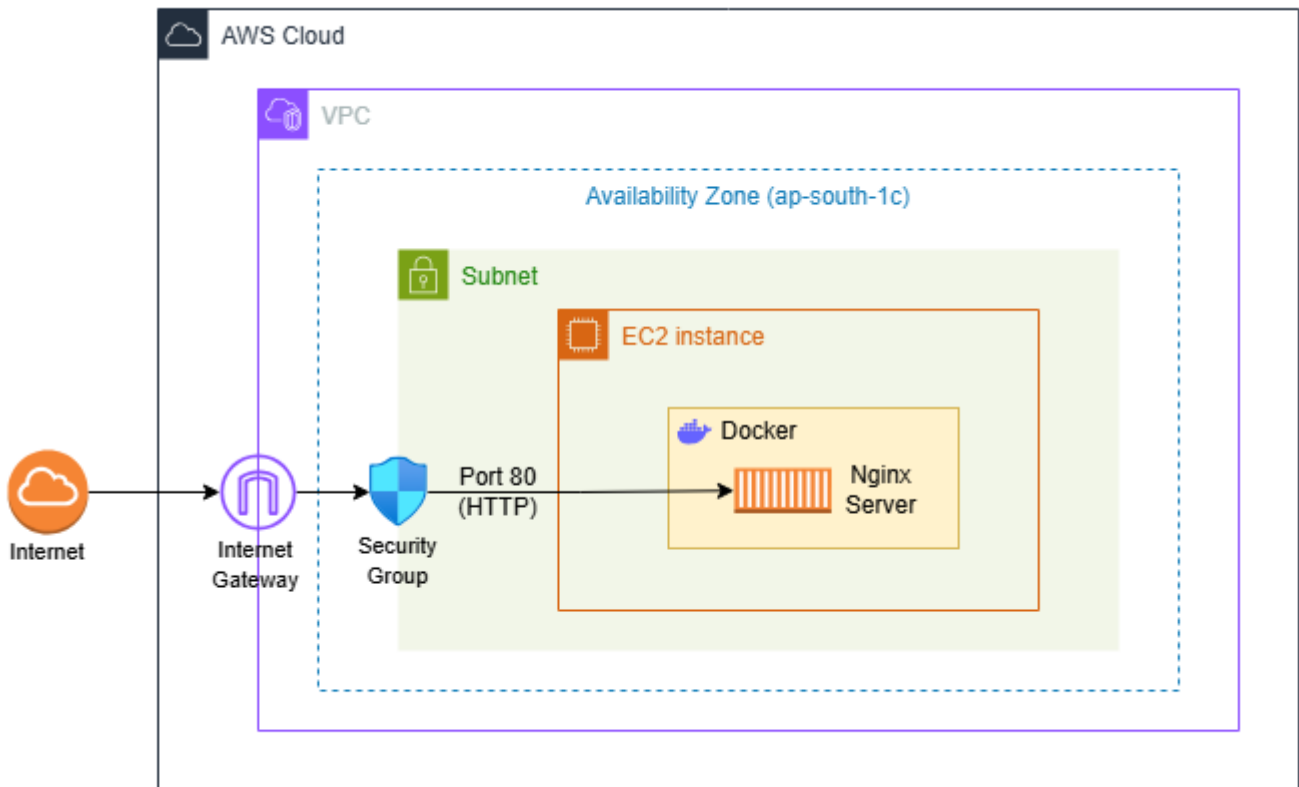


TASK 3 - AWS EC2 Deployment

Requirements: Launch EC2 (cost-optimized: t2.micro / t3.micro), Install Docker and Run containers on EC2.

Prerequisites: An AWS account with permission to create EC2 instances and Security Groups.

Architecture:



An EC2 instance is launched inside a VPC/subnet with an Internet Gateway attached, and a Security Group allows inbound HTTP (80) so users on the internet can access a NGINX server running inside a Docker container on the instance.

Components and roles:

VPC + Subnet (network boundary): The VPC is your isolated network in AWS, and the EC2 instance is placed inside a subnet within that VPC (diagram shows a single Availability Zone and subnet). Because this instance is reachable from the internet, the subnet effectively behaves as a public subnet (i.e., it has a route to the internet via an Internet Gateway and the instance has a public IP).

