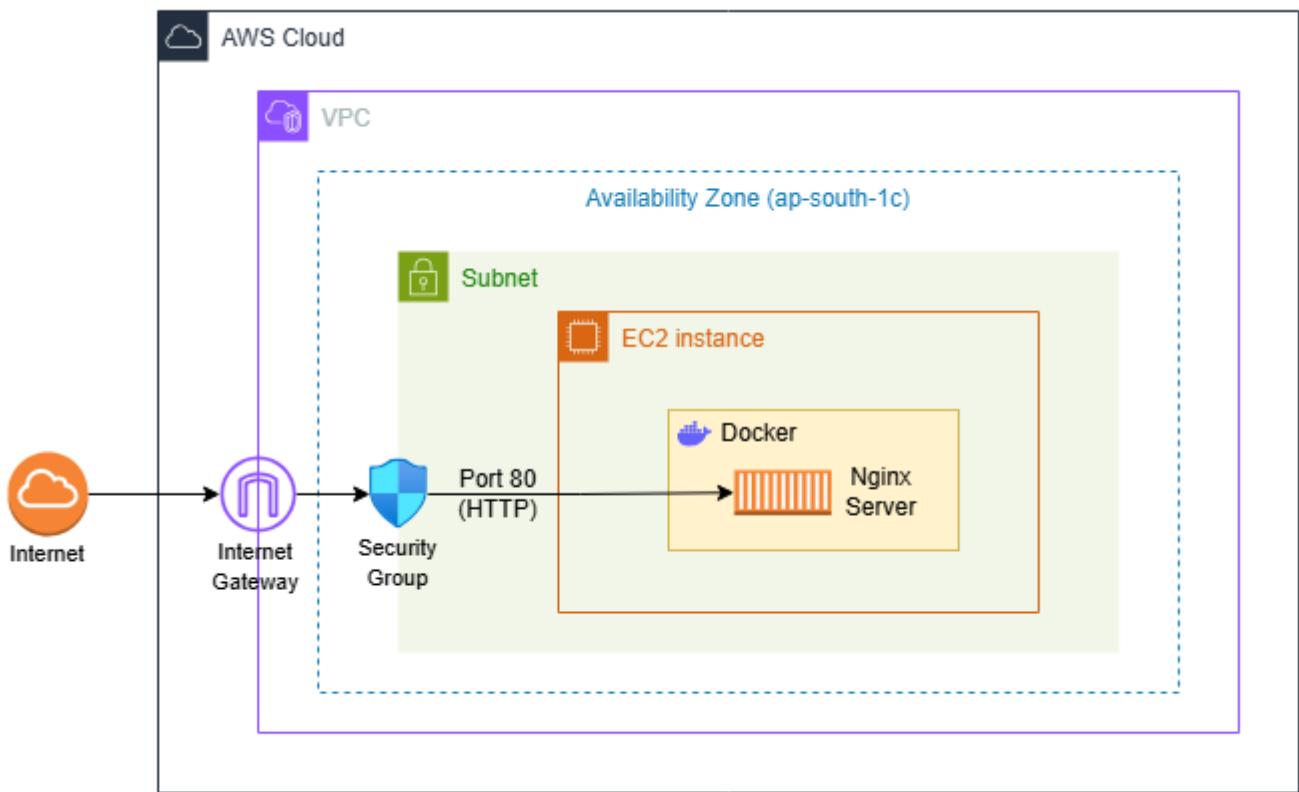


TASK 3 - AWS EC2 Deployment

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.

The screenshot shows the AWS Console home page. The navigation bar at the top includes links for IAM, VPC, S3, EC2, Aurora and RDS, Route 53, Certificate Manager, Lambda, Amazon EventBridge, DynamoDB, Simple Notification Service, Key Management Service, CloudFront, and CloudWatch. The EC2 icon is highlighted. The main content area features a sidebar titled "Recently visited" with links to various AWS services like IAM, VPC, and S3. To the right, there are sections for "Applications" (0), "AWS Health", and "Cost and usage". The EC2 section is expanded, showing a "Launch a virtual server" button and options like "Launch instance", "View dashboard", and "Get started walkthroughs". The status bar at the bottom indicates the user is in the Asia Pacific (Mumbai) region, logged in as Nilima, and the current time is 10:43 AM on 07-02-2026.

