**EXPERIMENT-07**

**NAME:** SHAIKH MUBASHIRA TUFEL AHMED

**ROLL NO:** 612055 **COURSE:** ADVANCE DEVOPS(ITL504)

**BRANCH:** T.E. INFORMATION TECHNOLOGY (SEM 5)

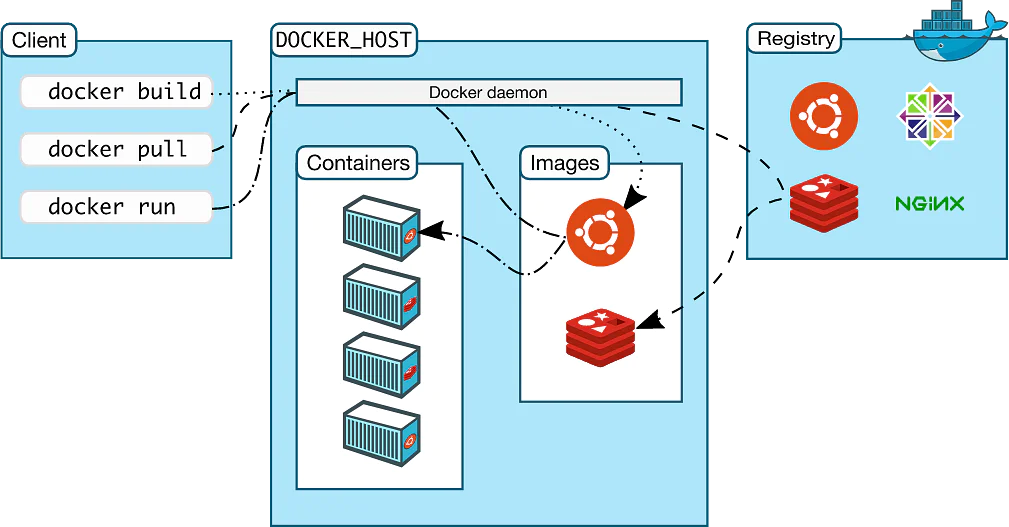
1. **What is Containerization / Docker? Explain Docker Architecture with the help of diagram.**

🡺Docker is an open-source platform that enables developers to build, deploy, run, update and manage containers—standardized, executable components that combine application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

[Containers](https://www.ibm.com/cloud/learn/containers) simplify development and delivery of distributed applications. They have become increasingly popular as organizations shift to [cloud-native](https://www.ibm.com/cloud/learn/cloud-native) development and hybrid [multicloud](https://www.ibm.com/cloud/learn/multicloud" \t "_blank) environments. It’s possible for developers to create containers without Docker, by working directly with capabilities built into Linux and other operating systems. But Docker makes [containerization](https://www.ibm.com/cloud/learn/containerization) faster, easier and safer.

**Docker Architecture:**

Docker follows Client-Server architecture, which includes the three main components that are **Docker Client**, **Docker Host**, and **Docker Registry**.

****

**1. Docker Client**

🡪Docker client uses **commands** and **REST APIs** to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

🡪Docker Client uses Command Line Interface (CLI) to run the following commands -

* docker build
* docker pull
* docker run

**2. Docker Host**

🡪Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

**3. Docker Registry**

Docker Registry manages and stores the Docker images.

There are two types of registries in the Docker -

* **Pubic Registry -** Public Registry is also called as **Docker hub**.
* **Private Registry -** It is used to share images within the enterprise.

**Docker Objects**

There are the following Docker Objects -

**🡺Docker Images**

Docker images are the **read-only binary templates** used to create Docker Containers. It uses a private container registry to share container images within the enterprise and also uses public container registry to share container images within the whole world. Metadata is also used by docket images to describe the container's abilities.

🡺**Docker Containers**

Containers are the structural units of Docker, which is used to hold the entire package that is needed to run the application. The advantage of containers is that it requires very less resources.

In other words, we can say that the image is a template, and the container is a copy of that template.

**🡺Docker Networking**

Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers -

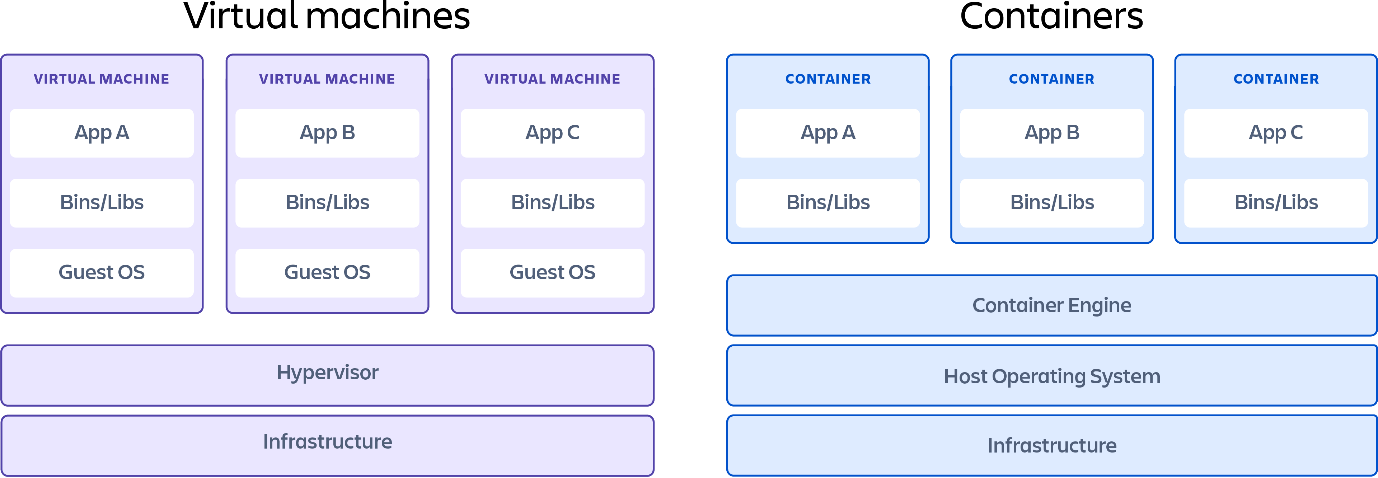
* **Bridge -** Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.
* **Host -** It is used when we don't need for network isolation between the container and the host.
* **None -** It disables all the networking.
* **Overlay -** Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.
* **Macvlan -** Macvlan is used when we want to assign MAC addresses to the containers.

**🡺Docker Storage**

Docker Storage is used to store data on the container. Docker offers the following options for the Storage -

* **Data Volume -** Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.
* **Directory Mounts -** It is one of the best options for docker storage. It mounts a host's directory into a container.
* **Storage Plugins -** It provides an ability to connect to external storage platforms.

1. **Compare Containers vs VMs.**



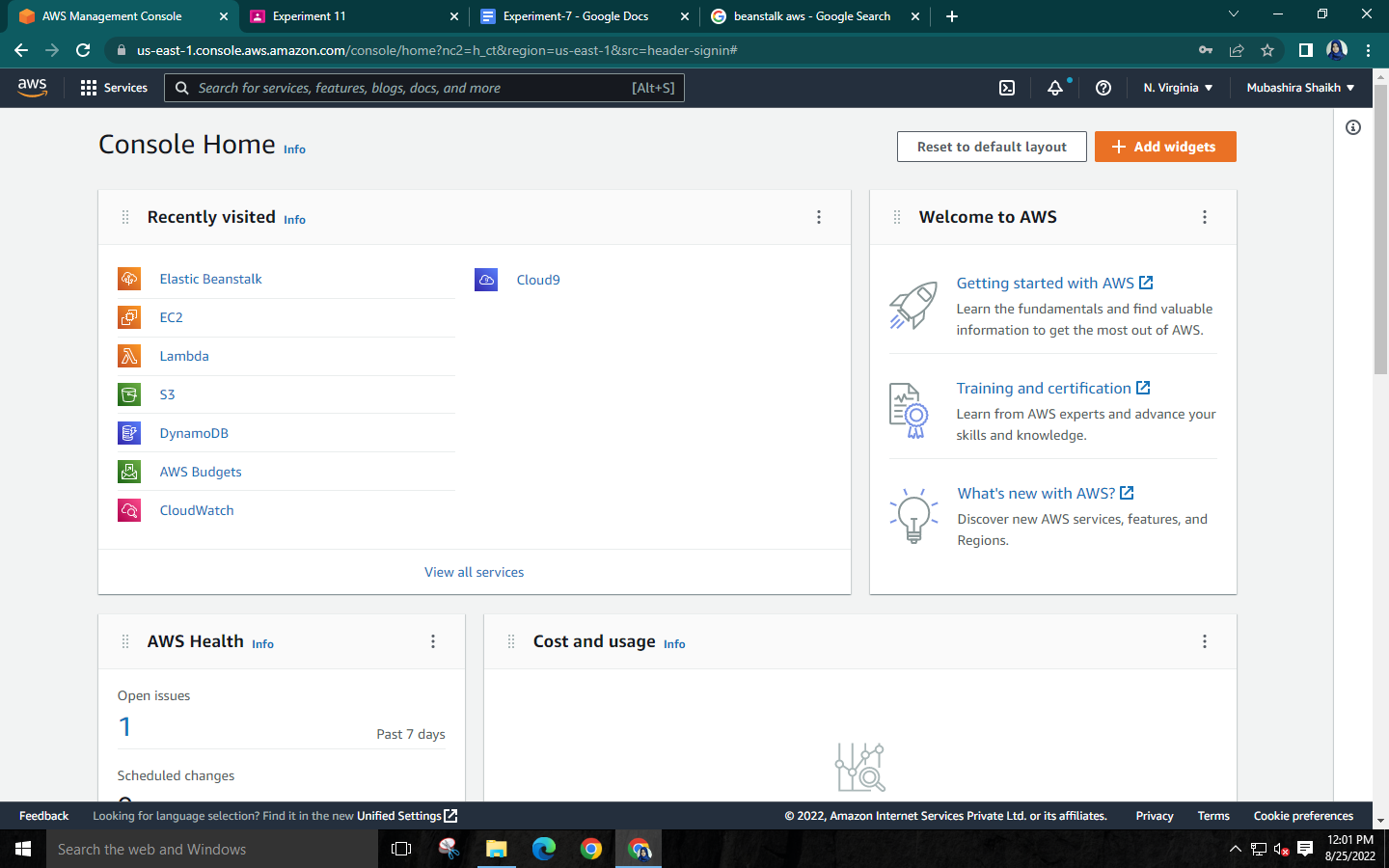
* + A virtual machine (VM) is a virtual environment that functions as a virtual computer system with its own CPU, memory, network interface, and storage, created on a physical hardware system (located off- or on-premises).
  + Containerization and virtualization are similar in that they both allow for full isolation of applications so that they can be operational in multiple environments. Where the main differences lie are in size and portability.
  + The key differentiator between containers and virtual machines is that virtual machines virtualize an entire machine down to the hardware layers and containers only virtualize software layers above the operating system level.
  + VMs are the larger of the two, typically measured by the gigabyte and containing their own OS, which allows them to perform multiple resource-intensive functions at once. The increased resources available to VMs allows them to abstract, split, duplicate, and emulate entire servers, operating systems, desktops, databases, and networks.
  + Containers are much smaller, typically measured by the megabyte and not packaging anything bigger than an app and its running environment.
  + Where VMs work well with traditional, monolithic IT architecture, containers were made to be compatible with newer and emerging technology like clouds, CI/CD, and DevOps.