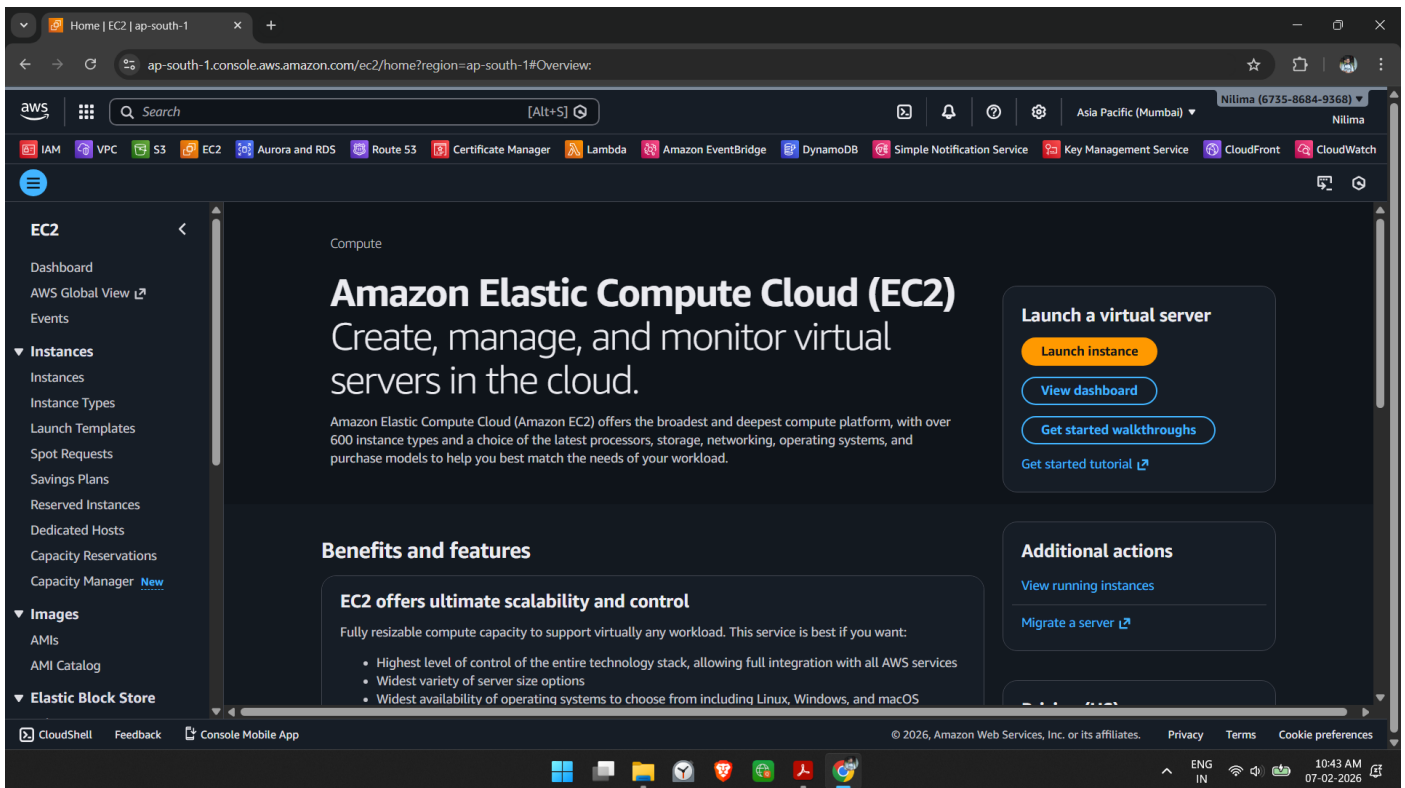
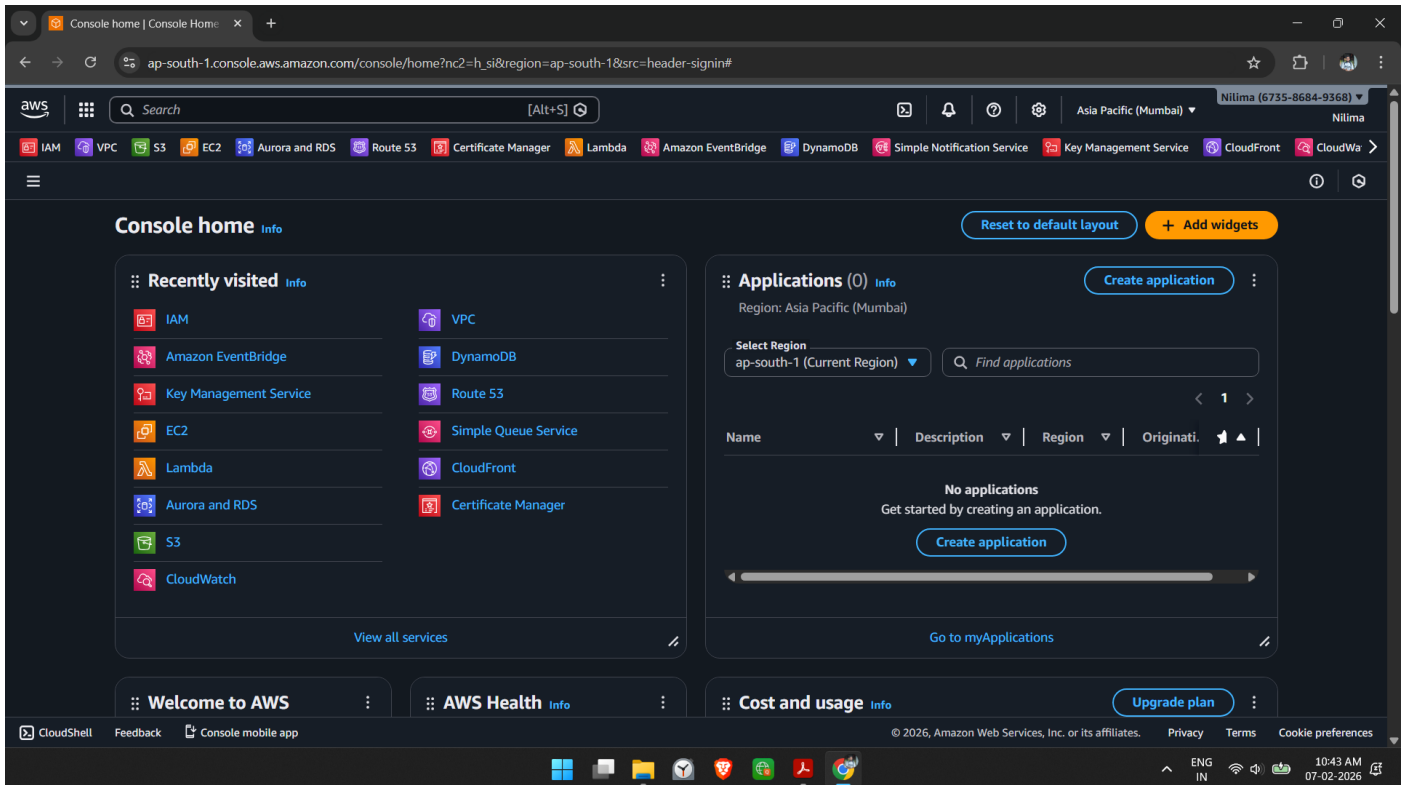
Internet Gateway (IGW) (internet connectivity): An Internet Gateway is the VPC component that enables communication between resources in your VPC and the public internet. For internet access to work, the subnet's route table typically includes a default route 0.0.0.0/0 pointing to the IGW, which sends non-local traffic out to the internet.

EC2 instance (compute host): The EC2 instance is the virtual machine that runs Docker and hosts your application workload. It has a public IPv4 address (shown in screenshots), which is what you use in the browser to reach the web server.

Step-by-step procedure:

I) Launch an EC2 instance

1. Open the AWS Console and go to EC2 → Launch instance.



2. Provide an instance name (shown as docker).

Launch an instance | EC2 | ap-south-1

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

Search [Alt+S]

EC2 > Instances > Launch an instance

Launch an instance

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags

Name: [Add additional tags](#)

Application and OS Images (Amazon Machine Image)

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose [Browse more AMIs](#).

Search our full catalog including 1000s of application and OS images

Recents Quick Start

[Browse more AMIs](#)
Including AMIs from AWS, Marketplace and the Community

Summary

Number of instances:

Software Image (AMI)
Amazon Linux 2023 AMI 2023.10...[read more](#)
ami-0ff5003538b60d5ec

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#) [Preview code](#)

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

10:44 AM 07-02-2026

3. Select Amazon Linux 2023 AMI and choose an instance type (shown as t3.micro).

Launch an instance | EC2 | ap-south-1

ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#LaunchInstances:

Search [Alt+S]

EC2 > Instances > Launch an instance

Amazon Machine Image (AMI)

Amazon Linux 2023 kernel-6.1 AMI Free tier eligible

ami-0ff5003538b60d5ec (64-bit (x86), uefi-preferred) / ami-0f0613925dab57b30 (64-bit (Arm), uefi)
Virtualization: hvm ENA enabled: true Root device type: ebs

Description

Amazon Linux 2023 (kernel-6.1) is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a secure, stable and high-performance execution environment to develop and run your cloud applications.