The screenshot shows the EC2 service page. The left sidebar lists categories: EC2 (selected), Dashboard, AWS Global View, Events, Instances (selected), Instances Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Capacity Manager, Images, AMIs, AMI Catalog, and Elastic Block Store. The main content area is titled "Amazon Elastic Compute Cloud (EC2)" and sub-titled "Create, manage, and monitor virtual servers in the cloud.". It features a callout for "Launch a virtual server" with buttons for "Launch instance", "View dashboard", and "Get started walkthroughs". Below this, there's a section titled "Benefits and features" with a sub-section "EC2 offers ultimate scalability and control". A bulleted list under this section includes: "Highest level of control of the entire technology stack, allowing full integration with all AWS services", "Widest variety of server size options", and "Widest availability of operating systems to choose from including Linux, Windows, and macOS". The status bar at the bottom indicates the user is in the Asia Pacific (Mumbai) region, logged in as Nilima, and the current time is 10:43 AM on 07-02-2026.

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

The screenshot shows the 'Launch an instance' wizard in the AWS Management Console. The current step is 'Name and tags'. A text input field contains the value 'docker'. To the right, there's a button labeled 'Add additional tags'. Below this section is a summary panel titled 'Summary' which includes fields for 'Number of instances' (set to 1), 'Software Image (AMI)' (Amazon Linux 2023 AMI 2023.10...), 'Virtual server type (instance type)' (t3.micro), 'Firewall (security group)' (New security group), and 'Storage (volumes)' (1 volume(s) - 8 GiB). At the bottom right of the summary panel are 'Cancel', 'Launch instance', and 'Preview code' buttons.

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

This screenshot shows the continuation of the 'Launch an instance' wizard. In the 'Amazon Machine Image (AMI)' step, it lists 'Amazon Linux 2023 kernel-6.1 AMI' as the selected option. This AMI is described as being 'Free tier eligible'. Below this, there's a 'Description' section for Amazon Linux 2023, followed by a table with details like Architecture (64-bit (x86)), Boot mode (uefi-preferred), AMI ID (ami-0ff5003538b60d5ec), Publish Date (2026-01-23), and Username (ec2-user). In the 'Verified provider' column, there's a green button. The 'Instance type' step follows, showing 't3.micro' as the selected instance type, also marked as 'Free tier eligible'. It provides detailed pricing information for On-Demand and On-Demand RHEL base pricing. Below this, there's a note about additional costs for pre-installed software. The summary panel on the right remains the same as in the previous step, including the 'Launch instance' button.

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

The screenshot shows the 'Create key pair' dialog box over the EC2 instance creation interface. In the 'Key pair name' field, 'docker-mumbai-key' is entered. Under 'Key pair type', 'RSA' is selected. Under 'Private key file format', '.pem' is selected. A note at the bottom of the dialog box states: 'When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance.' A 'Create key pair' button is visible at the bottom right of the dialog.

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).

The screenshot shows the 'Network settings' section of the EC2 instance creation interface. Under 'VPC - required', 'VPC: vpc-0387f1efb5c5fcfe' is selected. Under 'Subnet', 'subnet-0feb3c3994f12cf2b' is selected. The 'Enable' option is selected for 'Auto-assign public IP'. On the right side, a summary panel shows the selected AMI (Amazon Linux 2023 AMI 2023.10), instance type (t3.micro), and storage (1 volume(s) - 8 GiB). A 'Launch instance' button is visible at the bottom right of the summary panel.

