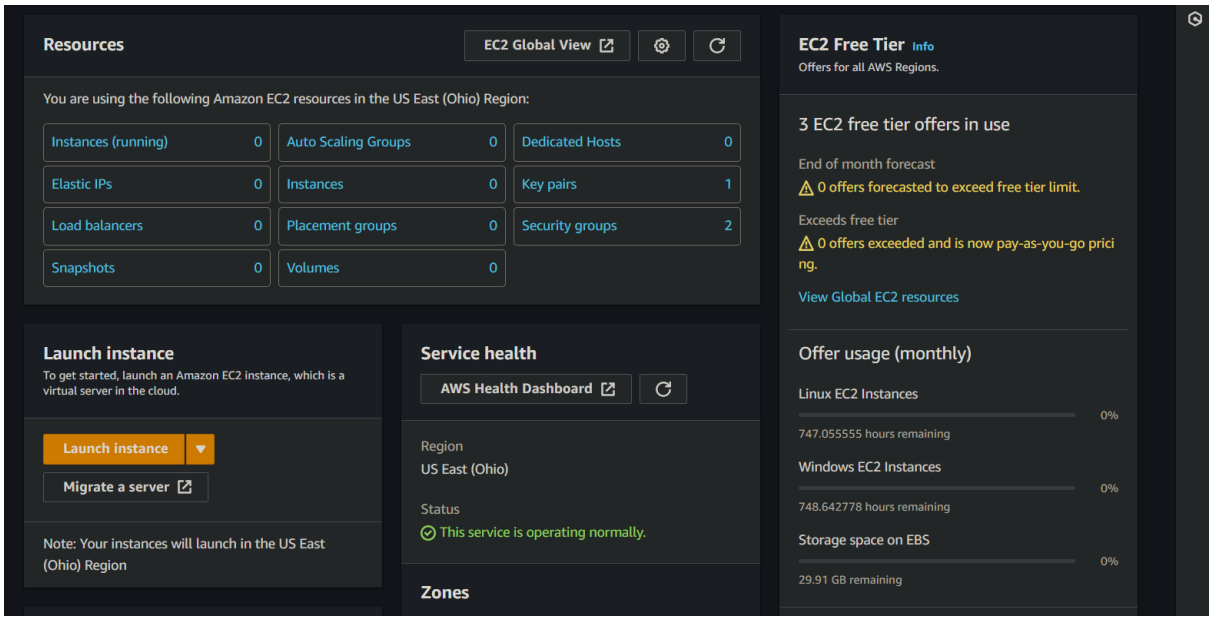
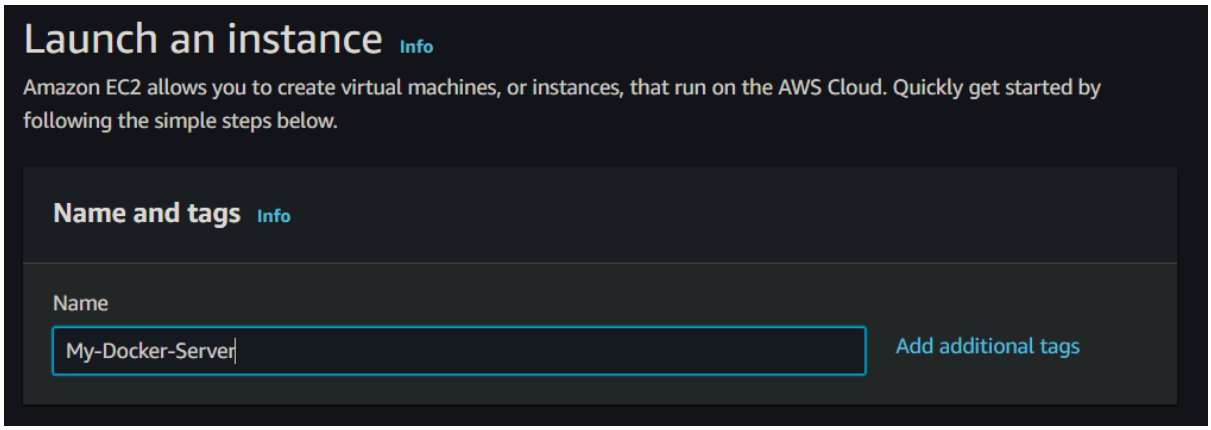
Project Title: - Deploy a Basic Webpage using a Docker Container