Amazon Linux 2023 AMI 2023.10.20260120.4 x86_64 HVM kernel-6.1

Architecture	Boot mode	AMI ID	Publish Date	Username
64-bit (x86)	uefi-preferred	ami-0ff5003538b60d5ec	2026-01-23	ec2-user Verified provider

Instance type

Instance type: Free tier eligible

Family: t3 2 vCPU 1 GiB Memory Current generation: true

On-Demand Linux base pricing: 0.0112 USD per Hour On-Demand SUSE base pricing: 0.0112 USD per Hour

On-Demand Windows base pricing: 0.0204 USD per Hour

On-Demand Ubuntu Pro base pricing: 0.0147 USD per Hour On-Demand RHEL base pricing: 0.04 USD per Hour

[Additional costs apply for AMIs with pre-installed software](#)

☐ All generations [Compare instance types](#)

Summary

Number of instances:

Software Image (AMI)
Amazon Linux 2023 AMI 2023.10...[read more](#)
ami-0ff5003538b60d5ec

Virtual server type (instance type)
t3.micro

Firewall (security group)
New security group

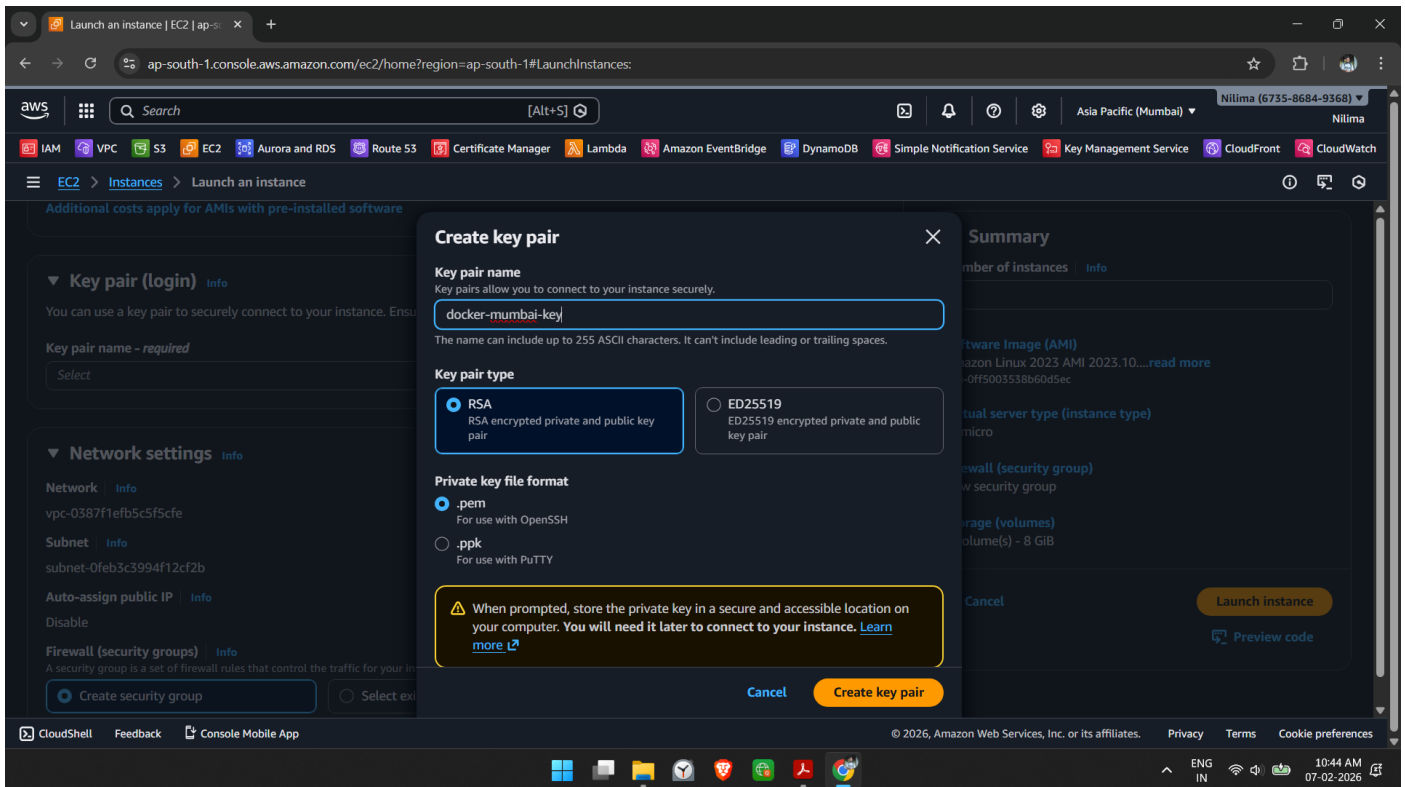
Storage (volumes)
1 volume(s) - 8 GiB

[Cancel](#) [Launch instance](#) [Preview code](#)

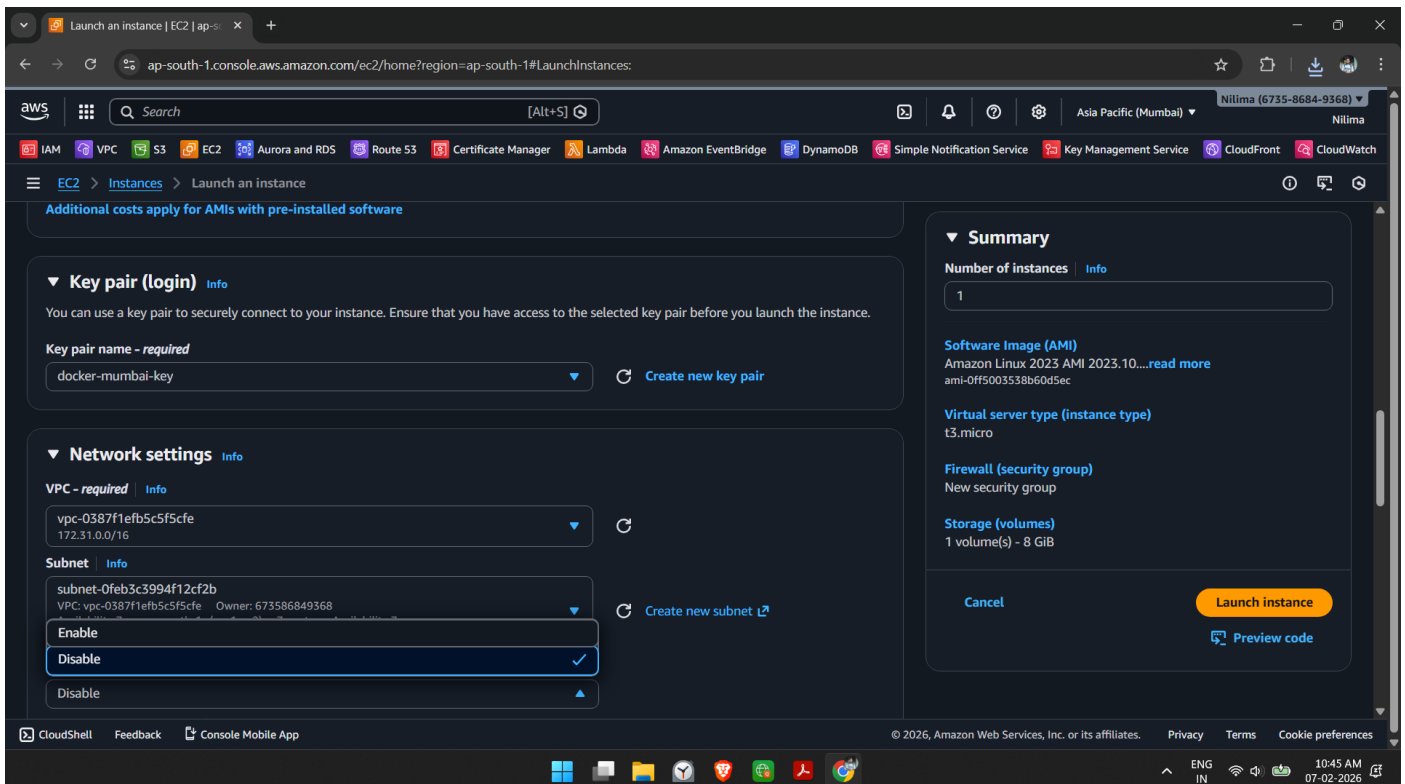
© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

