**Experiment-6**

**AIM:-Create a docker image for any application using docker file and push it to Docker Hub.**

**Step 1:-**[**Connecting AWS Instance Ubuntu using Mobaxterm**](#mobaxterm)

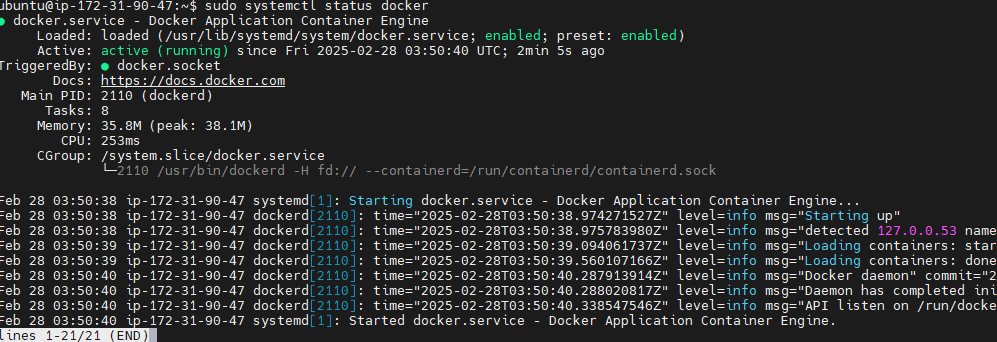
1. Login AWS(Amazon Web Services) Account
2. Lunch Instance name Docker
3. Connect to Ubuntu or mobaxterm

(note:-Follow this url for docker file and application—https://github.com/devisar/devopslab)

**Step 2:-**[**Create Docker Hub Account and create repository in Docker Hub**](#dockerhub)

**Step 3:-Install Docker and Check Status and Start Docker in mobaxterm**

1. sudo apt update -y
2. sudo apt install docker.io –y
3. sudo systemctl status docker(come outside use command ctl+z)



Above status command is docker running means no problem if not run use command below to run

1. sudo systemctl start docker

### Step 4:- Grant Access

### Why we give grant access means

A easy way to verify your Docker installation is by running the below command

docker run hello-world

If the output says:

### 

This can mean two things,

1. Docker deamon is not running.(start docker using “sudo systemctl start docker” )
2. Your user does not have access to run docker commands.

### Grant Access to your user to run docker commands

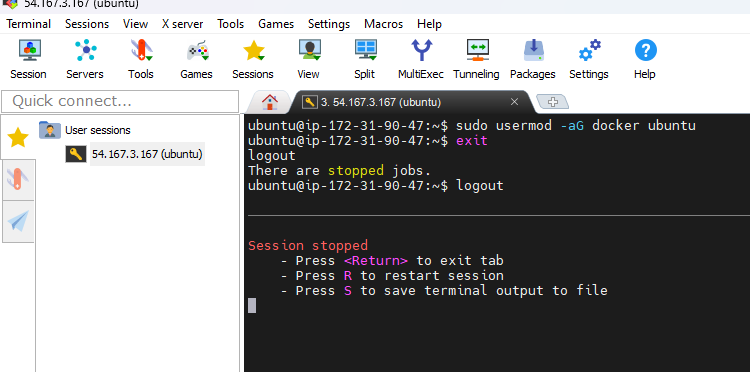
1. sudo usermod -aG docker ubuntu

In the above command ubuntu is the name of the user, you can change the username appropriately.

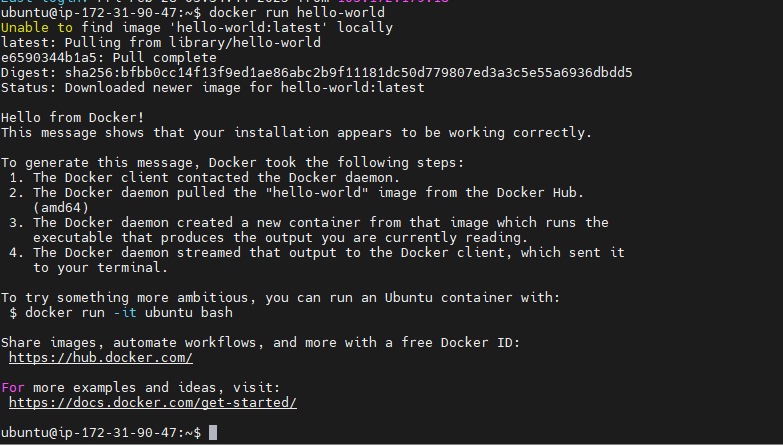
**NOTE:** : You need to logout and login back for the changes to be reflected.

1. Logout purpose use commands exit or logout

Again run command “docker run hello-world”



Start session

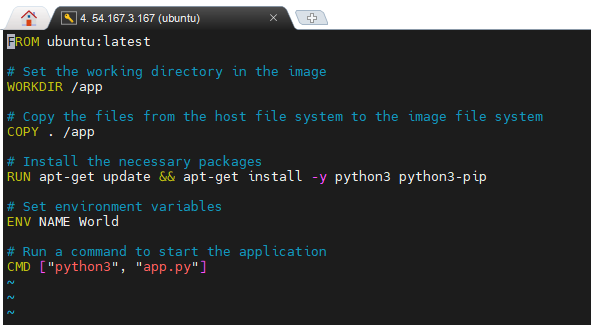


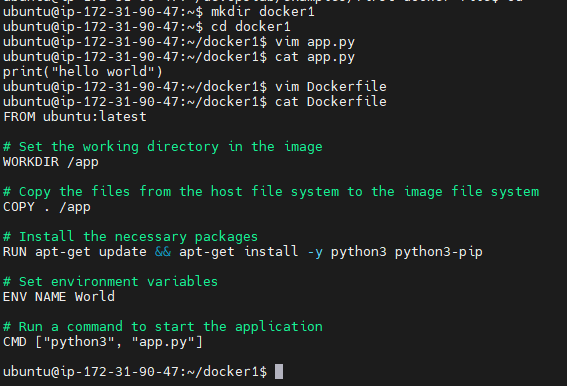
**Step 5:-Creating application and Docker file**

1. Mkdir docker1
2. cd docker1
3. vim app.py

print(“hello world”)

1. cat app.py
2. vim Dockerfile (below pic commands write lab record) typing command vim before click “i” for insert data after completion Docker file commands save before click esc use :wq!



**Below image for understanding purpose**se

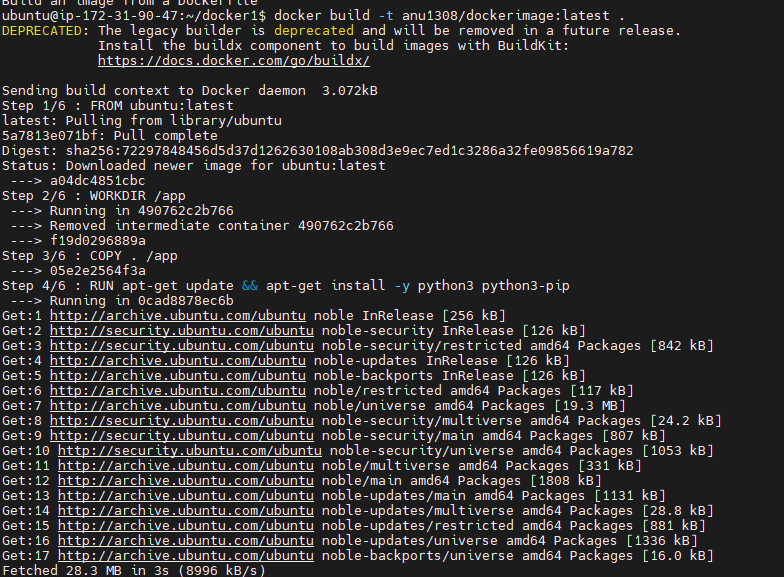
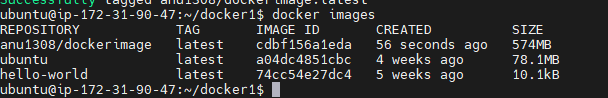
**Step 6:-** **Build and Check Docker image**

Syntax:- docker build –t dockerhub\_username/repositoryname:tag .