Step:-1 Go to AWS Management Console and logged using your credentials

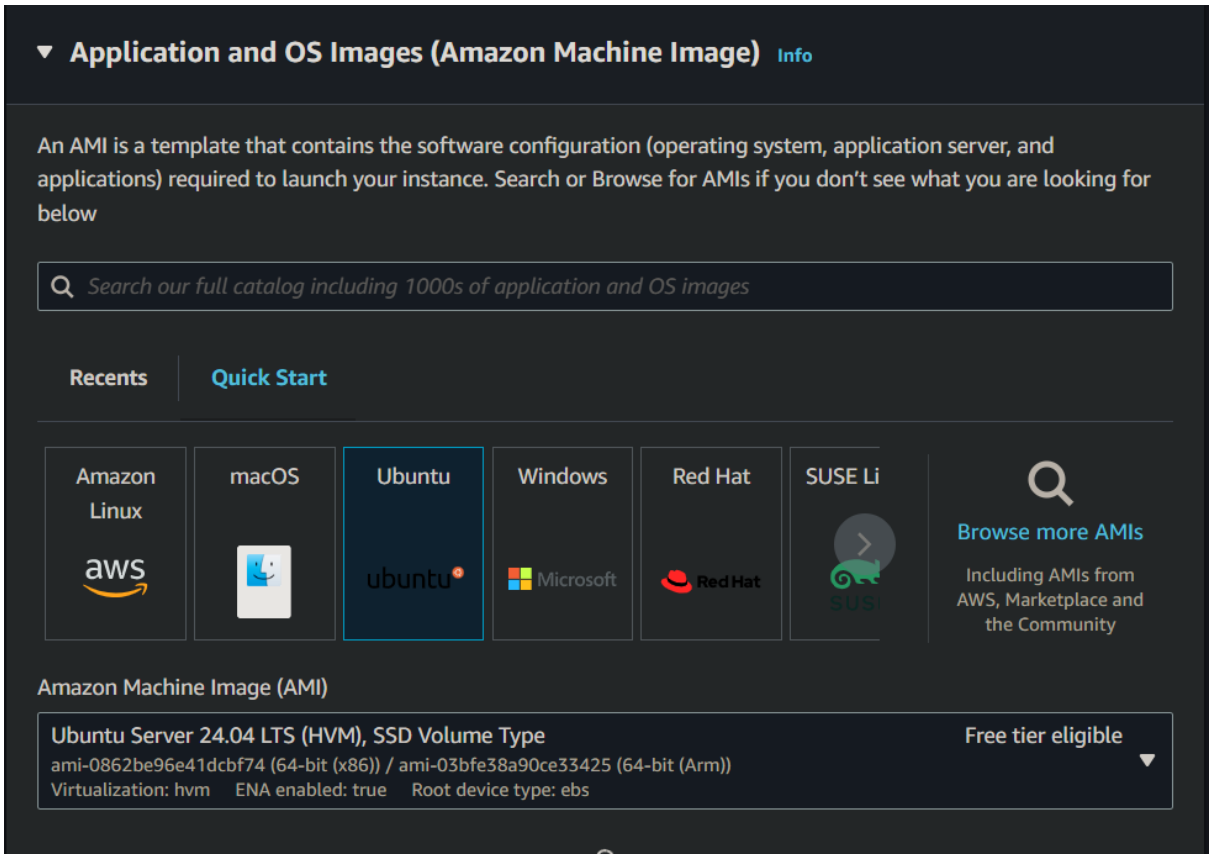
Step:-2 Go to EC2 dashboard and click on launch instance.



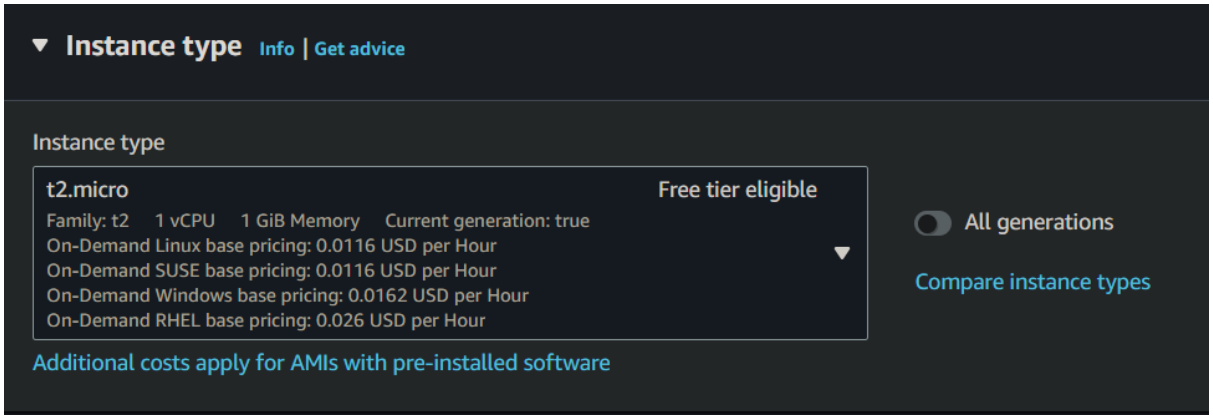
Step:-3 Launch an Instance → Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. [Name and tags]



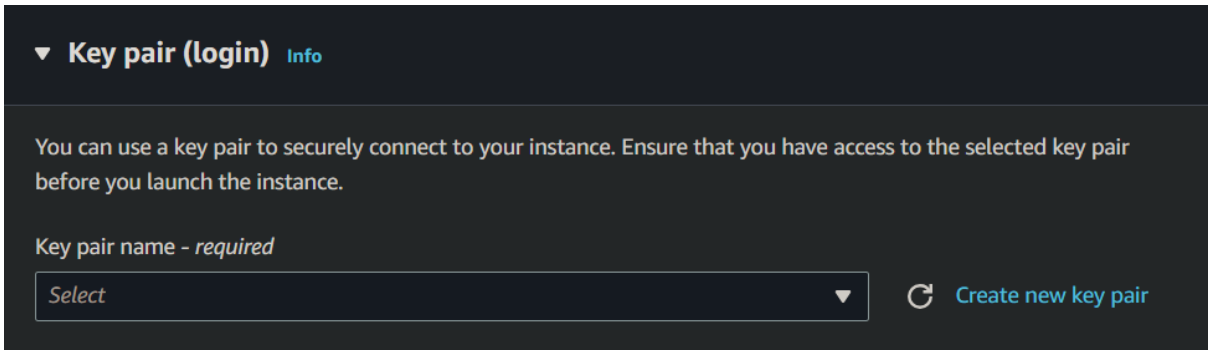
Step:-4 Select Application and OS Images [Amazon Machine Image] → An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or browse for AMIs if you don't see what you are looking for below.