10:44 AM 07-02-2026

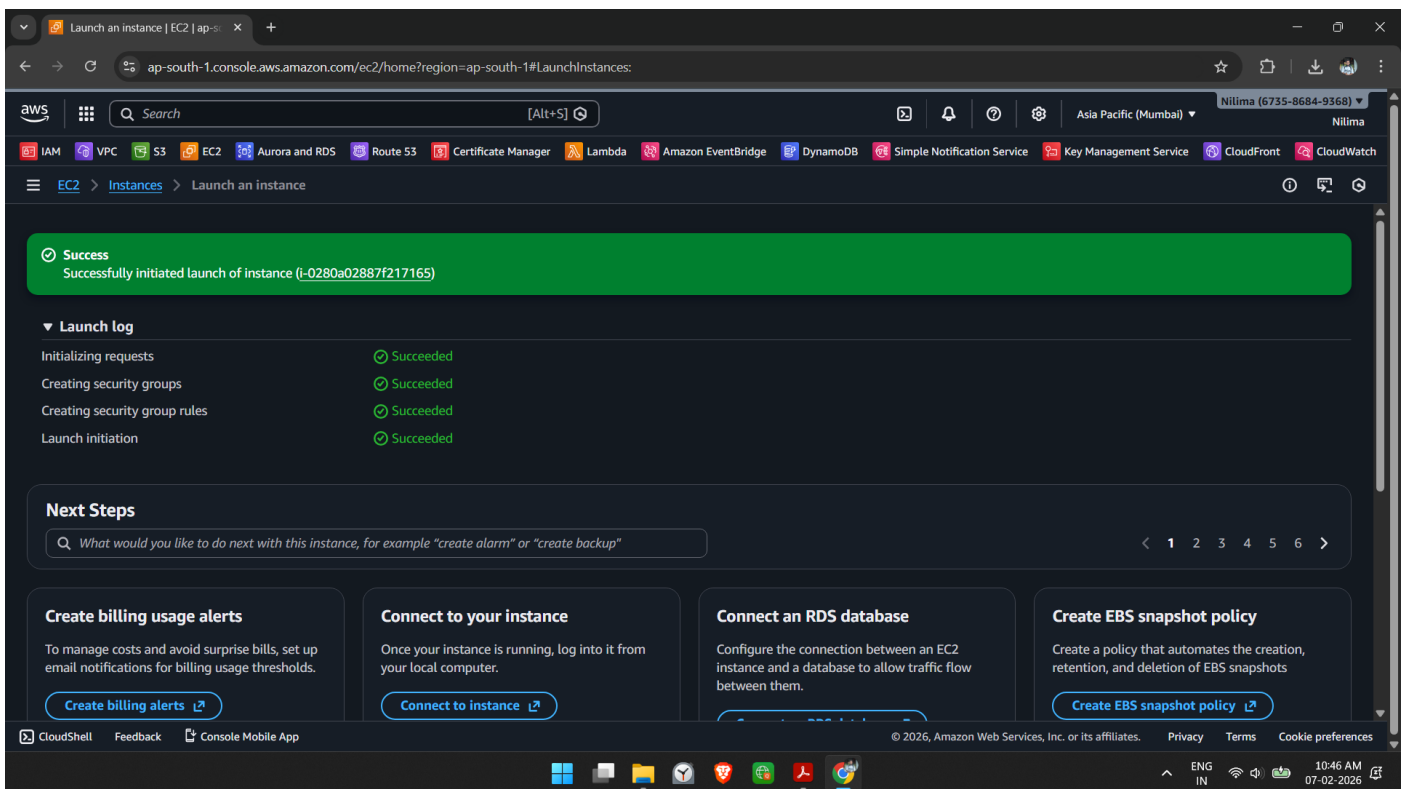
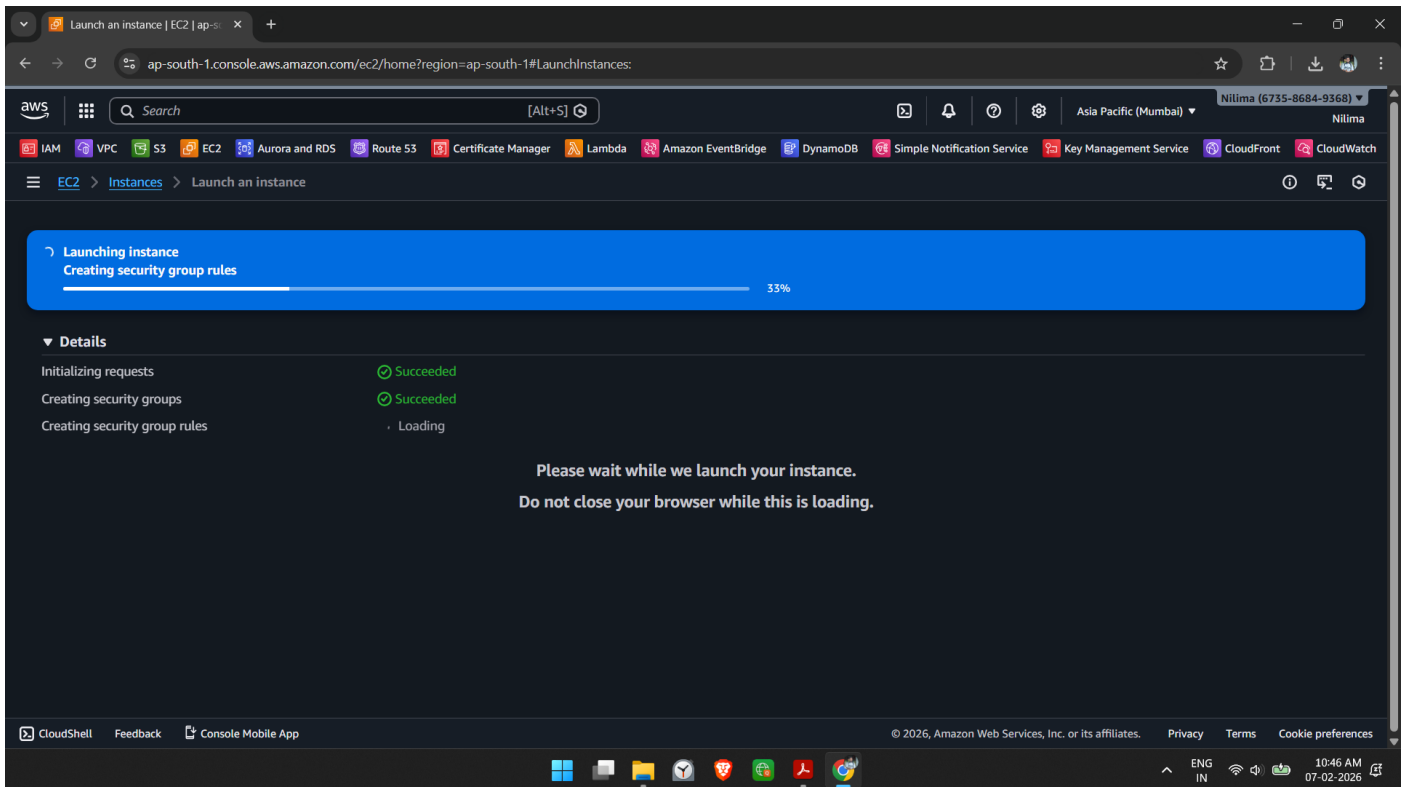
4. Create/select a key pair (shown as docker-mumbai-key) and download the .pem file.