6. Click Launch instance and confirm the success message.

The screenshot shows the AWS EC2 'Launch an instance' progress bar. The status bar indicates 'Creating security group rules' and is at 33% completion. Below the progress bar, there's a 'Details' section showing the status of three tasks: 'Initializing requests' (Succeeded), 'Creating security groups' (Succeeded), and 'Creating security group rules' (Loading). A central message says 'Please wait while we launch your instance. Do not close your browser while this is loading.'

The screenshot shows the AWS EC2 'Launch an instance' success page. A green success message states 'Successfully initiated launch of instance i-0280a02887f217165'. Below it, a 'Launch log' section shows the status of five tasks: 'Initializing requests' (Succeeded), 'Creating security groups' (Succeeded), 'Creating security group rules' (Succeeded), and 'Launch initiation' (Succeeded). The 'Next Steps' section includes links to 'Create billing usage alerts', 'Connect to your instance', 'Connect an RDS database', and 'Create EBS snapshot policy'.

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 with the EC2 service selected. In the left navigation pane, 'Instances' is expanded, and 'Instances' is selected. The main content area displays a table titled 'Instances (1/1) Info' with one row. The row details are:

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DN |
|--------|---------------------|----------------|---------------|--------------|--------------|-------------------|----------------|
| docker | i-0280a02887f217165 | Running | t3.micro | Initializing | | ap-south-1c | - |

Below the table, the instance details for 'i-0280a02887f217165 (docker)' are shown. The 'Details' tab is selected, displaying the following information:

| Instance ID | Public IPv4 address | Private IP4 addresses |
|---|--|-----------------------|
| i-0280a02887f217165 | 13.203.65.241 open address ↗ | 172.31.34.115 |
| IPv6 address | Instance state | Public DNS |
| - | Running | - |
| Hostname type | Private IP DNS name (IPv4 only) | |
| IP name: in-172-31-34-115.ap-south-1.compute.internal | in-172-31-34-115.ap-south-1.compute.internal | |

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 for the instance 'i-0280a02887f217165 (docker)'. The 'SSH client' tab is selected. The page displays the following information:

| Instance ID | VPC ID | Security groups | IAM role |
|------------------------------|-----------------------|---------------------------------------|----------|
| i-0280a02887f217165 (docker) | vpc-0387f1efb5c5f5fce | sg-0c5c7ffca64ed44e (launch-wizard-2) | - |

Below the table, there are four tabs: 'EC2 Instance Connect', 'SSM Session Manager', 'SSH client' (selected), and 'EC2 serial console'. Under the 'SSH client' tab, the instance ID is listed again, followed by a numbered list of steps:

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.
chmod 400 "docker-mumbai-key.pem"
4. Connect to your instance using its Public IP:
13.203.65.241

Below the steps, there is an 'Example:' section with a command:

```
ssh -i "docker-mumbai-key.pem" ec2-user@13.203.65.241
```

A note at the bottom 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."

sg-0c5c7ffca64ed44e - launch-wizard-2

Details

| | | | |
|--|---|---|--------------------------------|
| Security group name launch-wizard-2 | Security group ID sg-0c5c7ffca64ed44e | Description launched-wizard-2 created 2026-02-07T05:13:43.486Z | VPC ID vpc-0387f1efb5c5f5ce |
| Owner 673586849368 | Inbound rules count 1 Permission entry | Outbound rules count 1 Permission entry | |

Inbound rules (1)

| Name | Security group rule ID | IP version | Type | Protocol | Port range |
|------|------------------------|------------|------|----------|------------|
| - | sgr-019adc082ba7d7170 | IPv4 | SSH | TCP | 22 |

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).

