

NAME: MULLA AFRAH AKKAS ALI

ROLL NO.: 612038

BRANCH: T.E. – I.T.

SEMESTER: ODD SEMESTER 5

COURSE: Advance DevOPs (ITL504)

DATE: 24-08-2022

EXPERIMENT 7

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

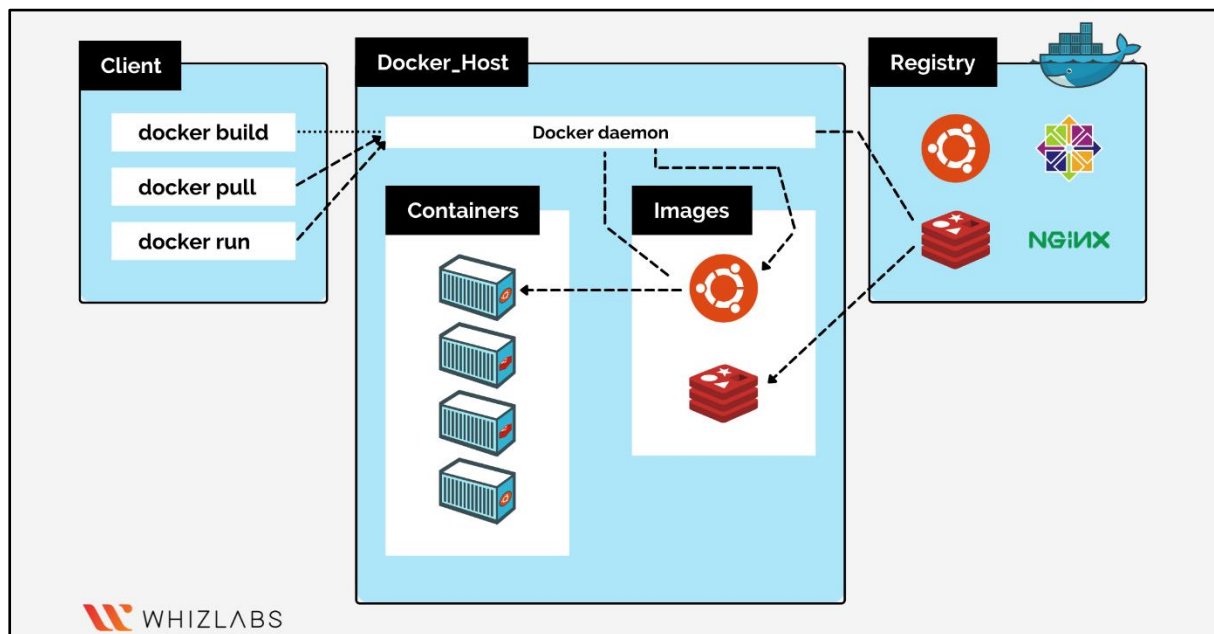
Containerization is the packaging together of software code with all its necessary components like libraries, frameworks, and other dependencies so that they are isolated in their own "container." The container acts as a kind of bubble or a computing environment surrounding the application and keeping it independent of its surroundings.

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.

Docker Architecture

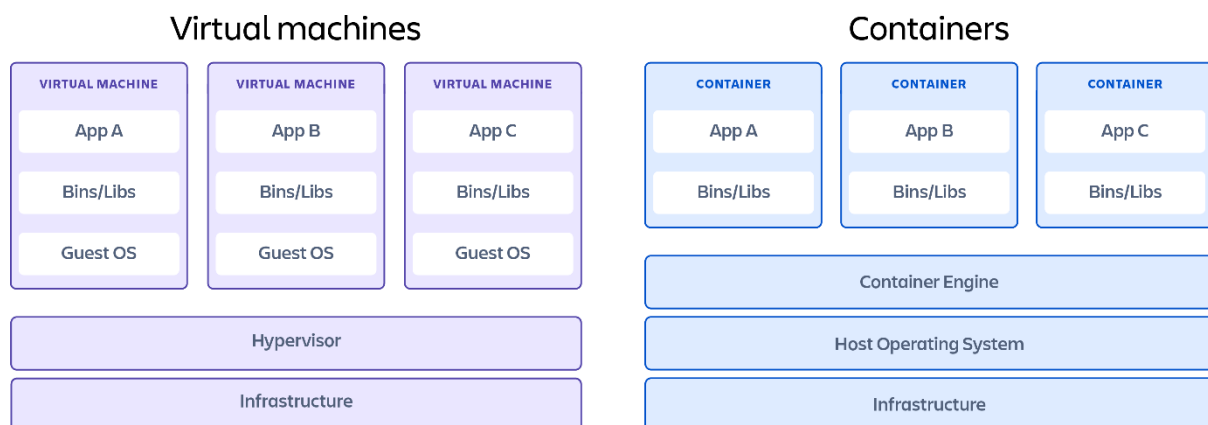
To allow for an application to be self-contained the Docker approach moves up the abstraction of resources from the hardware level to the Operating System level.

To further understand Docker, let us look at its architecture. It uses a client-server model and comprises of the following components:



- Docker daemon: The daemon is responsible for all container related actions and receives commands via the CLI or the REST API.
- Docker Client: A Docker client is how users interact with Docker. The Docker client can reside on the same host as the daemon or a remote host.
- Docker Objects: Objects are used to assemble an application. Apart from networks, volumes, services, and other objects the two main requisite objects are:
 - Images: The read-only template used to build containers. Images are used to store and ship applications.
 - Containers: Containers are encapsulated environments in which applications are run. A container is defined by the image and configuration options. At a lower level, you have containers, which is a core container runtime that initiates, and supervises container performance.
- Docker Registries: Registries are locations from where we store and download (or “pull”) images.

2. 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).

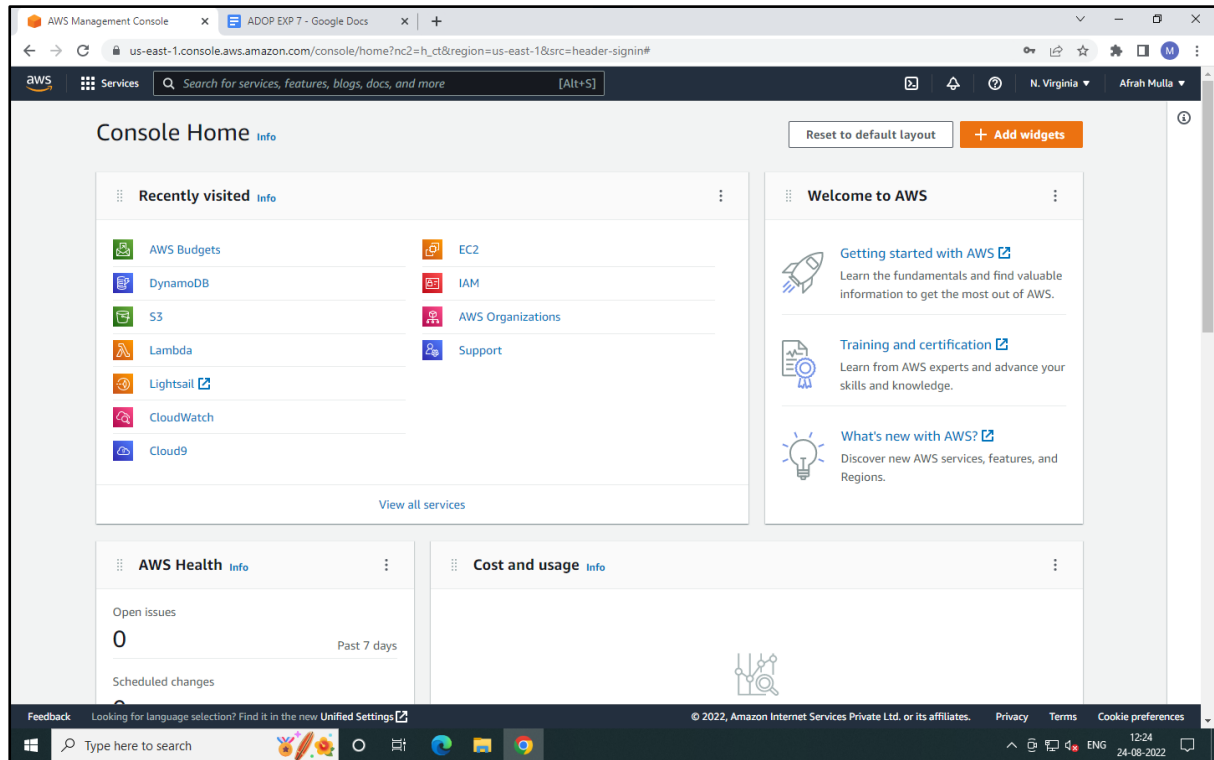
- Containerization and virtualization are similar in that they both allow for full isolation of applications so that they can be operational in multiple environments. Where the main differences lie are in size and portability.
- The key differentiator between containers and virtual machines is that virtual 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.

3. Why are Containers lightweight?

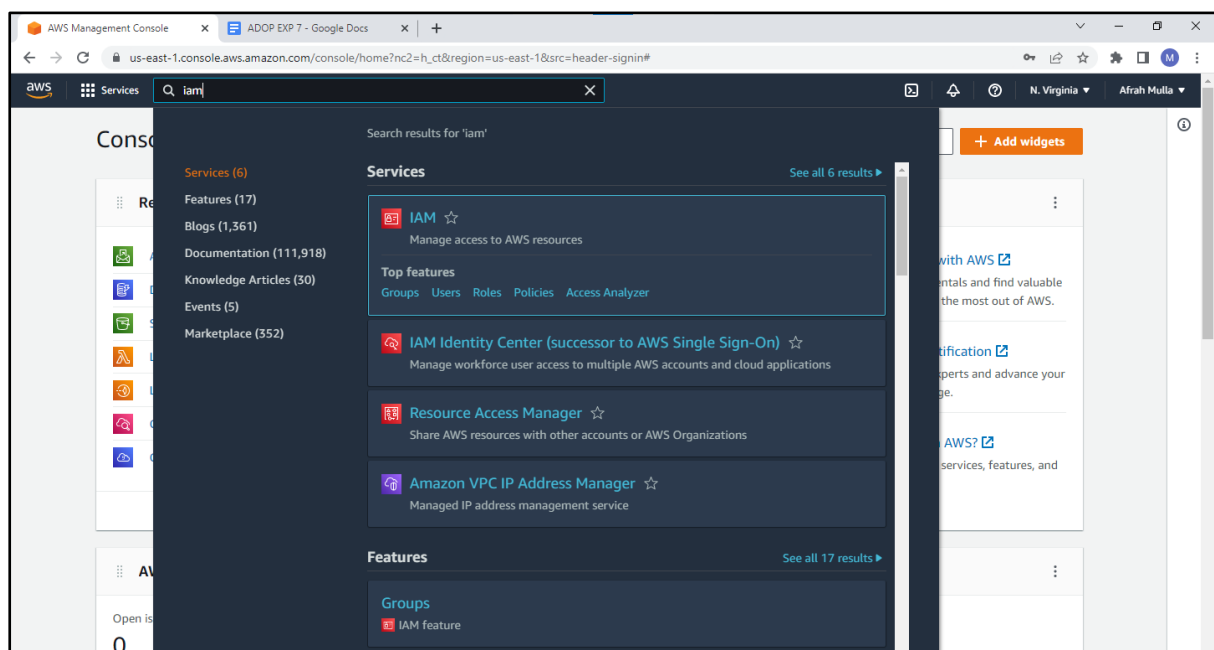
Inside a container are all the necessary executables, binary code, libraries, and configuration files. Compared to server or machine virtualization approaches, however, containers share the machine's OS system kernel and therefore do not require an OS per application, driving higher server efficiencies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

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: AWS Management Console Dashboard



Step 2: Search for IAM and click on it



Step 3: Select 'Roles'

The screenshot shows the AWS IAM Management Console dashboard. The left-hand navigation menu has the 'Roles' link circled in red. The main content area displays the IAM dashboard with the following sections:

- Security recommendations:** Includes a warning to 'Add MFA for root user' and a confirmation that 'Root user has no active access keys'.
- IAM resources:** A summary table showing the following counts:

User groups	Users	Roles	Policies	Identity providers
0	0	5	2	0
- What's new:** Updates for features in IAM, including 'Right-size permissions for more roles' and 'Amazon S3 Object Ownership'.
- AWS Account:** Information about the account, including Account ID (018099588167), Account Alias (018099588167), and the Sign-in URL.
- Quick Links:** Links to 'My security credentials', 'Policy simulator', and other tools.