# docker build –t anu1308/dockerimage:latest .

1. docker images

**Below images for understanding purpose**

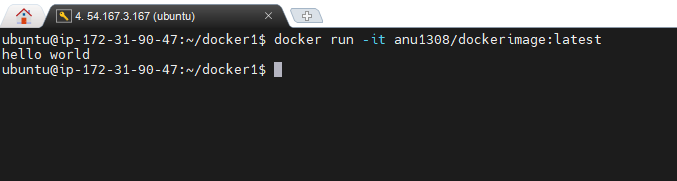
 

### Step 7:- Run your First Docker Container

1. docker run -it anu1308/dockerimage:latest

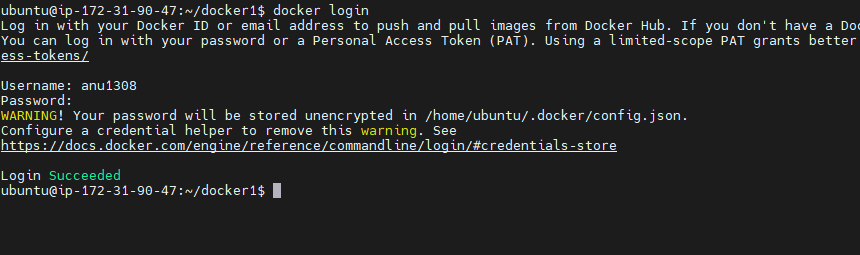
Output

Hello World



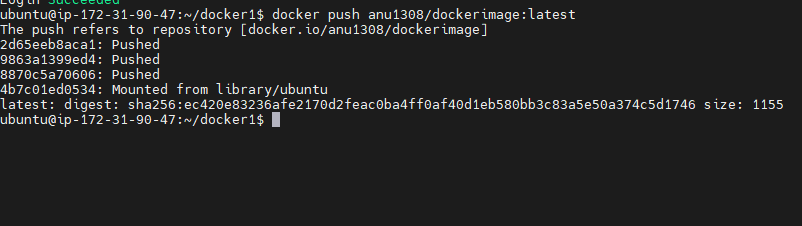
**Step 8:-Docker Login**

1. docker login

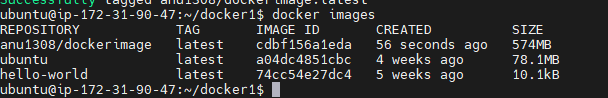


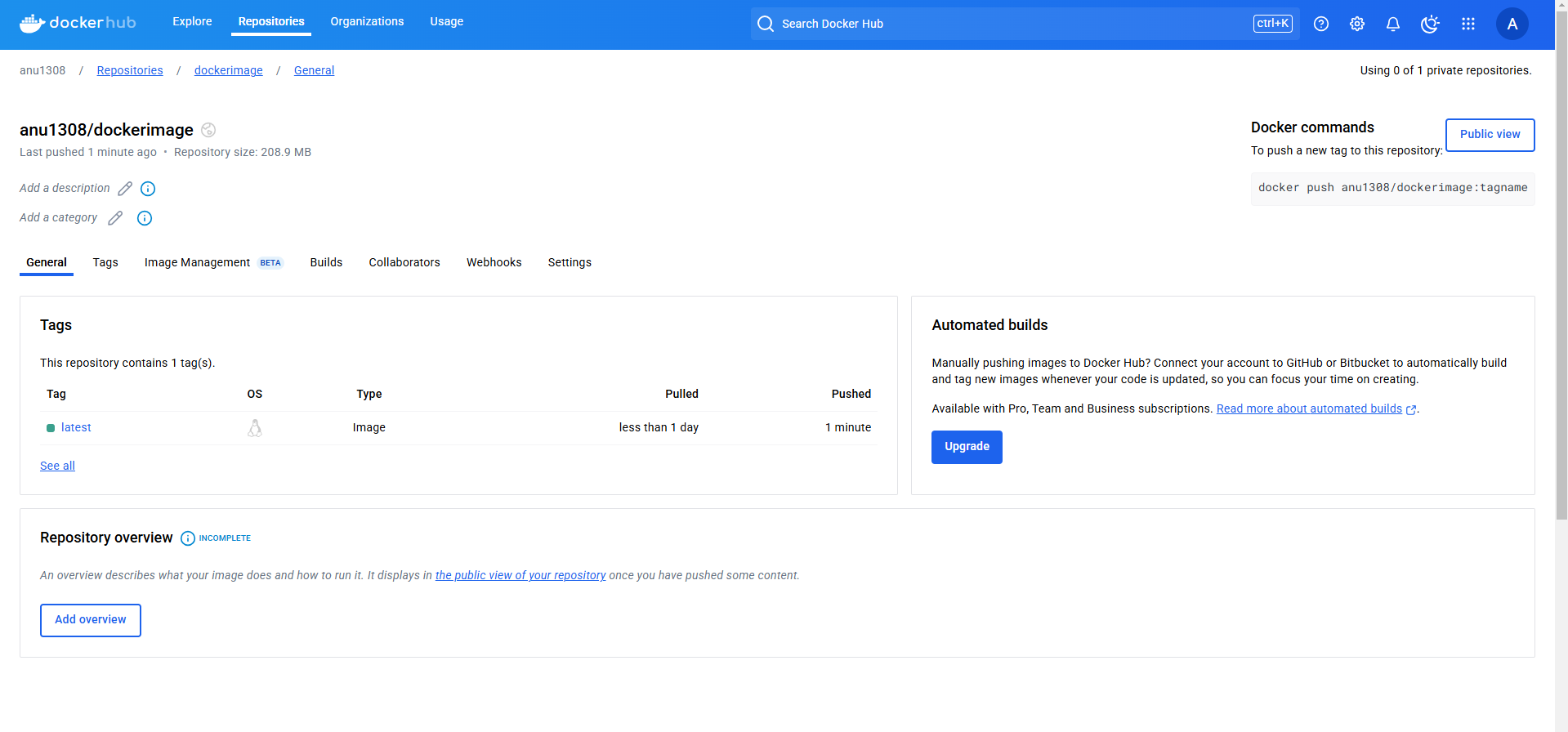
**Step 9:- Push the Image to DockerHub and share it with the world**

1. docker push anu1308/dockerimage:latest



**Output:-**

****



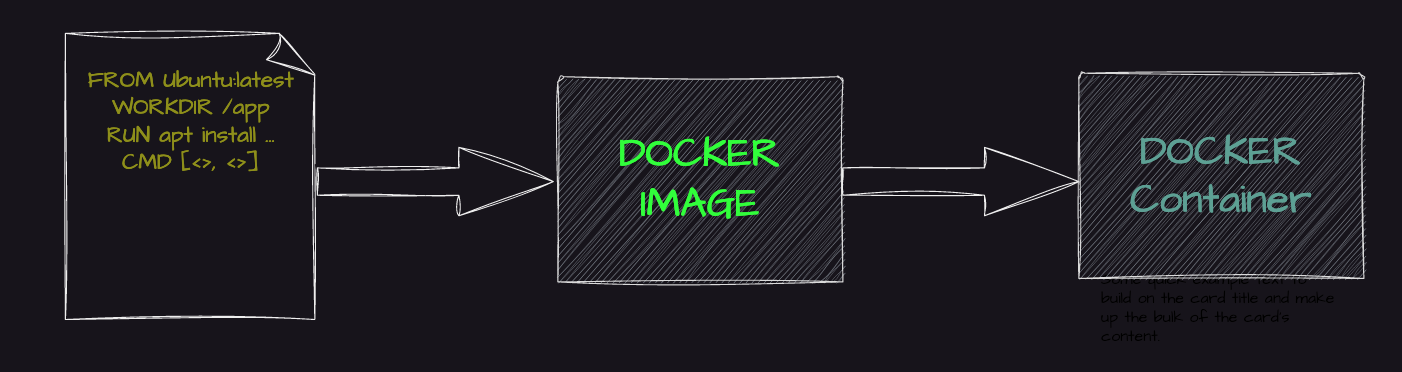
**Definitions:-**

**Docker LifeCycle**

We can use the above Image as reference to understand the lifecycle of Docker.

There are three important things,

1. docker build -> builds docker images from Dockerfile
2. docker run -> runs container from docker images
3. docker push -> push the container image to public/private regestries(docker hub) to share the docker images.

[](https://user-images.githubusercontent.com/43399466/217511949-81f897b2-70ee-41d1-b229-38d0572c54c7.png)

## What is a container ?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

## Why are containers light weight ?

Containers are lightweight because they use a technology called containerization, which allows them to share the host operating system's kernel and libraries, while still providing isolation for the application and its dependencies. This results in a smaller footprint compared to traditional virtual machines, as the containers do not need to include a full operating system. Additionally, Docker containers are designed to be minimal, only including what is necessary for the application to run, further reducing their size.

### What is Docker ?

Docker is a containerization platform that provides easy way to containerize your applications, which means, using Docker you can build container images, run the images to create containers and also push these containers to container regestries such as DockerHub

#### Docker daemon

The Docker daemon (dockerd) listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

#### Docker client

The Docker client (docker) is the primary way that many Docker users interact with Docker. When you use commands such as docker run, the client sends these commands to dockerd, which carries them out. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.

#### Docker Desktop

Docker Desktop is an easy-to-install application for your Mac, Windows or Linux environment that enables you to build and share containerized applications and microservices. Docker Desktop includes the Docker daemon (dockerd), the Docker client (docker), Docker Compose, Docker Content Trust, Kubernetes, and Credential Helper. For more information, see Docker Desktop.

#### Docker registries

A Docker registry stores Docker images. Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.

When you use the docker pull or docker run commands, the required images are pulled from your configured registry. When you use the docker push command, your image is pushed to your configured registry. Docker objects

When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects. This section is a brief overview of some of those objects.

#### Dockerfile

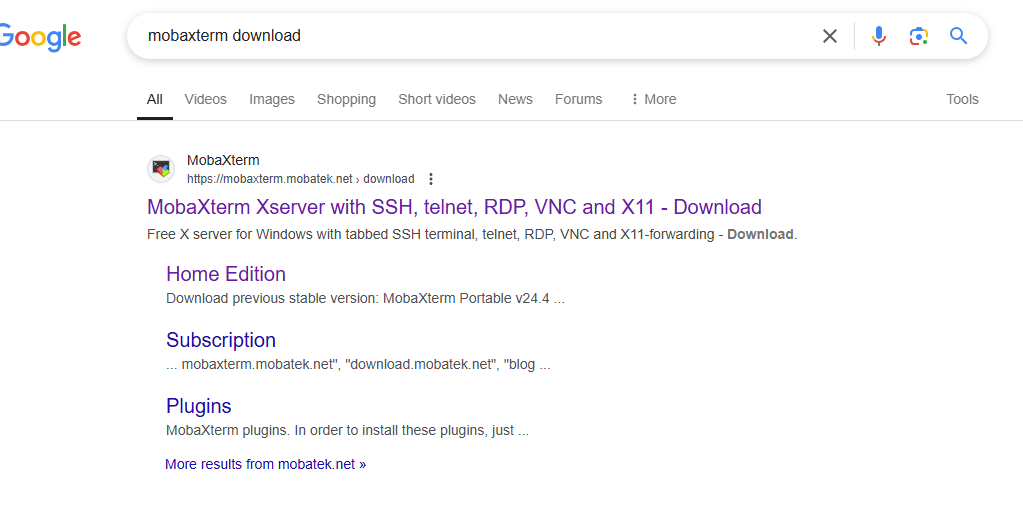
Dockerfile is a file where you provide the steps to build your Docker Image.

