**Advance DevOps lab**

**Experiment 7**

**Name: BALOCH MUZAMMIL HAFEEZ ROLL NO: 612012**

**Semester: V**

**Branch: Information Technology**

**1) What is Containerization / Docker? Explain Docker Architecture with the help of a**

**diagram.**

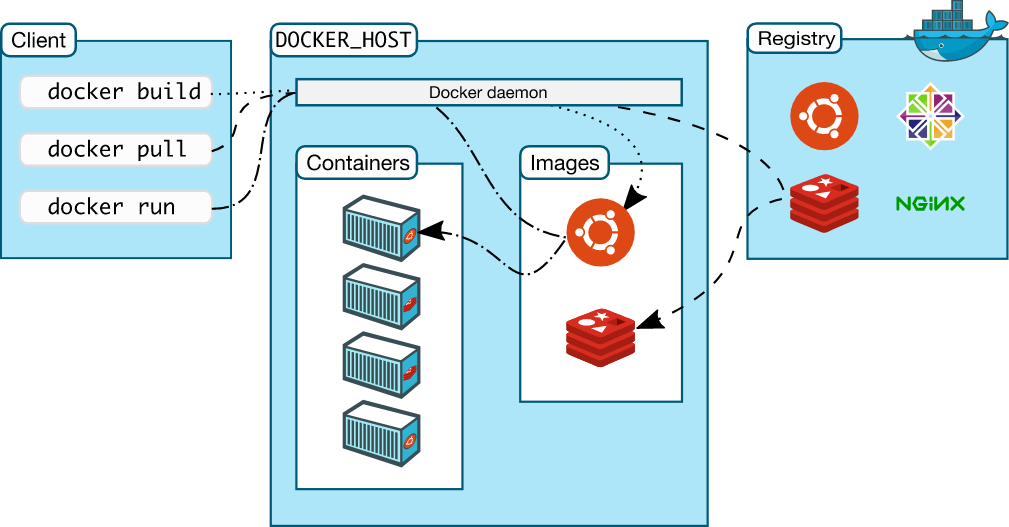
Ans:

Containerization: Containerization is an operating system-based virtualization technique that creates several virtual units in userspace known as Containers. Containers share the same host kernel but are separated by private namespaces and resource control methods at the operating system level. When compared to hypervisors, container-based virtualization provides a higher level of abstraction in terms of virtualization and isolation. Because hypervisors consume a lot of hardware, there is some overhead in terms of virtualizing hardware and virtual device drivers. In each virtual machine instance, a full operating system (for example, Linux or Windows) runs on top of this virtualized hardware.

Docker: Docker is a software platform that enables rapid development, testing, and deployment of programmes. Docker organises software into standardised units called containers, which contain everything the software requires to execute, such as libraries, system tools, code, and runtime. Docker allows you to swiftly deploy and scale apps into any environment while remaining certain that your code will run.

Docker on AWS gives developers and administrators a highly dependable, low-cost solution to build, ship, and run distributed applications of any size.

Docker Architecture: Docker is built on a client-server model. The Docker client communicates with the Docker daemon, which is in charge of building, operating, and distributing your Docker containers. Docker client and daemon can run on the same machine, or a Docker client can connect to a distant Docker daemon. The Docker client and daemon communicate with one another using a REST API, UNIX sockets, or a network interface. Docker Compose is another Docker client that allows you to deal with applications made up of a collection of containers.



**2) Compare Containers vs VMs**

Ans:

| **Virtual Machines(VM)** | **Containers** |
| --- | --- |
| VM is piece of software that allows you to install other software inside of it so you basically control it virtually as opposed to installing the software directly on the computer. | While a container is a software that allows different functionalities of an application independently. |
| Applications running on VM systems can run different OS. | While applications running in a container environment share a single OS. |
| VM virtualizes the computer system. | While containers virtualize the operating system only. |
| VM’s are useful when we require all of the OS resources to run various applications. | While containers are useful when we are required to maximise the running applications using minimal servers. |
| VM is more secure. | While containers are less secure. |
| VM uses a lot of system memory. | While containers require very less memory. |
| Examples of VM are: KVM, Xen, VMware. | While examples of containers are:RancherOS Containers by Docker. |

**3) Why are Containers lightweight?**

Ans: Containers are a form of operating system virtualization. A single container might be

used to run anything from a small microservice or software process to a larger application.