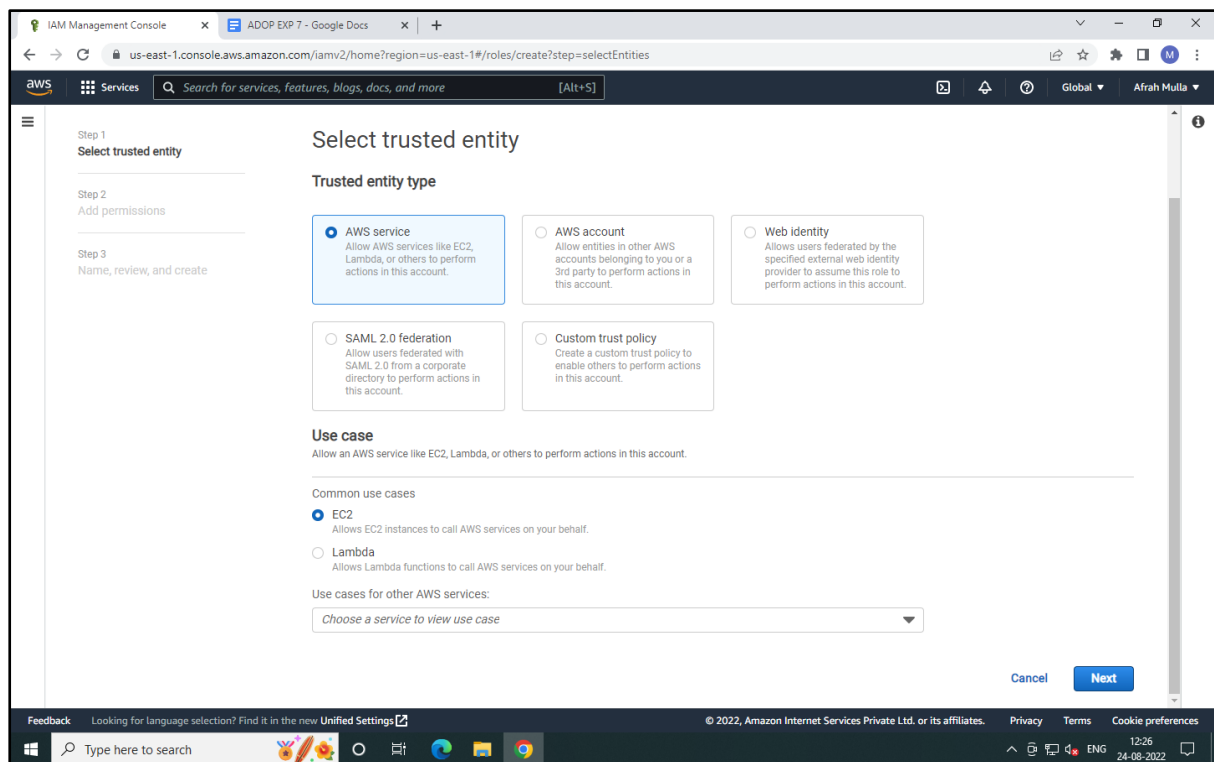
Step 4: Click on 'Create role'

The screenshot shows the AWS IAM Management Console 'Roles' page. The 'Create role' button is highlighted in blue. The page displays a list of existing roles and a section for 'Roles Anywhere'.

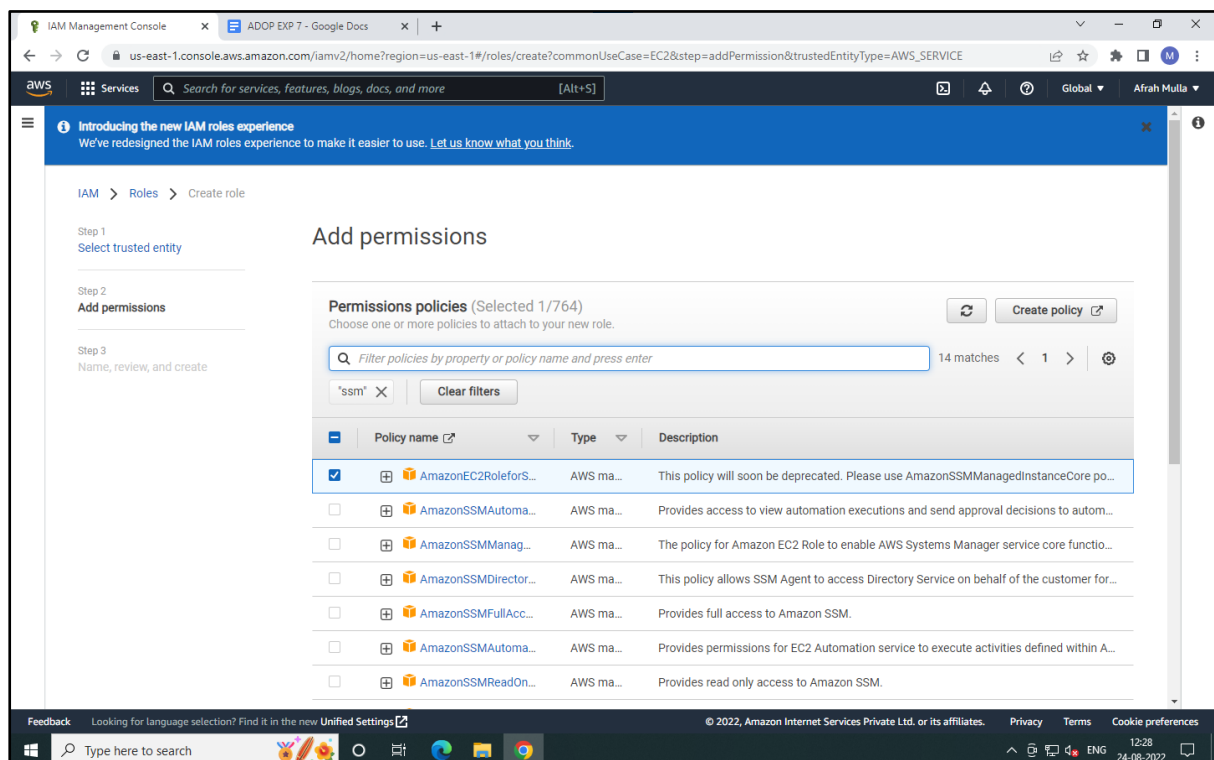
Role name	Trusted entities	Last acti...
<input type="checkbox"/> afrah-mulla	AWS Service: lambda	13 days ago
<input type="checkbox"/> AWSServiceRoleForApplicationAutoScaling_DynamoDBTable	AWS Service: dynamodb.application-autoscaling (Service-Linked R...	6 days ago
<input type="checkbox"/> AWSServiceRoleForAWSCloud9	AWS Service: cloud9 (Service-Linked Role)	21 days ago
<input type="checkbox"/> AWSServiceRoleForSupport	AWS Service: support (Service-Linked Role)	-
<input type="checkbox"/> AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service-Linked Role)	-

The 'Roles Anywhere' section is also visible, with a 'Manage' button.

Step 5: Set Trusted entity type to AWS service, use case to EC2 -> Next



Step 6: Search for SSH in Permission policies -> Select AmazonEC2RoleforSSH -> Next



Step 7: Assign a name to your role -> Create role

The screenshot shows the 'Name, review, and create' step in the AWS IAM console. The left sidebar indicates the current step is 'Step 3: Name, review, and create'. The main content area is titled 'Name, review, and create' and contains the following sections:

- Role details**
 - Role name**: A text input field containing 'mulla-afrah-38'. Below it, a note states: 'Maximum 64 characters. Use alphanumeric and '+', '@', '-' characters.'
 - Description**: A text area containing 'Allows EC2 instances to call AWS services on your behalf.' Below it, a note states: 'Maximum 1000 characters. Use alphanumeric and '+', '@', '-' characters.'
- Step 1: Select trusted entities**: A button labeled 'Edit' is visible.
- JSON Policy**: A code editor showing the following JSON policy:

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "sts:AssumeRole"
8       ],
9       "Principal": {
10        "Service": [
11          "ec2.amazonaws.com"
12        ]
13      }
14    }
15  ]
16 }
```

The bottom of the console shows the footer with '© 2022, Amazon Internet Services Private Ltd. or its affiliates.' and a system clock showing 12:29 on 24-08-2022.

The screenshot shows the 'Step 2: Add permissions' section of the AWS IAM console. The left sidebar indicates the current step is 'Step 2: Add permissions'. The main content area is titled 'Step 2: Add permissions' and contains the following sections:

- Permissions policy summary**: A table showing the attached policy.
- Tags**: A section for adding tags, with a note: 'Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.' Below it, a note states: 'No tags associated with the resource.' and an 'Add tag' button.
- Buttons**: 'Cancel', 'Previous', and 'Create role' buttons are at the bottom right.

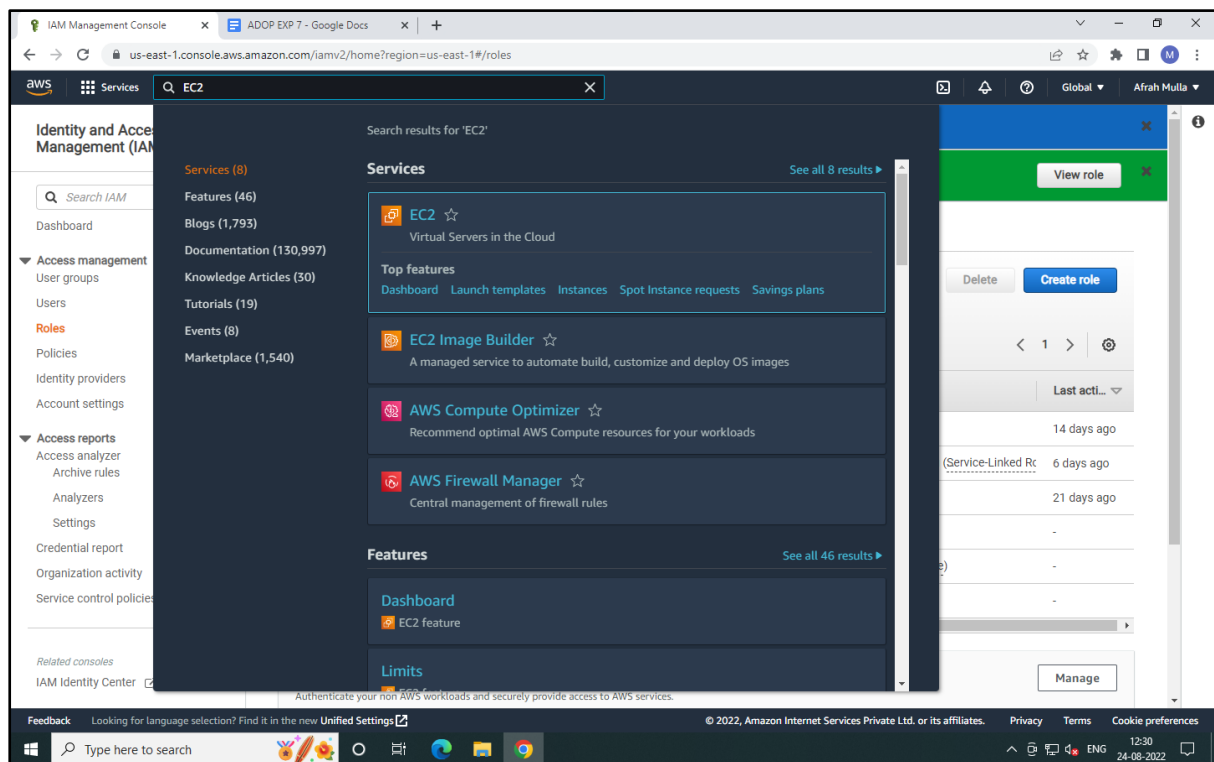
The table 'Permissions policy summary' has the following data:

Policy name	Type	Attached as
AmazonEC2RoleforSSM	AWS managed	Permissions policy

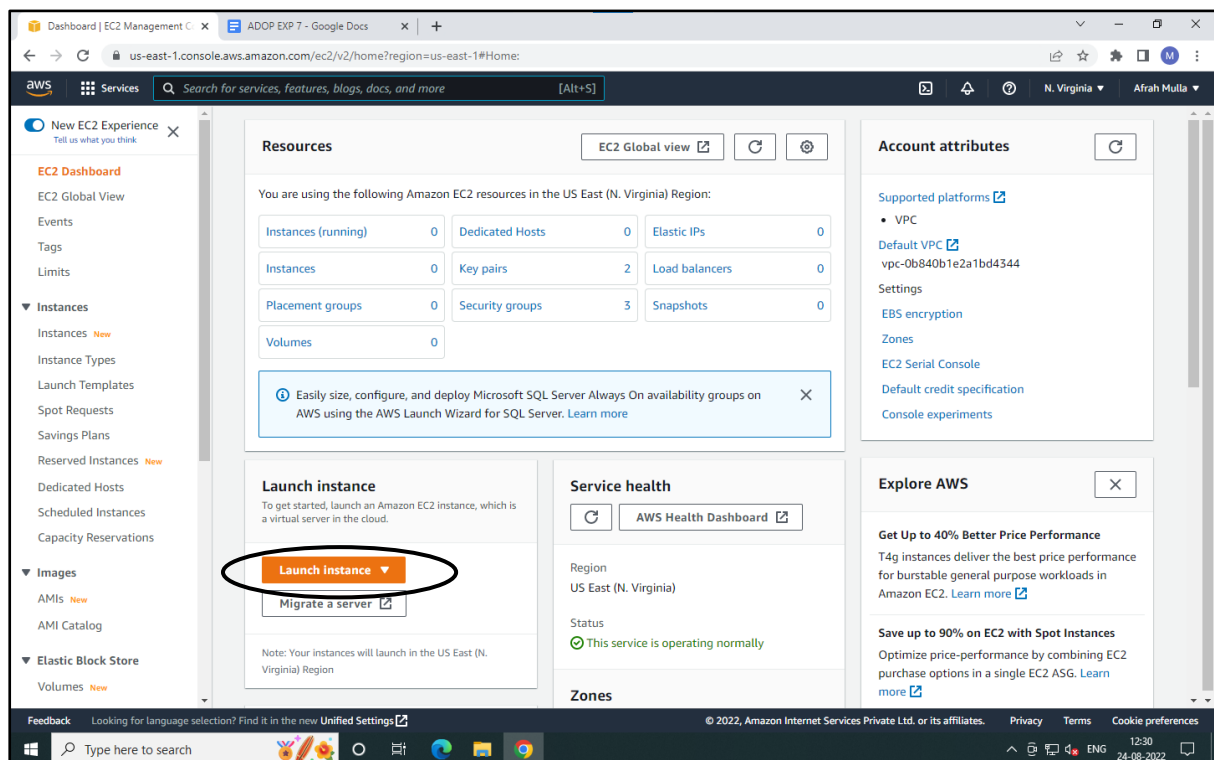