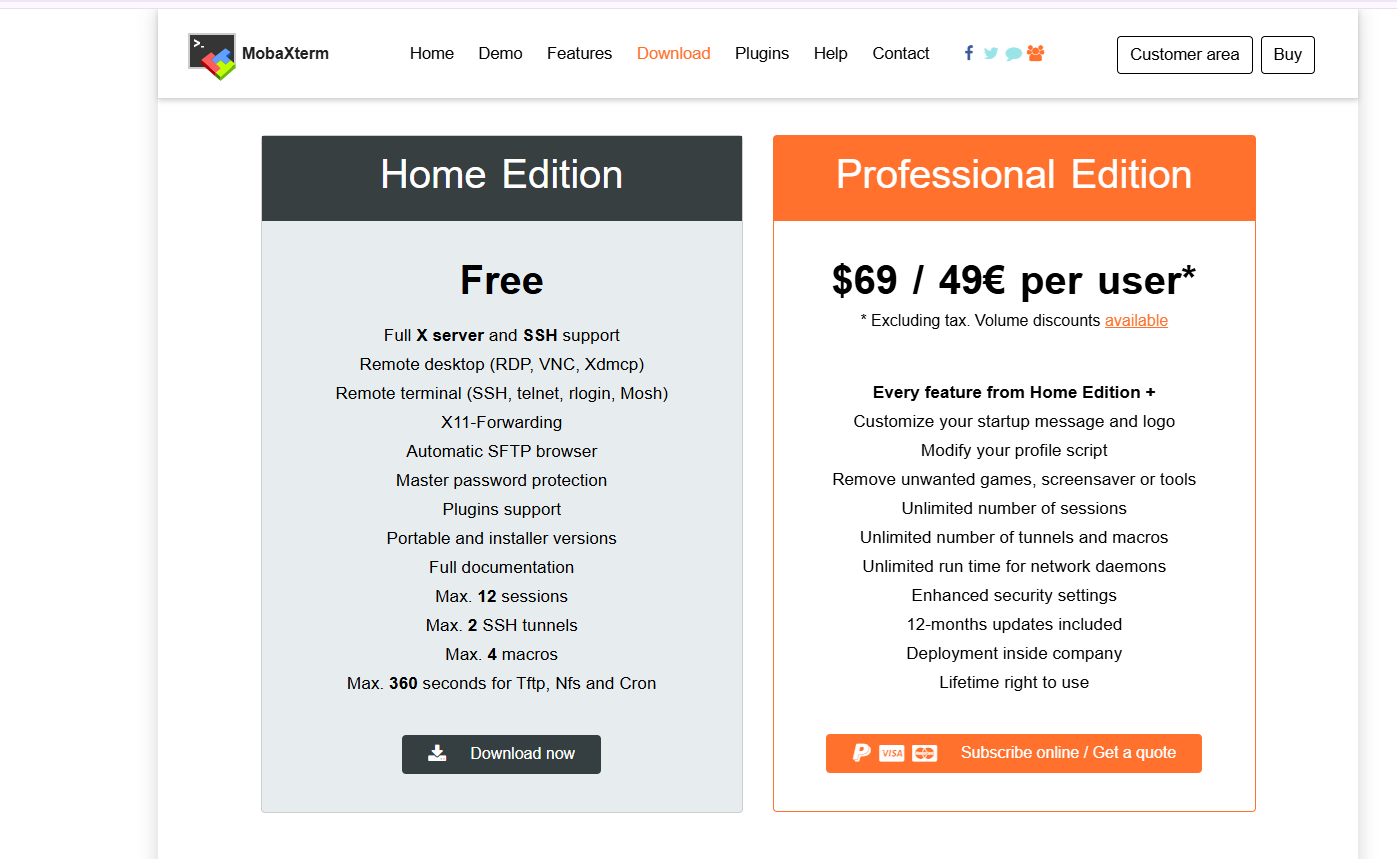
#### Images

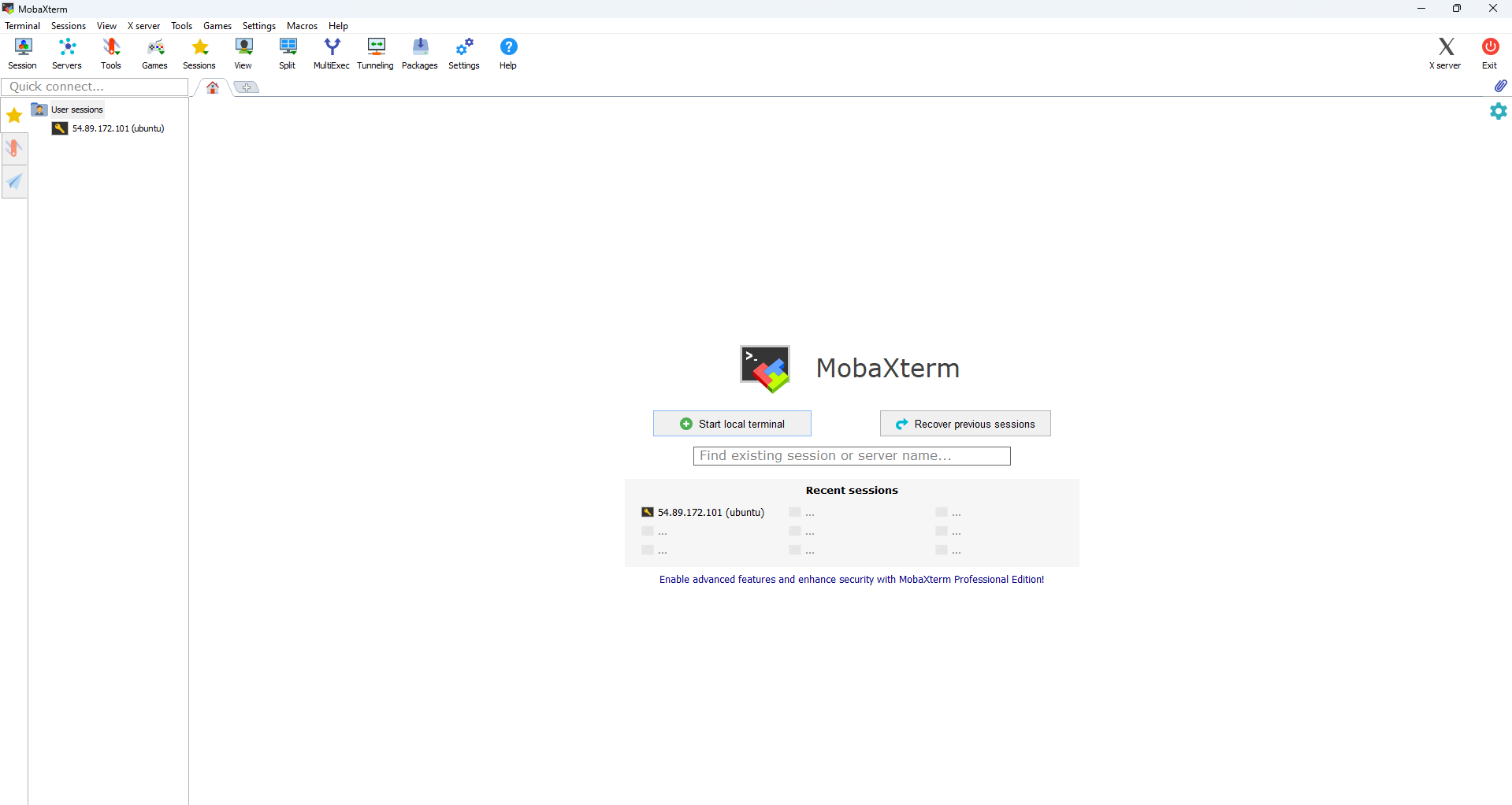
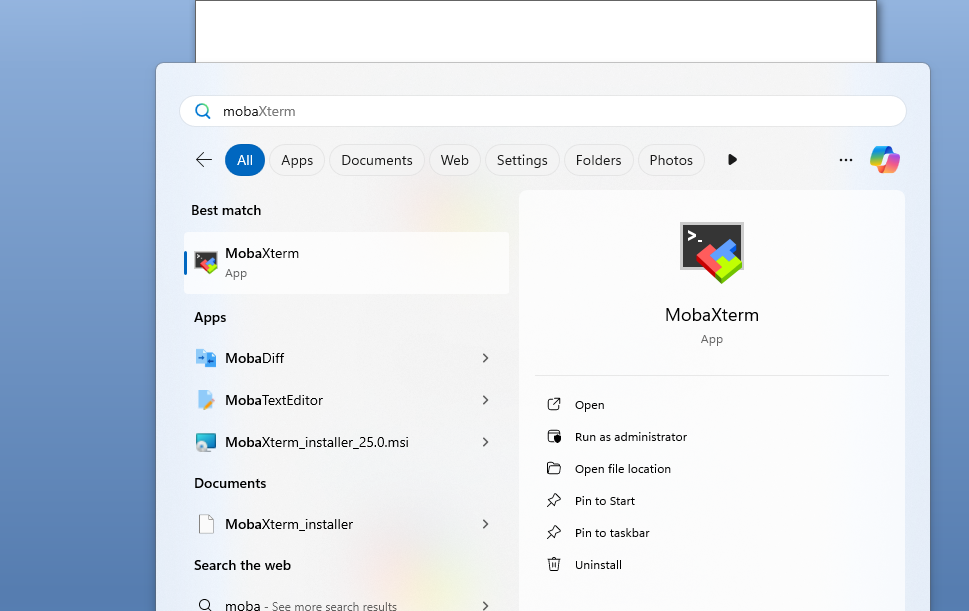
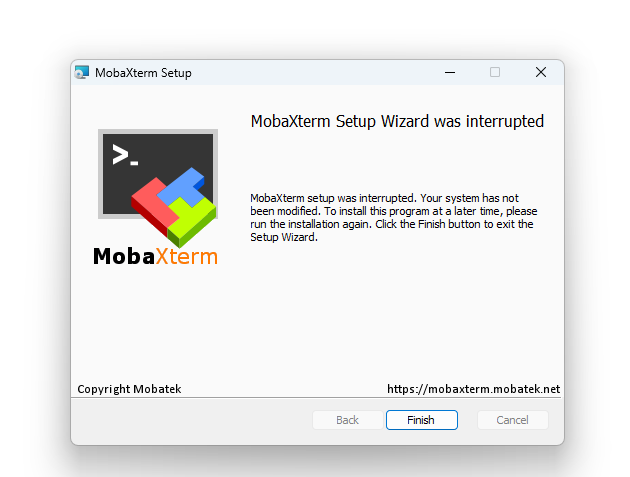
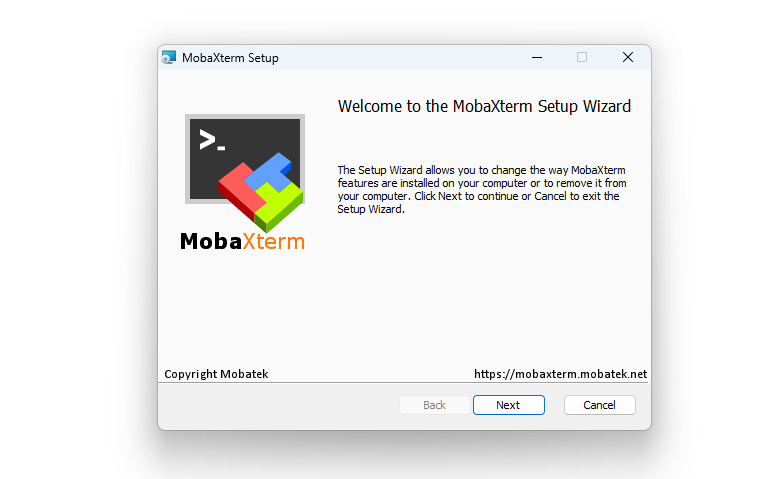