Inside a container are all the necessary executables, binary code, libraries, and

configuration files. Compared to server or machine virtualization approaches, however,

containers do not contain operating system images. This makes them more lightweight

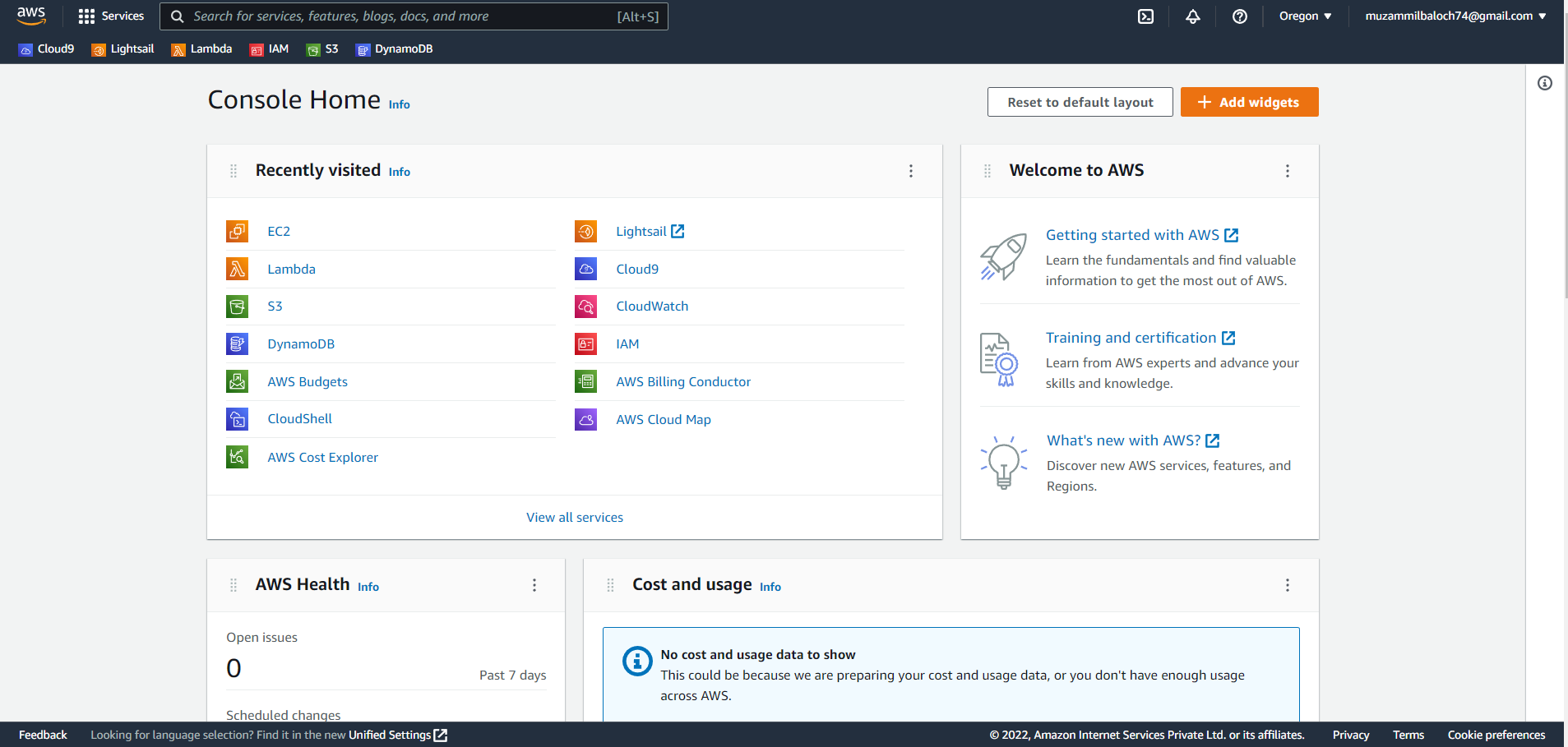
and portable, with significantly less overhead. In larger application deployments, multiple

containers may be deployed as one or more container clusters. Such clusters might be