1. **Why are Containers lightweight?**

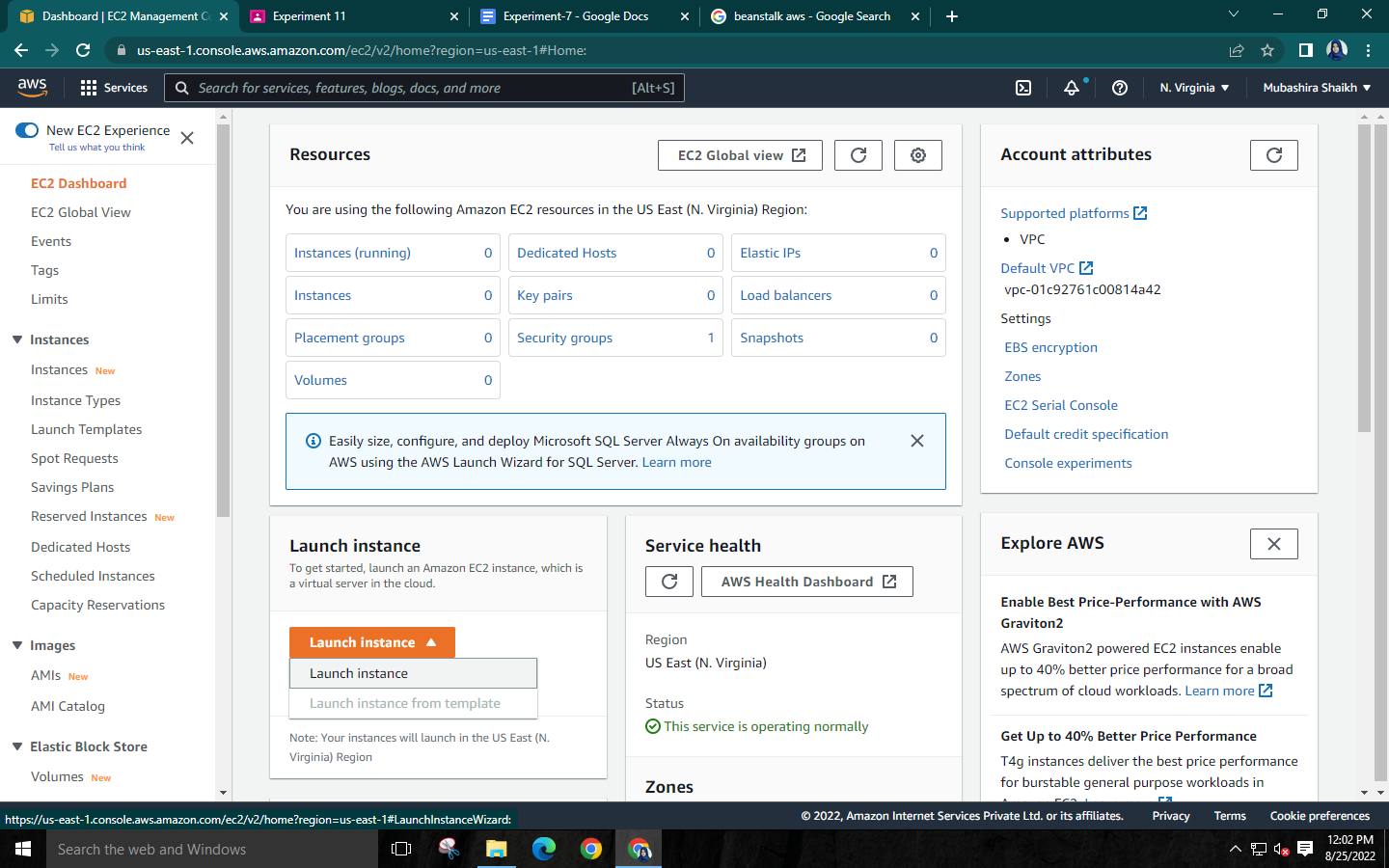
🡺Inside a container are all the necessary executables, binary code, libraries, and configuration files. Compared to server or machine virtualization approaches, however, containers share the machine’s OS system kernel and therefore do not require an OS per application, driving higher server efficiencies and reducing server and licensing costs. This makes them more lightweight and portable, with significantly less overhead.

1. **Deploy a containerized web Application on AWS EC2 Linux. [Install Docker, pull nginx image and run it]. Pull python images and run the command to list all the locally stored docker images.**

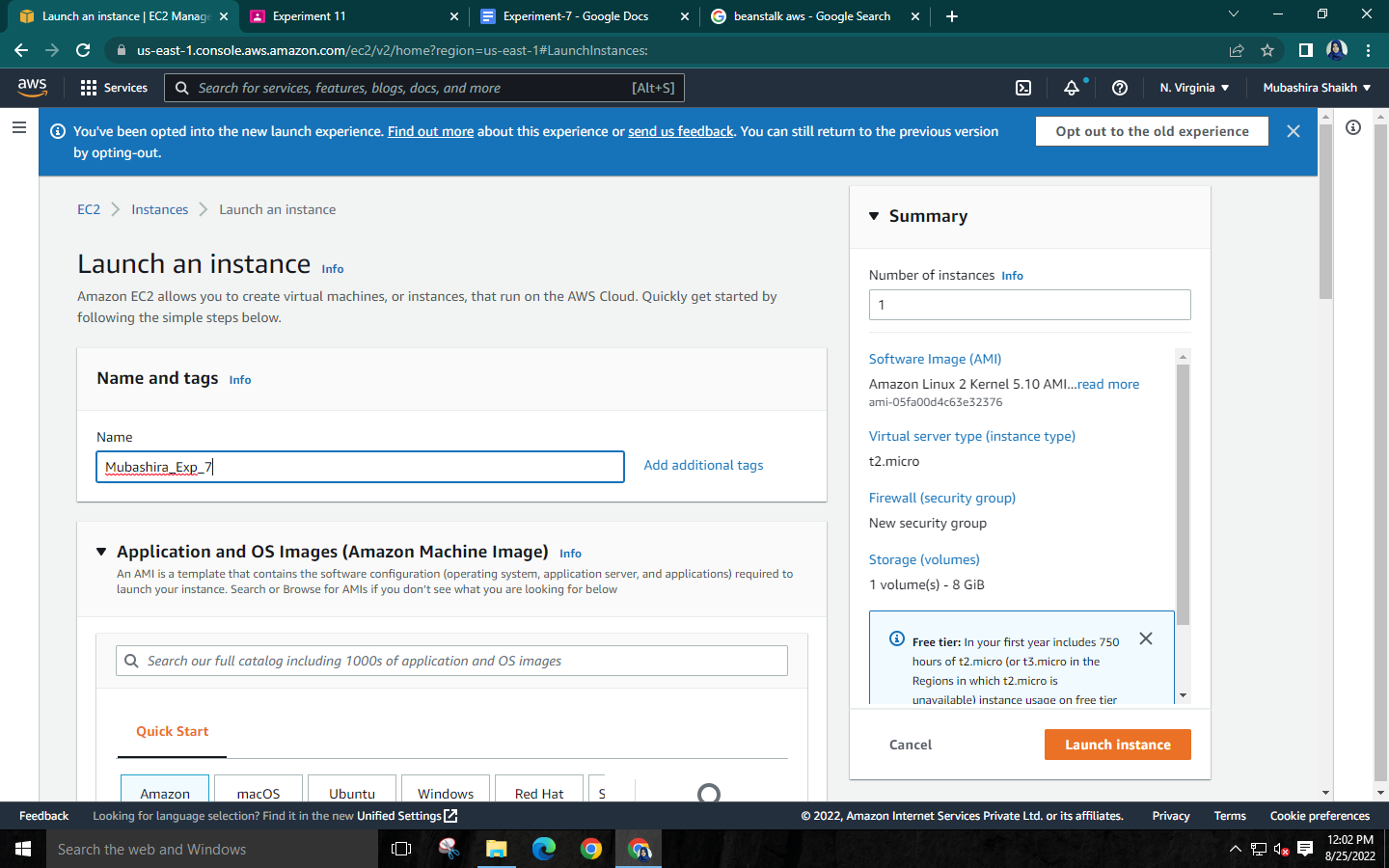
**Step 1: AWS Management Console Dashboard🡪Click on ‘EC2’**