The bottom of the console shows the footer with '© 2022, Amazon Internet Services Private Ltd. or its affiliates.' and a system clock showing 12:29 on 24-08-2022.

The screenshot shows the AWS IAM console after the role has been created. The left sidebar indicates the current step is 'Step 3: Name, review, and create'. The main content area is titled 'Identity and Access Management (IAM)'. A green banner at the top of the main content area displays the message: 'Role mulla-afrah-38 created'. A 'View role' button is located to the right of the message. The bottom of the console shows the footer with '© 2022, Amazon Internet Services Private Ltd. or its affiliates.' and a system clock showing 12:29 on 24-08-2022.

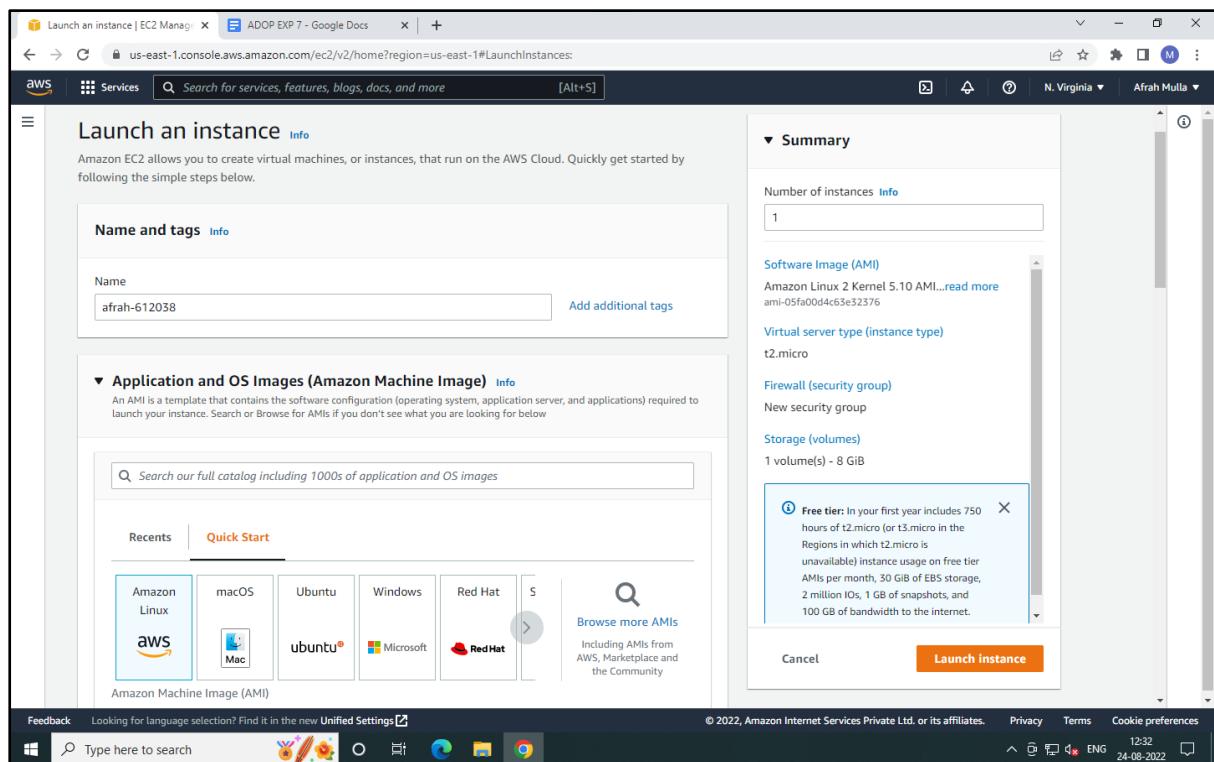
Step 8: Search for 'EC2' -> Select it



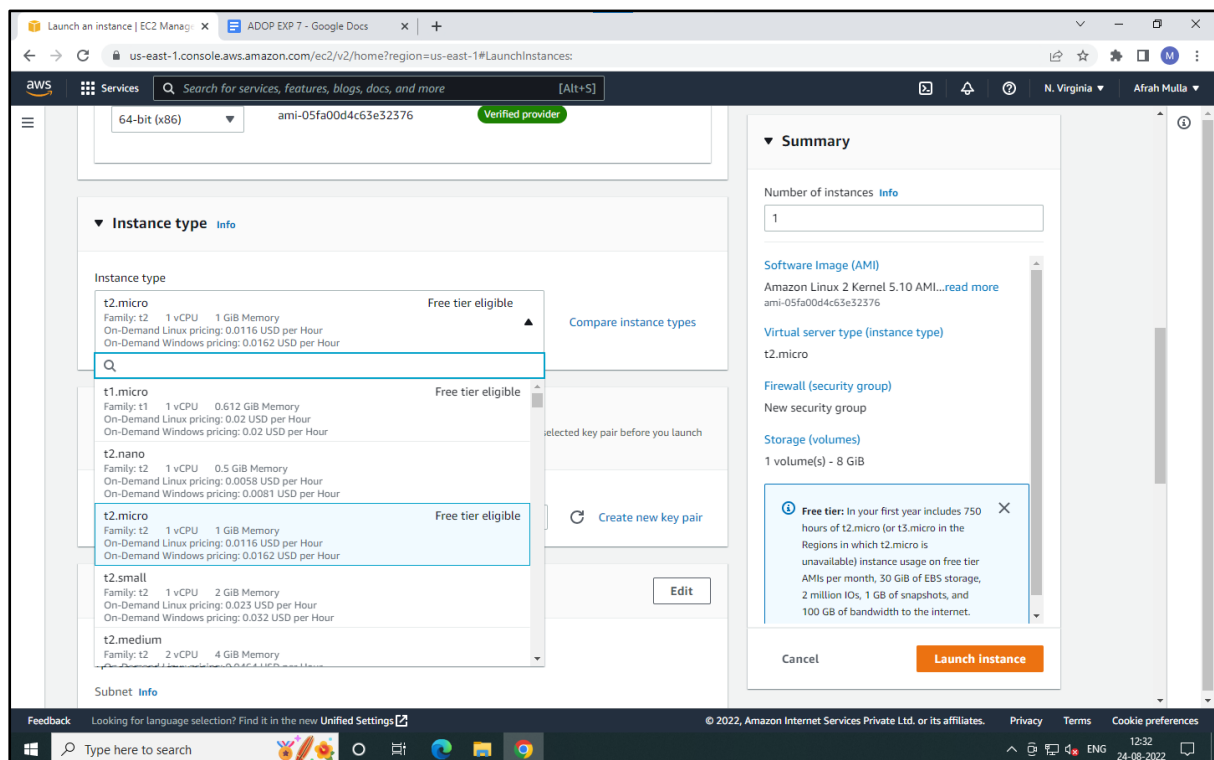
Step 9: Launch instance



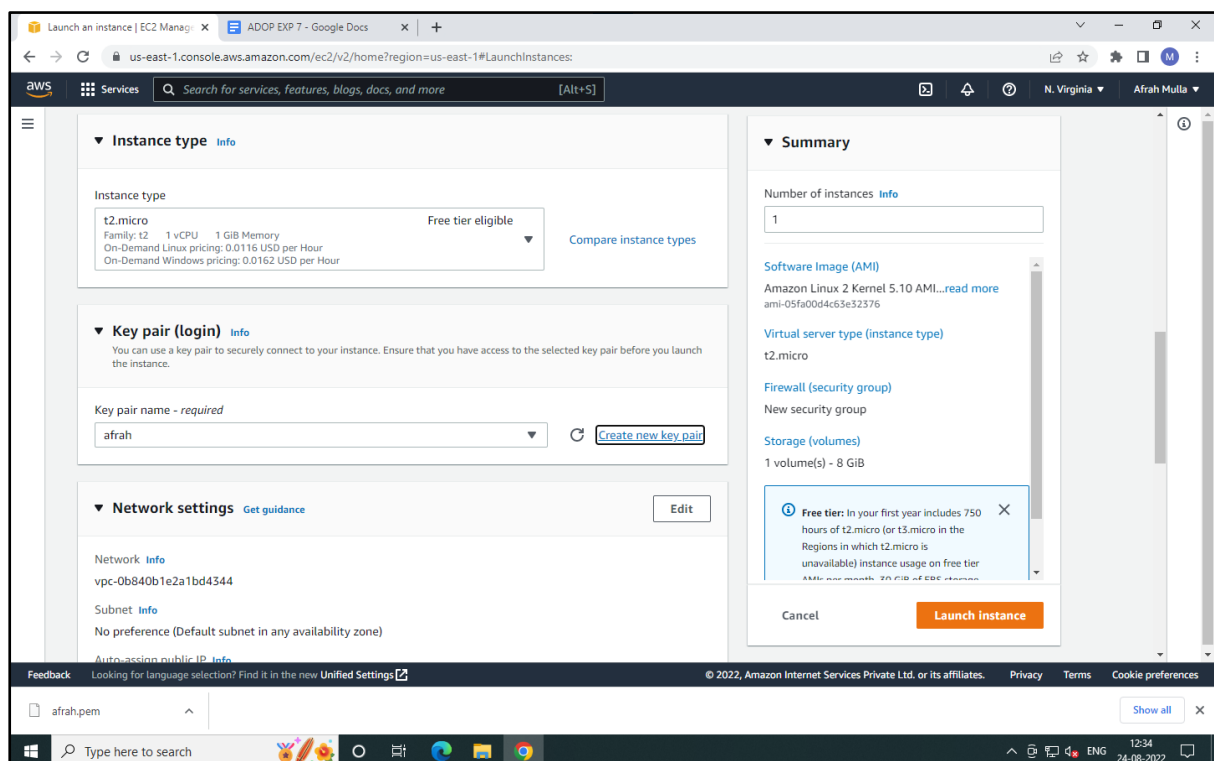
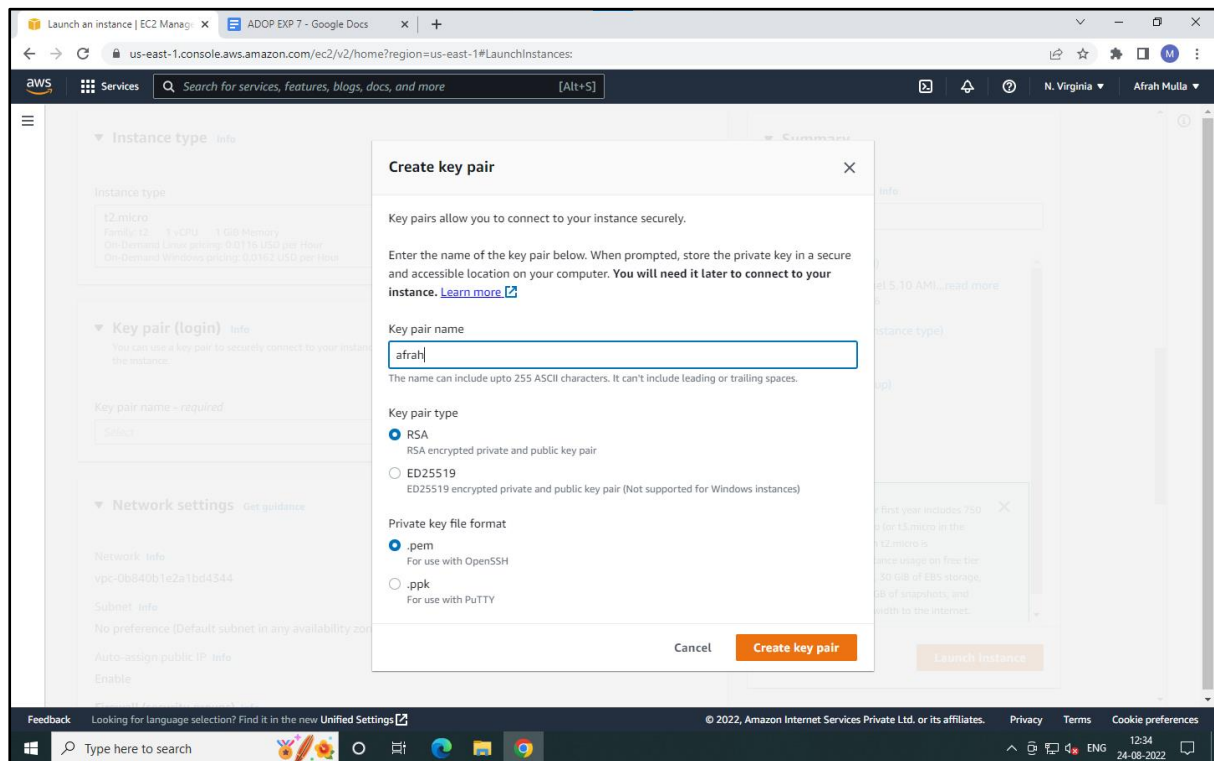
Step 10: Give a name to your instance and select Amazon Linux



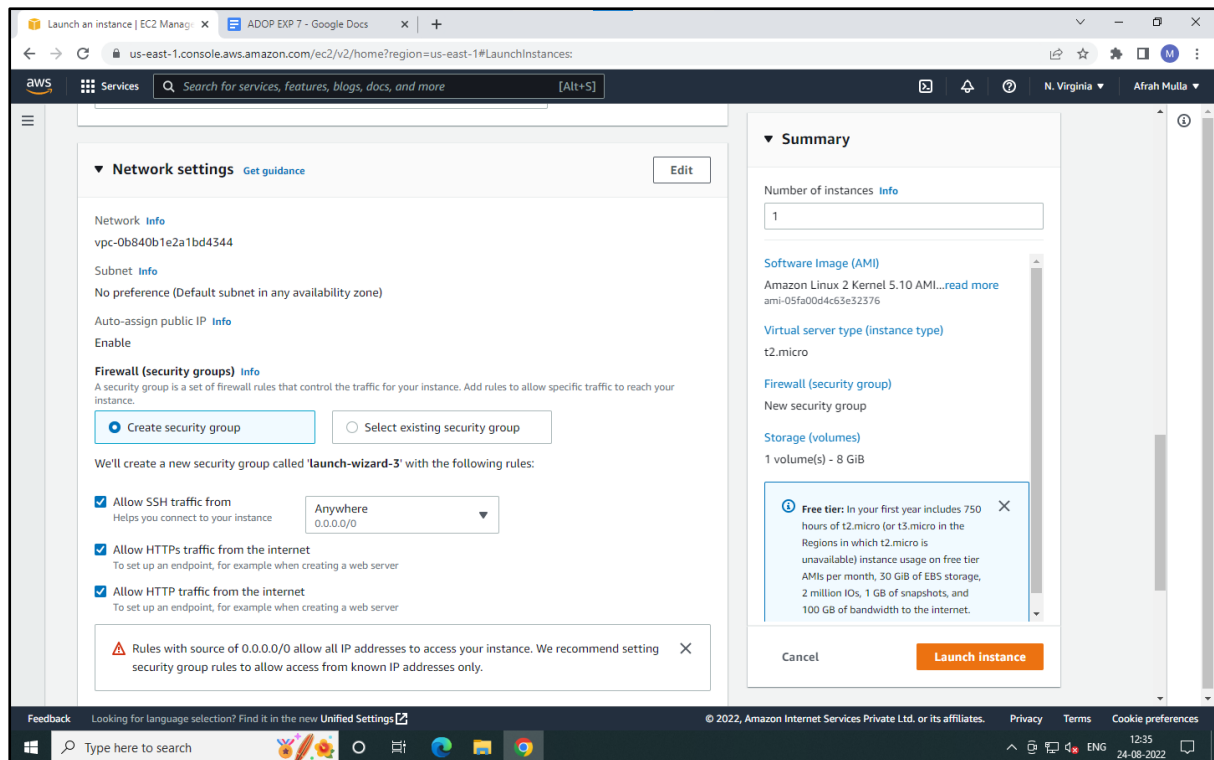
Step 11: Select instance type with free tier eligibility (t2.micro)



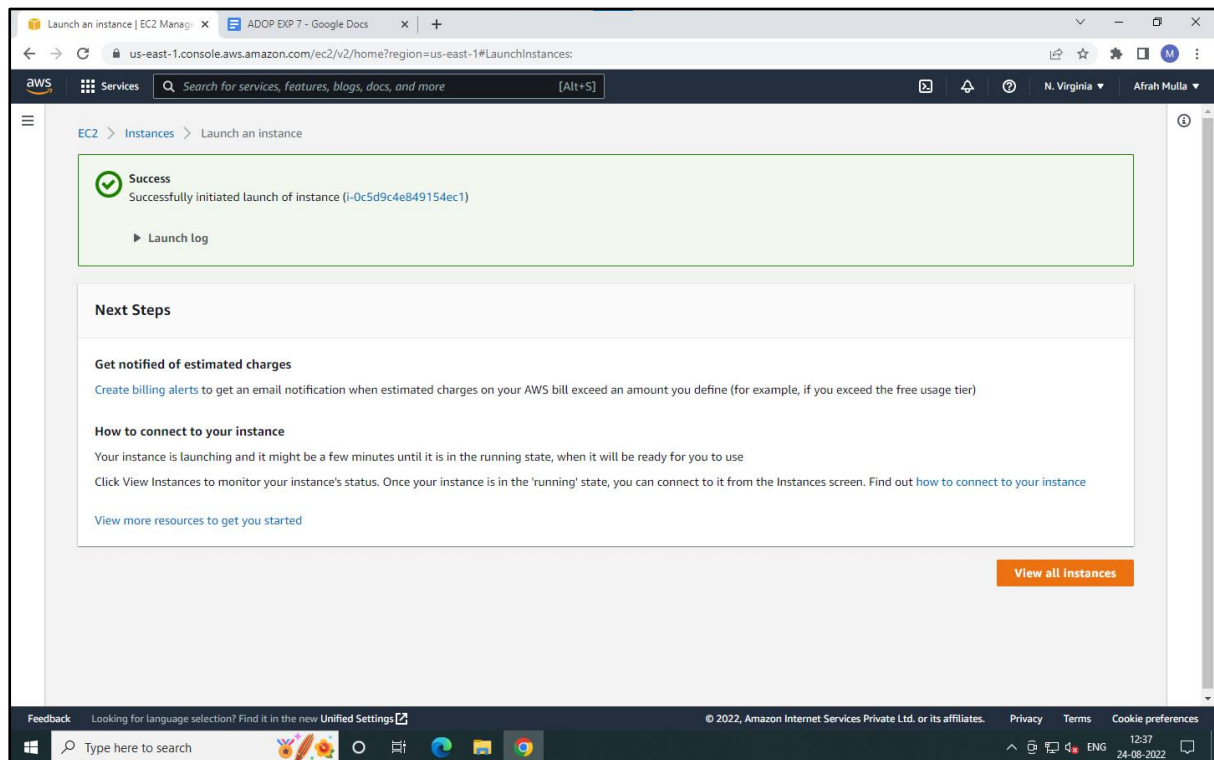
Step 12: Create key pair. A .pem file will be downloaded which will be later used to connect to the instance