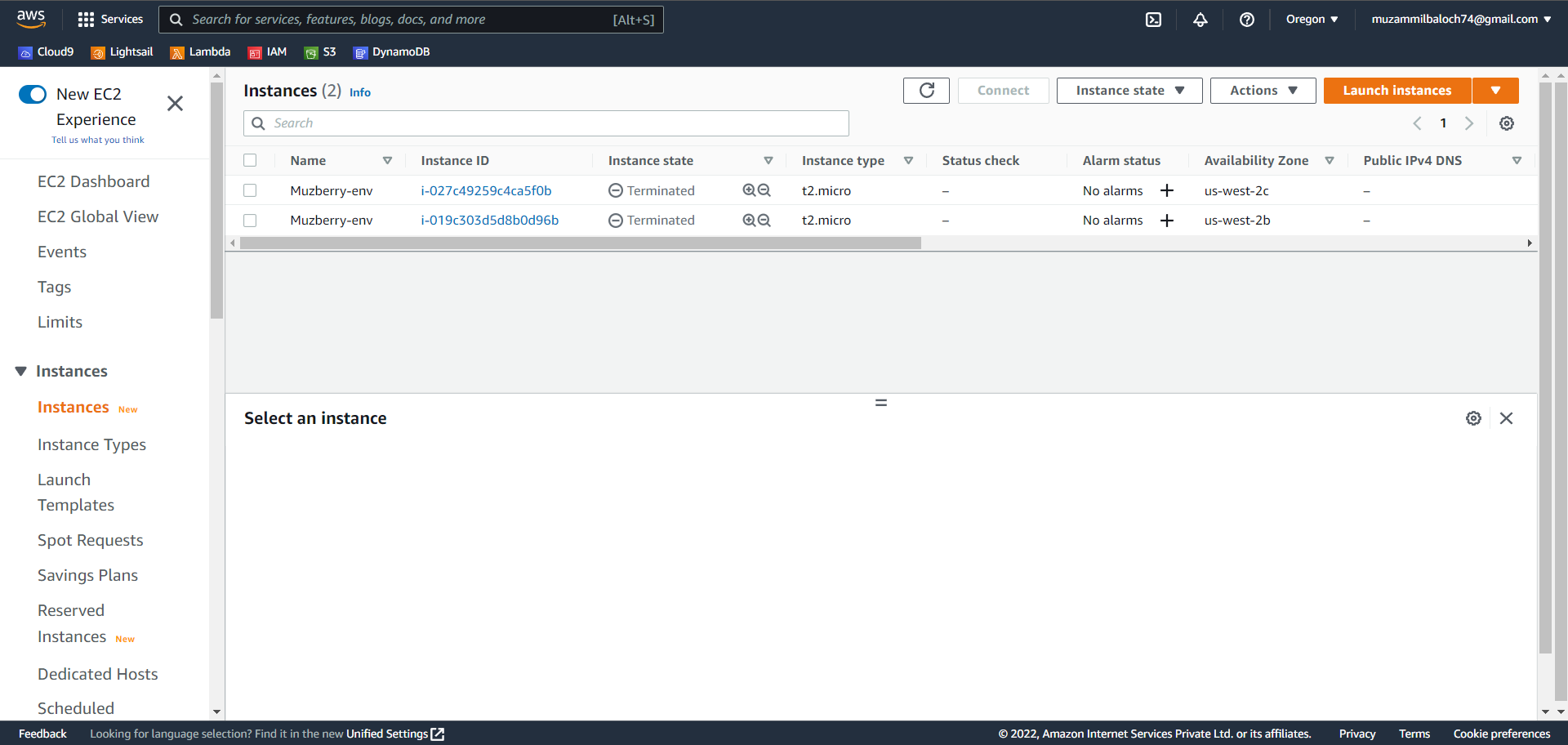
managed by a container orchestrator such as Kubernetes.

**4) 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 .**

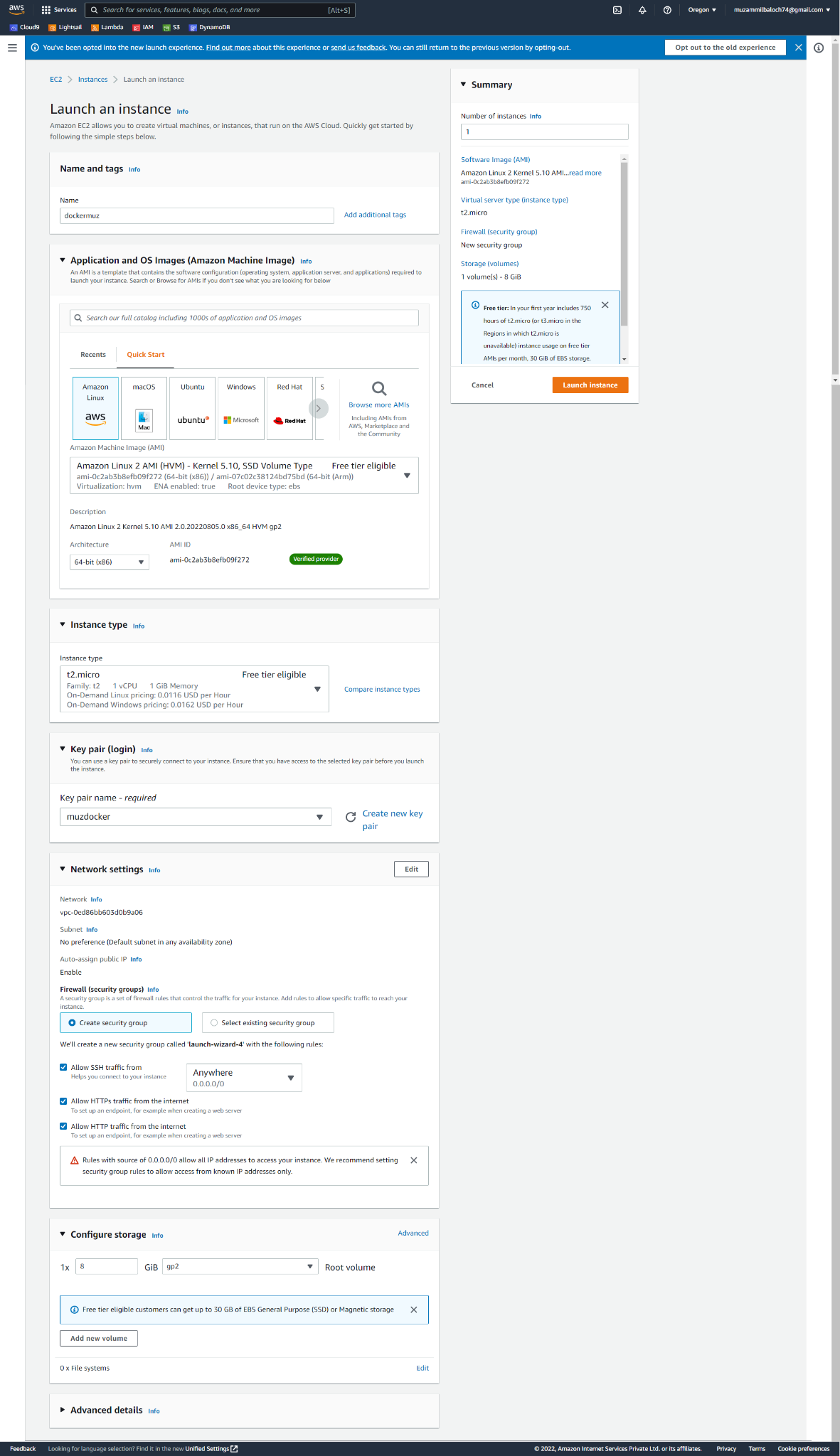
Step 1: Login to your AWS Console.