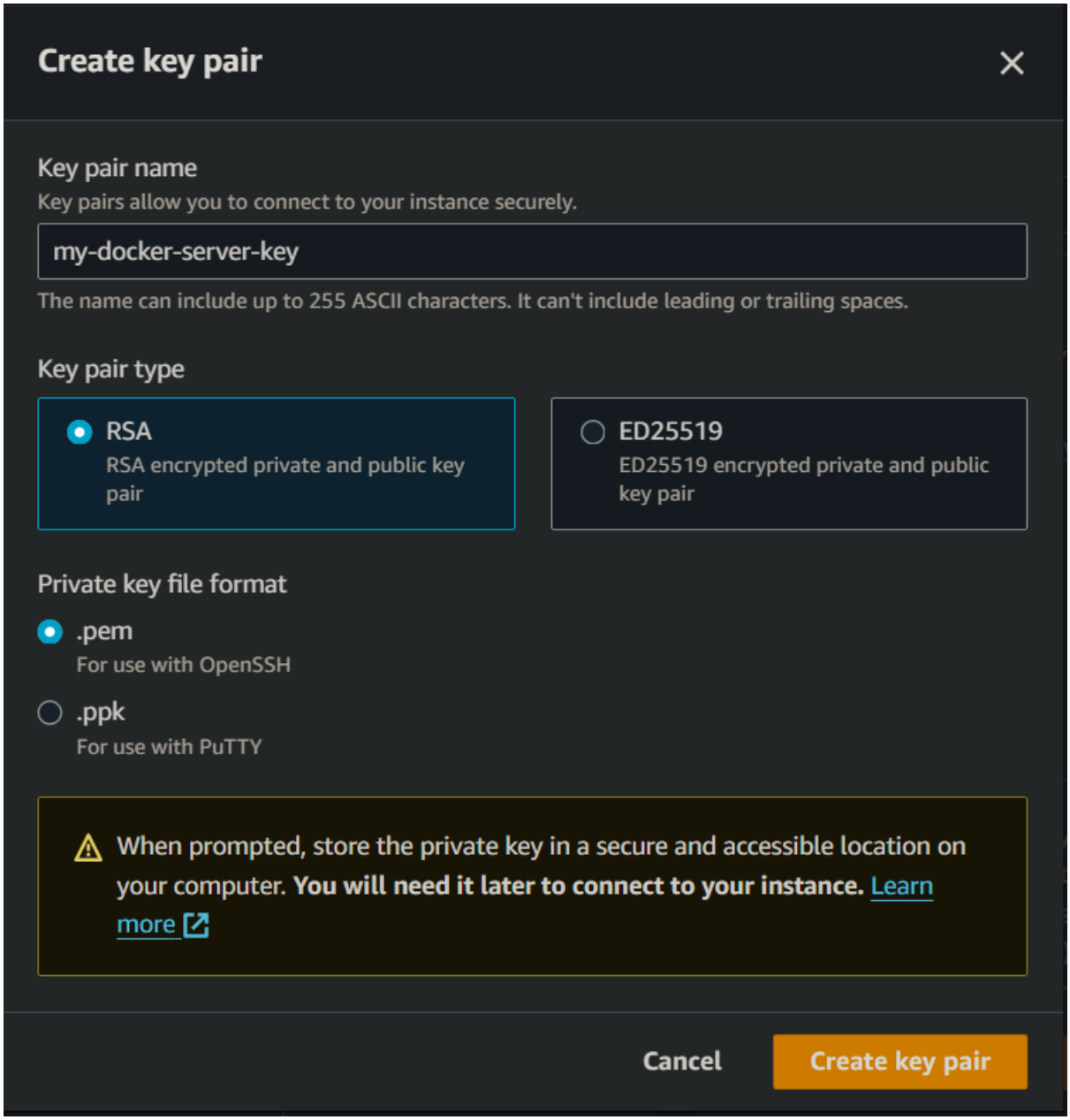


Step:-5 Select Instance Type

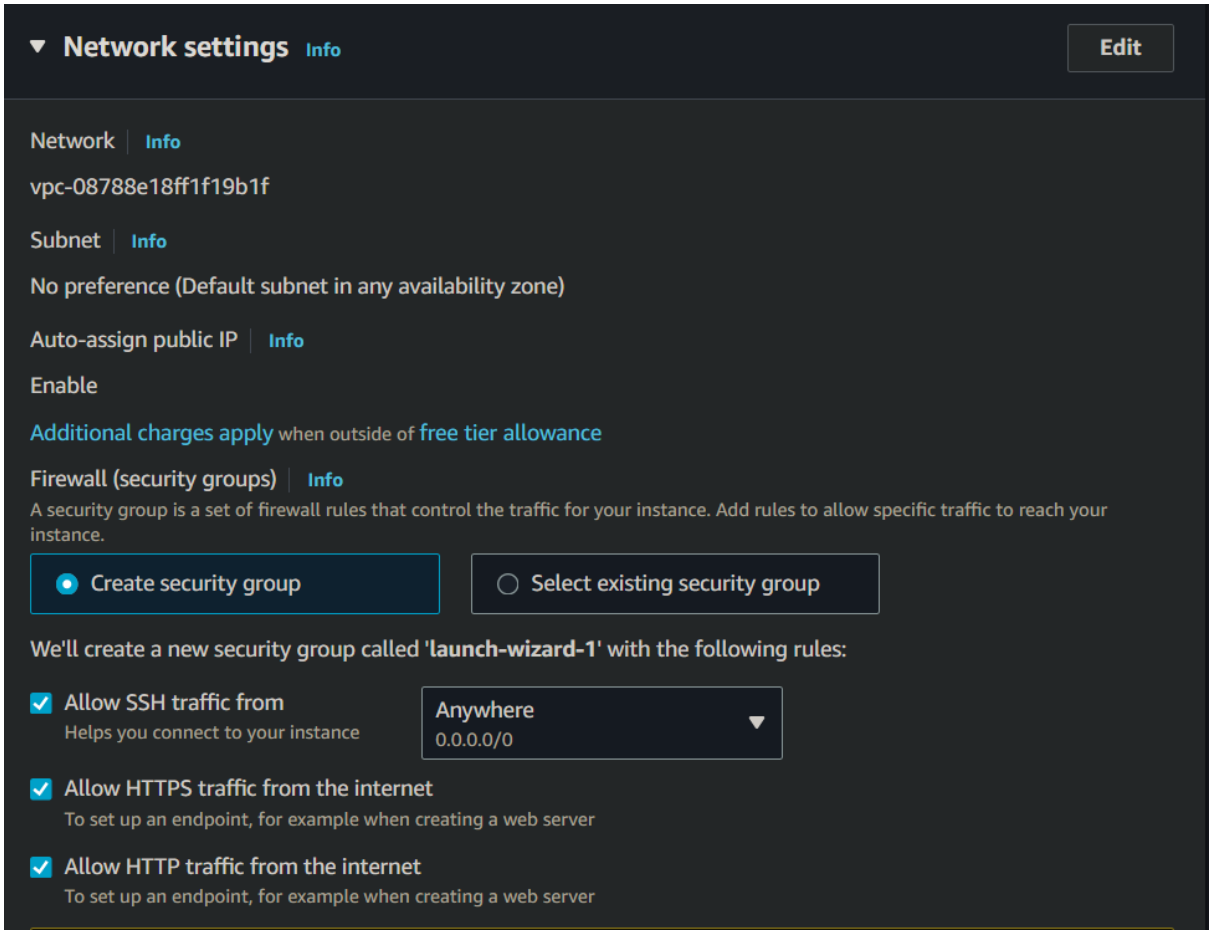


Step:-6 Key pair (login) [Create new key pair] you can use key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.





Step:-7 Network Settings → Click on edit button → Create new security group name as "**my-docker-server-sg**"



VPC - *required* | Info

vpc-08788e18ff1f19b1f (default)
172.31.0.0/16

Subnet | Info

No preference

Create new subnet

Auto-assign public IP | Info

Enable

Additional charges apply when outside of free tier allowance

Firewall (security groups) | Info

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

Security group name - *required*

my-docker-server-sg