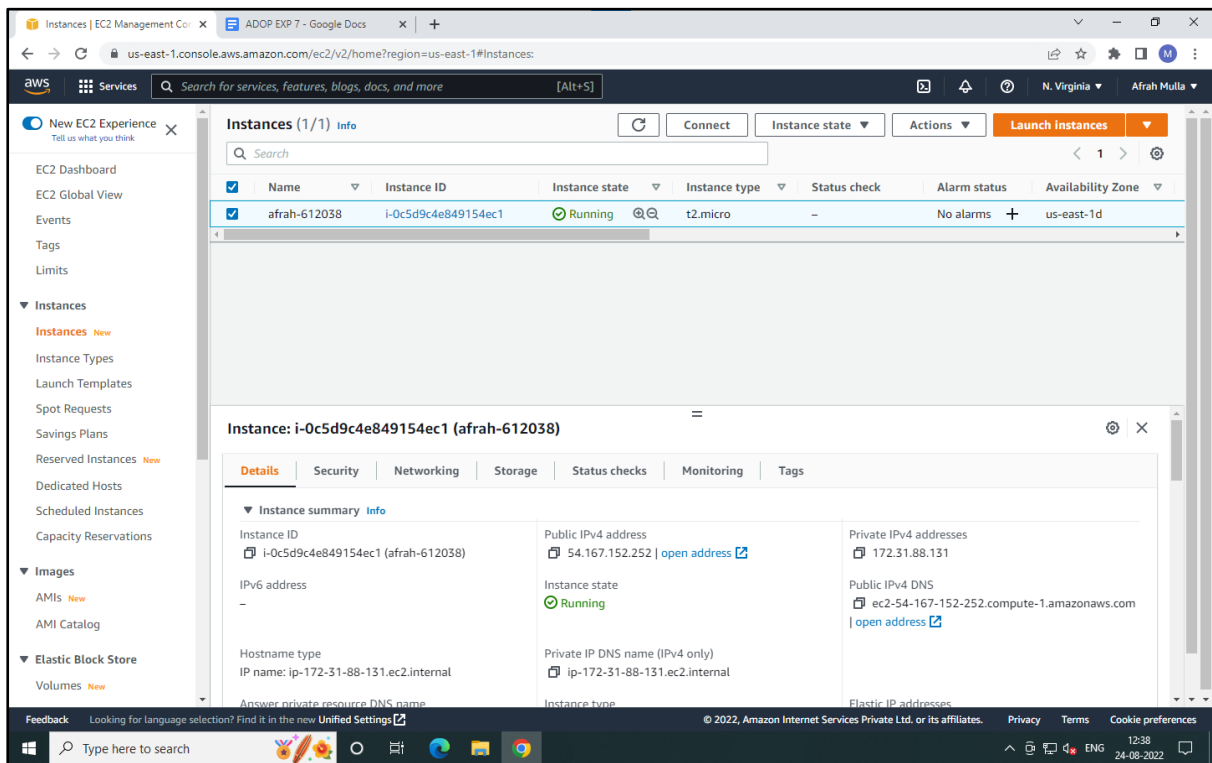
Step 13: Network Settings: Select 'Allow SSH Traffic from' Anywhere, Allow HTTPs traffic from the internet and Allow HTTP traffic from the internet. Then Launch Instance



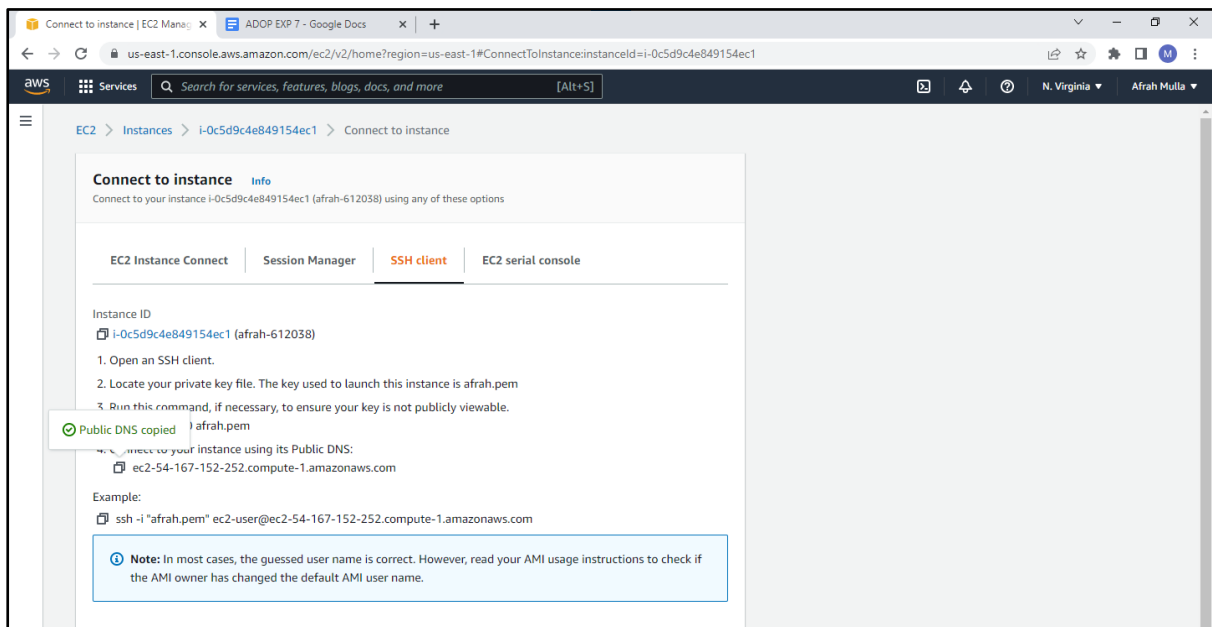
Success message will be shown after successful creation of instance



Step 14: Launch Linux instance to get the remote host and username for SSH in MobaXterm

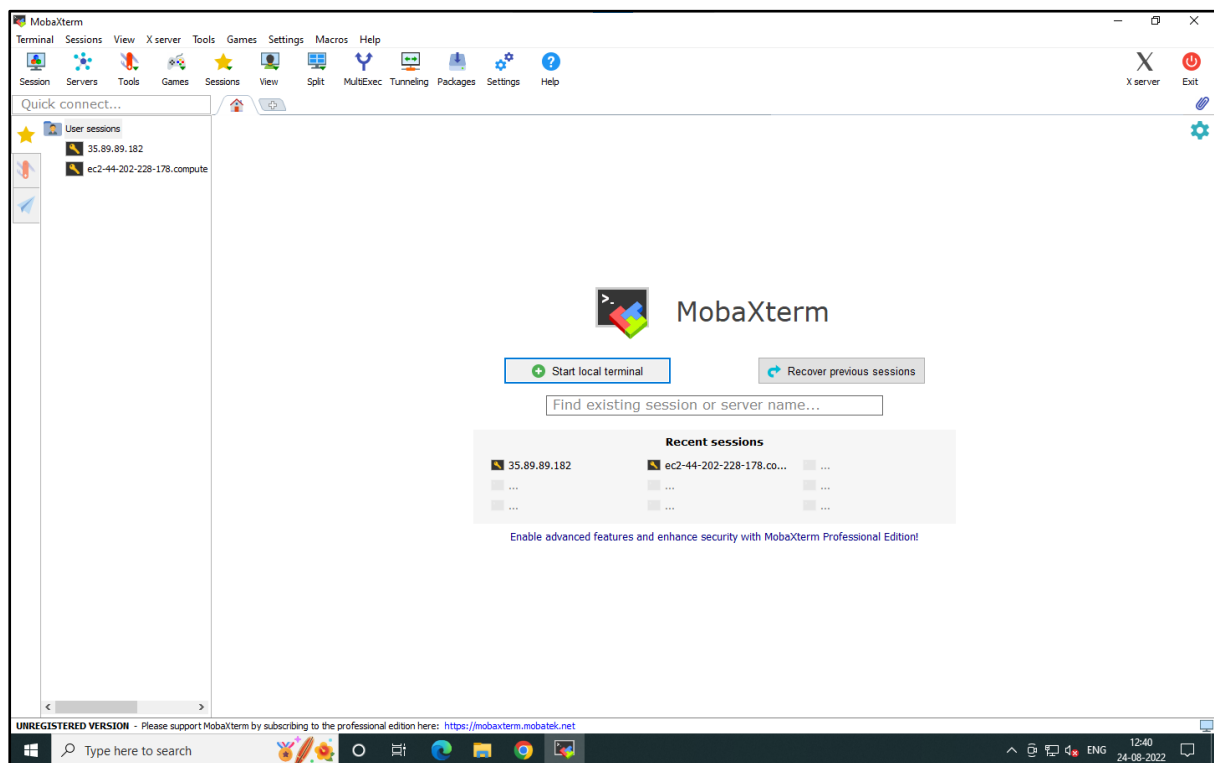


Go to SSH Client to get the remote host and username

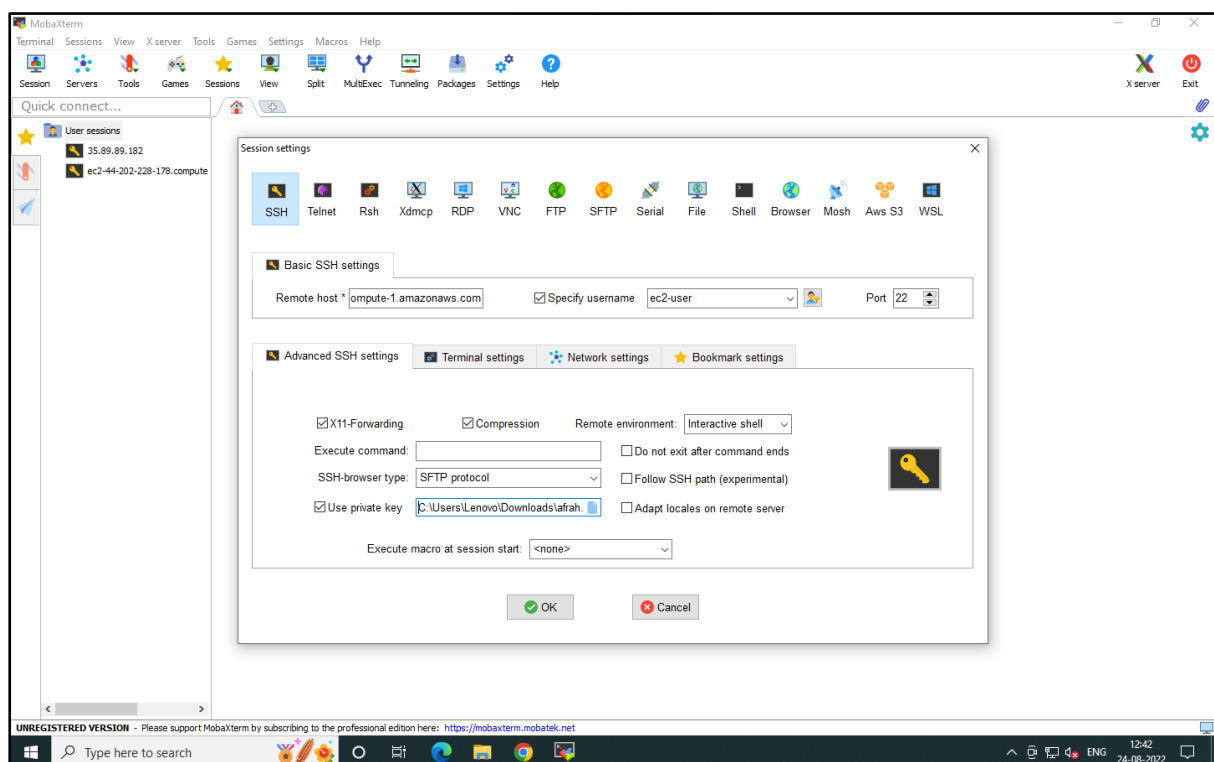


Copy the Public DNS which is your remote host and the username is the word before '@'

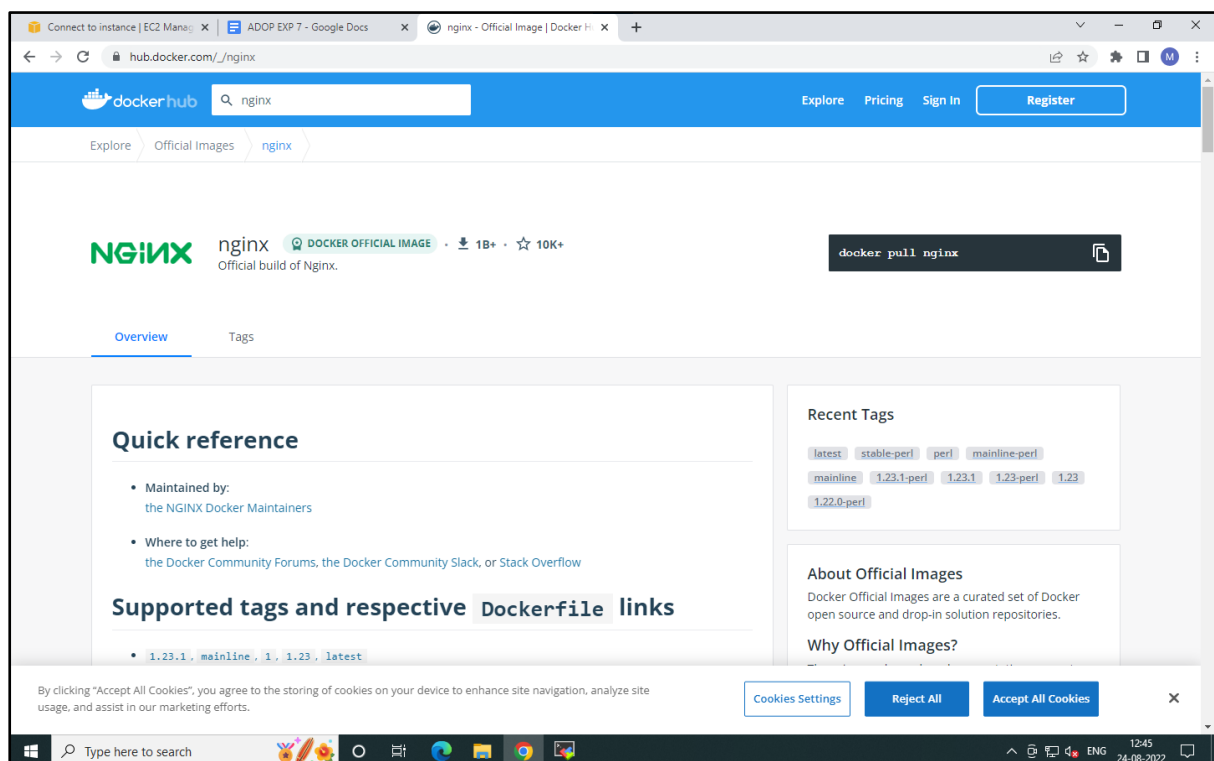
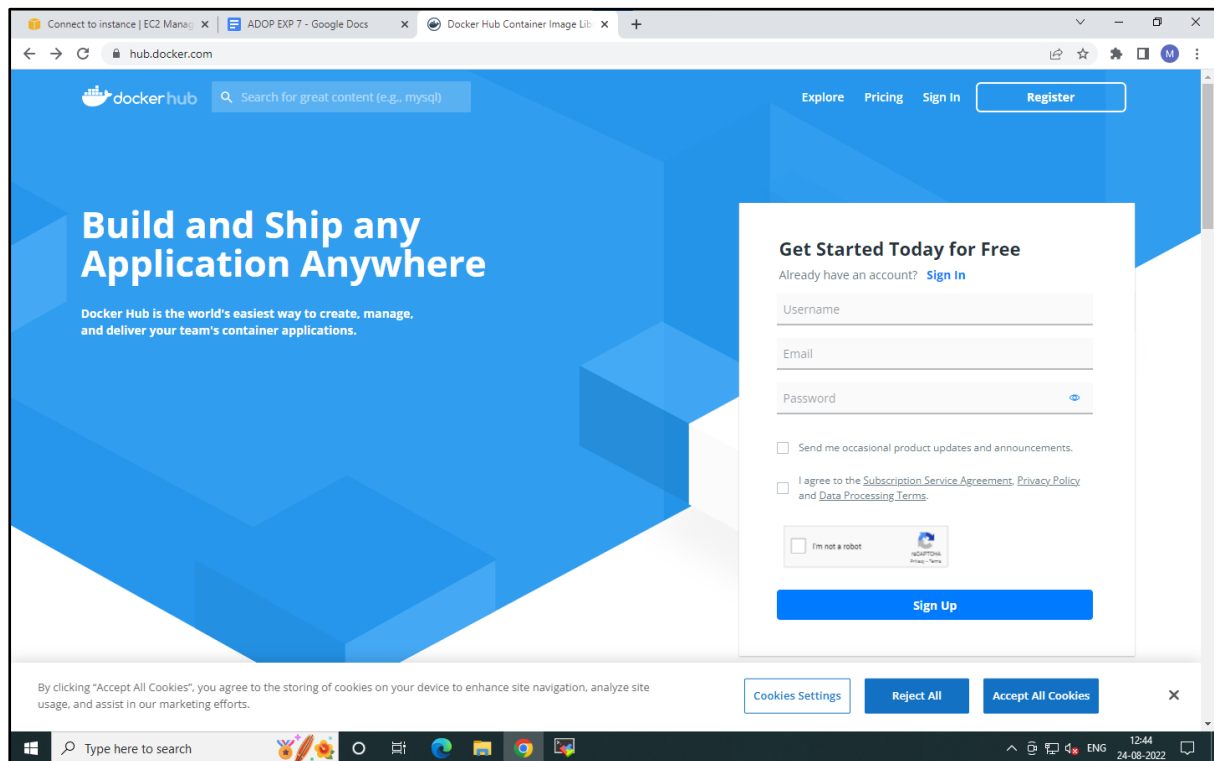
Step 15: Launch MobaXterm to connect the instance. Go to Sessions -> New Session



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



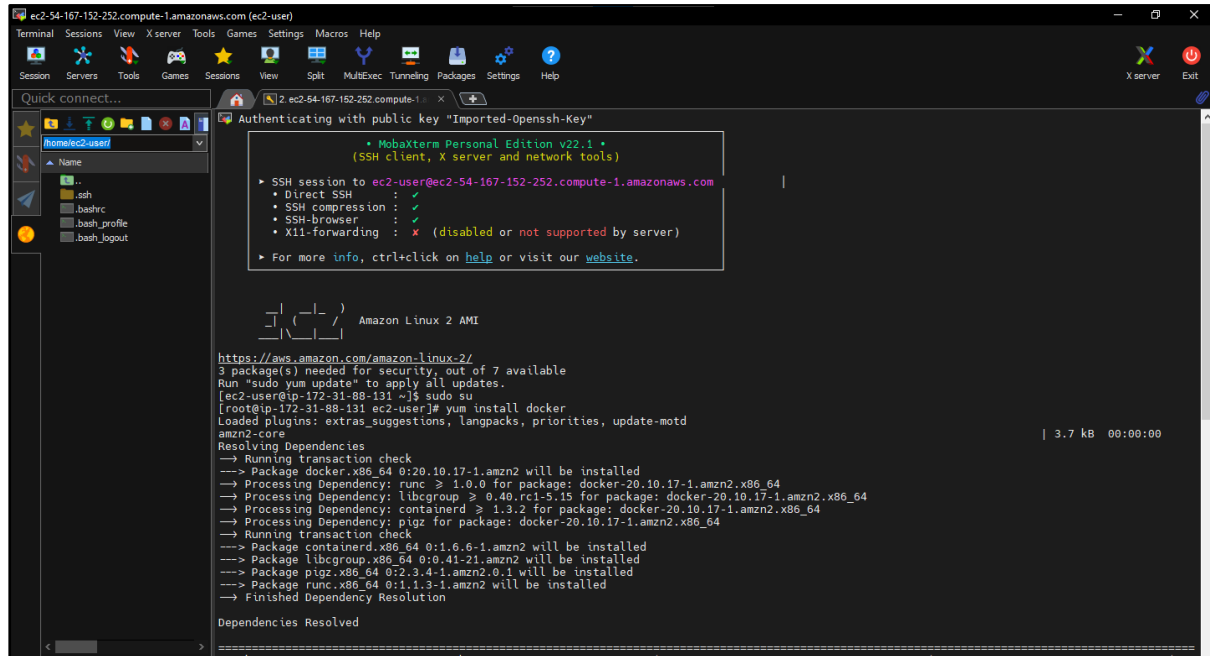
Step 16: Go to <https://hub.docker.com/>, search 'nginx' to get the commands to install Docker, pull nginx image and run it.



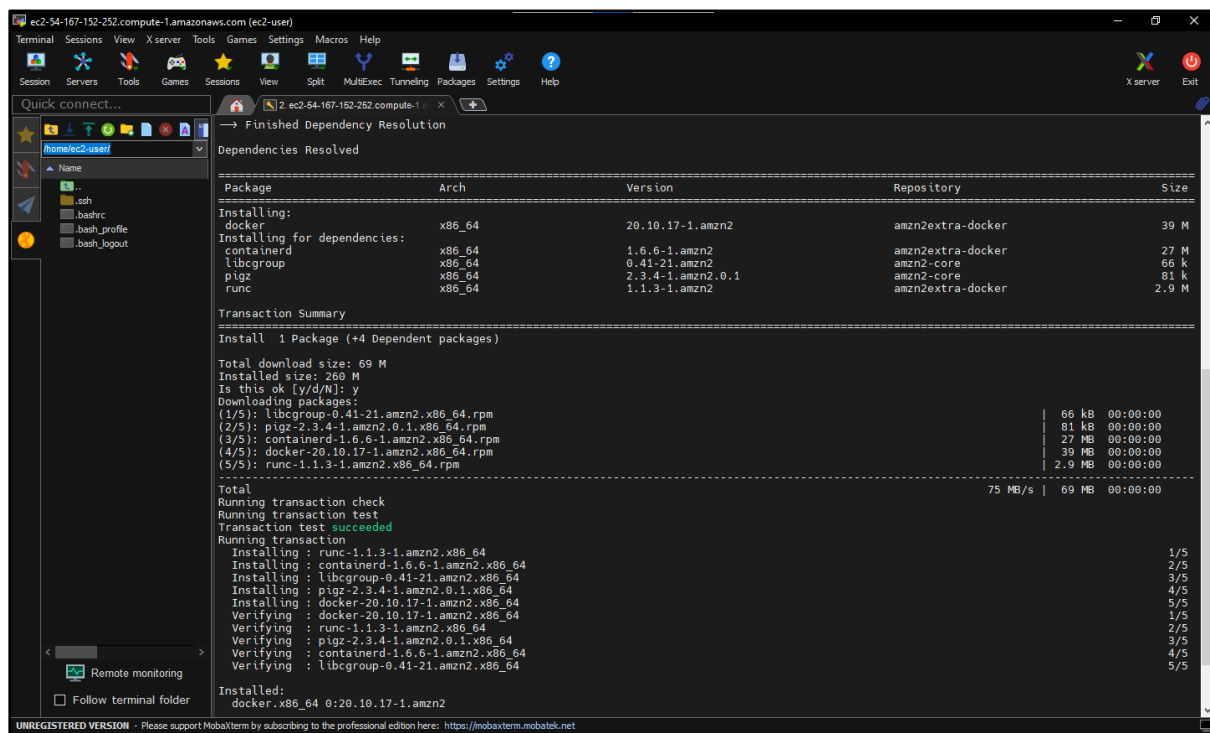
Execute the following commands –

`sudo su`

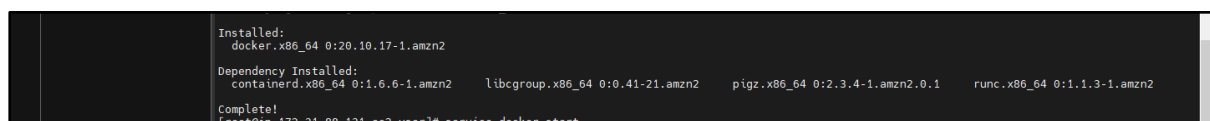
`yum install docker`