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

| Security group rule ID | Type | Protocol | Port range | Source | Description - optional |
|------------------------|-------|----------|------------|----------|------------------------|
| sgr-019adc082ba7d7170 | SSH | TCP | 22 | Custom | 0.0.0.0/0 |
| - | HTTP | TCP | 80 | Anywh... | 0.0.0.0/0 |
| - | HTTPS | TCP | 443 | Anywh... | 0.0.0.0/0 |

Add rule

⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel **Preview changes** **Save rules**

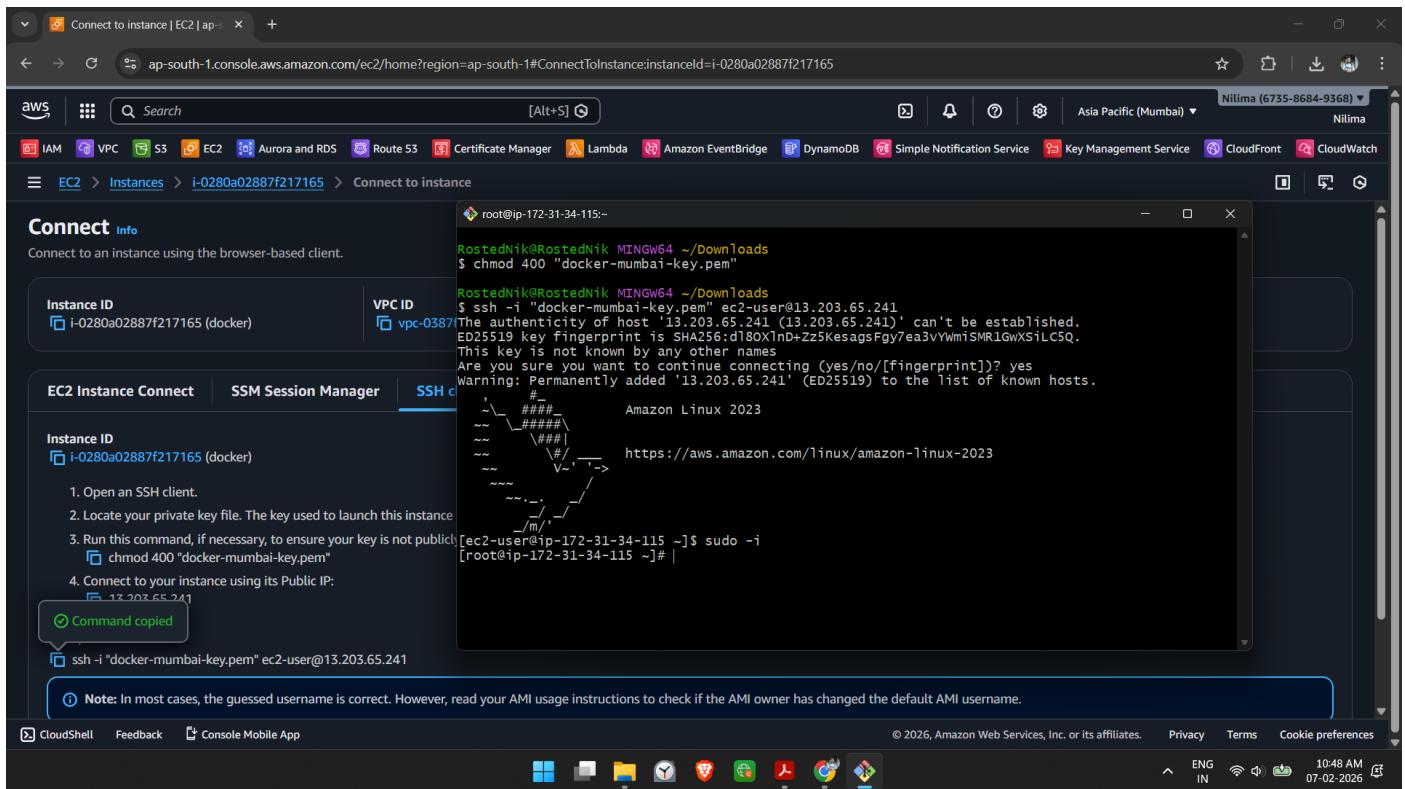
Save the rules (your screenshots show sources set to 0.0.0.0/0, meaning open to the internet).

