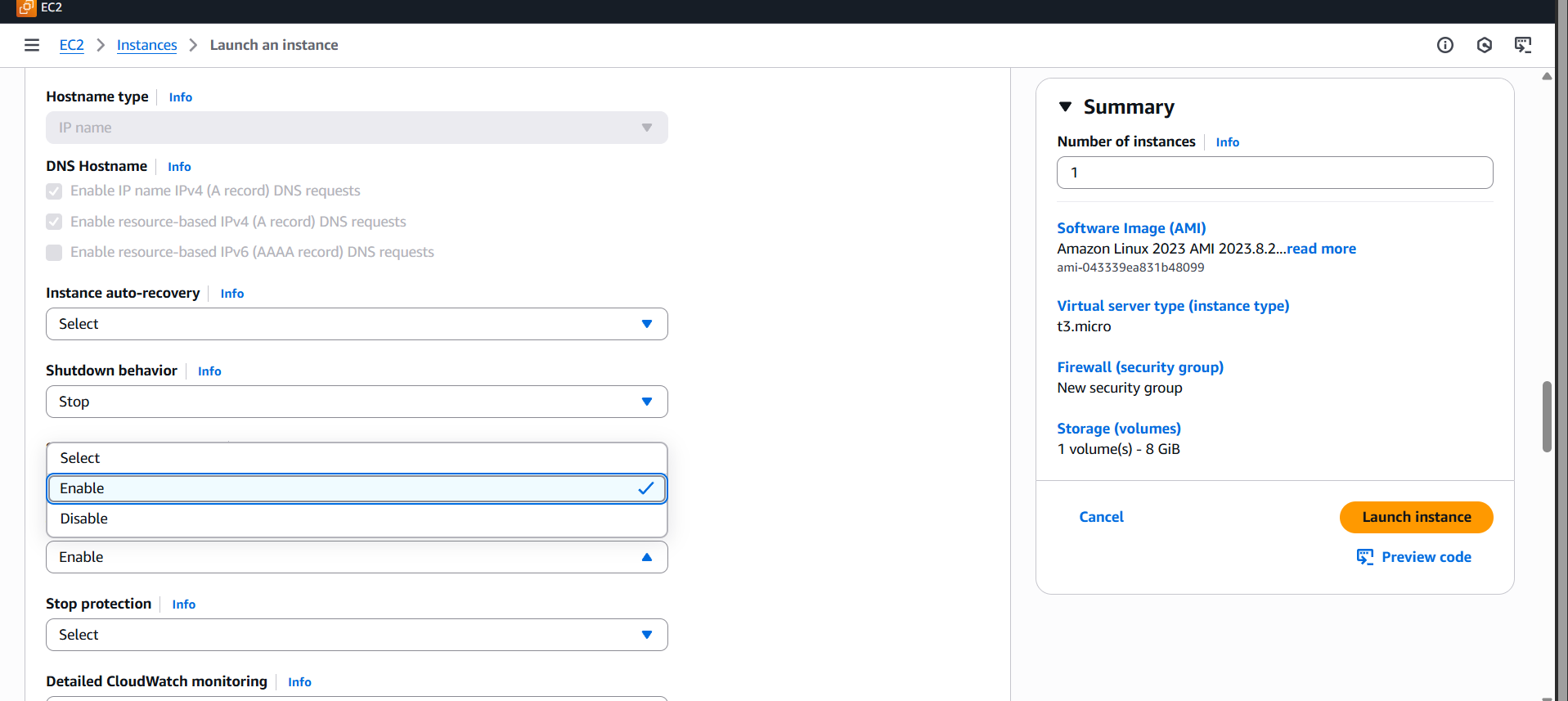
* By default, any EC2 instance can be **terminated** (deleted) from the AWS Console, CLI, or API.
* **Termination protection** prevents **accidental termination** of your EC2 instance.
* Even with this enabled, you can **still stop/restart** the instance, but you **cannot terminate** it unless you disable protection first.