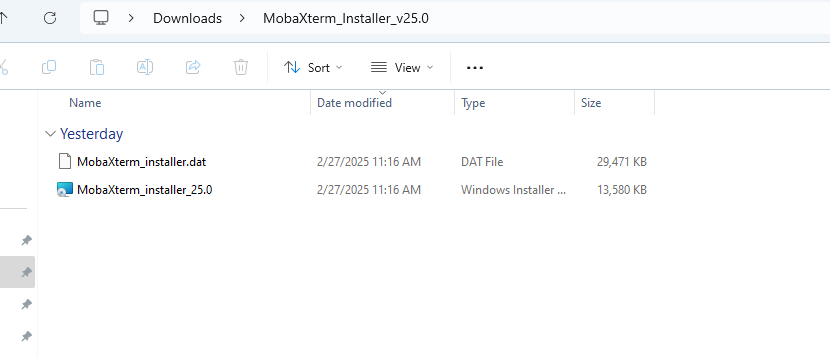
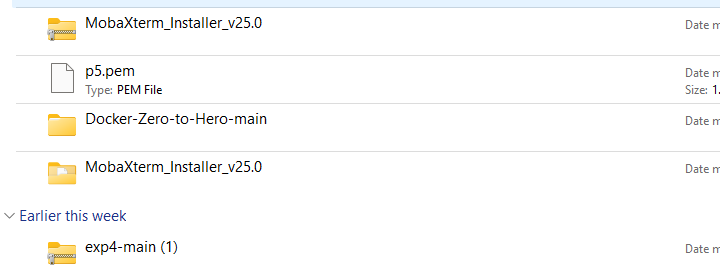
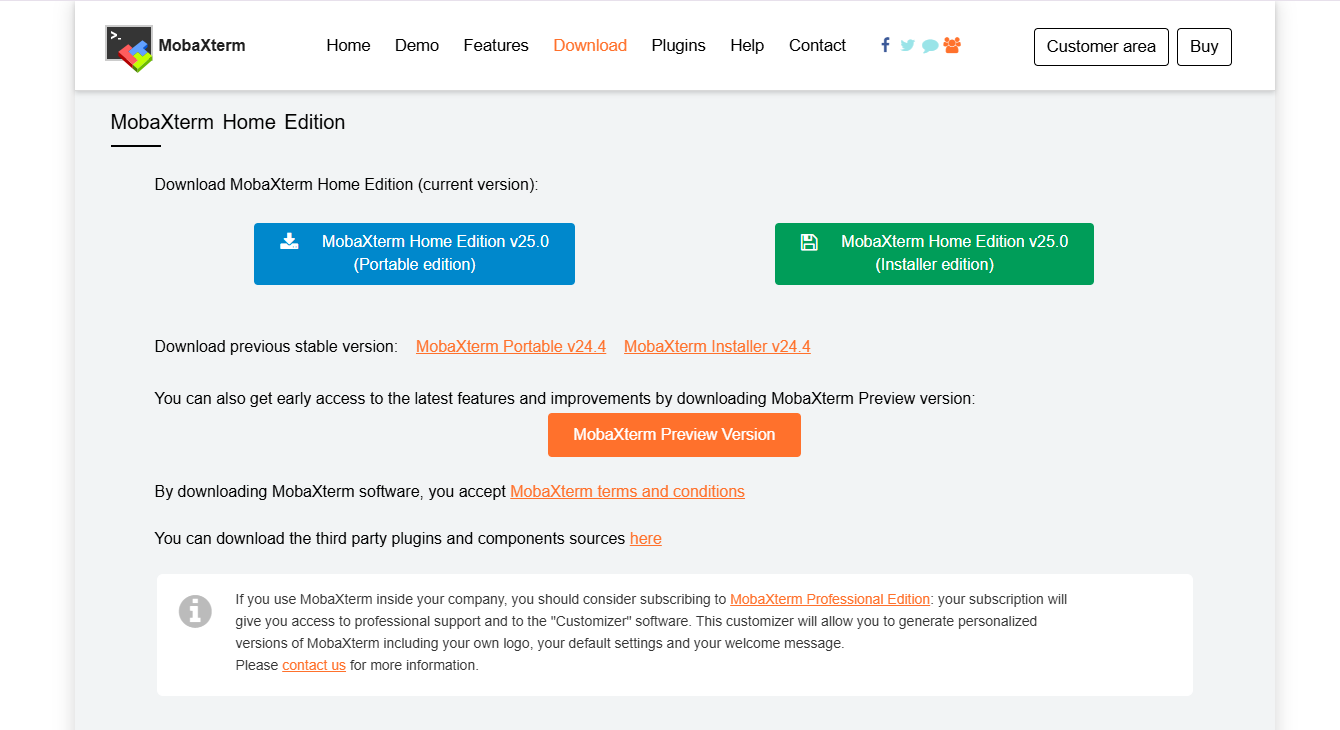
An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization. For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.