```
ec2-54-167-152-252.compute-1.amazonaws.com (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
Authenticating with public key "Imported-OpenSSH-Key"
MobaXterm Personal Edition v22.1
(SSH client, X server and network tools)
SSH session to ec2-user@ec2-54-167-152-252.compute-1.amazonaws.com
Direct SSH
SSH compression
SSH-browser
X11-forwarding (disabled or not supported by server)
For more info, ctrl+click on help or visit our website.
Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
3 package(s) needed for security, out of 7 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-88-131 ~]$ sudo su
[root@ip-172-31-88-131 ec2-user]# yum install docker
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
Resolving Dependencies
--> Running transaction check
--> Package docker.x86_64 0:20.10.17-1.amzn2 will be installed
--> Processing Dependency: runc >= 1.0.0 for package: docker-20.10.17-1.amzn2.x86_64
--> Processing Dependency: libcgrouper >= 0.40.rc1-5.15 for package: docker-20.10.17-1.amzn2.x86_64
--> Processing Dependency: containerd >= 1.3.2 for package: docker-20.10.17-1.amzn2.x86_64
--> Processing Dependency: pigz for package: docker-20.10.17-1.amzn2.x86_64
--> Running transaction check
--> Package containerd.x86_64 0:1.6.6-1.amzn2 will be installed
--> Package libcgrouper.x86_64 0:0.41-21.amzn2 will be installed
--> Package pigz.x86_64 0:2.3.4-1.amzn2.0.1 will be installed
--> Package runc.x86_64 0:1.1.3-1.amzn2 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
=====
Package Arch Version Repository Size
-----
Installing:
docker x86_64 20.10.17-1.amzn2 amzn2extra-docker 39 M
Installing for dependencies:
containerd x86_64 1.6.6-1.amzn2 amzn2extra-docker 27 M
libcgrouper x86_64 0.41-21.amzn2 amzn2-core 66 k
pigz x86_64 2.3.4-1.amzn2.0.1 amzn2-core 81 k
runc x86_64 1.1.3-1.amzn2 amzn2extra-docker 2.9 M
Transaction Summary
-----
Install 1 Package (+4 Dependent packages)
Total download size: 69 M
Installed size: 260 M
Is this ok [y/d/N]: y
Downloading packages:
(1/5): libcgrouper-0.41-21.amzn2.x86_64.rpm | 66 kB 00:00:00
(2/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm | 81 kB 00:00:00
(3/5): containerd-1.6.6-1.amzn2.x86_64.rpm | 27 MB 00:00:00
(4/5): docker-20.10.17-1.amzn2.x86_64.rpm | 39 MB 00:00:00
(5/5): runc-1.1.3-1.amzn2.x86_64.rpm | 2.9 MB 00:00:00
Total 75 MB/s | 69 MB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : runc-1.1.3-1.amzn2.x86_64 1/5
Installing : containerd-1.6.6-1.amzn2.x86_64 2/5
Installing : libcgrouper-0.41-21.amzn2.x86_64 3/5
Installing : pigz-2.3.4-1.amzn2.0.1.x86_64 4/5
Installing : docker-20.10.17-1.amzn2.x86_64 5/5
Verifying : docker-20.10.17-1.amzn2.x86_64 1/5
Verifying : runc-1.1.3-1.amzn2.x86_64 2/5
Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64 3/5
Verifying : containerd-1.6.6-1.amzn2.x86_64 4/5
Verifying : libcgrouper-0.41-21.amzn2.x86_64 5/5
Installed:
docker.x86_64 0:20.10.17-1.amzn2
Complete!
[root@ip-172-31-88-131 ec2-user]# service docker start
```