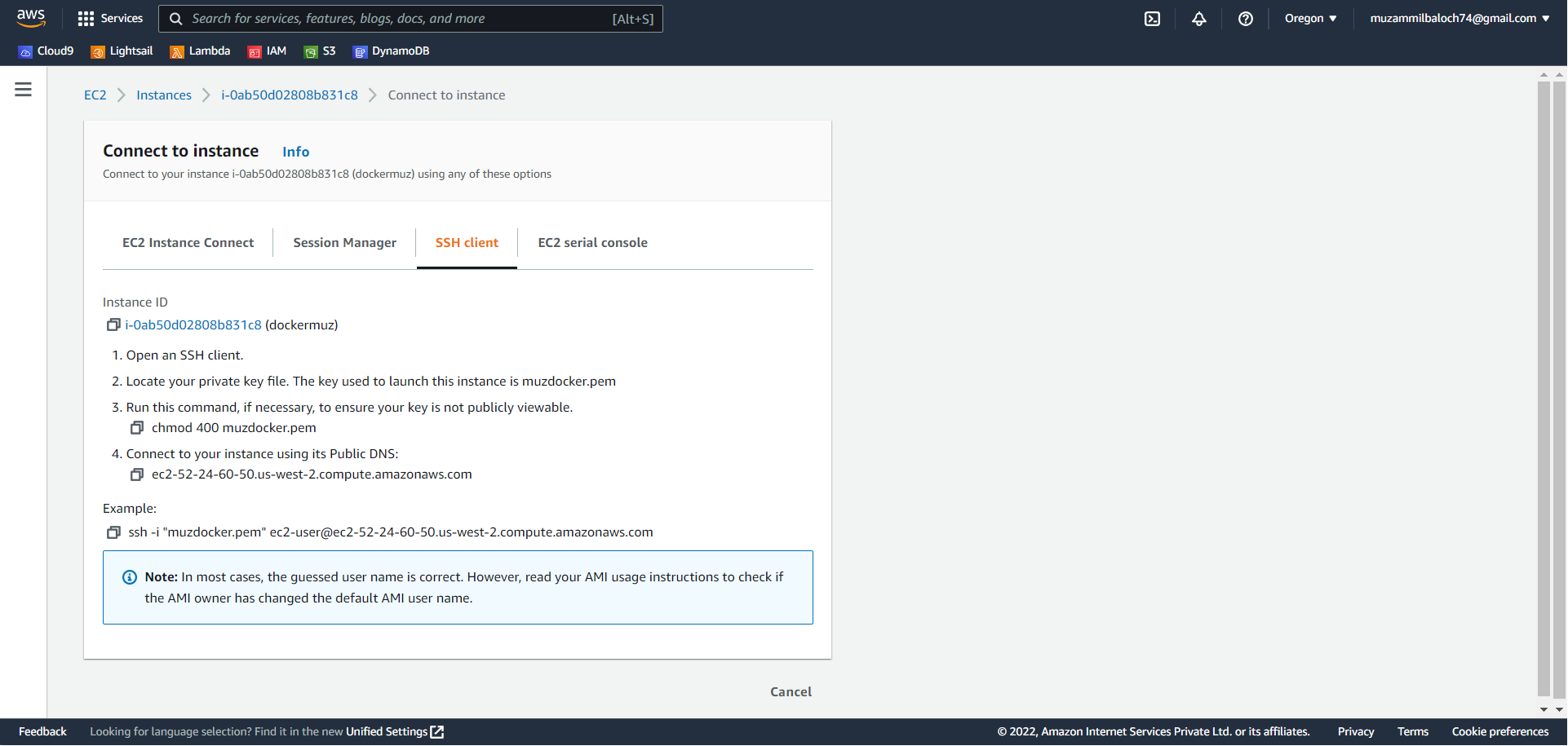
Step 2: Create a instance.