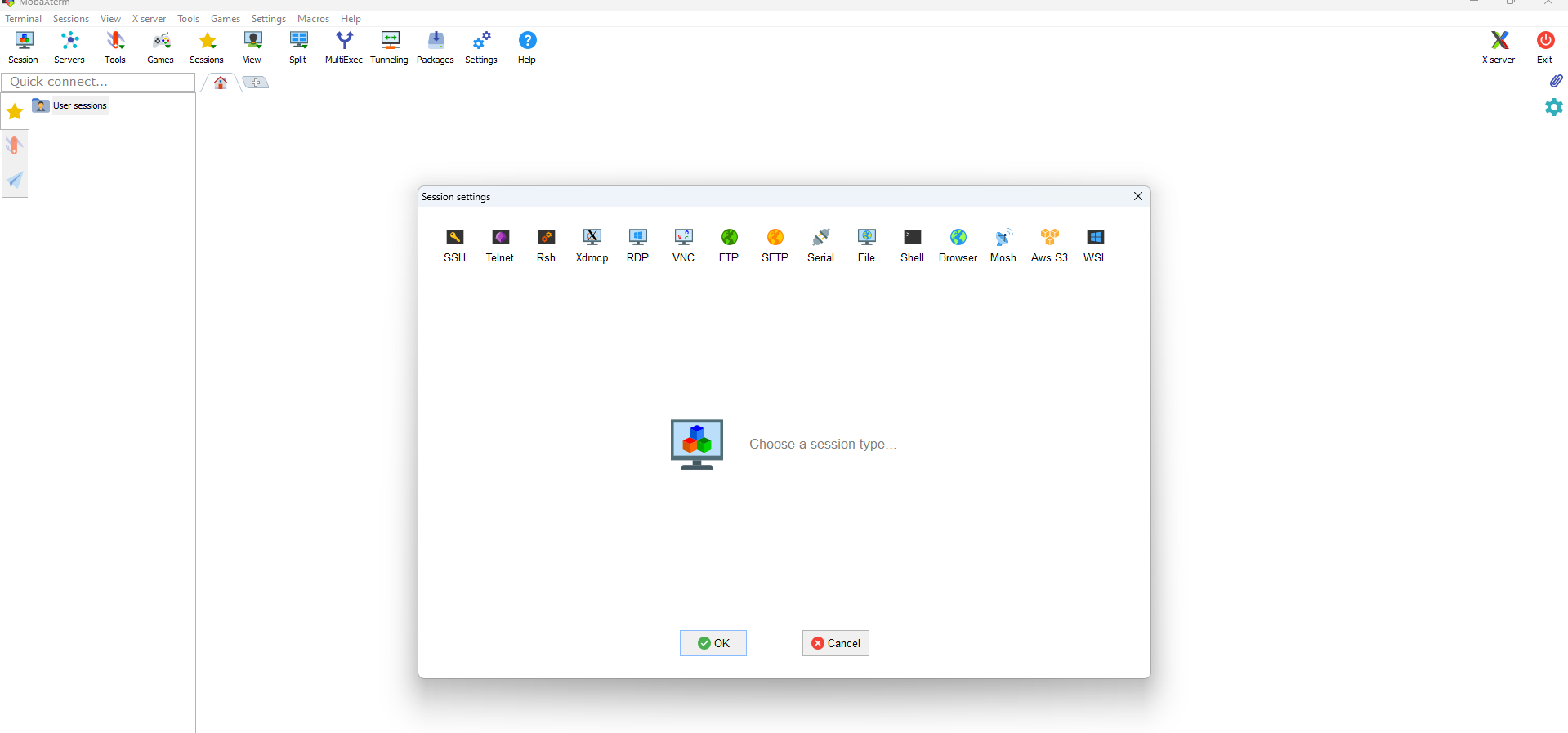
Below Images for Understanding purpose

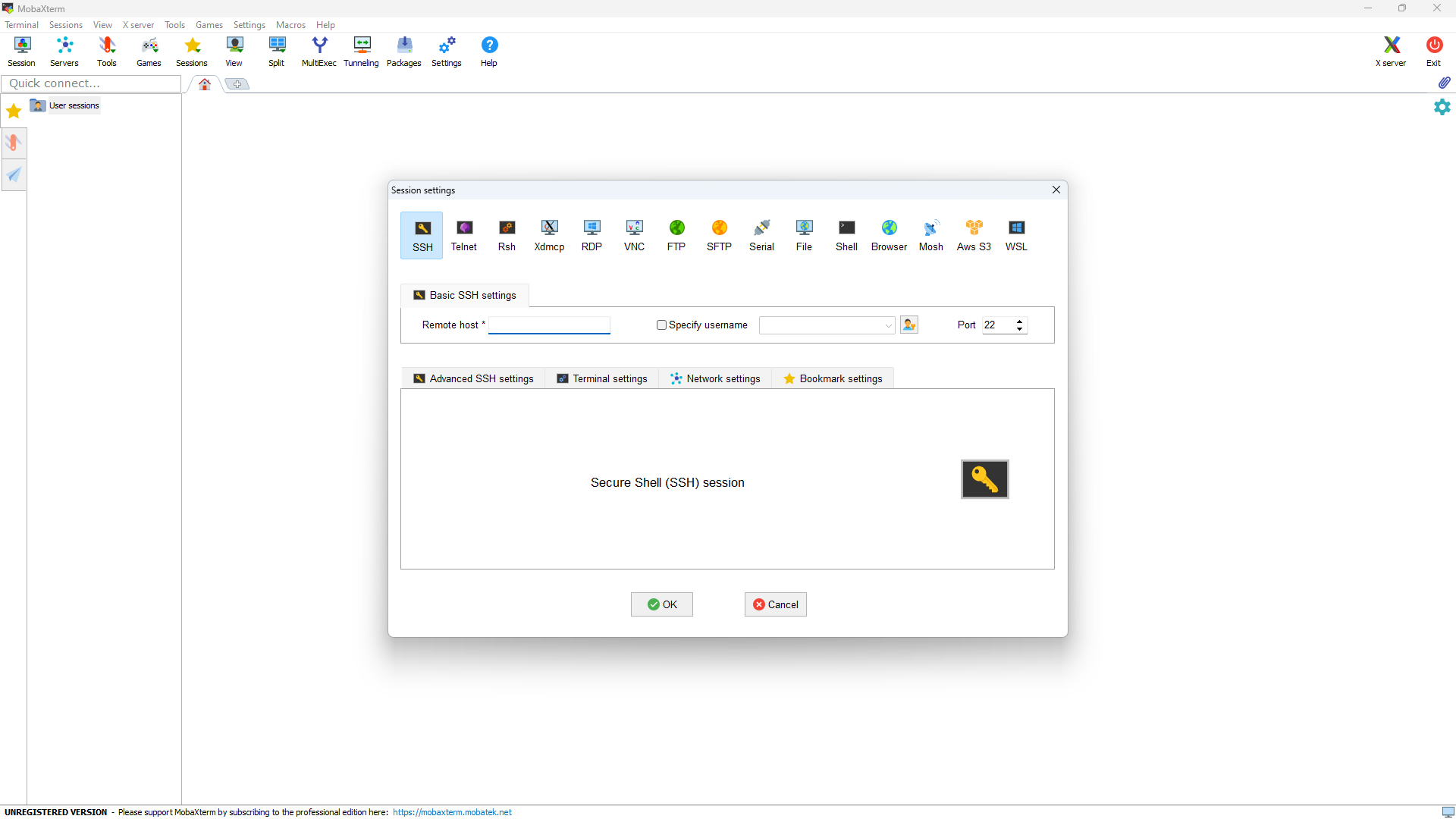
**Connecting AWS Instance to Mobaxterm**