```
ec2-54-167-152-252.compute-1.amazonaws.com (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Quick connect...
Finished Dependency Resolution
Dependencies Resolved
=====
Package Arch Version Repository Size
-----
Installing:
docker x86_64 20.10.17-1.amzn2 amzn2extra-docker 39 M
Installing for dependencies:
containerd x86_64 1.6.6-1.amzn2 amzn2extra-docker 27 M
libcgrouper x86_64 0.41-21.amzn2 amzn2-core 66 k
pigz x86_64 2.3.4-1.amzn2.0.1 amzn2-core 81 k
runc x86_64 1.1.3-1.amzn2 amzn2extra-docker 2.9 M
Transaction Summary
-----
Install 1 Package (+4 Dependent packages)
Total download size: 69 M
Installed size: 260 M
Is this ok [y/d/N]: y
Downloading packages:
(1/5): libcgrouper-0.41-21.amzn2.x86_64.rpm | 66 kB 00:00:00
(2/5): pigz-2.3.4-1.amzn2.0.1.x86_64.rpm | 81 kB 00:00:00
(3/5): containerd-1.6.6-1.amzn2.x86_64.rpm | 27 MB 00:00:00
(4/5): docker-20.10.17-1.amzn2.x86_64.rpm | 39 MB 00:00:00
(5/5): runc-1.1.3-1.amzn2.x86_64.rpm | 2.9 MB 00:00:00
Total 75 MB/s | 69 MB 00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Installing : runc-1.1.3-1.amzn2.x86_64 1/5
Installing : containerd-1.6.6-1.amzn2.x86_64 2/5
Installing : libcgrouper-0.41-21.amzn2.x86_64 3/5
Installing : pigz-2.3.4-1.amzn2.0.1.x86_64 4/5
Installing : docker-20.10.17-1.amzn2.x86_64 5/5
Verifying : docker-20.10.17-1.amzn2.x86_64 1/5
Verifying : runc-1.1.3-1.amzn2.x86_64 2/5
Verifying : pigz-2.3.4-1.amzn2.0.1.x86_64 3/5
Verifying : containerd-1.6.6-1.amzn2.x86_64 4/5
Verifying : libcgrouper-0.41-21.amzn2.x86_64 5/5
Installed:
docker.x86_64 0:20.10.17-1.amzn2
Complete!
[root@ip-172-31-88-131 ec2-user]# service docker start
```