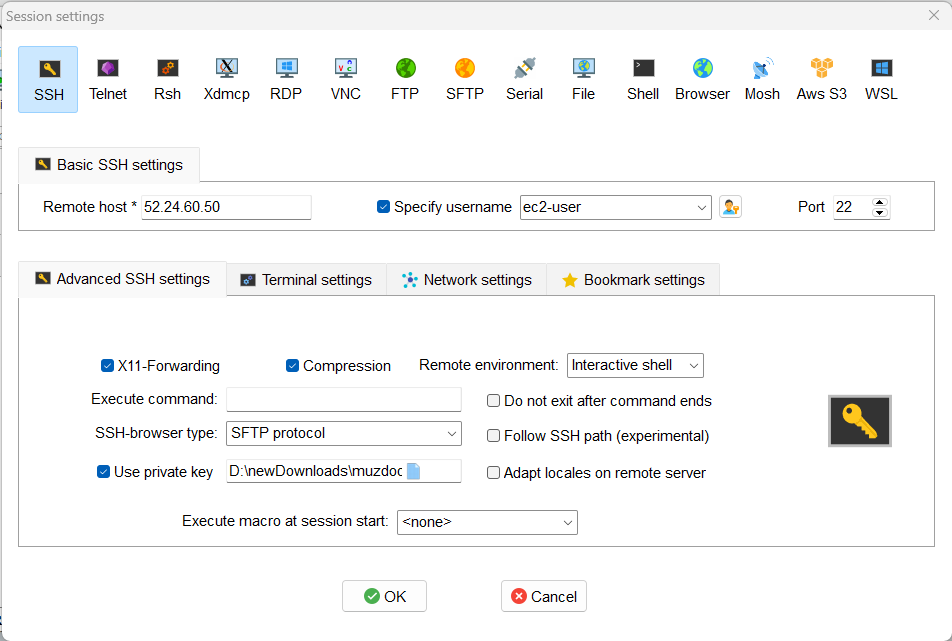
Step 3: Give the instance a name and configure its settings.



Step 4: Connect to the instance



Step 5: connect through MobaXterm.



Step 6: After connecting, run the following command in order to set up docker.

