

# Docker Assignment – Assignment 8

---

Author: Ayush Singh

Roll No: 22bcse63



## Step-by-Step Docker Instructions

### 1. Install Docker on Your Local Machine

If you're using **Ubuntu** or an **Ubuntu-based EC2 instance**, run the following:

```
bash
CopyEdit
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
sudo systemctl status docker
```

Ensure it shows: **active (running)**.

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-8-53:~$ sudo systemctl start docker
ubuntu@ip-172-31-8-53:~$ sudo systemctl enable docker
ubuntu@ip-172-31-8-53:~$ 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 2025-06-22 12:50:40 UTC; 1min 6s ago
      TriggeredBy: • docker.socket
        Docs: https://docs.docker.com
     Main PID: 2285 (dockerd)
       Tasks: 8
      Memory: 29.3M (peak: 29.5M)
        CPU: 259ms
       CGroup: /system.slice/docker.service
               └─2285 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

### 2. Pull Nginx Image and Run as a Container

**Step 1:** Pull the image:

```
docker pull nginx
```

**Step 2:** Run the container:

```
docker run -it -d --name my-nginx -p 80:80 nginx
```

This creates a container named `my-nginx` and maps port 80 of your machine to the container.

```
ubuntu@ip-172-31-8-53:~$ docker pull nginx
Using default tag: latest
permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://<ip>/v1.47/images/create?fromImage=nginx&tag=latest": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-8-53:~$ sudo su
root@ip-172-31-8-53:/home/ubuntu# cd
root@ip-172-31-8-53:# docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
dad67da3f26b: Pull complete
3b00567da964: Pull complete
56b61cfa547d: Pull complete
lbc5cdc8b475d: Pull complete
979e6233a40a: Pull complete
d2a7ba8dbfee: Pull complete
32e44235e1d5: Pull complete
Digest: sha256:6784fb0834aa7dbbe12e3d7471e69c290df3e6ba810dc38b34ae33d3c1c05f7d
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

## 3. Create a Custom Docker Network and Connect Nginx

**Step 1:** Create a bridge network:

```
docker network create my-custom-net
```

**Step 2:** Connect the container to the network:

```
docker network connect my-custom-net my-nginx
```

Now, `my-nginx` is connected to both the default bridge and custom network.



## 4. Verify Network Connection

Inspect the custom network:

```
docker network inspect my-custom-net
```

Look for:

- Container ID of my-nginx
- Assigned internal IP

```
root@ip-172-31-8-53:~# docker network create my-custom-net
5eac5509b73aa7dd5af0b587c638242301ccb64c8358bbb398e7d2b693976600
root@ip-172-31-8-53:~# docker network connect my-custom-net my-nginx
```

```
    "ConfigOnly": false,
    "Containers": {
        "44f0069dd4491414f613f07f55add61a3e75f76cf9f2bd85605ca36126703c4": {
            "Name": "my-nginx",
            "EndpointID": "a3f51a0933b79df395c5ca34c89e4a81c6b3a904fe04726d47dda68cc355ffee",
            "MacAddress": "02:42:ac:12:00:02",
            "IPv4Address": "172.18.0.2/16",
            "IPv6Address": ""
        }
    },
    "Options": {},
    "Labels": {}
```



## Docker & Virtualization – Theory



### What is a Hypervisor?

A hypervisor allows multiple virtual machines (VMs) to run on a single physical machine. It allocates resources like CPU, memory, and disk.

#### ❖ Types of Hypervisors:

Type	Description	Examples
Type 1	Bare-metal (runs on hardware)	VMware ESXi, Hyper-V
Type 2	Runs on OS as app	VirtualBox, VMware Workstation



### What is Containerization?

Containerization packages software + dependencies into a container. Unlike VMs, containers share the host OS kernel.

**Tool:** Docker

## Docker Use Case – Node.js Deployment

### Traditional VM-based Setup:

1. Create VM
2. Install Node.js
3. Place project files
4. Run `npm install`
5. Use Nginx/Apache as reverse proxy

 **Problems:** Version mismatch, dependency errors, inconsistent environments.

### Docker-based Node.js Setup:

1. Create a Dockerfile
2. Inside Dockerfile:

```
Dockerfile
CopyEdit
FROM node
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

### Benefits:

- Consistent environments
- No manual installs
- Shareable images
- Better resource efficiency

## Docker Installation on Ubuntu EC2

```
sudo su
cd
sudo apt update
```

```
sudo apt install docker.io -y  
sudo systemctl start docker  
sudo systemctl enable docker  
sudo systemctl status docker
```

## Running Docker Containers – Nginx Example

```
bash  
CopyEdit  
docker run -it -d -p 80:80 nginx
```

### Detach Mode (-d)

- Runs in background
- Doesn't show logs
- Use for long-running services
- Access logs via:

```
bash  
CopyEdit  
docker logs <container_id>
```

## Managing Containers

### Stopping:

```
bash  
CopyEdit  
docker container ls  
docker stop <container_id>
```

### Listing:

```
bash
```

```
CopyEdit
docker ps      # Running only
docker ps -a   # All containers
```

## Removing:

```
bash
CopyEdit
docker rm <container_id>
```

## Allow Non-Root User to Use Docker

```
bash
CopyEdit
sudo su - ubuntu
sudo usermod -aG docker $USER
sudo su
sudo su - ubuntu
docker
```

If no error on running docker, the setup is successful.

## Running Multiple Containers

Run Nginx again:

```
docker run -it -d -p 80:80 nginx
```

Run Ubuntu container:

```
docker run -it -d ubuntu
```

## Check Containers

```
docker container ls  
docker ps -a
```

You should see containers for both nginx and ubuntu.

## Creating a Docker Image with Dockerfile

### 9.1 Create Dockerfile:

```
vi Dockerfile
```

Inside:

```
Dockerfile  
FROM nginx:1.27.5
```

Save with :wq.

### 9.2 Build the Docker Image

```
docker build -t test .
```

### 9.3 Verify Image

```
docker image ls
```

Should show the image test.

### 9.4 Run Container from Custom Image

```
docker run -it -d -p 80:80 test
```

### 9.5 Check Running Containers

```
docker container ls
```