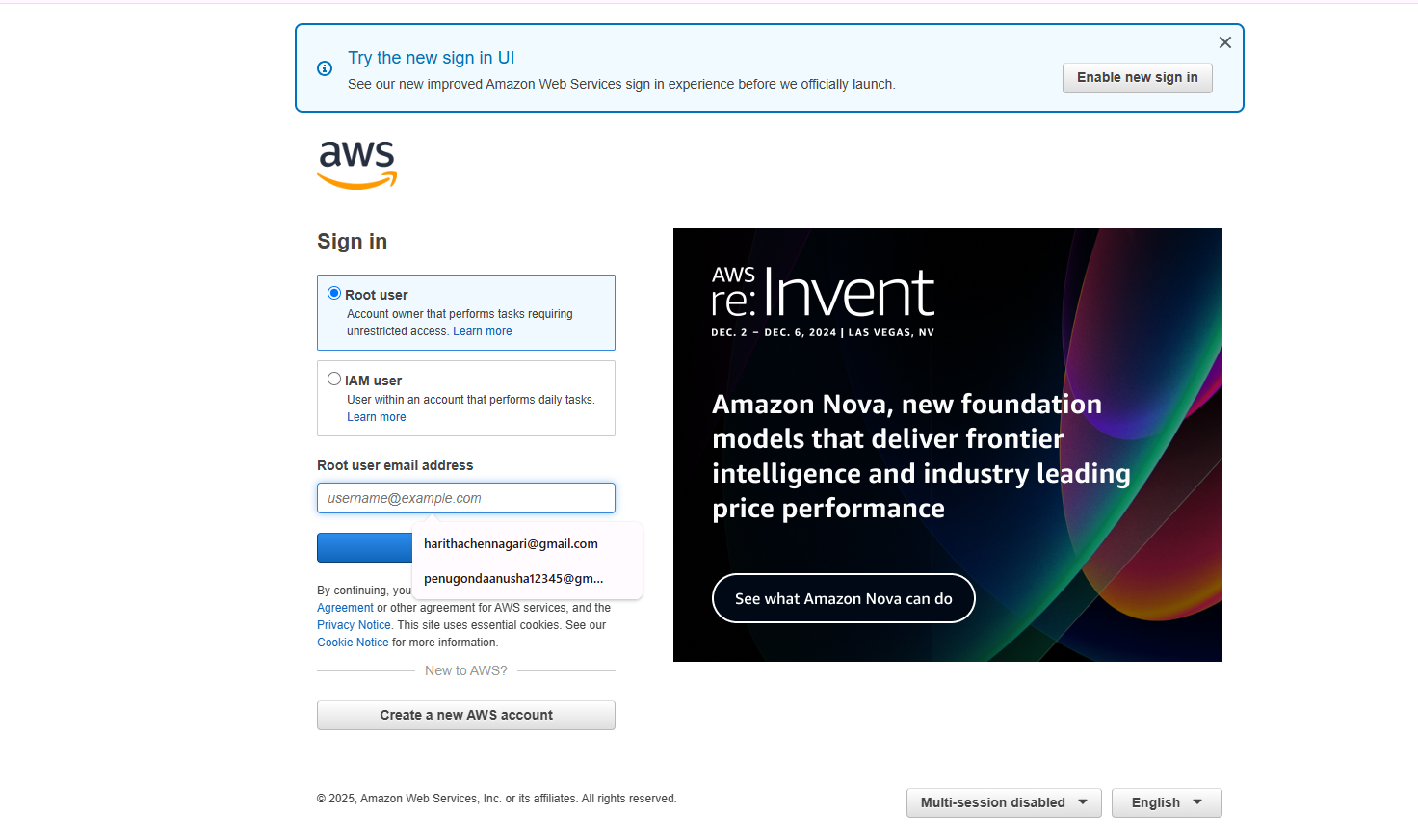


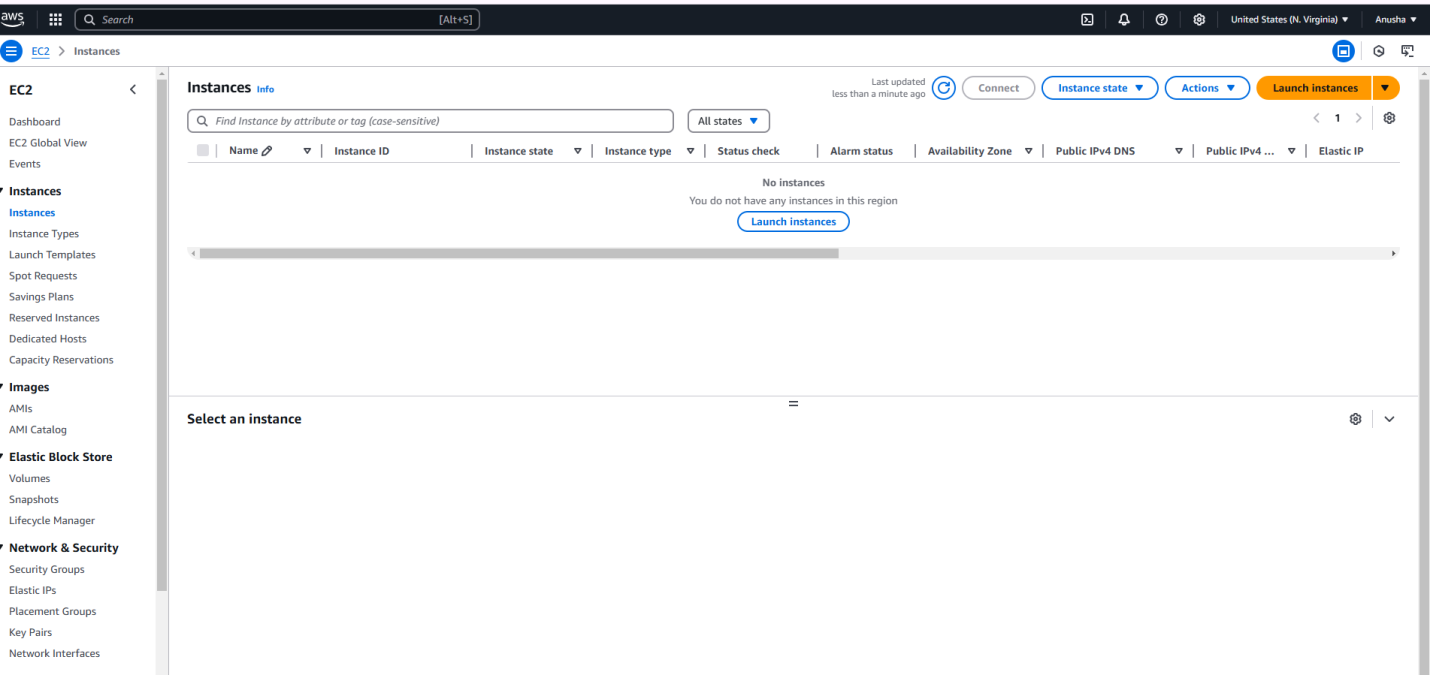


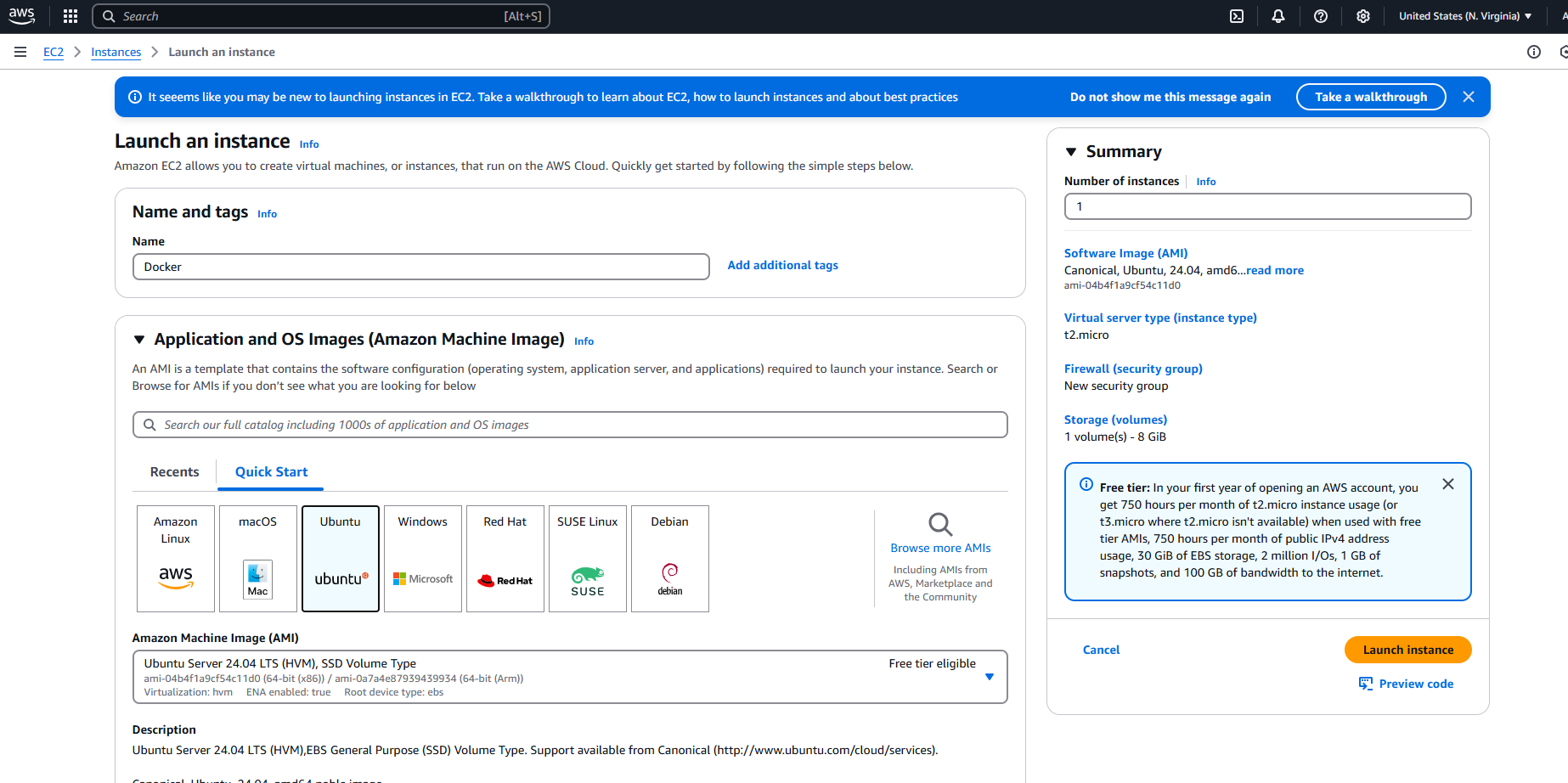


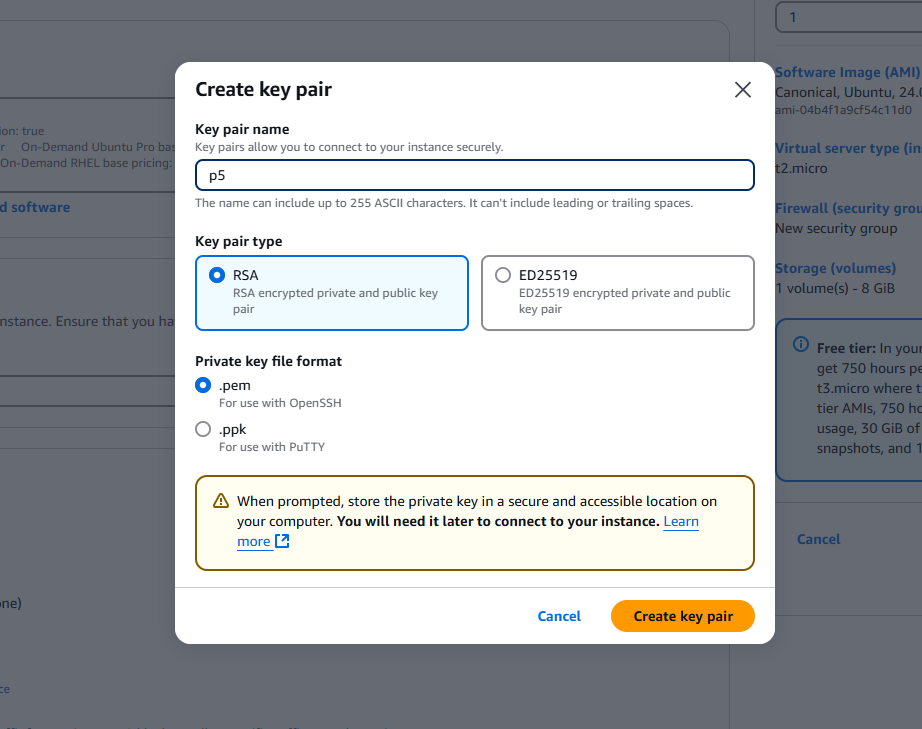


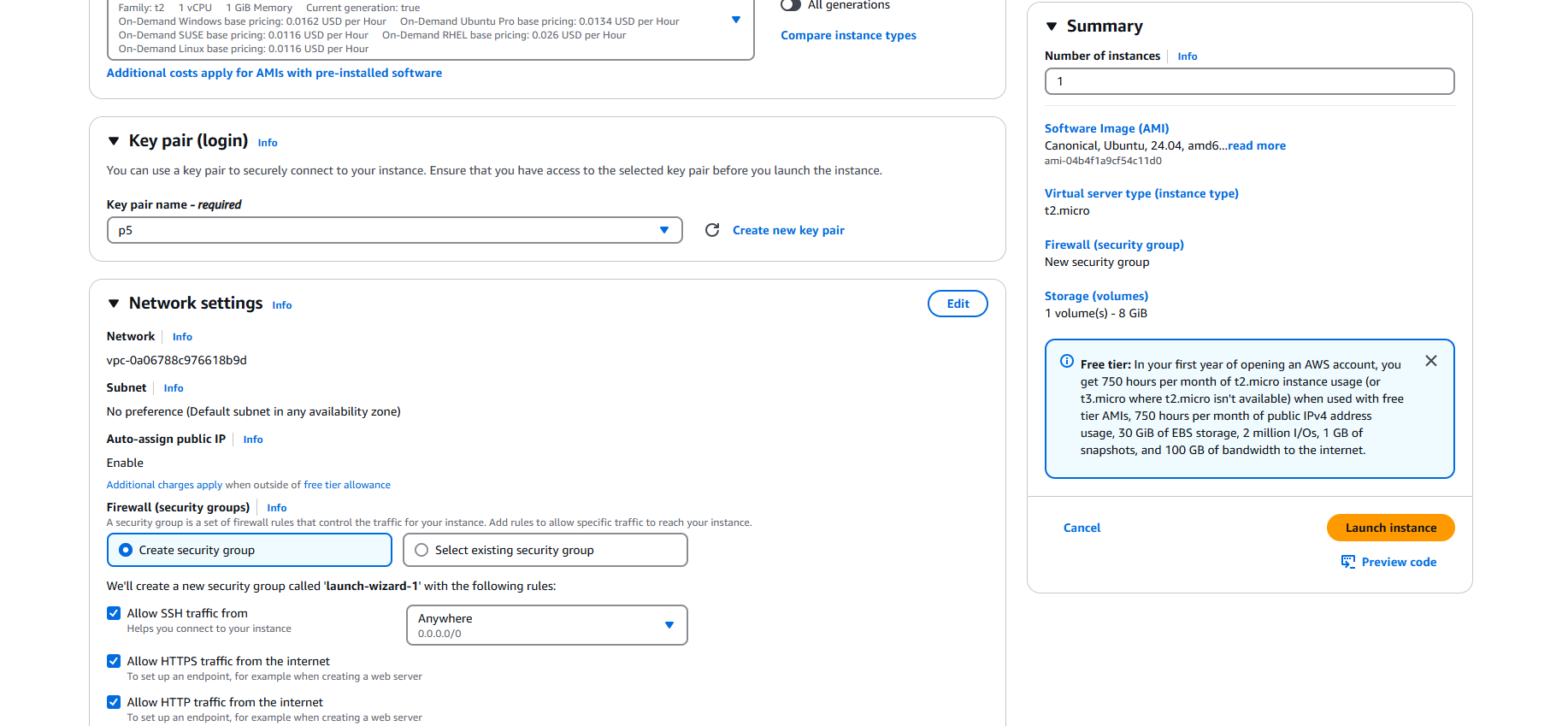
Remote host we want public key and .pem file also so login to AWS Account