```
Installed:
docker.x86_64 0:20.10.17-1.amzn2
Dependency Installed:
containerd.x86_64 0:1.6.6-1.amzn2 libcgrouper.x86_64 0:0.41-21.amzn2 pigz.x86_64 0:2.3.4-1.amzn2.0.1 runc.x86_64 0:1.1.3-1.amzn2
Complete!
[root@ip-172-31-88-131 ec2-user]# service docker start
```


service docker start

docker pull nginx

```
Complete!
[root@ip-172-31-88-131 ec2-user]# service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-172-31-88-131 ec2-user]# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
7a6db449b51b: Pull complete
ca1981074b58: Pull complete
d4019c921e20: Pull complete
7cb804d746d4: Pull complete
67a561026262: Pull complete
7247f6e5c102: Pull complete
Digest: sha256:b95a99feebf7797479e0c5eb5ec0bdfa5d9f504bc94da550c2f58e839ea6914f
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
[root@ip-172-31-88-131 ec2-user]#
```

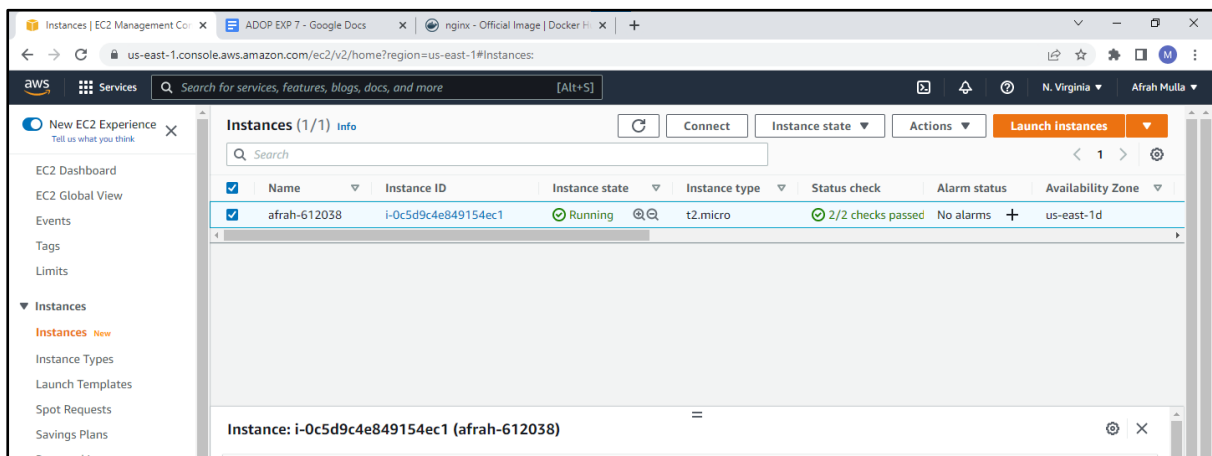
docker images

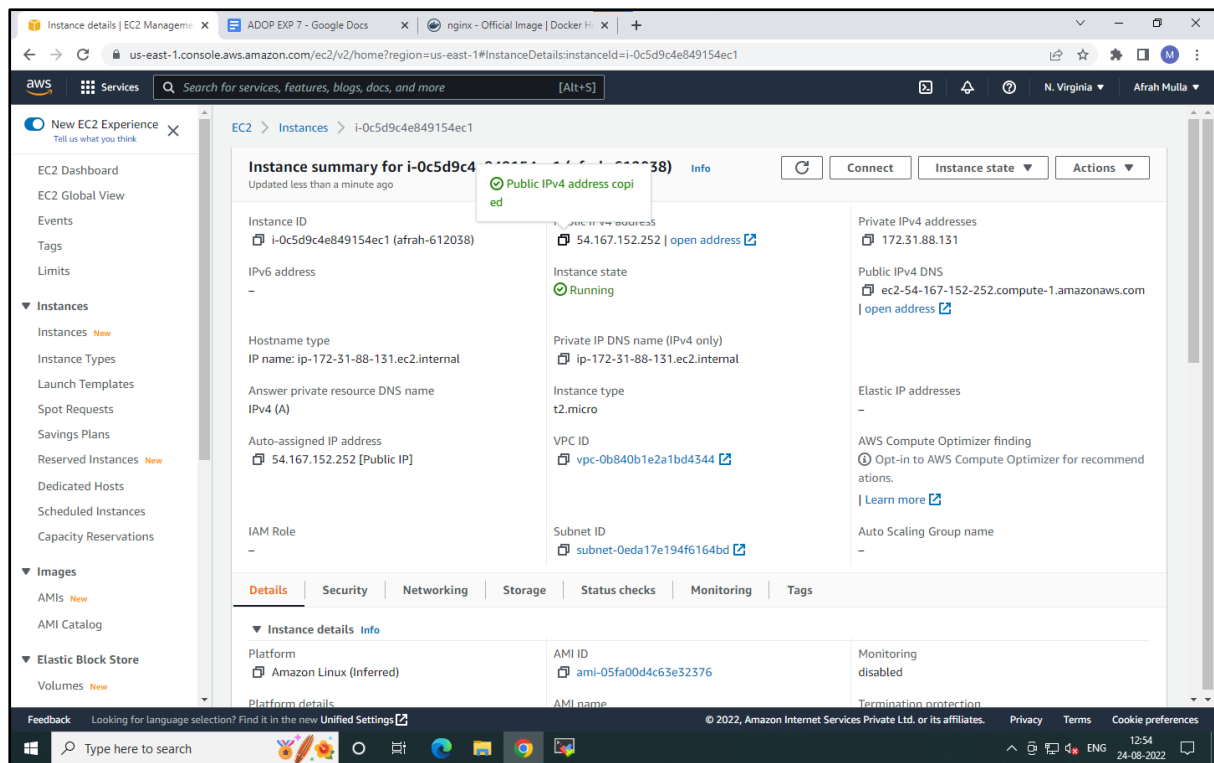
```
docker.io/library/nginx:latest
[root@ip-172-31-88-131 ec2-user]# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
nginx latest 2b7d6430f78d 27 hours ago 142MB
[root@ip-172-31-88-131 ec2-user]#
```

docker run -p 80:80 nginx

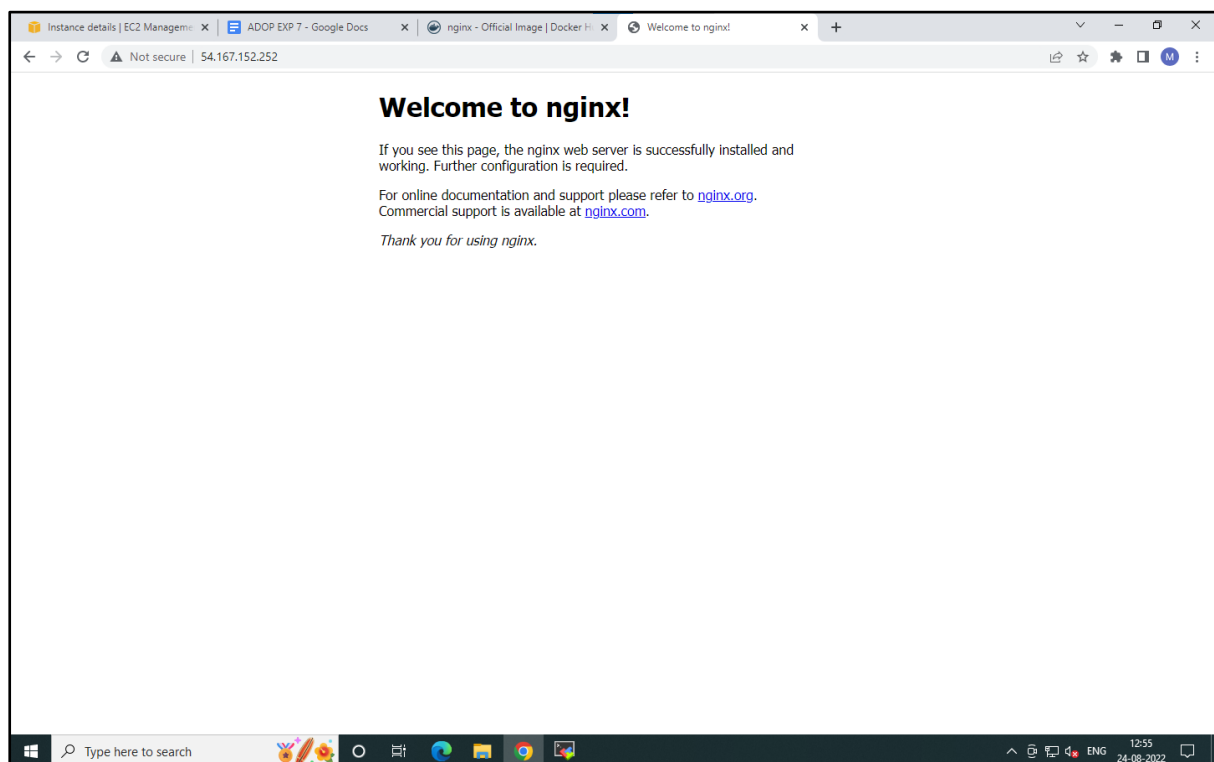
```
nginx latest 2b7d6430f78d 27 hours ago 142MB
[root@ip-172-31-88-131 ec2-user]# docker run -p 80:80 nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2022/08/24 07:23:10 [notice] #1: using the "epoll" event method
2022/08/24 07:23:10 [notice] #1: nginx/1.23.1
2022/08/24 07:23:10 [notice] #1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2022/08/24 07:23:10 [notice] #1: OS: Linux 5.10.130-110.517.amzn2.x86_64
2022/08/24 07:23:10 [notice] #1: getrlimit(RLIMIT_NOFILE): 32768:65536
2022/08/24 07:23:10 [notice] #1: start worker processes
2022/08/24 07:23:10 [notice] #1: start worker process 31
```

Step 17: To see if Nginx Web Server is successfully installed and working, go to Linux Instance and copy the 'Public IPv4 address'

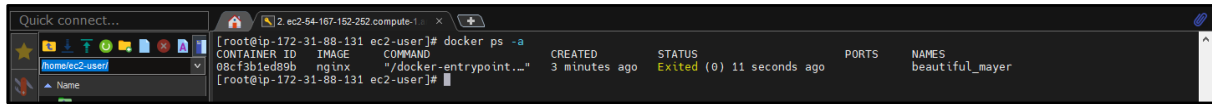




Step 18: Paste the Public IPv4 address in a web browser and the following web page will be displayed which means you have successfully installed Nginx Web Server



Step 19: To see the Container ID run the command –
docker ps -a



```
[root@ip-172-31-98-131 ec2-user]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS      PORTS       NAMES
08cf3b1ed89b   nginx    "/docker-entrypoint..." 3 minutes ago    Exited (0) 11 seconds ago           beautiful_mayer
[root@ip-172-31-98-131 ec2-user]#
```

Step 20: Lastly, terminate the EC2 instance and delete the IAM role that you have created

