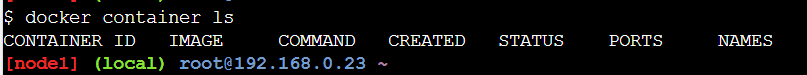
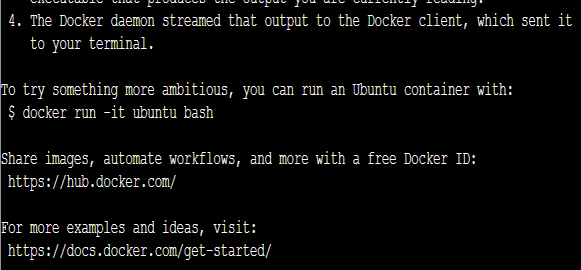
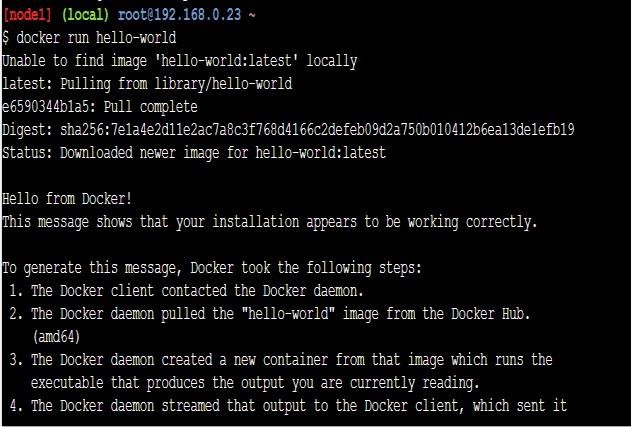
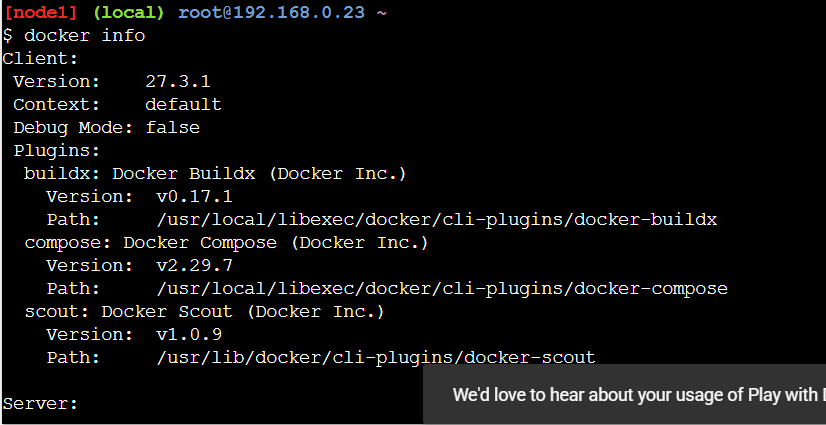
docker info

To view detailed information about your Docker installation



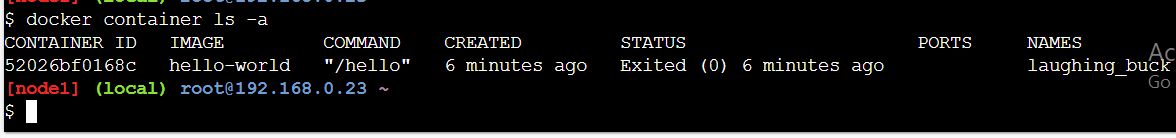
creating hello world image

docker container ls

lists all **running** Docker containers on your system.