****

**Step 2: Click on Launch instance**

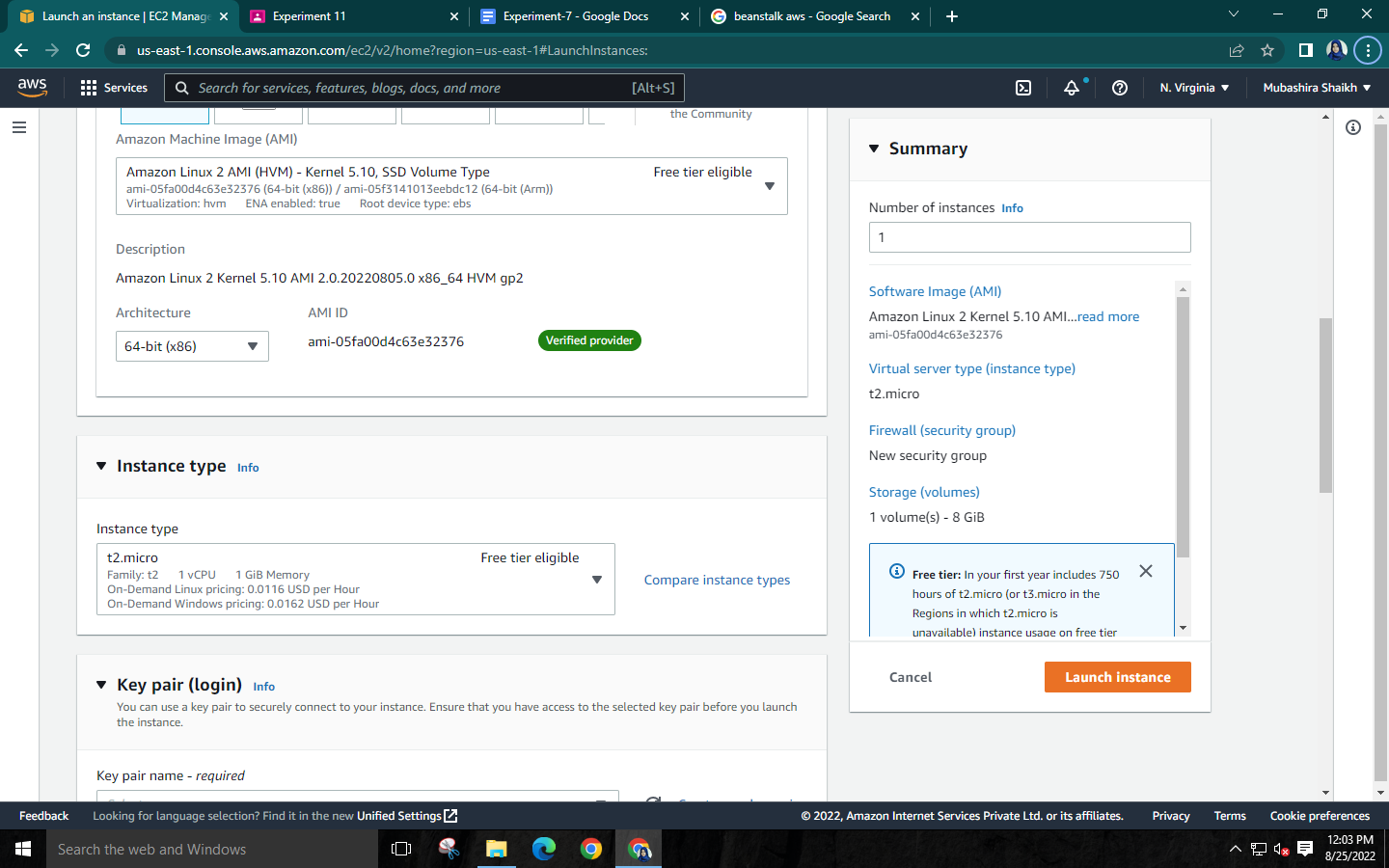
****

**Step 3: Give a name to your instance and select Amazon Linux**

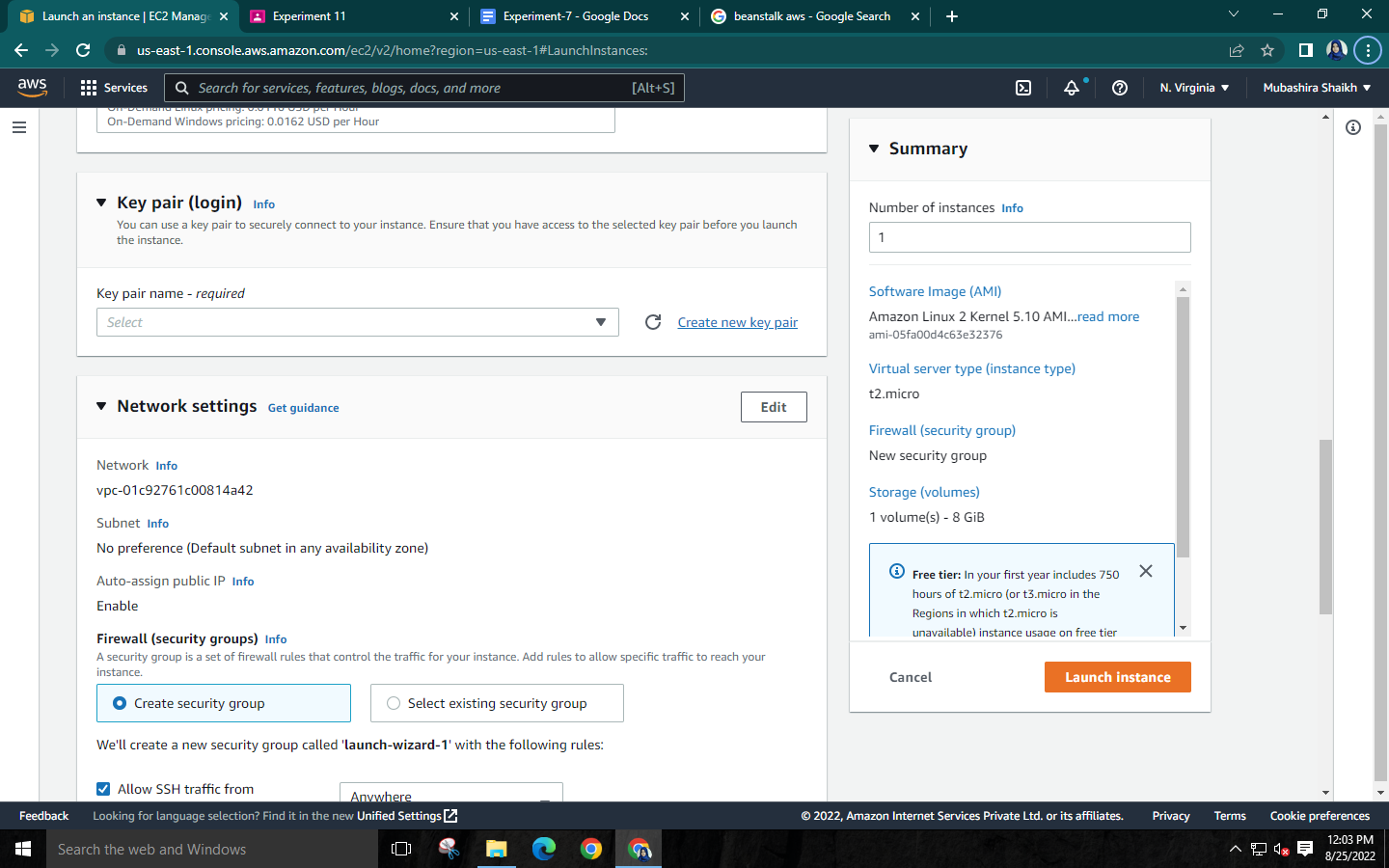
****

****

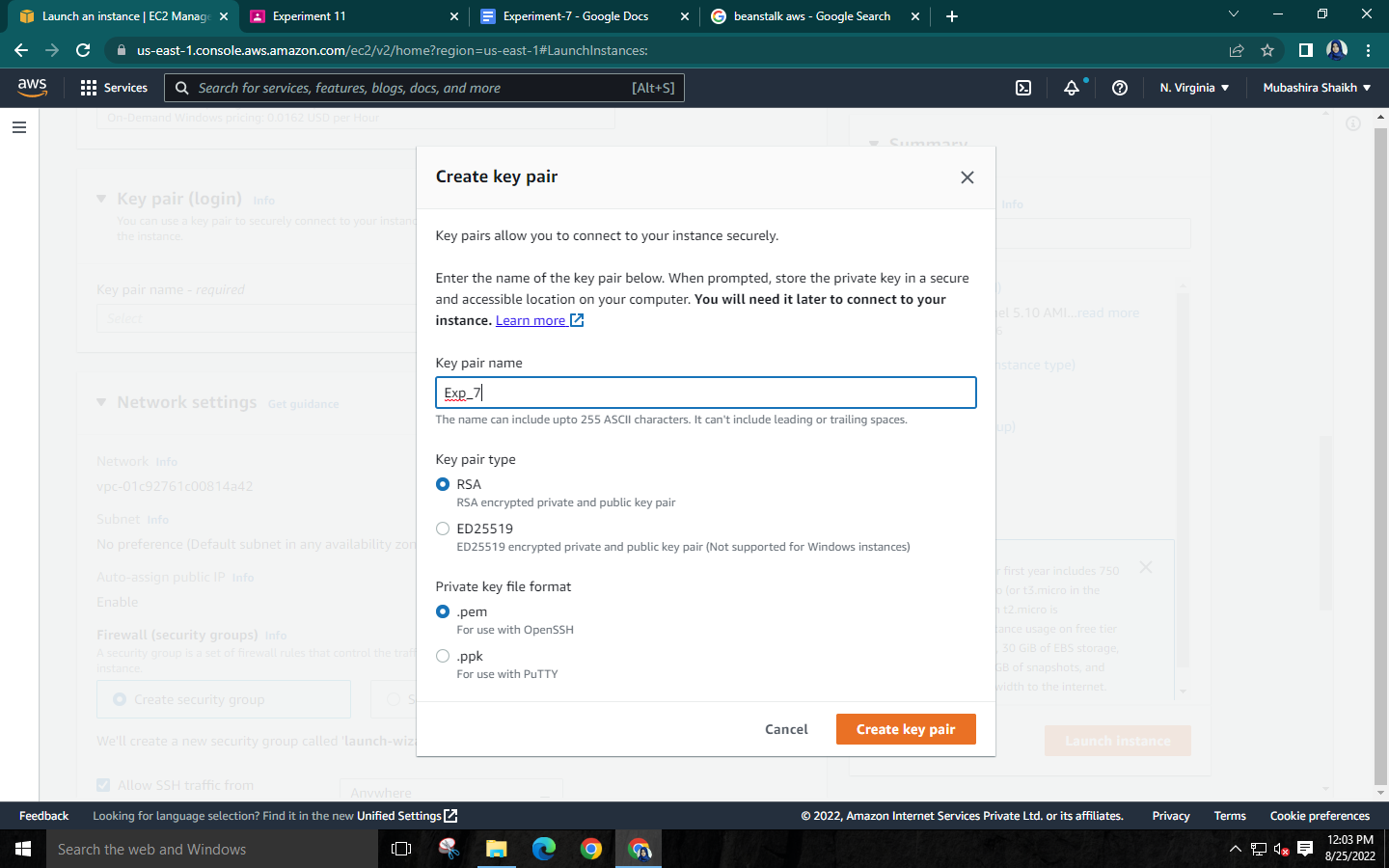
**Step 4: Select instance type with free tier eligibility (t2.micro)**