Sudo su

Yum install docker

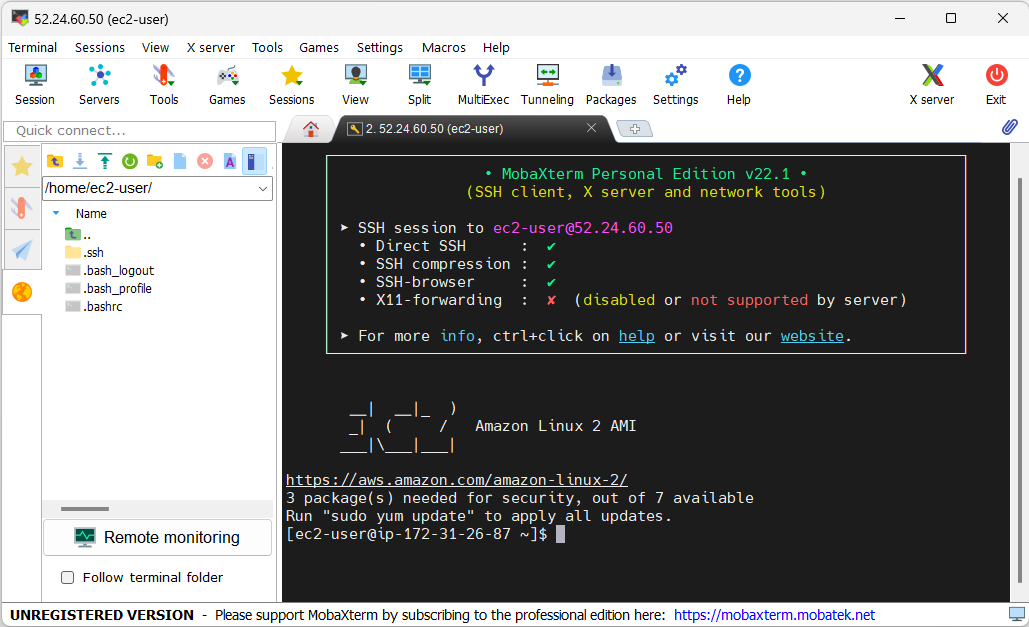
Service docker start

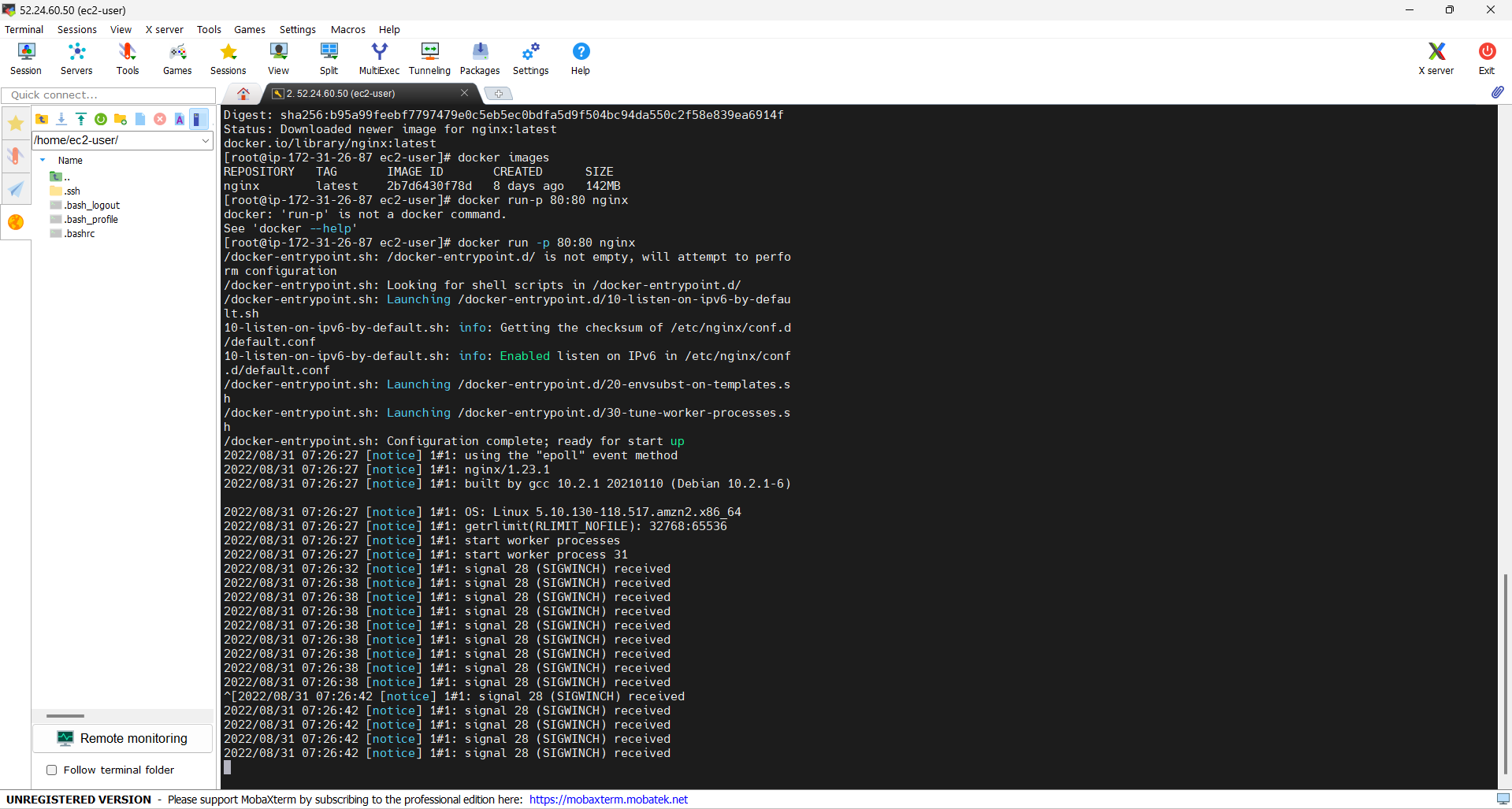
Docker images

Docker pull nginx

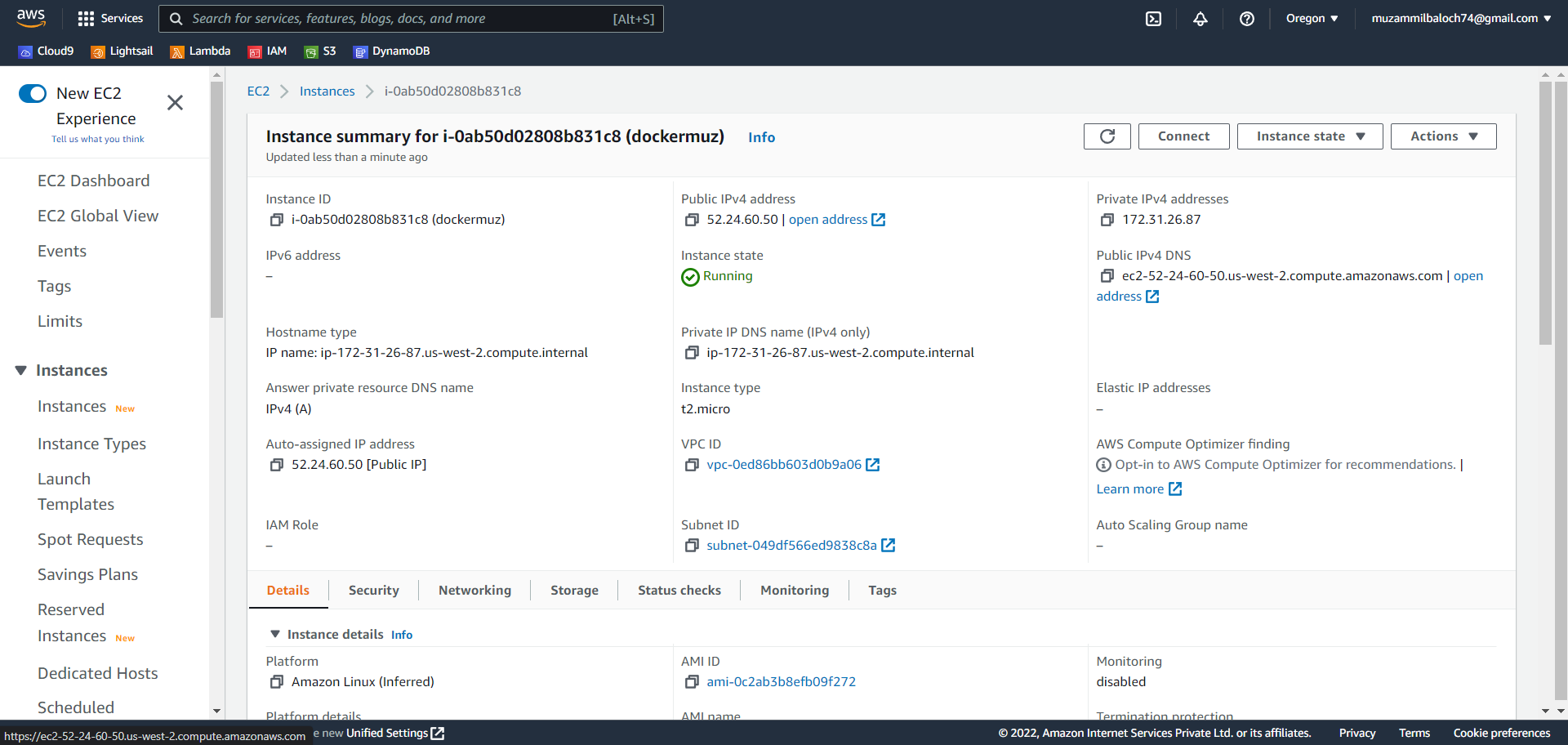