This security group will be added to all network interfaces. The name can't be edited after the security group is created. Max length is 255 characters. Valid characters: a-z, A-Z, 0-9, spaces, and ._-:/()#,@[]+=&;{}!$*

Description - *required* | Info

launch-wizard-1 created 2024-07-28T14:01:37.700Z

Allow the following protocol SSH to anywhere because to connect to remote server from local machine, also allow HTTP & HTTPS to anywhere for inbound traffic on your webpage.

Inbound Security Group Rules

Security group rule 1 (TCP, 22, 0.0.0.0/0)

Remove

Type	Protocol	Port range
ssh	TCP	22

Source type	Source	Description - *optional*
Anywhere	Add CIDR, prefix list or security 0.0.0.0/0	e.g. SSH for admin desktop

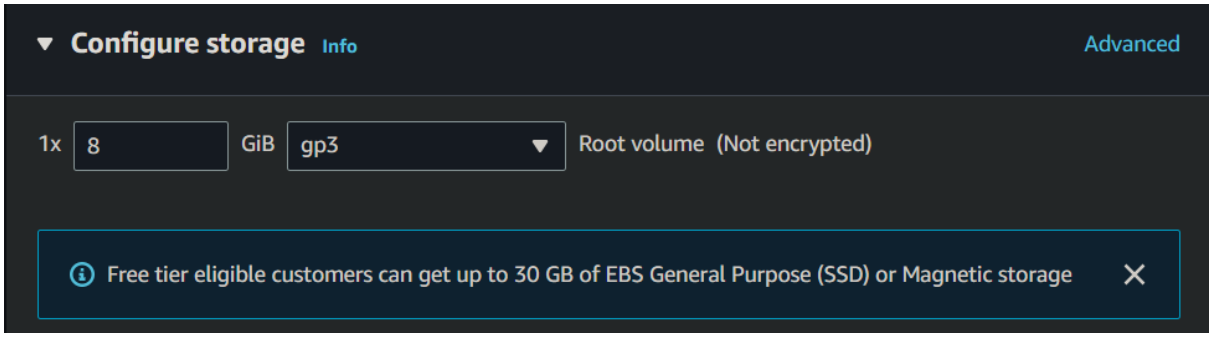
Security group rule 2 (TCP, 443, 0.0.0.0/0)