docker container ls –a

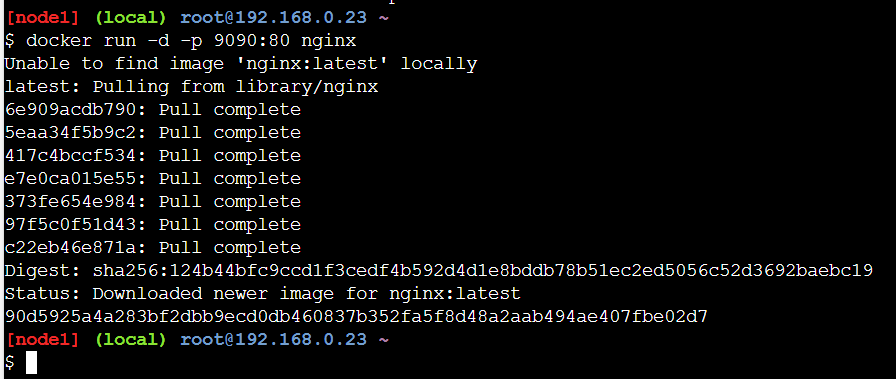
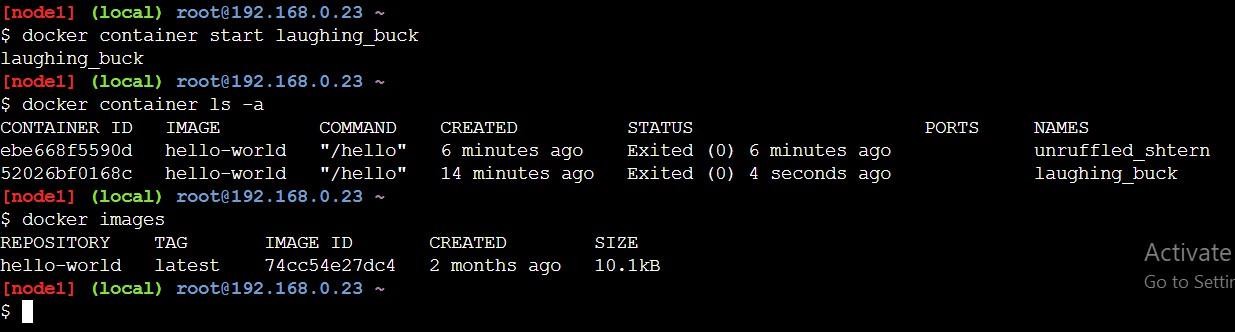
lists **all Docker containers** on your system — both **running and stopped**.



**docker container start laughing\_buck**

This command starts an **already created and exited container** named laughing\_buck. Docker confirms it by echoing back the container name.

**docker images**



Shows the available Docker images on your system.

docker run -d -p 9090:80 nginx

**Explanation:**

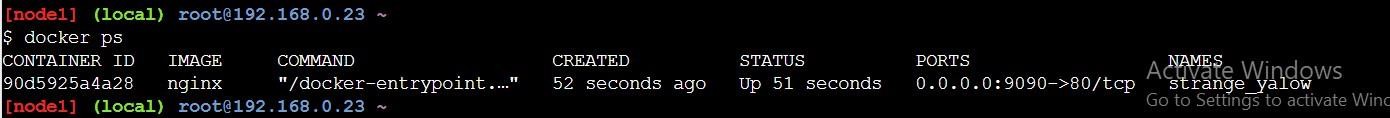
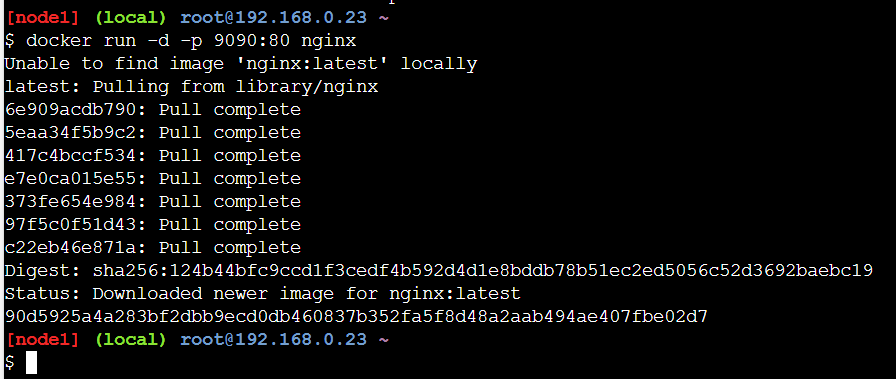
* docker run: Start a new Docker container.
* -d: Run it in **detached mode** (in the background).
* -p 9090:80: **Map port 9090** on your **host machine** to port **80 inside the container**

(where nginx serves web content).

* nginx: The **image name**. You’re asking Docker to run a container using the nginx image.