Docker images

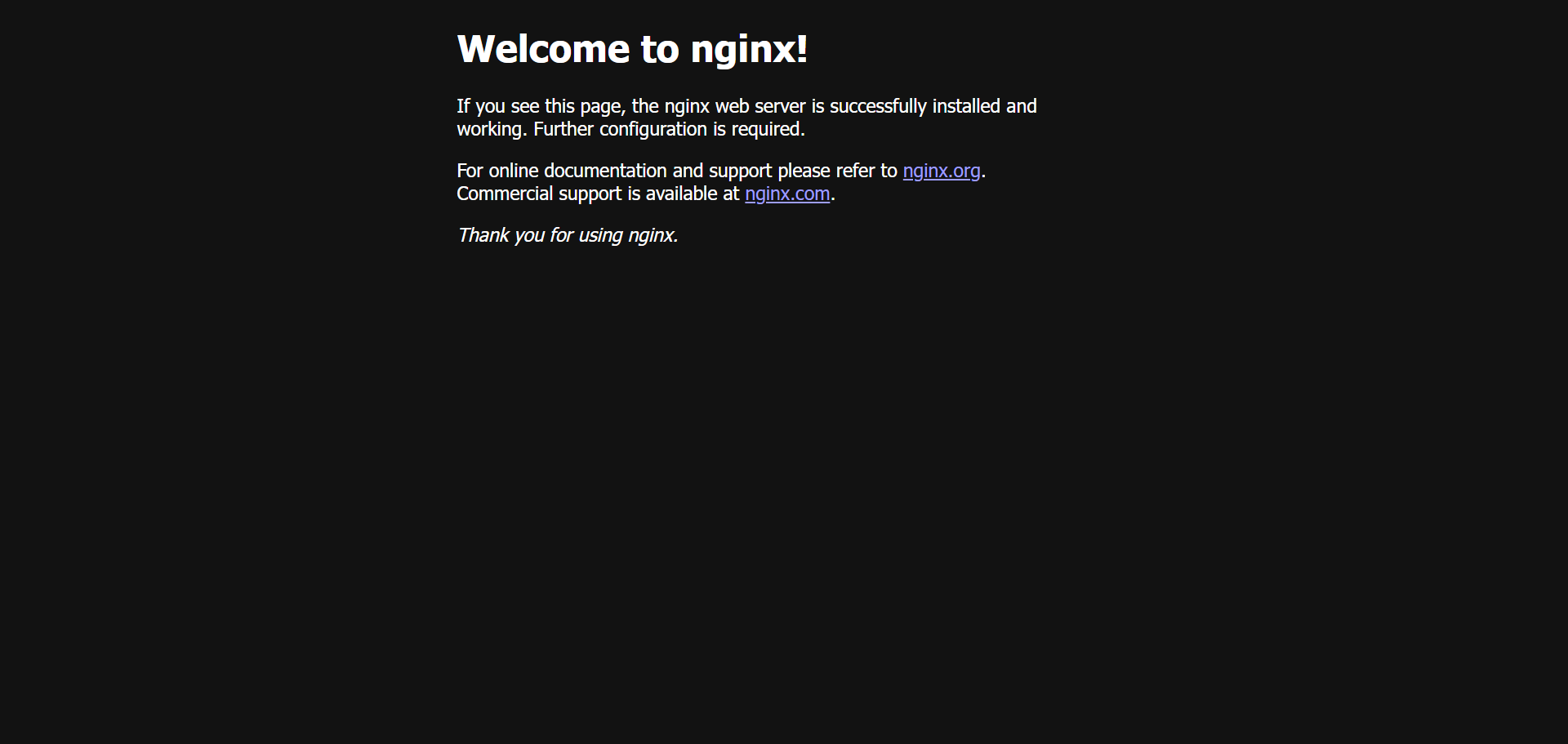
Docker run -p 80:80 nginx





Step 7:To assess the website, copy the Public IPv4 DNS and paste it into a new tab.