Remove

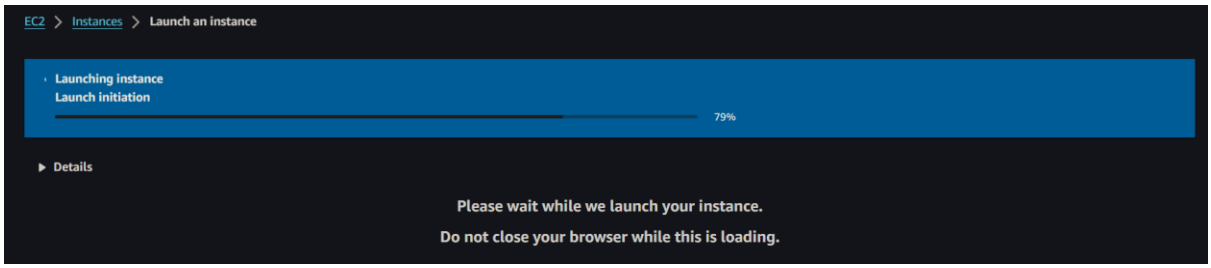
Type	Protocol	Port range
HTTPS	TCP	443

Source type	Source	Description - *optional*
Anywhere	Add CIDR, prefix list or security 0.0.0.0/0	e.g. SSH for admin desktop

Step:-8 Configure storage → free tier eligible customers can get up to 30 GB of EBS (Elastic Block Store) of General Purpose (SSD) or magnetic storage.

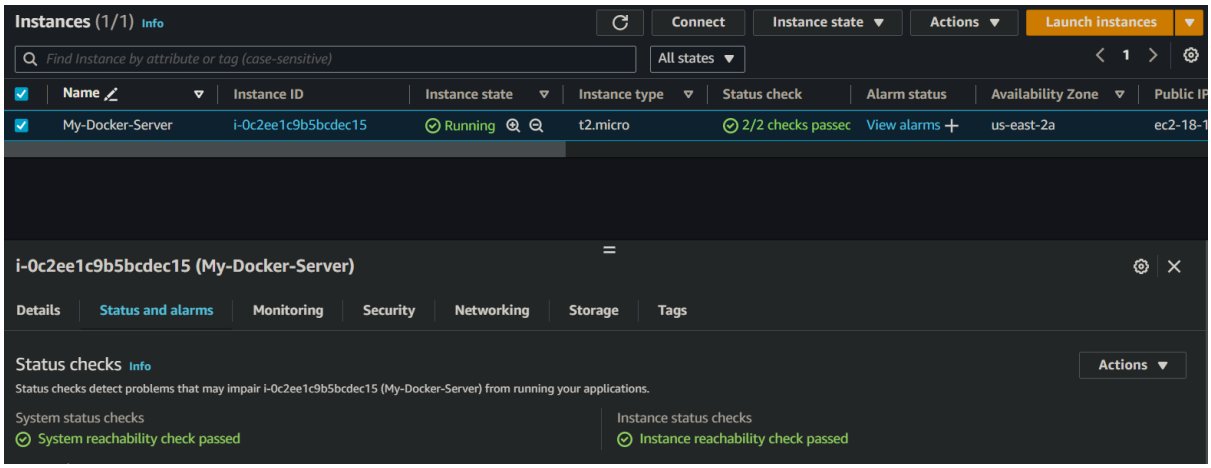


Step:-9 Launch instance



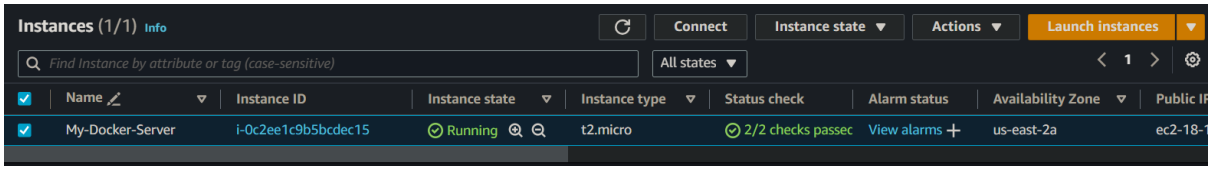
Step:-10 Instance is launch successfully click on instances to go on running instances.