5. In Network settings, ensure the instance will receive a public IPv4 address (Auto-assign public IP must be enabled so the website can be reached from the internet).



6. Click Launch instance and confirm the success message.



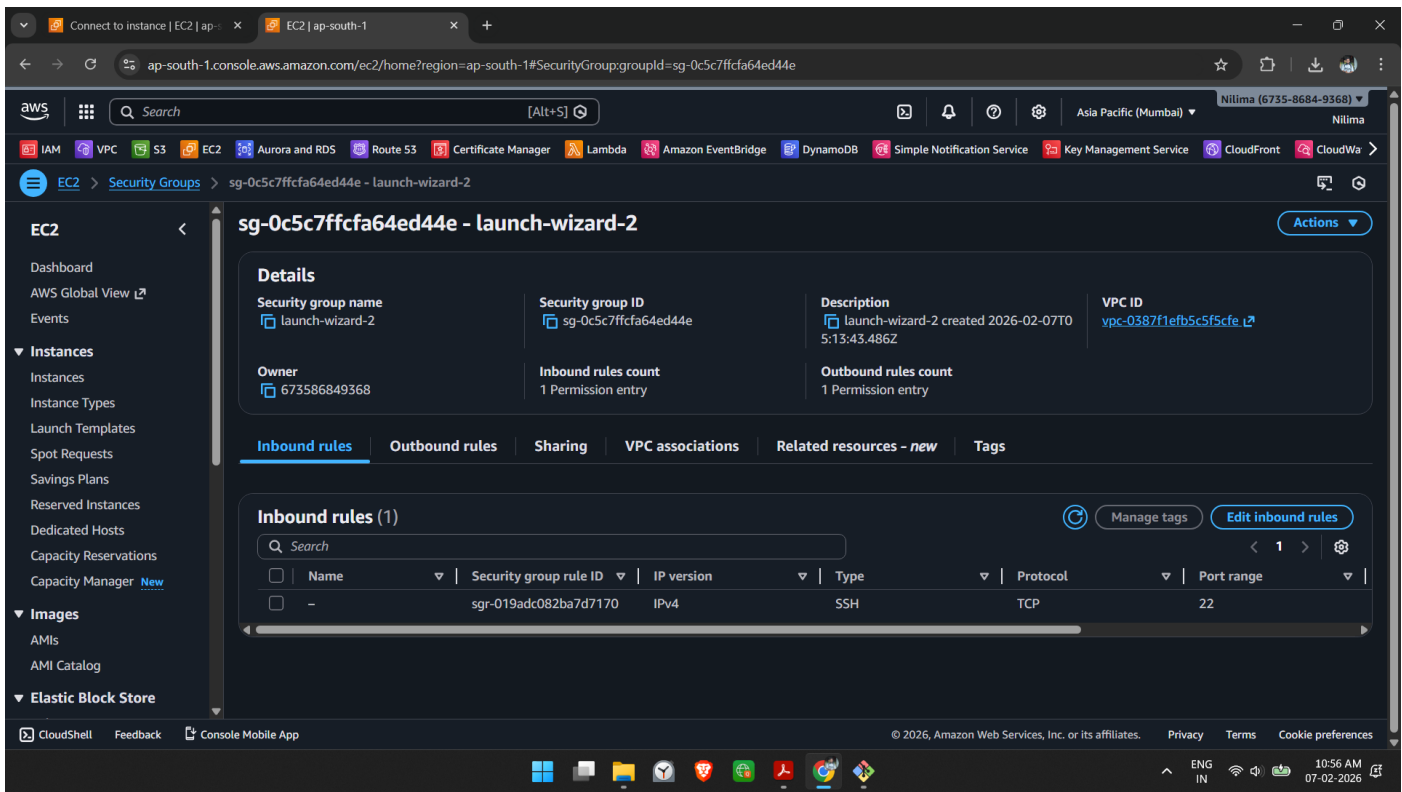
7. Open EC2 → Instances, select the instance, and copy the Public IPv4 address (shown as 13.203.65.241).

The screenshot shows the AWS Management Console for the 'ap-south-1' region. The 'Instances' page is active, displaying a table with one instance: 'docker' (ID: i-0280a02887f217165). The instance is in a 'Running' state, using the 't3.micro' type, and is located in the 'ap-south-1c' Availability Zone. The Public IPv4 address is 13.203.65.241. Below the table, the 'Details' tab for the selected instance shows the 'Instance summary' with the same information.

This public IP will be used later to test NGINX in the browser.