****

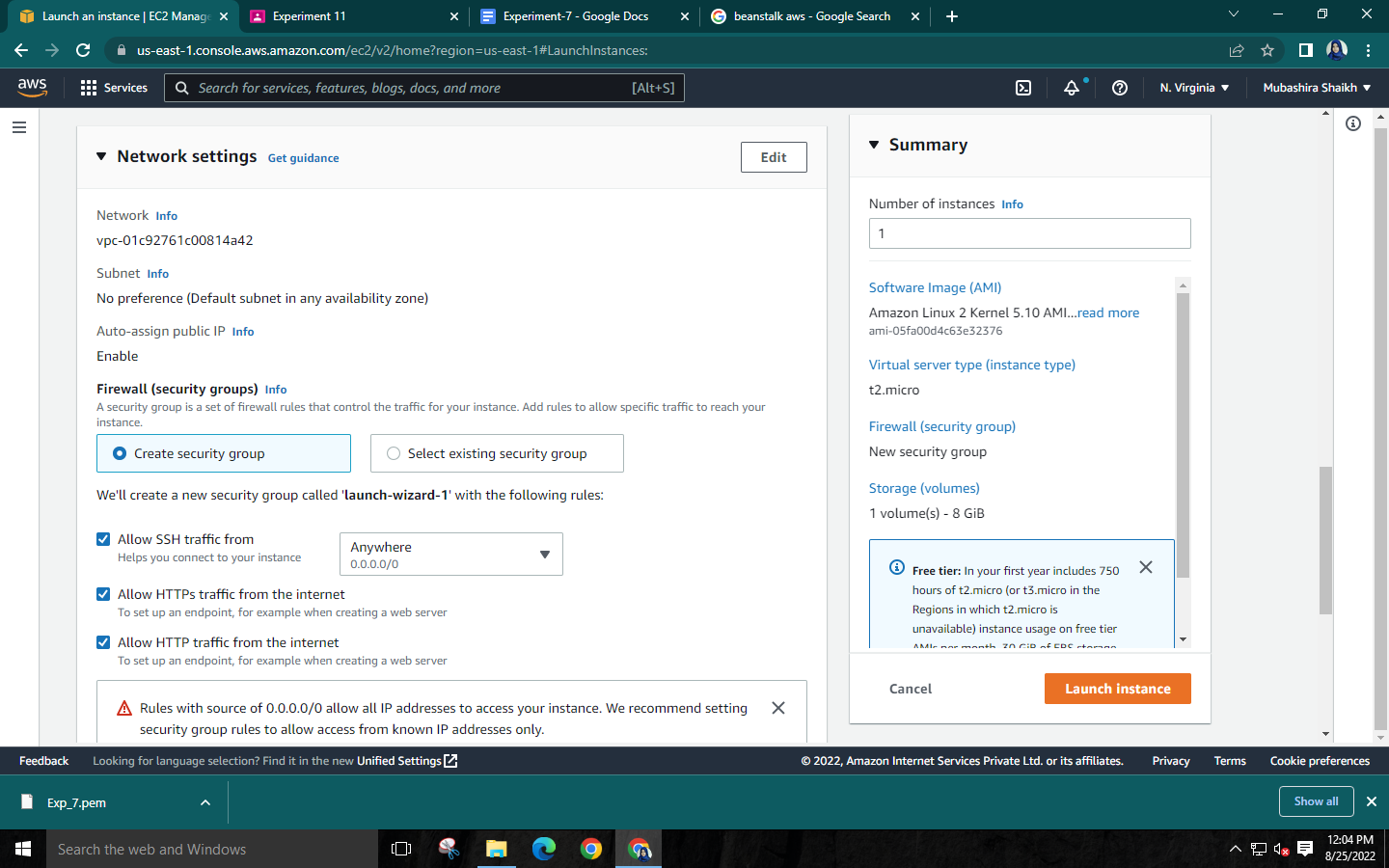
**Step 5: Create key pair. A .pem file will be downloaded which will be later used to connect to the instance**