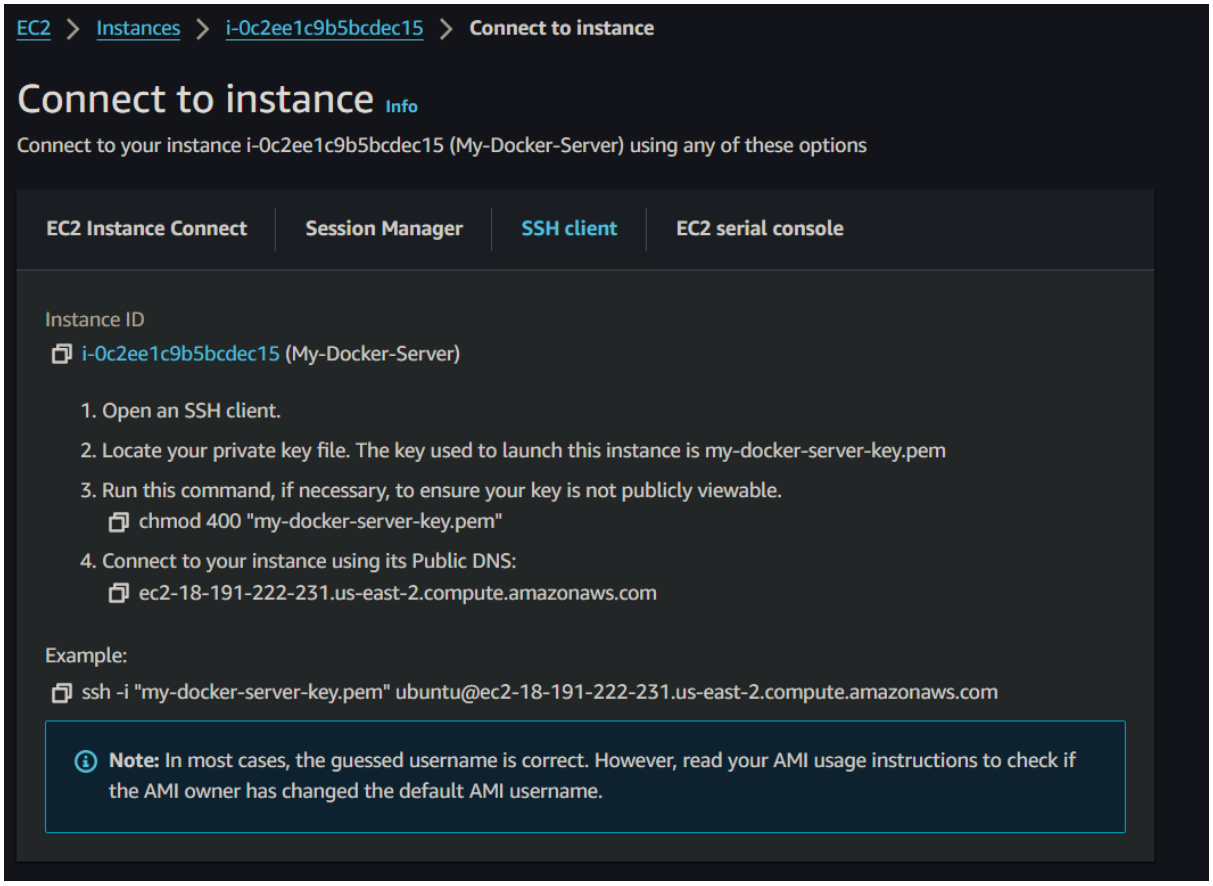
Step:-11 go to instance and click on the server name and check for status check as 2/2 checked passed that is means **system reachability checked passed** and **instance reachability checked passed**. If those are passed then your instance is ready to connect.



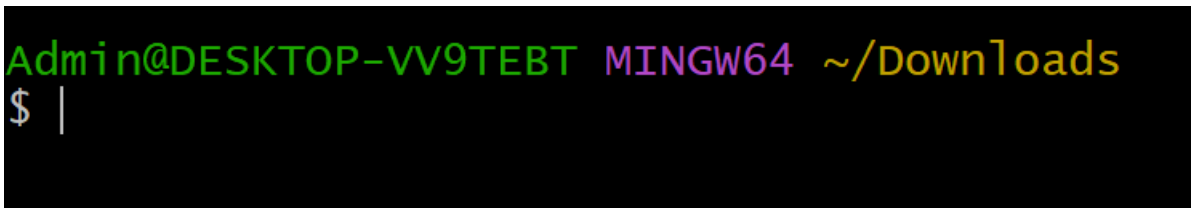
Step:-12: click on name i.e. My-Docker-Server → Connect → SSH Client → Copy the SSH command shown in the below diagram.

Command: - “ssh -i my-docker-server-key.pem Ubuntu@ec2-18-191-222-231.us-east-2.compute.amazonaws.com”





Step:-13 open your terminal from local machine and connect remote machine using SSH command “ssh -i my-docker-server-key.pem Ubuntu@ec2-18-191-222-231.us-east-2.compute.amazonaws.com”, here in my case I am using Git Bash to connect to my remote server from local server.



Change directory to ~/downloads folder where your “my-docker-server-key.pem” file is located and run here and here run SSH command to connect to remote server **“My-Docker-Server”**

```
Admin@DESKTOP-VV9TEBT MINGW64 ~/Downloads
$ ssh -i "my-docker-server-key.pem" ubuntu@ec2-18-191-222-231.us-east-2.compute.amazonaws.com
The authenticity of host 'ec2-18-191-222-231.us-east-2.compute.amazonaws.com (18.191.222.231)' can't be establi
shed.
ED25519 key fingerprint is SHA256:A14cNqaChdpNJJelg69u+lA3Wl7RujxBvSaxDQgDZ/8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? |
```

Step:-14 Verify from which user you are logged in to remote server [type command: **whoami**]

```
ubuntu@ip-172-31-8-152:~$ whoami
ubuntu
ubuntu@ip-172-31-8-152:~$ |
```

Step: - 15 update the My-Docker-Server host OS using [command: **sudo apt-get update**] after updation is completed install Docker on the host OS using [command: **sudo apt-get install docker.io**].