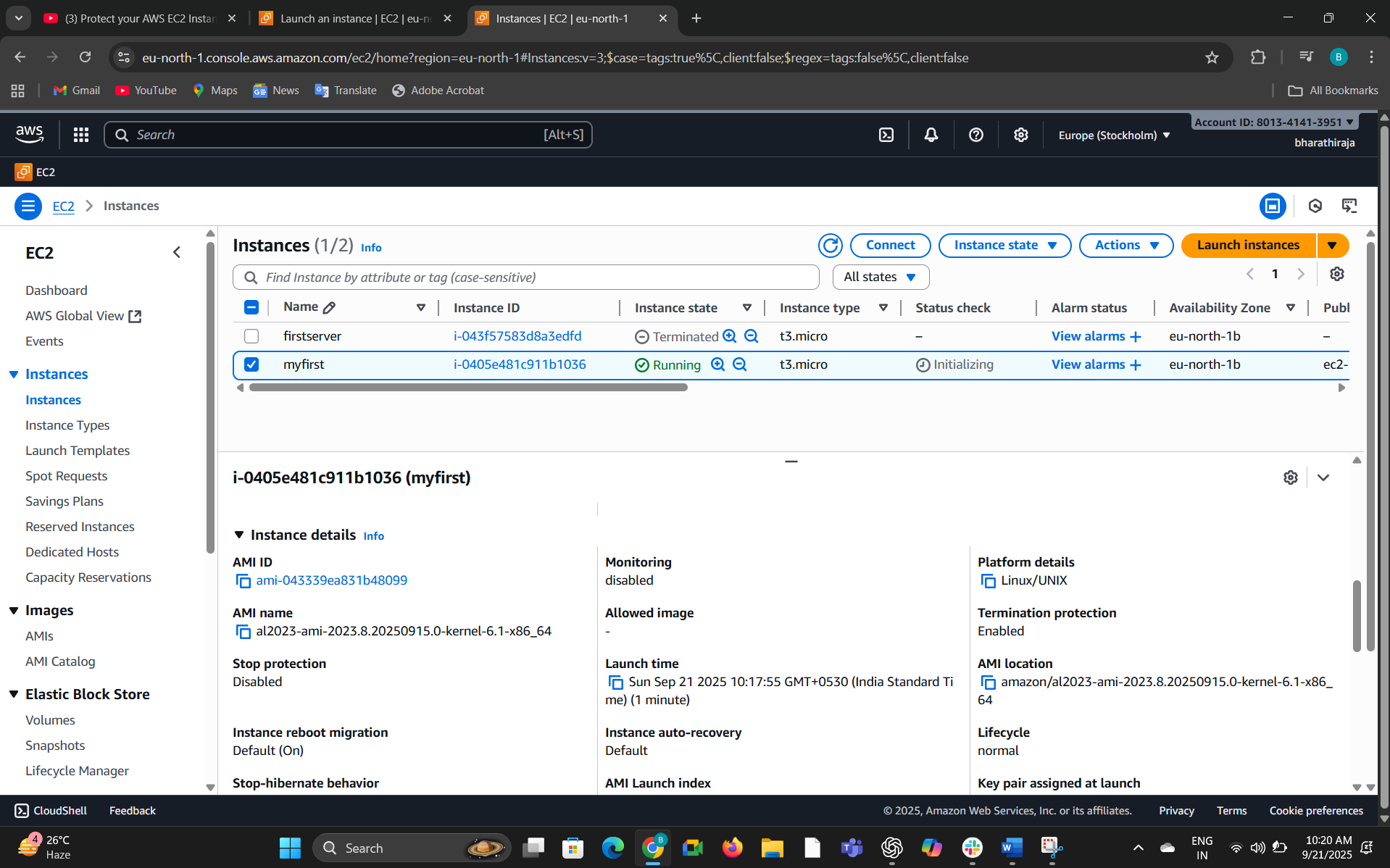
**EC2 instance** you created.  
 From the **Actions menu** (top), choose:  
**Instance settings → Change termination protection**.

A popup appears → Select **Enable**.

Click **Save**.