****

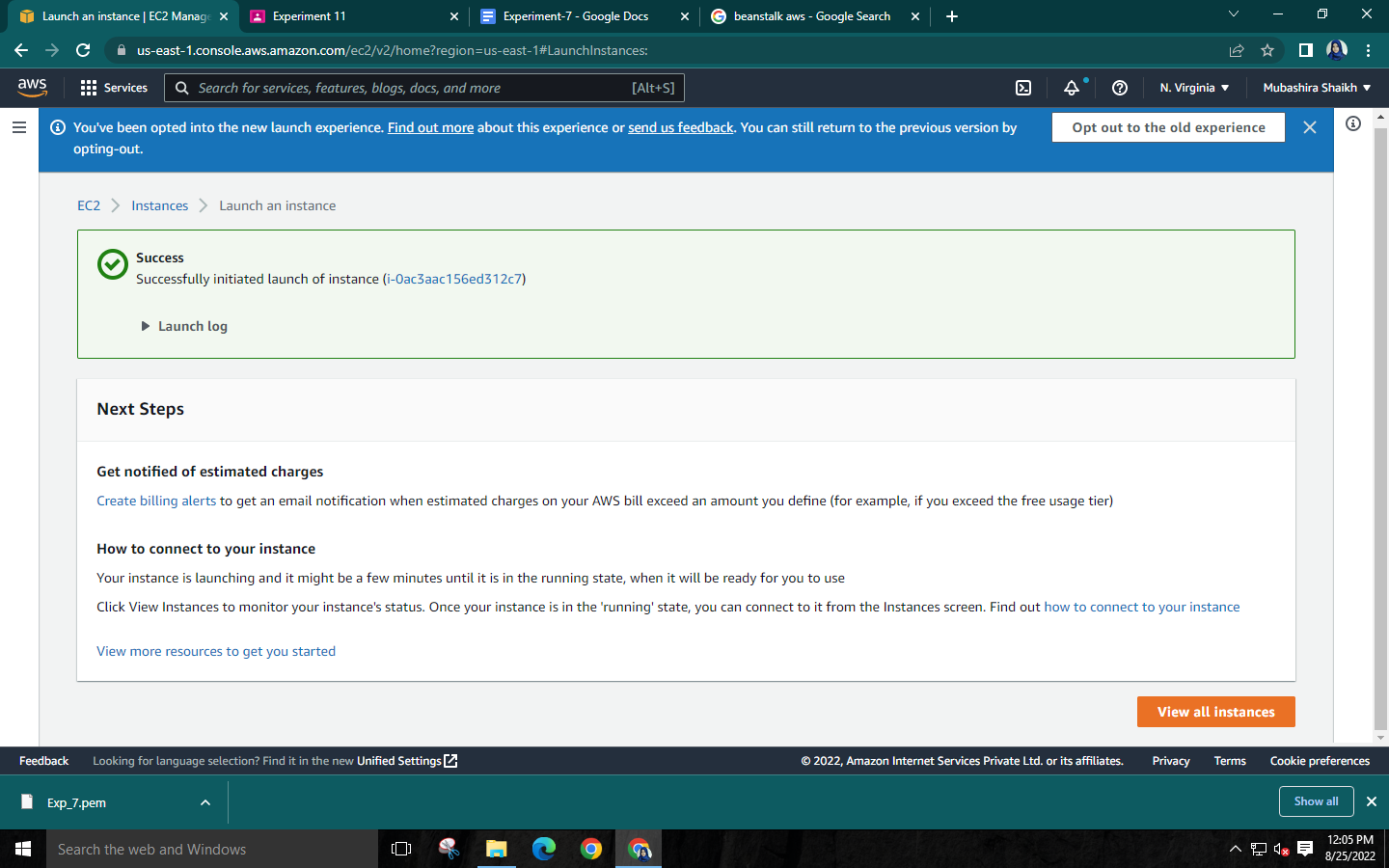
* **Give name to key pair**

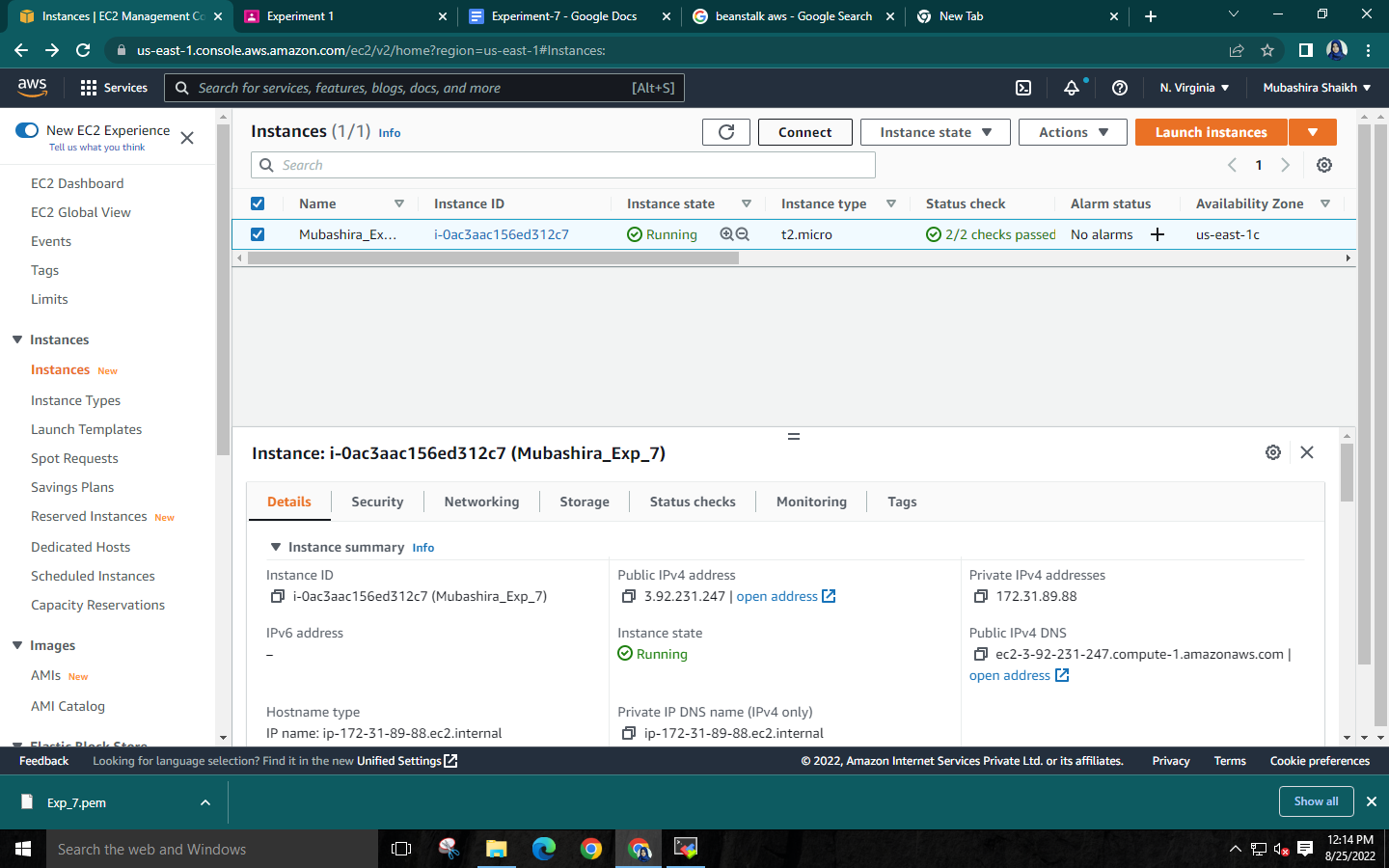
****

**Step 6: Network Settings🡪Select ‘Allow SSH Traffic from’ Anywhere, Allow HTTPs traffic from the internet and Allow HTTP traffic from the internet🡪Launch Instance**

****

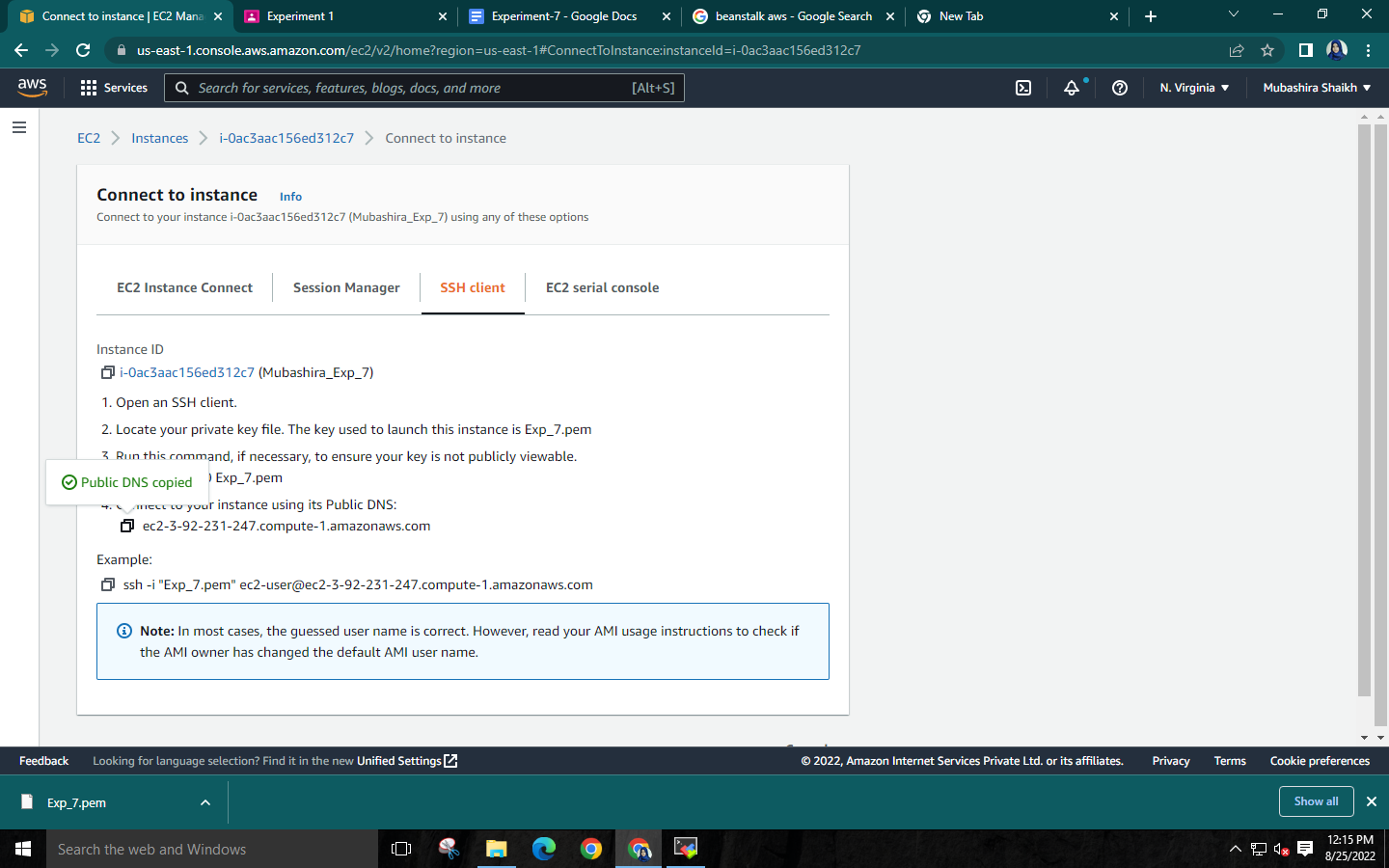
* **Success message will be shown after successful creation of instance**