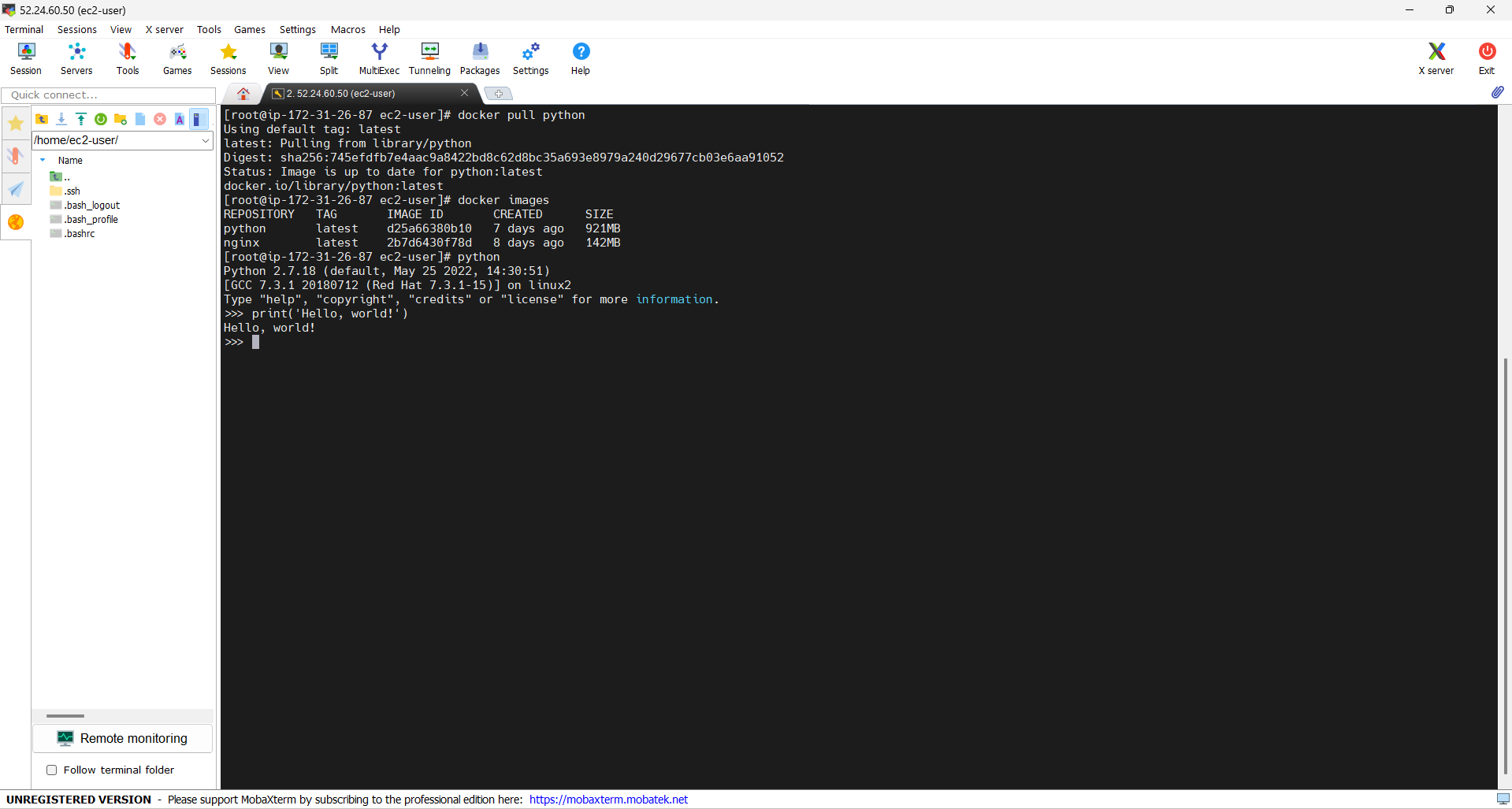


Step 8: run the following command in order to set up python.

Docker pull python

Docker images

Python



Step 9: Now terminate the instance.