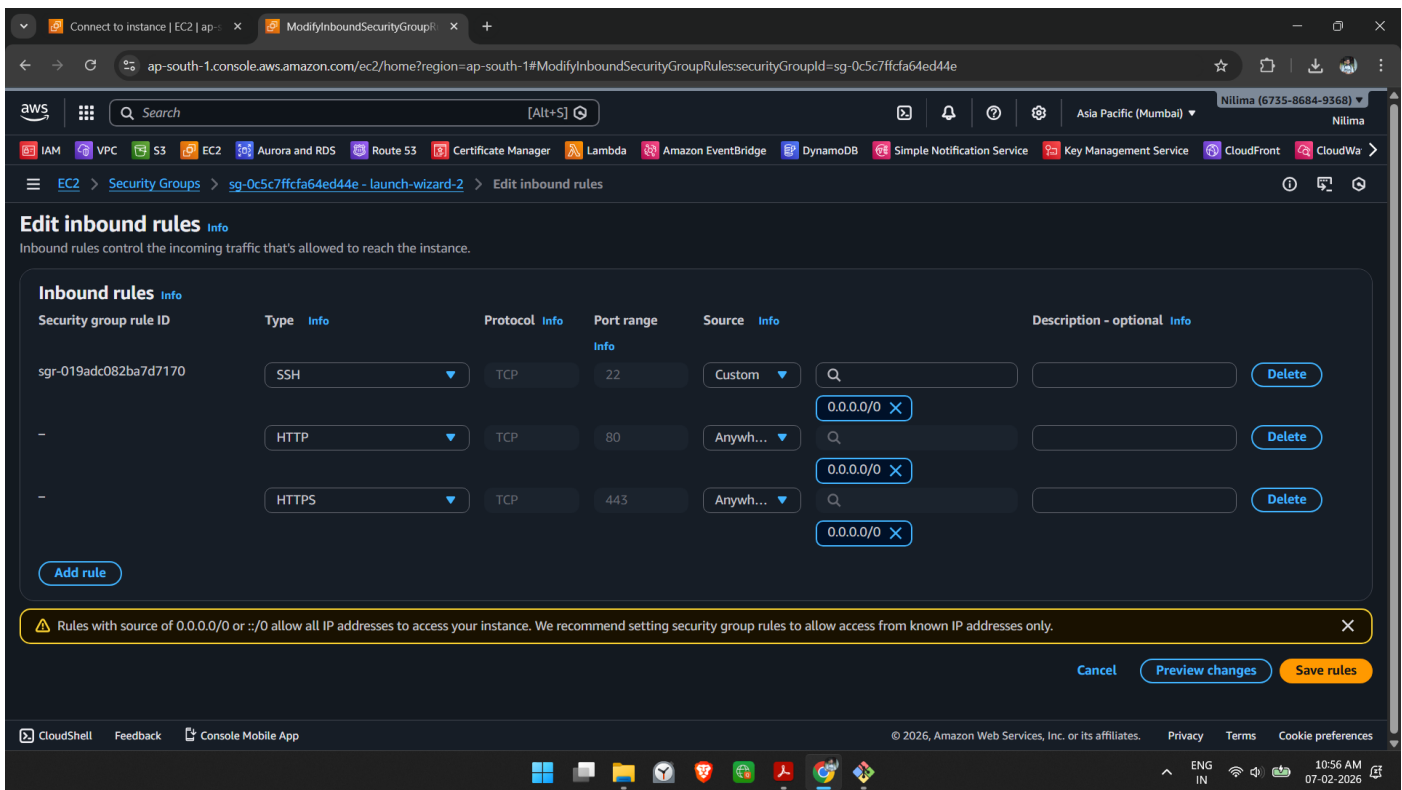
8. Open the instance's Security Group (shown as launch-wizard-2).

The screenshot shows the 'Connect to instance' page in the AWS Management Console. The page displays the instance ID 'i-0280a02887f217165 (docker)', VPC ID 'vpc-0387f1efb5c5f5cfe', Security groups 'sg-0c5c7ffcfa64ed44e (launch-wizard-2)', and IAM role '-'. The 'SSH client' tab is selected, showing instructions for connecting to the instance using SSH. The instructions include: 1. Open an SSH client. 2. Locate your private key file. The key used to launch this instance is docker-mumbai-key.pem. 3. Run this command, if necessary, to ensure your key is not publicly viewable. 4. Connect to your instance using its Public IP: 13.203.65.241. An example command is provided: `ssh -i "docker-mumbai-key.pem" ec2-user@13.203.65.241`. A note states: 'Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.'



9. Edit inbound rules to allow:

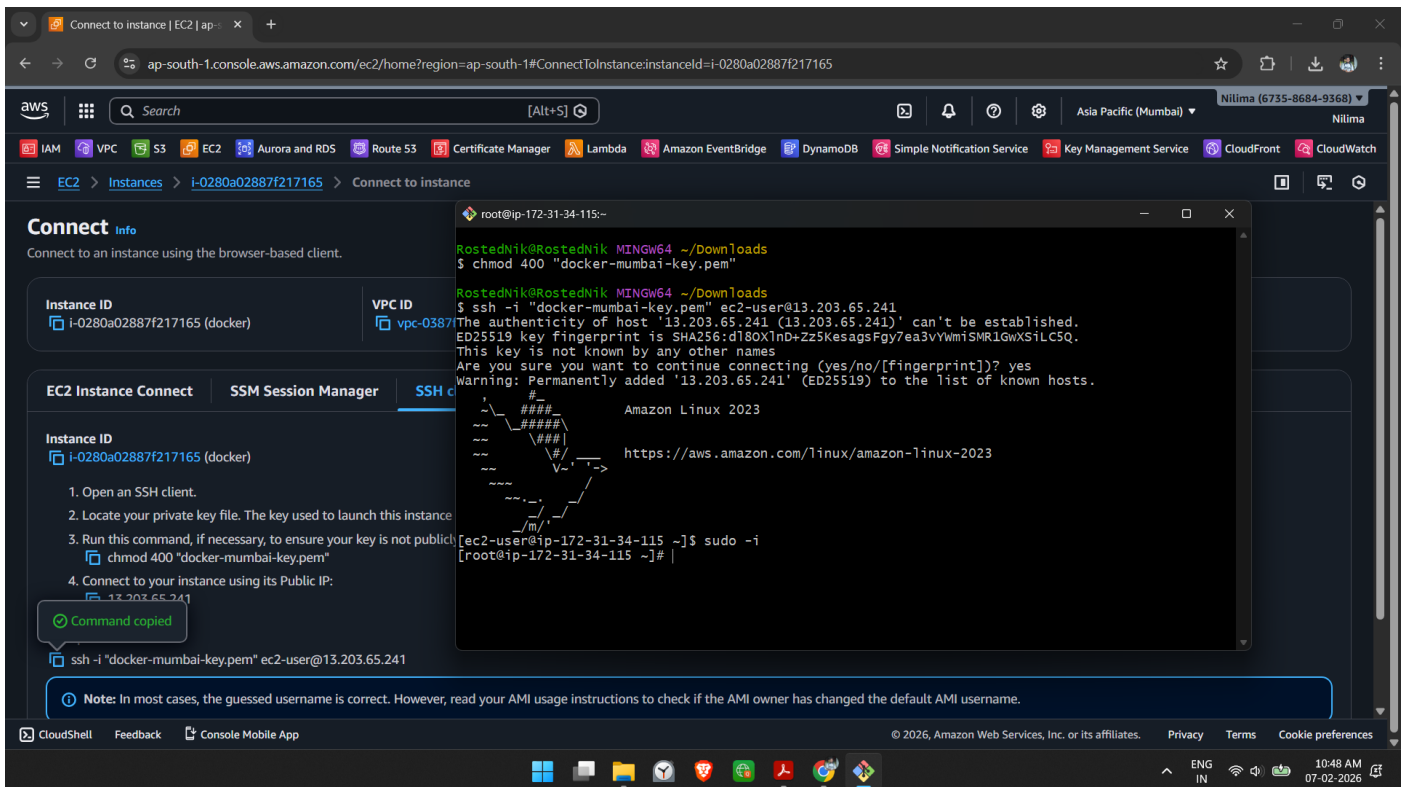
- SSH (22) for administration,
- HTTP (80) for the NGINX website,
- HTTPS (443) optional (as shown in rule edit screen).