****

**Step 7: Launch Linux instance to get the remote host and username for SSH in MobaXterm **

* **Go to SSH Client to get the remote host and username**

**Copy the Public DNS which is your remote host and the username is the word before ‘@’**

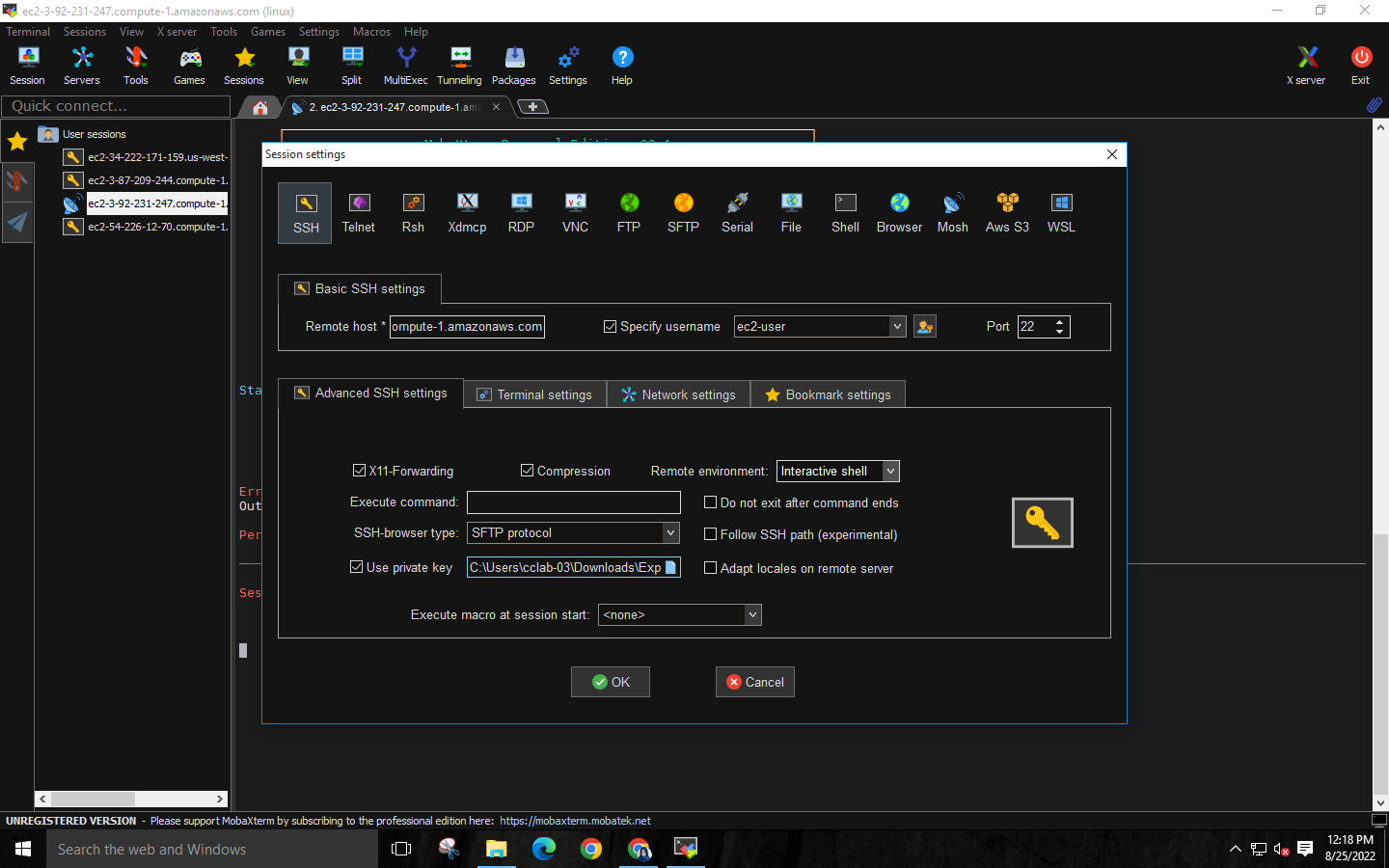
****

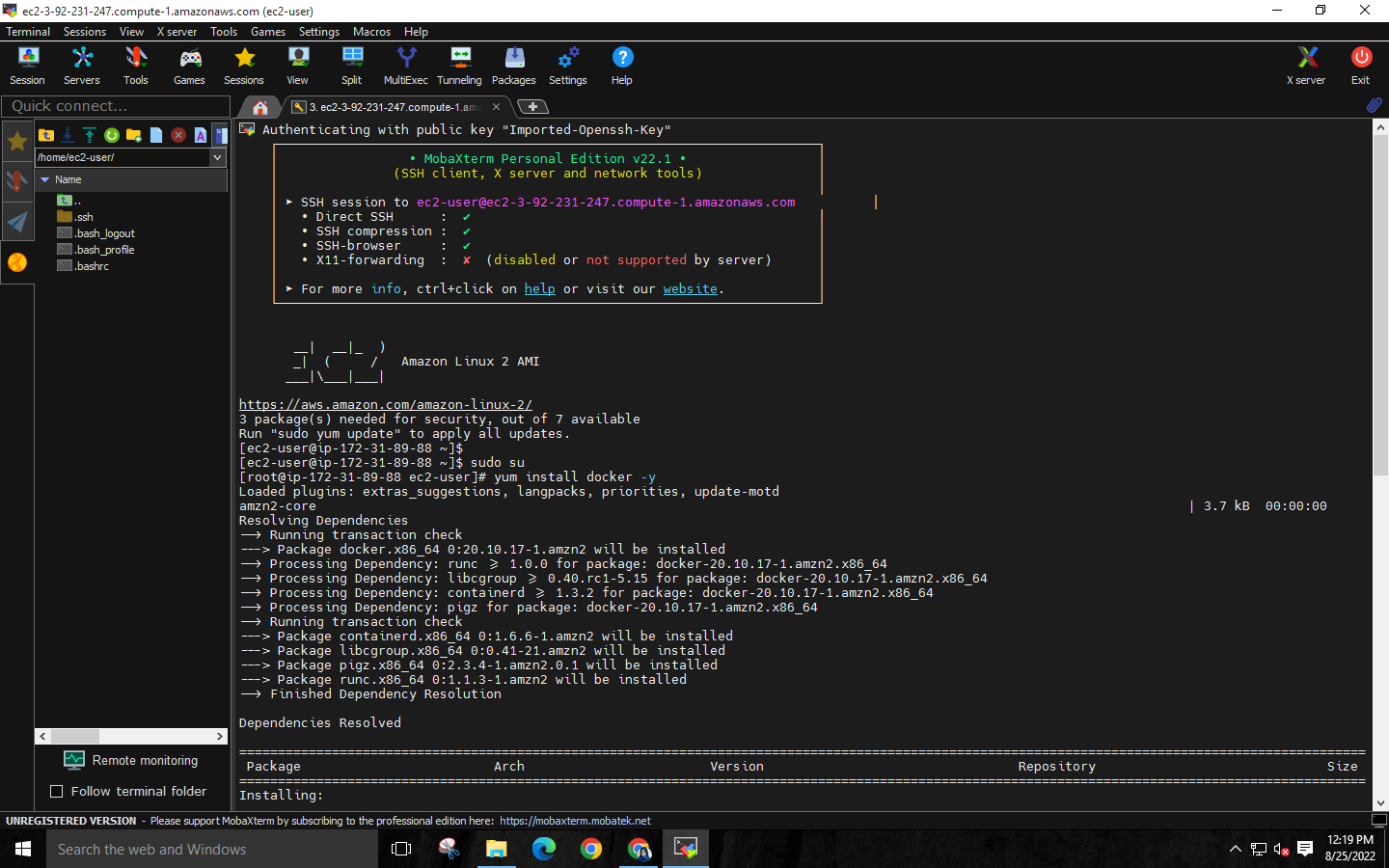
**Step 8: Launch MobaXterm to connect the instance. Go to Sessions 🡪New Session**

****



* **Then select SSH. Fill the basic SSH settings and attach the .pem file downloaded earlier in advanced SSH settings ‘Use private key’ section. Then OK. Your Linux Instance will be running**

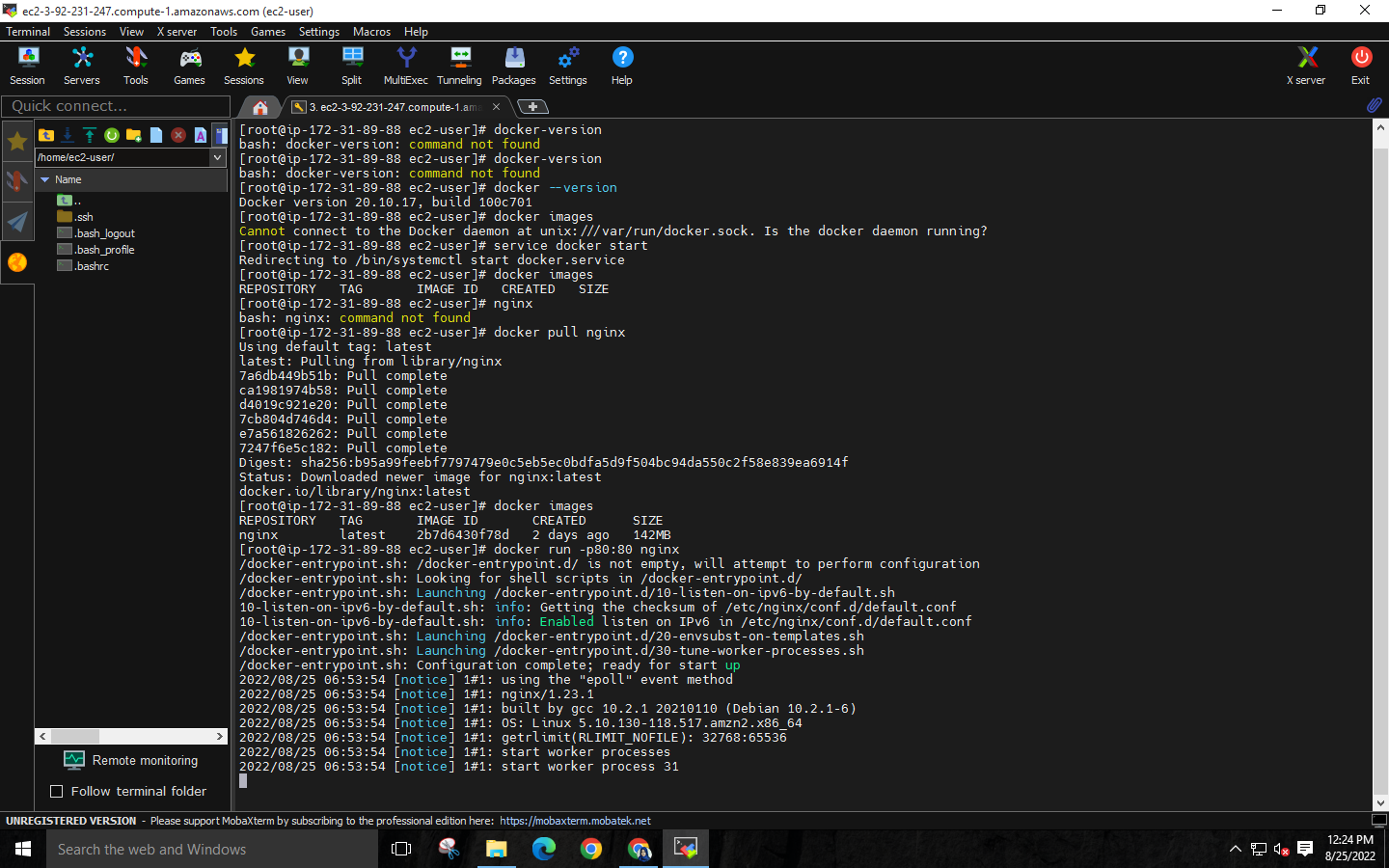
****

****

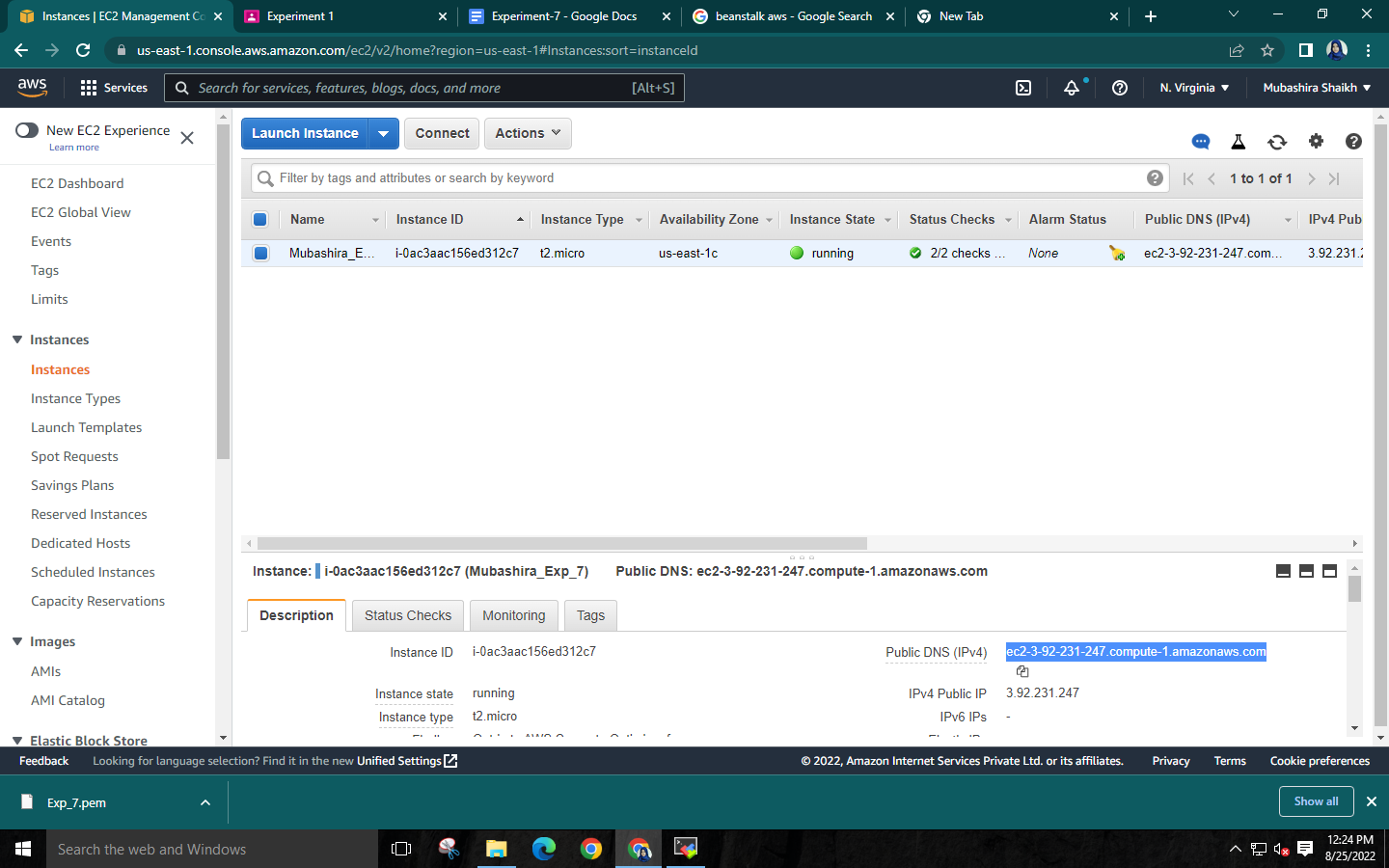
**Step 9: Run the command following commands:**