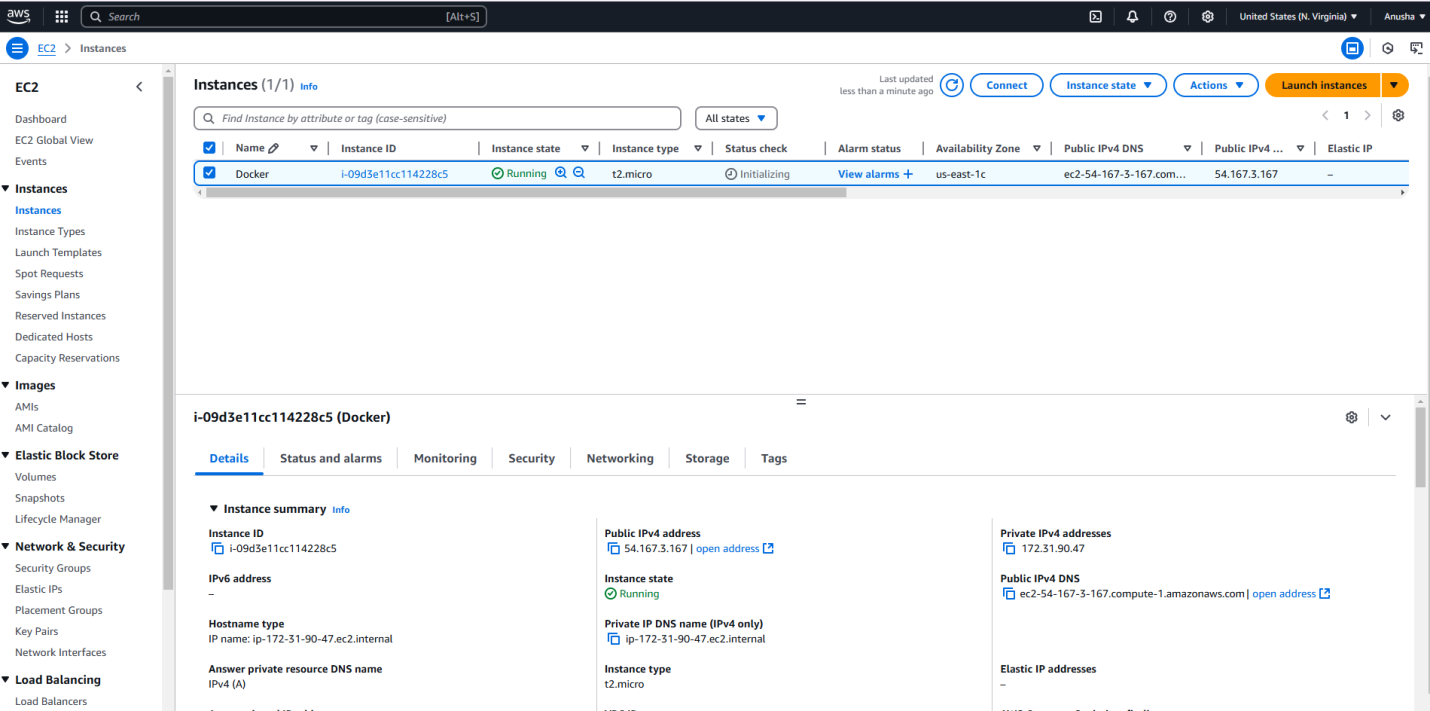


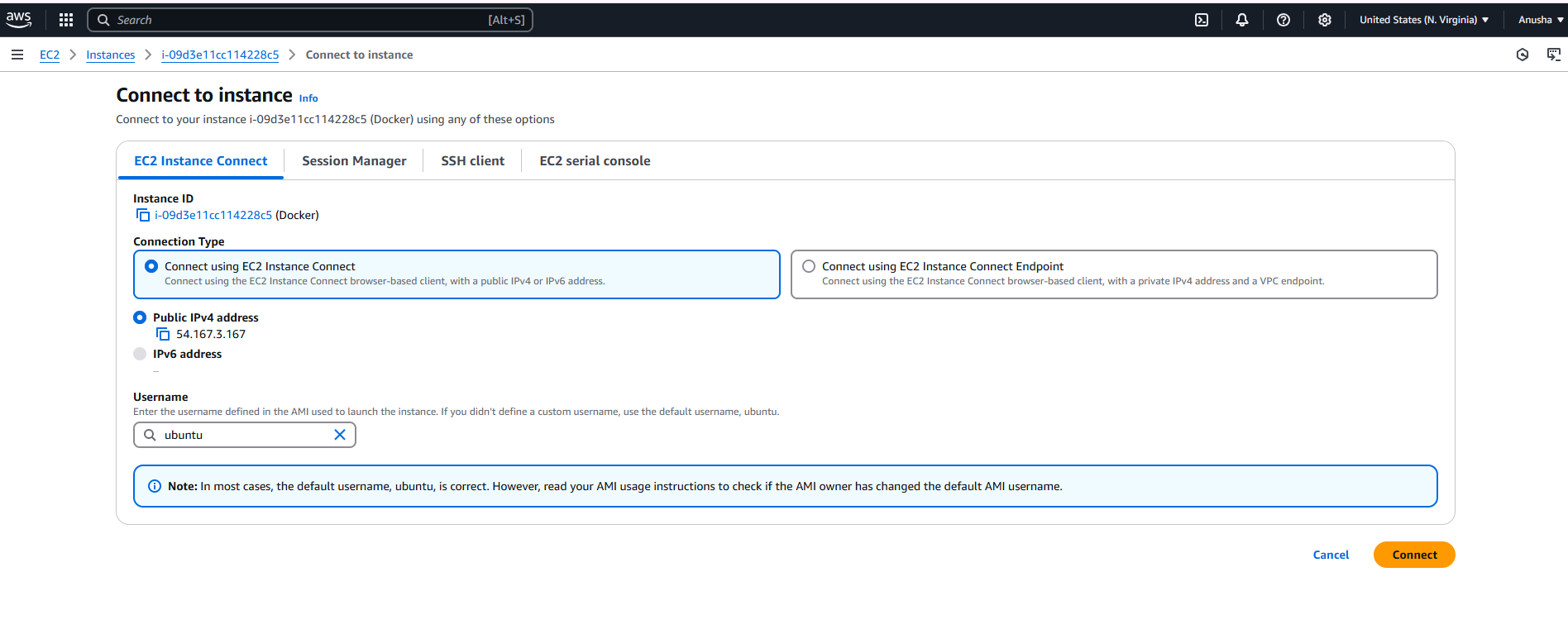


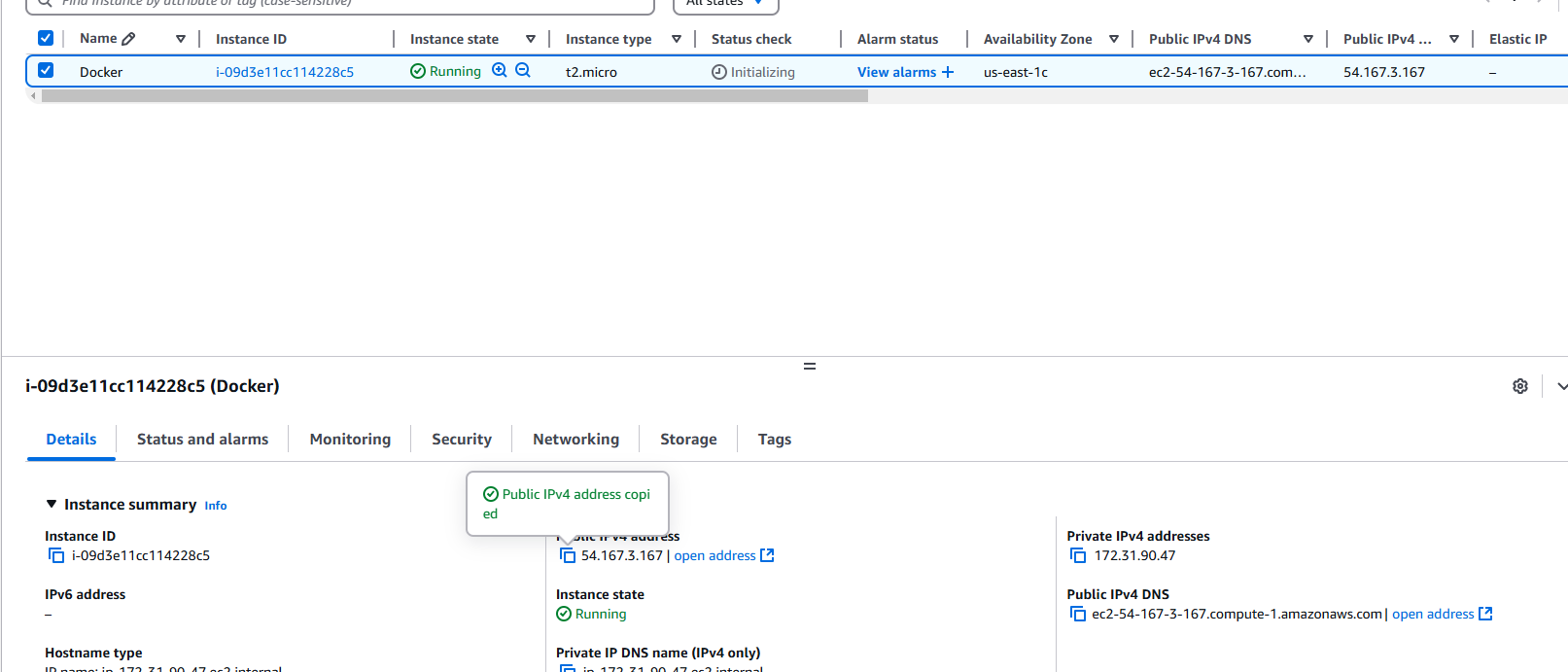


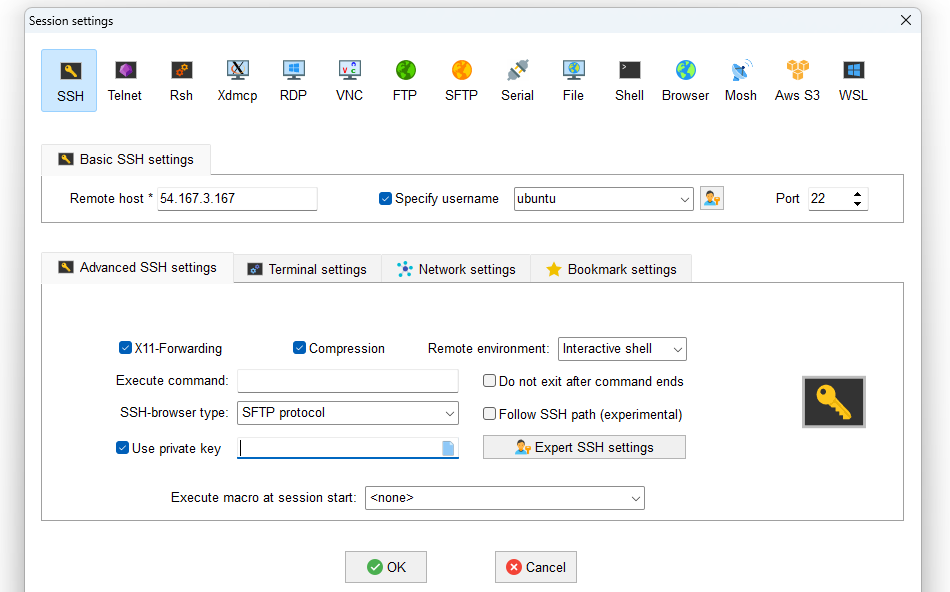


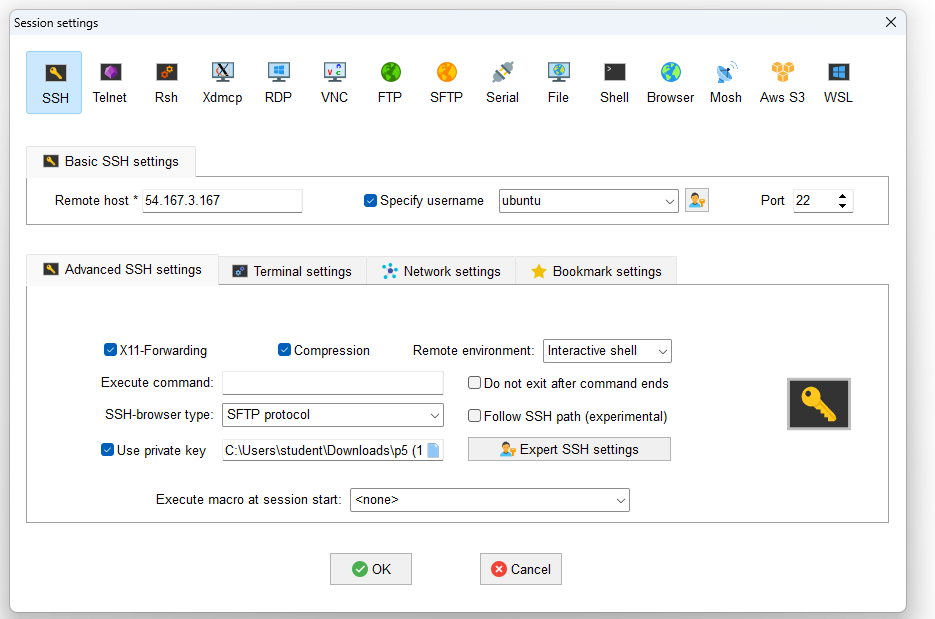
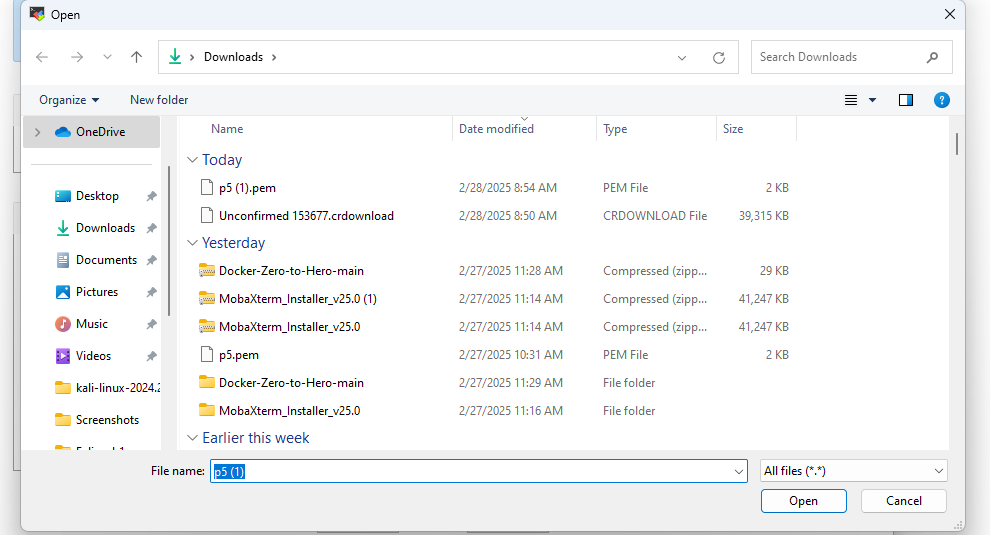