Save the rules (your screenshots show sources set to 0.0.0.0/0, meaning open to the internet).

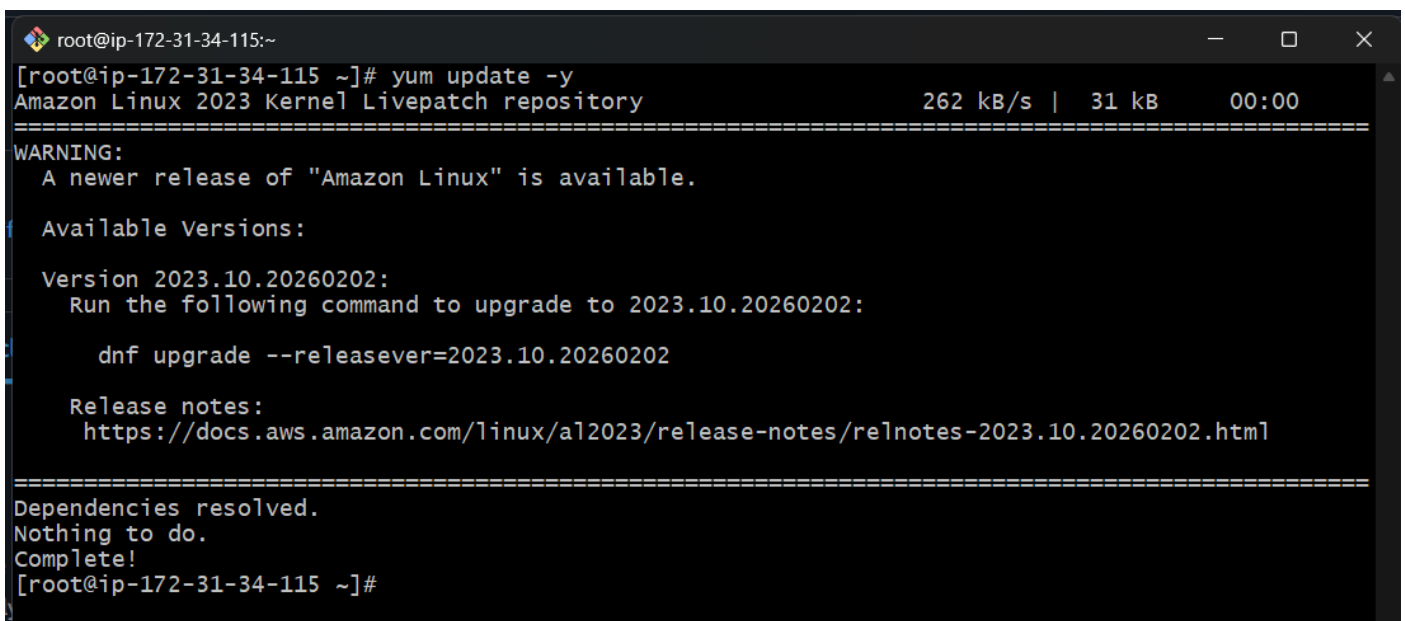
Note: Security groups are virtual firewalls that control inbound/outbound traffic to EC2, and allowing HTTP/HTTPS enables public web access.

10. From your local machine, set correct permissions on the key file:
`chmod 400 docker-mumbai-key.pem`
11. Connect using the SSH command provided by the EC2 console (pattern shown in your screenshot):
`ssh -i "docker-mumbai-key.pem" ec2-user@13.203.65.241`
12. Accept the host fingerprint prompt on first connection (the “yes” confirmation is shown).



II) Install and start Docker

13. Switch to root (shown as `sudo -i`).
14. Update packages (shown as `yum update -y`).



15. Install Docker (shown as `yum install docker -y`).

```
root@ip-172-31-34-115:~
Complete!
[root@ip-172-31-34-115 ~]# yum install docker -y
Last metadata expiration check: 0:00:32 ago on Sat Feb  7 05:18:51 2026.
Dependencies resolved.
=====
Package                                Architecture Version                                Repository                                Size
=====
Installing:
docker                                x86_64      25.0.14-1.amzn2023.0.1                  amazonlinux                                46 M
Installing dependencies:
container-selinux                     noarch      4:2.242.0-1.amzn2023                    amazonlinux                                58 k
containerd                            x86_64      2.1.5-1.amzn2023.0.4                    amazonlinux                                23 M
iptables-libs                         x86_64      1.8.8-3.amzn2023.0.2                    amazonlinux                                401 k
iptables-nft                         x86_64      1.8.8-3.amzn2023.0.2                    amazonlinux                                183 k
libcgroup                             x86_64      3.0-1.amzn2023.0.1                      amazonlinux                                75 k
libnetfilter_conntrack               x86_64      1.0.8-2.amzn2023.0.2                    amazonlinux                                58 k
libnfnetlink                         x86_64      1.0.1-19.amzn2023.0.2                   amazonlinux                                30 k
libnftnl                             x86_64      1.2.2-2.amzn2023.0.2                    amazonlinux                                84 k
pigz                                  x86_64      2.5-1.amzn2023.0.3                      amazonlinux                                83 k
runc                                  x86_64      1.3.4-1.amzn2023.0.1                    amazonlinux                                3.9 M
=====
Transaction Summary
=====
Install 11 Packages

Total download size: 74 M
Installed size: 281 M
Downloading Packages:
```

16. Start Docker and enable it on boot (shown as `systemctl start docker` and `systemctl enable docker`).

17. (Optional) Add user to the docker group (shown as `usermod -aG docker ec2-user`).