Once installation is done check and verifies docker is running or not using [command: **sudo systemctl status docker**]

```
ubuntu@ip-172-31-8-152:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: enabled)
     Active: active (running) since Sun 2024-07-28 14:25:49 UTC; 1min 4s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 1913 (dockerd)
      Tasks: 8
     Memory: 32.5M (peak: 33.0M)
        CPU: 282ms
     CGroup: /system.slice/docker.service
             └─1913 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Check and verifies docker installation run a [command: **docker ps**] to check running containers → permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker/sock.

To resolve this error add the current user into the docker group.

```
ubuntu@ip-172-31-8-152:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get
 "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect: perm
ission denied
ubuntu@ip-172-31-8-152:~$ |
```

Add current logged in user into the docker group using the [command: **sudo usermod –aG docker $USER**].

Check and verifies is current logged in user Ubuntu is added or not in the docker group [command: **cat /etc/group**].

```
ubuntu@ip-172-31-8-152:~$ sudo usermod -aG docker $USER
ubuntu@ip-172-31-8-152:~$ |
```

Check and verifies docker installation run a [command: **docker ps**] to check running containers → permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker/sock. Again this error came so you need to reboot the system and post reboot SSH into remote server [command: **sudo reboot**].

```
ubuntu@ip-172-31-8-152:~$ docker ps
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock:
 "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/containers/json": dial unix /var/run/docker.sock: connect:
ission denied
ubuntu@ip-172-31-8-152:~$ sudo reboot |
```

```
Admin@DESKTOP-VV9TEBT MINGW64 ~/Downloads
$ ssh -i "my-docker-server-key.pem" ubuntu@ec2-18-191-222-231.us-east-2.compute.amazonaws.com
```

Verify the docker installation → Note: Dokcer installed successfully.

```
ubuntu@ip-172-31-8-152:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
```

Step:-16 create a directory for your project name the folder as "**my-docker-projects**".

```
ubuntu@ip-172-31-8-152:~/my-docker-projects$ cd ..
ubuntu@ip-172-31-8-152:~$ ls
my-docker-projects
ubuntu@ip-172-31-8-152:~$ cd my-docker-projects/
ubuntu@ip-172-31-8-152:~/my-docker-projects$ |
```

Create a index.html into "**my-docker-projects**" directory and write your webpage code into it.

```
ubuntu@ip-172-31-8-152:~/my-docker-projects$ vim index.html
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Web Page</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background-color: #f0f0f0;
        }
        h1 {
            background-color: #4CAF50;
            color: white;
            padding: 20px;
            border-radius: 10px;
        }
    </style>
</head>
<body>
        <h1>Deploy a basic web server using a Docker container.!!!</h1>
</body>
</html>
```

For save your code type → esc → :wq → save index.html

Create a dockerfile into "**my-docker-projects**" directory. To containerize the webpage, use the following Dockerfile. This example uses an Nginx server to serve the static HTML content.

```
ubuntu@ip-172-31-8-152:~/my-docker-projects$ vim dockerfile
```

```dockerfile
# Use an official Nginx image as a base
FROM nginx:alpine

# Copy the static HTML file to the Nginx directory
COPY index.html /usr/share/nginx/html/

# Expose port 80 to access the webpage
EXPOSE 80

# Start Nginx server
CMD ["nginx", "-g", "daemon off;"]
```

1. **Build and run the docker image**
   In the terminal, navigate to the directory containing your index.html and Dockerfile and run: [command: **docker build –t my-webpage .]**

```
ubuntu@ip-172-31-8-152:~/my-docker-projects$ docker build -t my-webpage .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  3.584kB
Step 1/4 : FROM nginx:alpine
alpine: Pulling from library/nginx
46b060cc2620: Pull complete
21af147d2ad5: Pull complete
b3ee43e51ca6: Pull complete
b17a9d410da1: Pull complete
542e3e75411d: Pull complete
2b2faad386df: Pull complete
a5e22afba545: Pull complete
fb923a41dc10: Pull complete
Digest: sha256:208b70eefac13ee9be00e486f79c695b15cef861c680527171a27d253d834be9
Status: Downloaded newer image for nginx:alpine
 ---> 1ae23480369f
Step 2/4 : COPY index.html /usr/share/nginx/html/
 ---> 5c2315d9f4d2
Step 3/4 : EXPOSE 80
 ---> Running in 1170ec761dec
Removing intermediate container 1170ec761dec
 ---> 7a939d8cf35c
Step 4/4 : CMD ["nginx", "-g", "daemon off;"]
 ---> Running in 66322ec32e6b
Removing intermediate container 66322ec32e6b
 ---> e2deb7a8eff6
Successfully built e2deb7a8eff6
Successfully tagged my-webpage:latest
```

See your image using the [command: **docker images**]