* **“docker –version” to check the version of docker**
* **“docker images” to see the list of Docker images on the system**
* **“docker service start” to start one or more stopped containers.**
* **“docker pull nginx” to download a Docker image**
* **“docker run -p80:80 nginx” to map TCP port 80 in the container to**

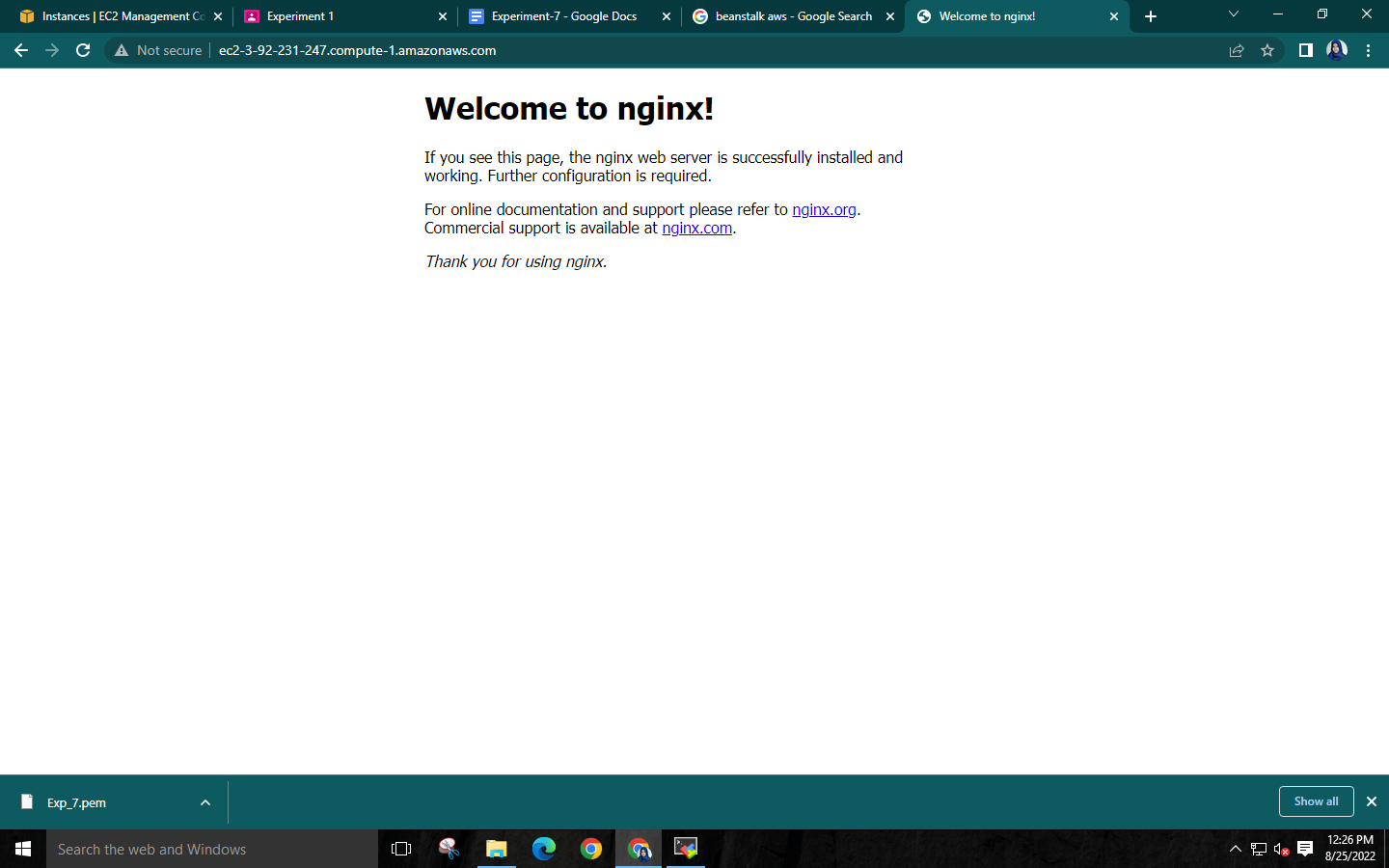
**port 8080 on the Docker host**

****

* **Copy Public DNS(IPv4)**