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

- From your local machine, set correct permissions on the key file:

```
chmod 400 docker-mumbai-key.pem
```
- 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
```
- Accept the host fingerprint prompt on first connection (the “yes” confirmation is shown).



The screenshot shows the AWS CloudShell interface. On the left, there's a sidebar with 'EC2' selected, showing 'Instances' and 'I-0280a02887f217165'. Below it are tabs for 'EC2 Instance Connect', 'SSM Session Manager', and 'SSH connect' (which is currently active). The main area is a terminal window titled 'root@ip-172-31-34-115~'. It displays the following SSH session:

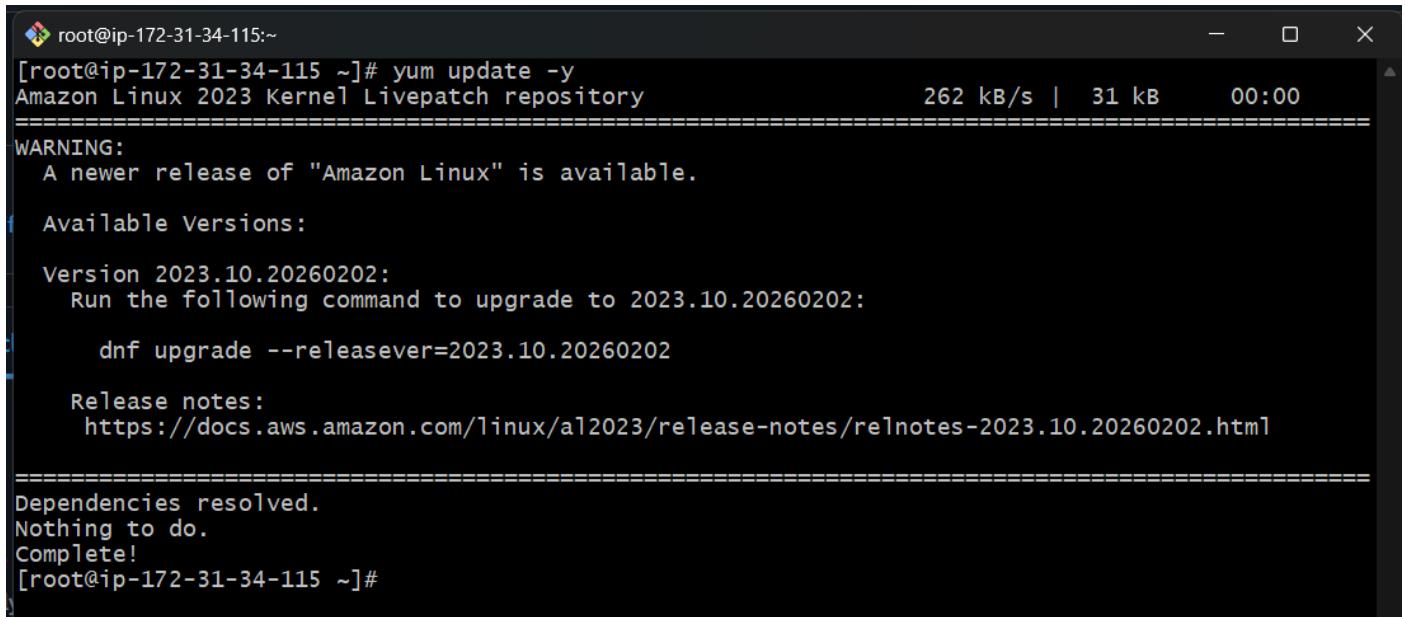
```
RostedNik@RostedNik MINGW64 ~/Downloads
$ chmod 400 "docker-mumbai-key.pem"
RostedNik@RostedNik MINGW64 ~/Downloads
$ ssh -i "docker-mumbai-key.pem" ec2-user@13.203.65.241
The authenticity of host '13.203.65.241 (13.203.65.241)' can't be established.
ED25519 key fingerprint is SHA256:d180X1nD+Zz5KesagsFgy7ea3vYwmisMRlGwxsLc5Q.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.203.65.241' (ED25519) to the list of known hosts.