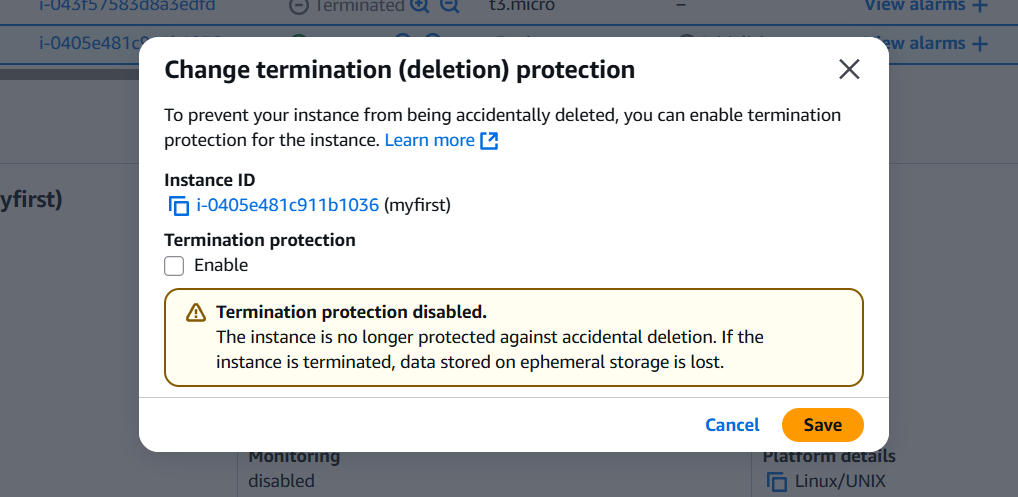
Advance option and click enable option

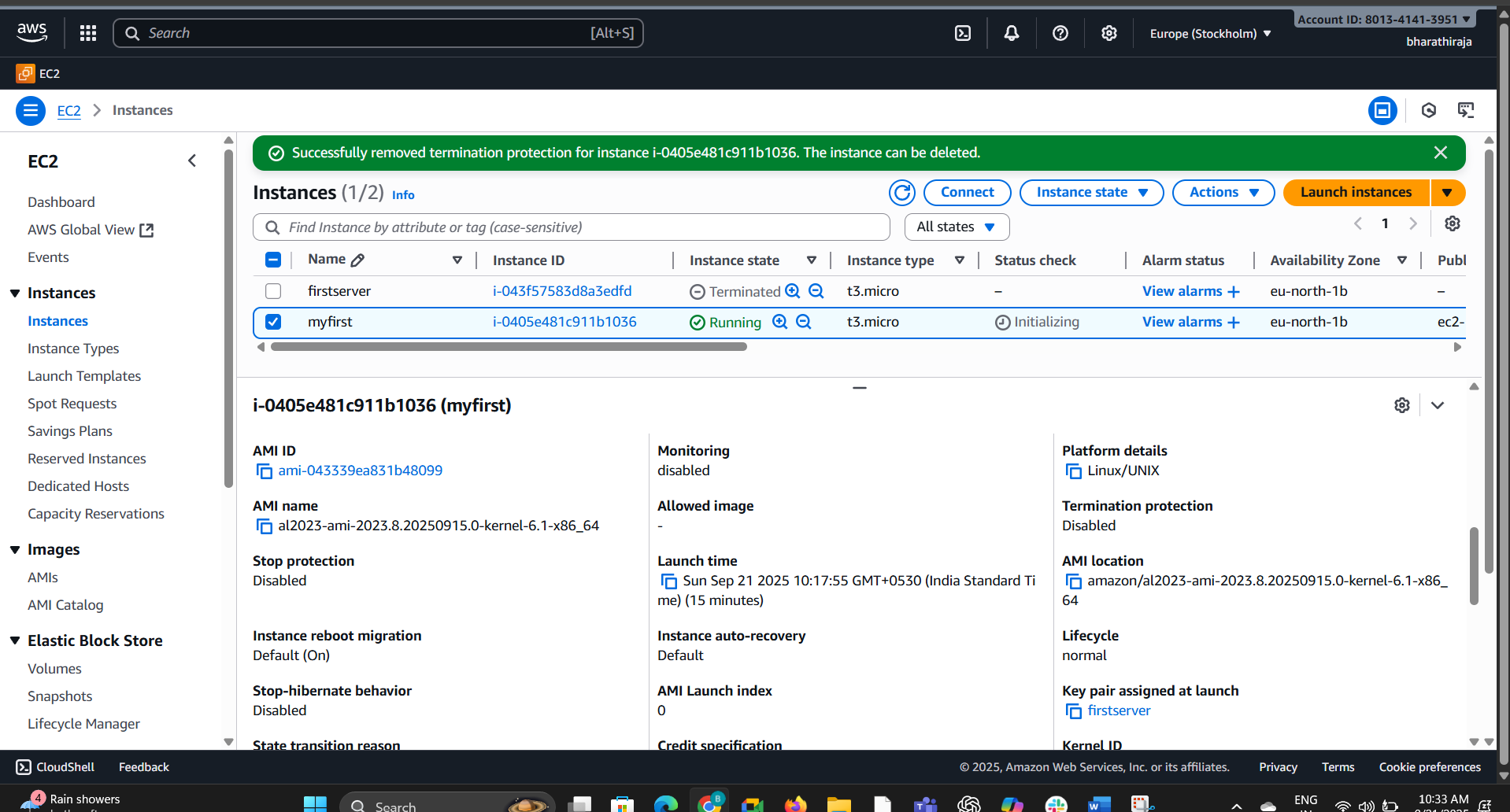






If you don’t want to enable, you can disable the option



  
We can see here I disabled the option

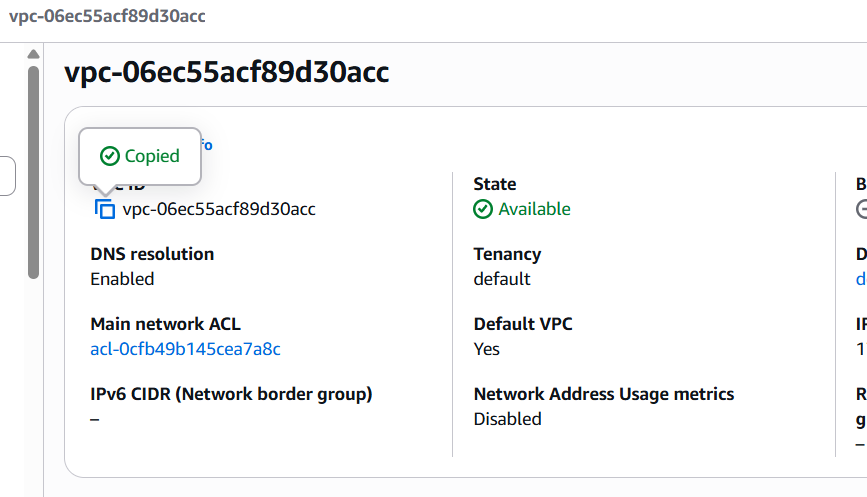


1. Launch one EC2 using AWS CLI.

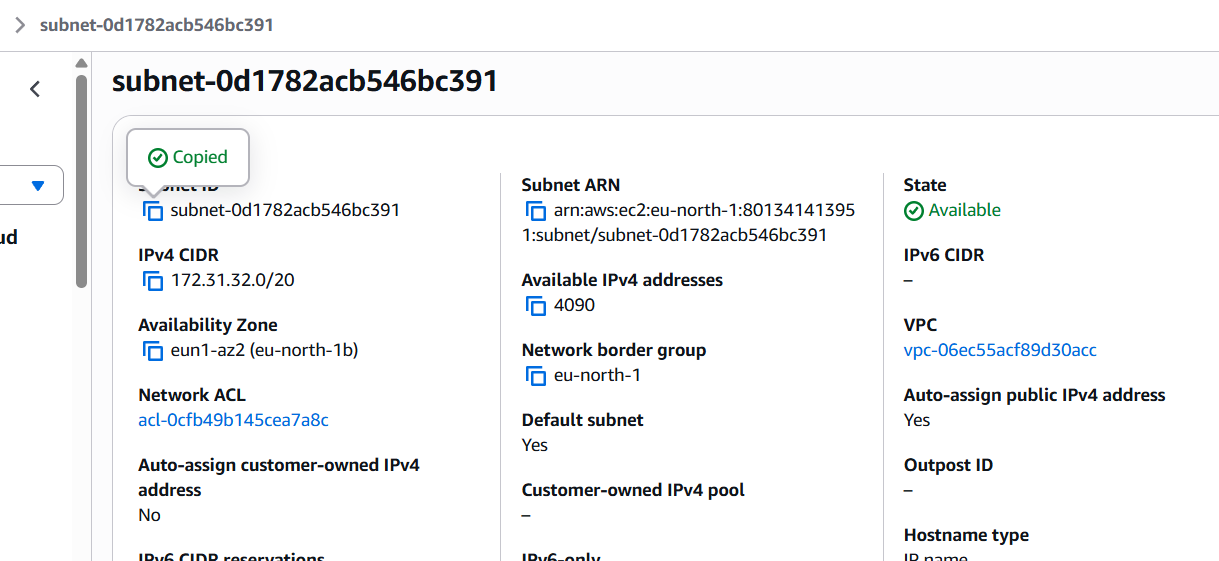
**What it means**

Normally, you launch an **EC2 instance** using the **AWS Management Console** (web UI).  
But here, the question wants you to **launch it using AWS CLI (Command Line Interface)** instead of clicking in the console.

To create an ec2 instance using CLI, you need the following.

1. Security group ID
2. Key pair name
3. AMI Id
4. Subnet ID
5. **VPC ID:** To create a security group
6. **One** **Subnet ID:**To launch ec2 instance.
7. Vpc : vpc-06ec55acf89d30

subnet-0d1782acb546bc391



AMI ID could be a base image AMI Id or ID of a custom image created by you or your team.

To get the AMI Id, Go to ec2 Dashboard --> AMI Catalog

Ami catlog 043339ea831b48099