Here you can Directly connect Ubuntu also but better use mobaxterm

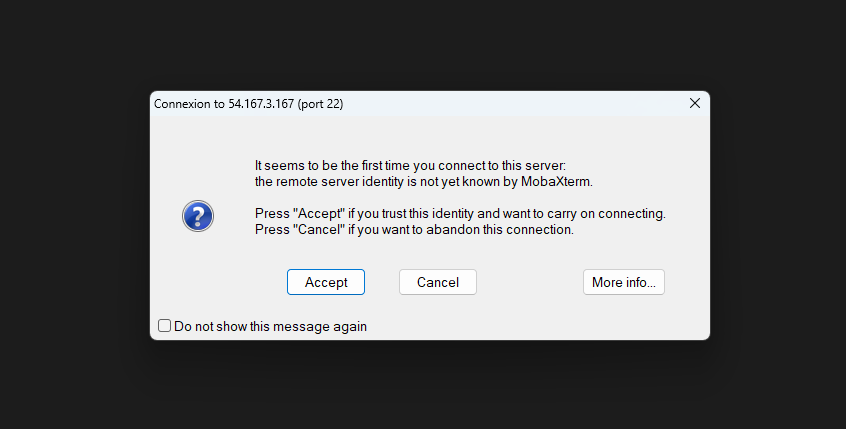


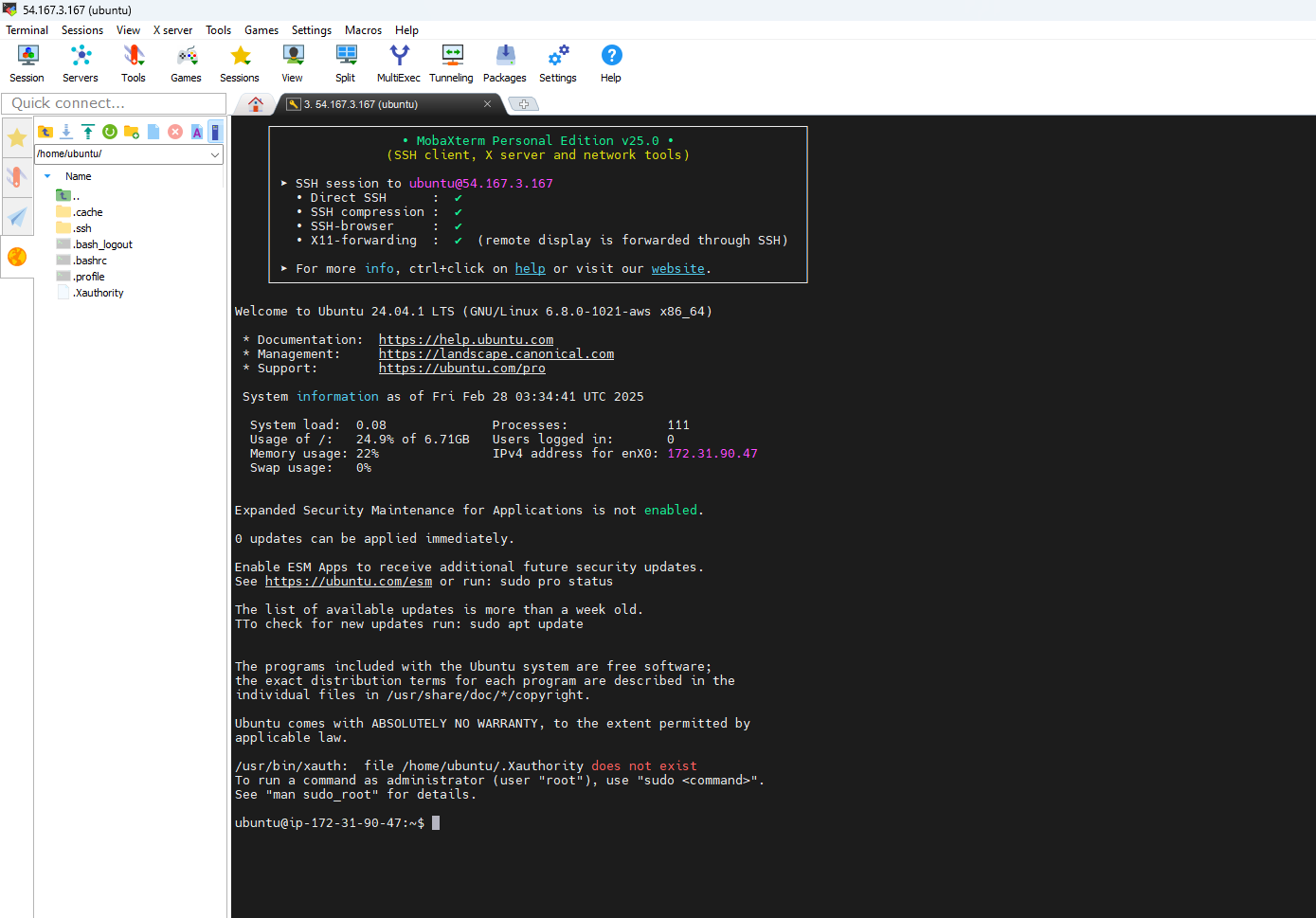












**Create Docker Hub Account and create repository in Docker Hub**