[...]
[ec2-user@ip-172-31-34-115 ~]$ sudo -i
[root@ip-172-31-34-115 ~]# |
```

A green box highlights the command copied: `ssh -i "docker-mumbai-key.pem" ec2-user@13.203.65.241`. A note at the bottom says: "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."

II) Install and start Docker

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



The screenshot shows the AWS CloudShell terminal with the root user. The command `yum update -y` is run, and the output is as follows:

```
[root@ip-172-31-34-115 ~]# yum update -y
Amazon Linux 2023 Kernel Livepatch repository
=====
WARNING:
A newer release of "Amazon Linux" is available.

Available Versions:
Version 2023.10.20260202:
Run the following command to upgrade to 2023.10.20260202:
dnf upgrade --releasever=2023.10.20260202

Release notes:
https://docs.aws.amazon.com/linux/al2023/release-notes/relnotes-2023.10.20260202.html

=====
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-34-115 ~]#
```

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.

Transaction Summary
=====
Install  11 Packages

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

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
  libnfnetwork     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/al2023/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 libnfnetwork-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:~#
~~..-.-/-
-/m/`-
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 systemd[1]: Started docker.service >

[root@ip-172-31-34-115 ~]#

```

III) Run NGINX as a Docker container

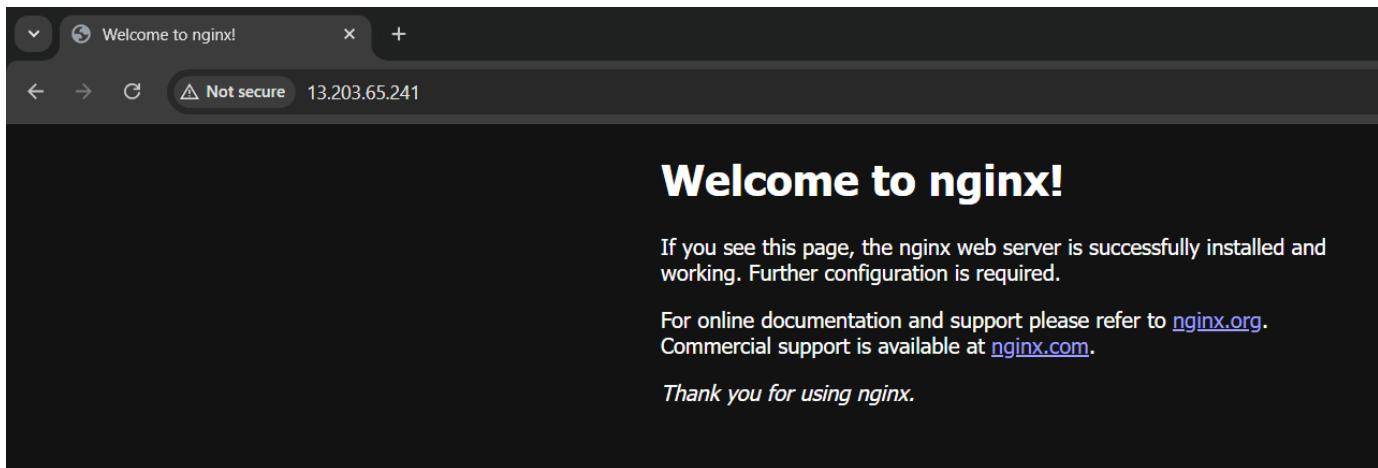
19. List containers (shown as `docker ps -a`) to confirm current state.
20. Run the official NGINX image in detached mode and map host port 80 to container port 80:
`docker run -d -p 80:80 --name my-web-server nginx`
21. Confirm the container is running and ports are mapped (shown by `docker ps` with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