18. Verify Docker service and/or version (shown via `systemctl status docker` and `docker --version`).

```
MINGW64:/c/Users/Nikhilesh Sakhare/Downloads
https://docs.aws.amazon.com/linux/a12023/release-notes/relnotes-2023.10.20260202.html
=====
Installed:
container-selinux-4:2.242.0-1.amzn2023.noarch      containerd-2.1.5-1.amzn2023.0.4.x86_64
docker-25.0.14-1.amzn2023.0.1.x86_64              iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
iptables-nft-1.8.8-3.amzn2023.0.2.x86_64          libcgroup-3.0-1.amzn2023.0.1.x86_64
libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64 libnfnetlink-1.0.1-19.amzn2023.0.2.x86_64
libnftnl-1.2.2-2.amzn2023.0.2.x86_64              pigz-2.5-1.amzn2023.0.3.x86_64
runc-1.3.4-1.amzn2023.0.1.x86_64
=====
Complete!
[root@ip-172-31-34-115 ~]# systemctl start docker
[root@ip-172-31-34-115 ~]# systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
[root@ip-172-31-34-115 ~]# usermod -aG docker ec2-user
[root@ip-172-31-34-115 ~]# docker --version
Docker version 25.0.14, build 0bab007
[root@ip-172-31-34-115 ~]# exit
logout
[ec2-user@ip-172-31-34-115 ~]$ exit
logout
Connection to 13.203.65.241 closed.

RostedNik@RostedNik MINGW64 ~/Downloads
$ |
```

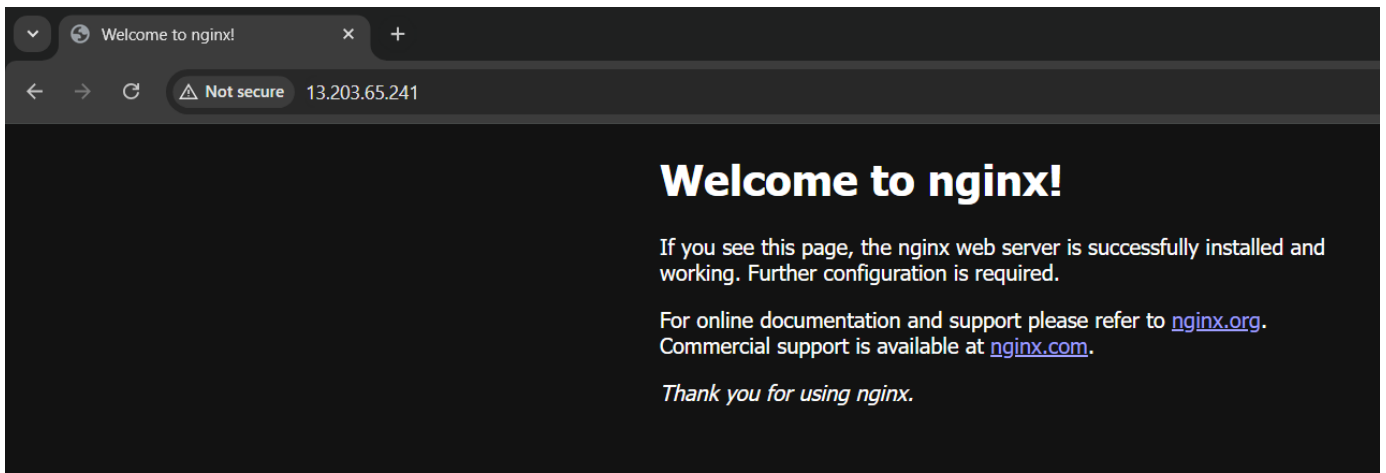
```
root@ip-172-31-34-115:~  
Last login: Sat Feb  7 05:17:54 2026 from 103.121.71.174  
[ec2-user@ip-172-31-34-115 ~]$ sudo -i  
[root@ip-172-31-34-115 ~]# systemctl status docker  
● docker.service - Docker Application Container Engine  
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)  
   Active: active (running) since Sat 2026-02-07 05:20:49 UTC; 6min ago  
TriggeredBy: ● docker.socket  
   Docs: https://docs.docker.com  
  Main PID: 27062 (dockerd)  
    Tasks: 9  
  Memory: 29.5M  
   CPU: 377ms  
  CGroup: /system.slice/docker.service  
          └─27062 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --de  
Feb 07 05:20:48 ip-172-31-34-115.ap-south-1.compute.internal systemd[1]: Starting docker.service>  
Feb 07 05:20:48 ip-172-31-34-115.ap-south-1.compute.internal dockerd[27062]: time="2026-02-07T05>  
Feb 07 05:20:48 ip-172-31-34-115.ap-south-1.compute.internal dockerd[27062]: time="2026-02-07T05>  
Feb 07 05:20:49 ip-172-31-34-115.ap-south-1.compute.internal dockerd[27062]: time="2026-02-07T05>  
Feb 07 05:20:49 ip-172-31-34-115.ap-south-1.compute.internal dockerd[27062]: time="2026-02-07T05>  
Feb 07 05:20:49 ip-172-31-34-115.ap-south-1.compute.internal dockerd[27062]: time="2026-02-07T05>  
Feb 07 05:20:49 ip-172-31-34-115.ap-south-1.compute.internal dockerd[27062]: time="2026-02-07T05>  
Feb 07 05:20:49 ip-172-31-34-115.ap-south-1.compute.internal systemd[1]: Started docker.service >  
[root@ip-172-31-34-115 ~]#
```

III) Run NGINX as a Docker container

19. List containers (shown as `docker ps -a`) to confirm current state.
20. Run the official NGINX image in detached mode and map host port 80 to container port 80:
`docker run -d -p 80:80 --name my-web-server nginx`
21. Confirm the container is running and ports are mapped (shown by `docker ps` with `0.0.0.0:80->80/tcp`).

```
root@ip-172-31-34-115:~  
[root@ip-172-31-34-115 ~]# docker ps -a  
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES  
[root@ip-172-31-34-115 ~]# docker run -d -p 80:80 --name my-web-server nginx  
Unable to find image 'nginx:latest' locally  
latest: Pulling from library/nginx  
0c8d55a45c0d: Pull complete  
46bf3a120c8e: Pull complete  
4f4efe02d542: Pull complete  
7b6cb8ccac7b: Pull complete  
f73400a233fd: Pull complete  
47cd406a84ef: Pull complete  
bae5a1799a80: Pull complete  
Digest: sha256:341bf0f3ce6c5277d6002cf6e1fb0319fa4252add24ab6a0e262e0056d313208  
Status: Downloaded newer image for nginx:latest  
fd35f9c067d32d51c292d1e507b3a6d650f6812133d32c99ae10f5a3a4be71eb  
[root@ip-172-31-34-115 ~]# docker ps  
CONTAINER ID   IMAGE     COMMAND                  CREATED    STATUS    PORTS    NAMES  
fd35f9c067d3   nginx     "/docker-entrypoint...."  25 seconds ago    Up 24 seconds    0.0.0.0:80->80  
/tcp, :::80->80/tcp    my-web-server  
[root@ip-172-31-34-115 ~]#
```

22. Open a browser and access the site using the instance public IP:
`http://13.203.65.241`



The “Welcome to nginx!” default page confirms the NGINX container is serving traffic publicly.