```
ubuntu@ip-172-31-8-152:~/my-docker-projects$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
my-webpage    latest    e2deb7a8eff6   52 seconds ago   43.2MB
nginx         alpine    1ae23480369f   5 weeks ago      43.2MB
hello-world   latest    d2c94e258dcb   15 months ago    13.3kB
ubuntu@ip-172-31-8-152:~/my-docker-projects$ |
```

2. **Run the docker container**
   This command maps port 80 on your local machine to port 80 on the container. [command: **docker run –d –p 80:80 my-webpage**]

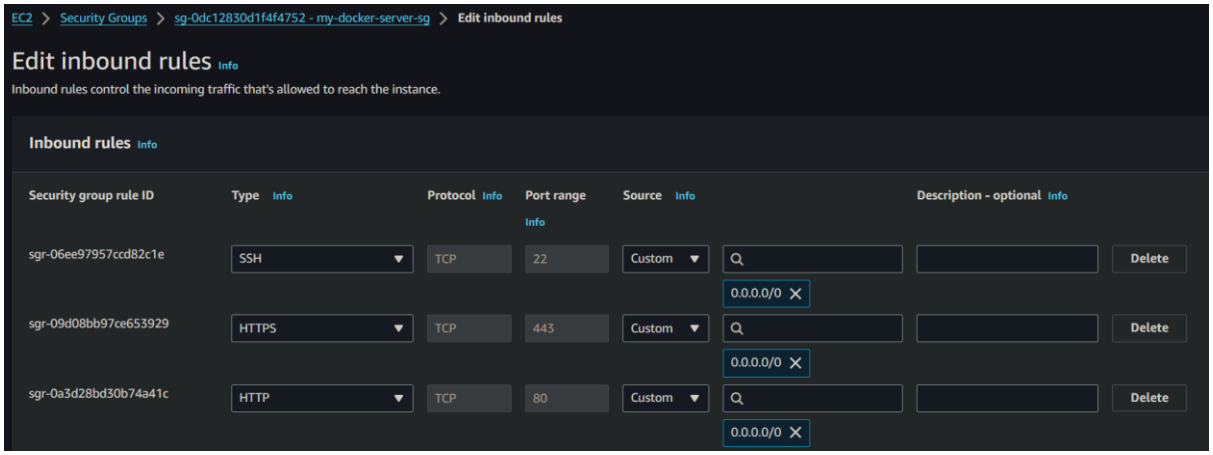
```
ubuntu@ip-172-31-8-152:~/my-docker-projects$ docker run -d -p 80:80 my-webpage
20de310a85eeb8b52b7b971b60ba3eb9abb9f8b24b7f9cbce7cbb5c7d4ff11df
ubuntu@ip-172-31-8-152:~/my-docker-projects$ docker ps
CONTAINER ID   IMAGE        COMMAND                  CREATED         STATUS         PORTS
               NAMES
20de310a85ee   my-webpage   "/docker-entrypoint.…"   5 seconds ago   Up 4 seconds   0.0.0.0:80->80/tcp, :
::80->80/tcp   cool_mayer
ubuntu@ip-172-31-8-152:~/my-docker-projects$ |
```

3. **Access the webpage**

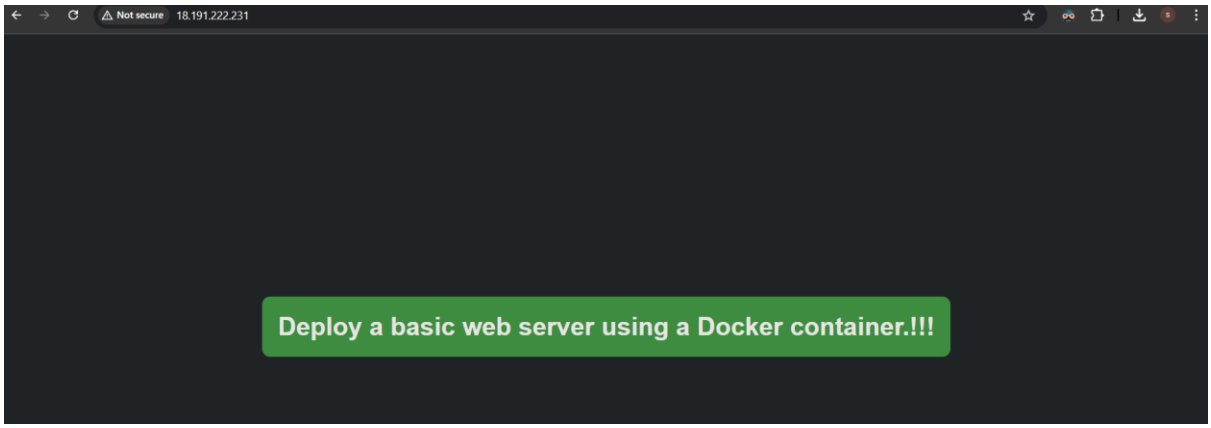
To access the webpage using your EC2 instance's public IP address, follow these steps:

1. **Ensure the Security Group allows inbound traffic on port 80:** Go to your EC2 instance's security group settings in the AWS console and make sure there is a rule allowing inbound traffic on port 80 from anywhere (0.0.0.0/0) or from your specific IP range.

2. **Find your EC2 instance's public IP address:** You can find the public IP address of your EC2 instance in the AWS console.



Access the webpage using <**EC2_PUBLIC_IP:HostPort**> example:- 18.191.222.231:80



Follow me on **LinkedIn & GitHub** for project updates and insights!

~ Shankar Chavhan