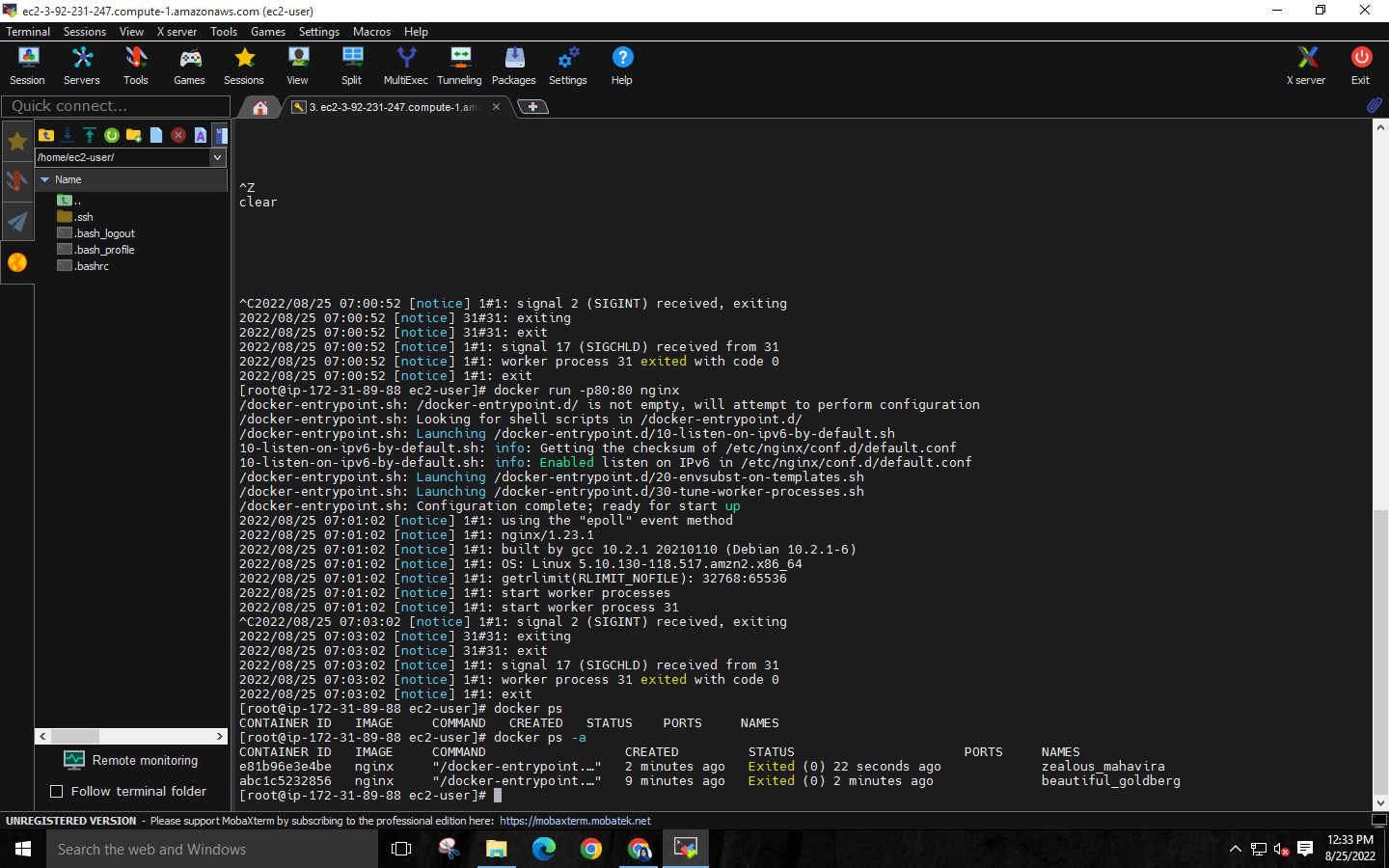
****

* **Paste it on chrome**

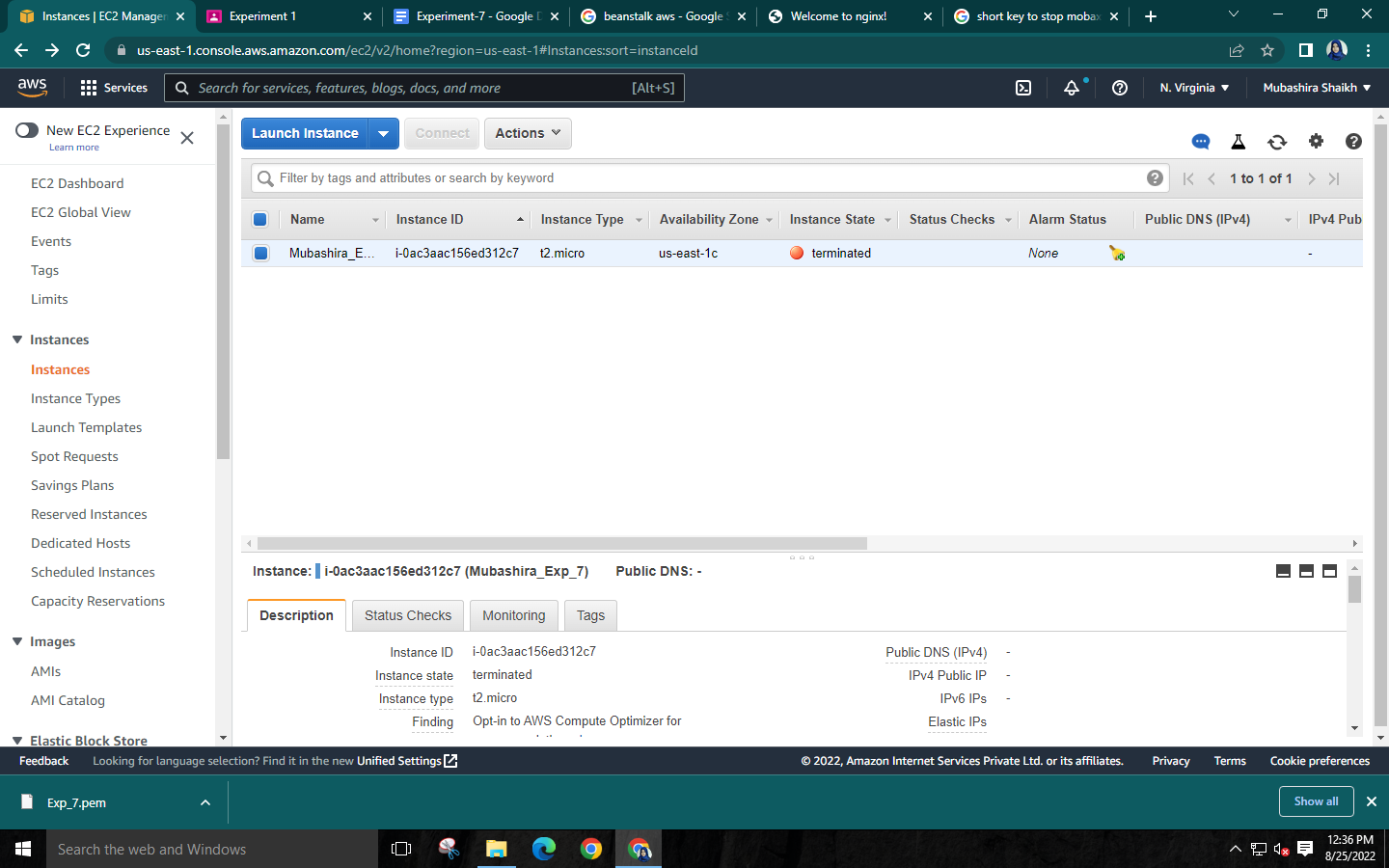
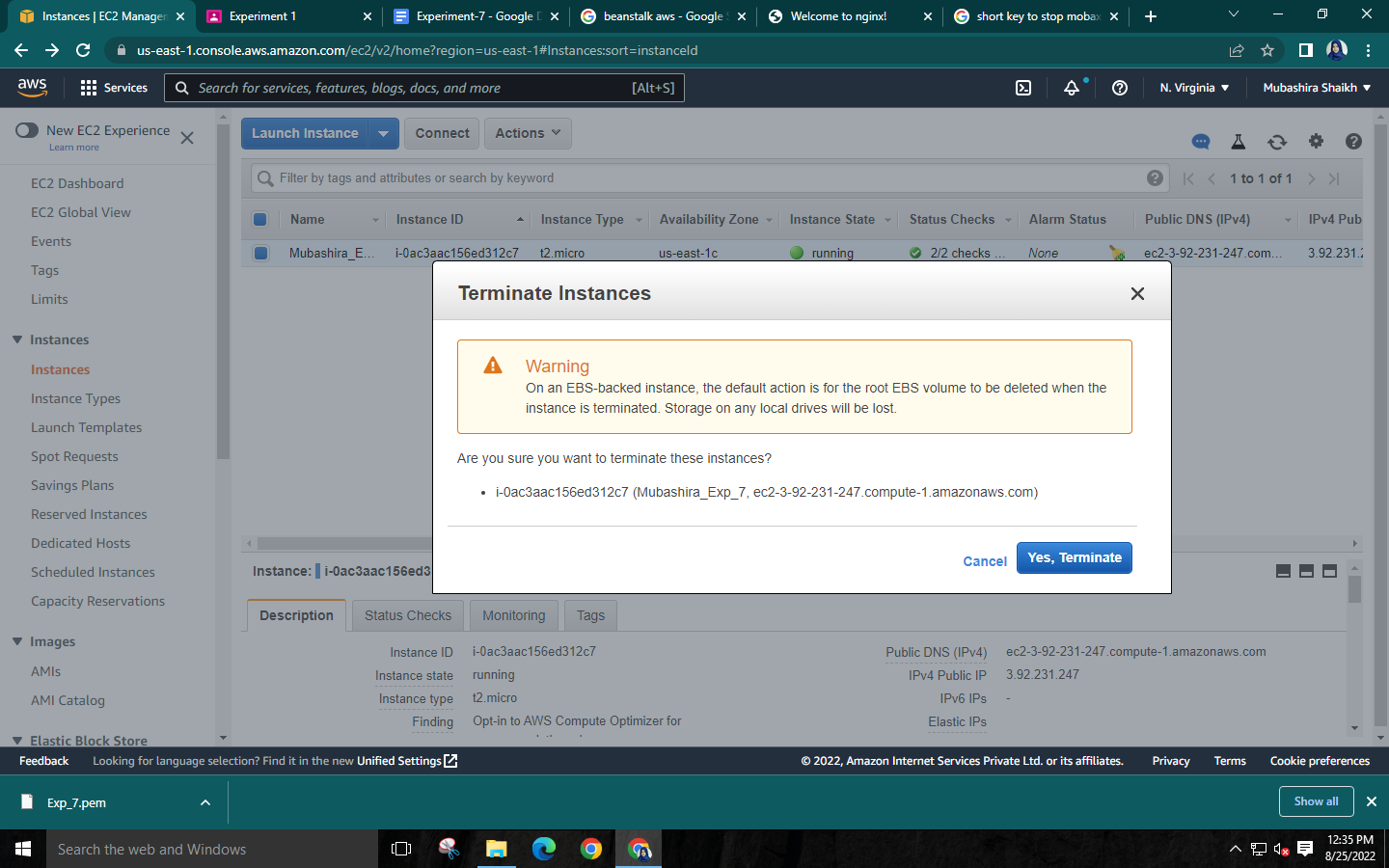
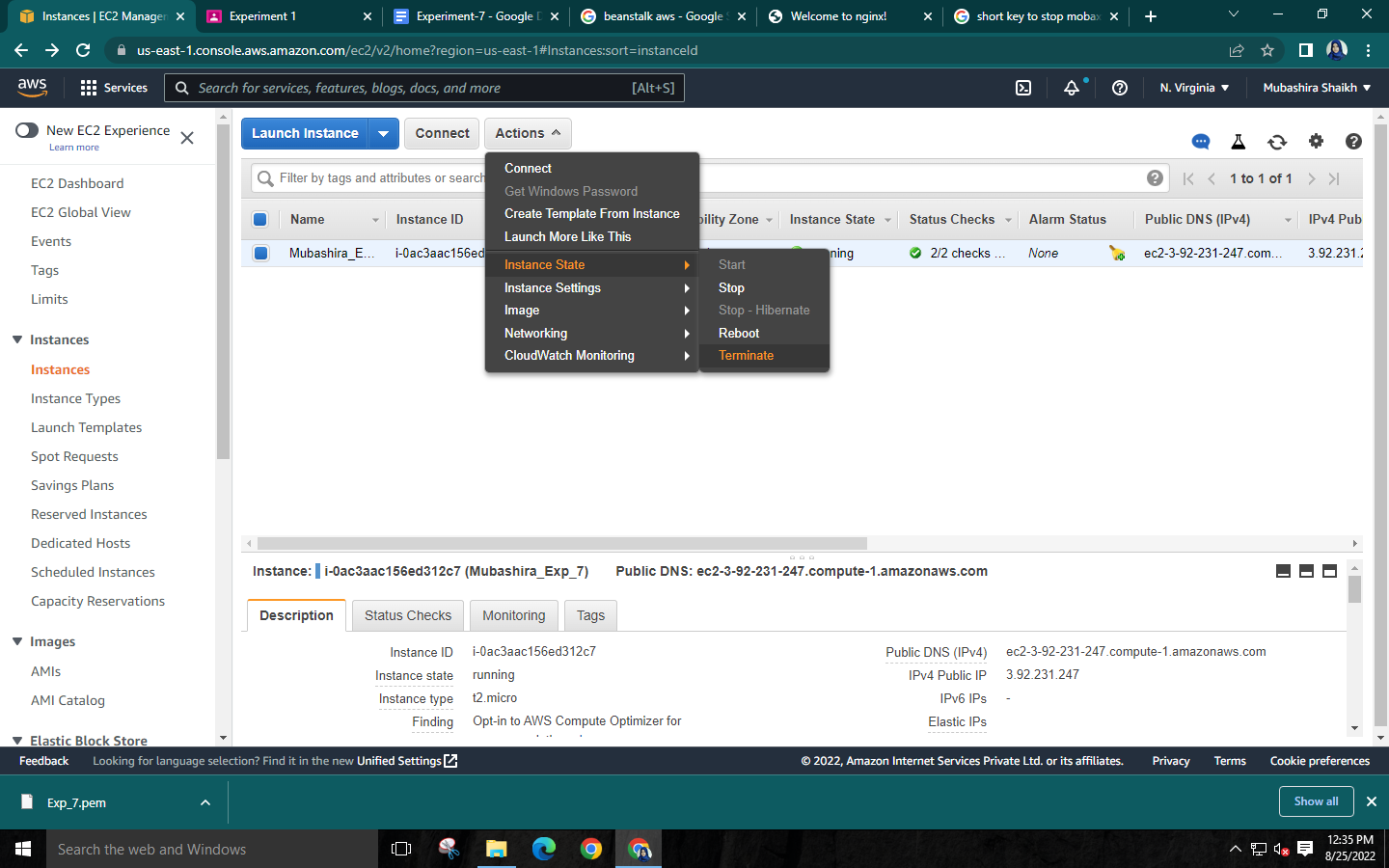
****

**Run command “docker ps” to see the status of the process**

**and “docker ps -a” to show all the containers both stopped and running**

****

**Step 10: Select the instance and terminate it**

****