## What Docker did:

* It **couldn’t find the nginx image locally**, so it pulled the latest version from Docker Hub (library/nginx).
* You can see multiple image layers being downloaded (Pull complete).
* Finally, it **downloaded and started the nginx container**.



docker ps

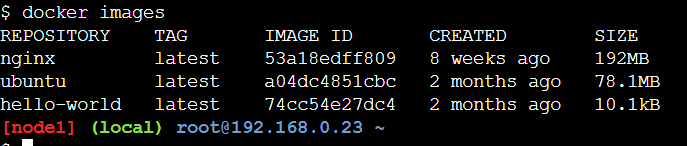
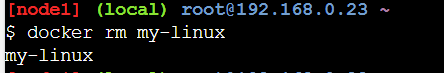
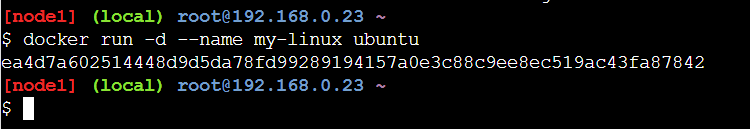
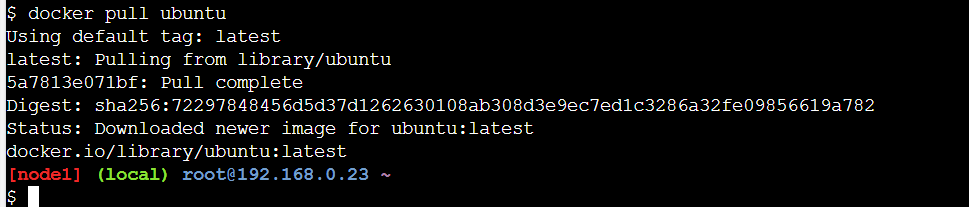
Now click on open port and type 9090

Nginx will start

## Pulling the Ubuntu image

docker pull Ubuntu

Pulling the latest **Ubuntu** image using Docker



lets check the image created or not using docker images

Now we are runnung an Ubuntu container in **detached mode**

-d: Detached mode (runs the container in the background).

--name my-linux: You named the container **my-linux**. ubuntu: You're using the Ubuntu image.

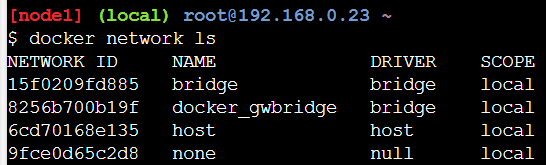
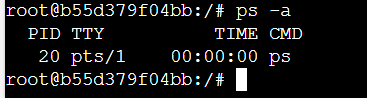
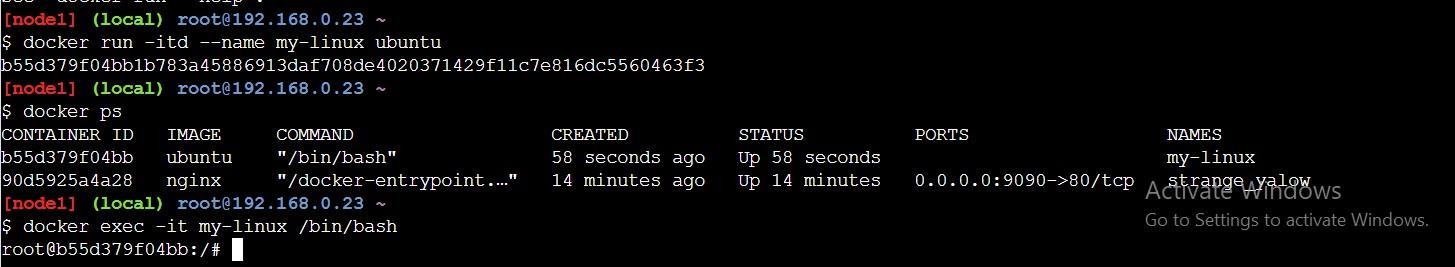
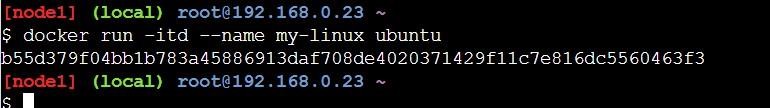
Now:

docker rm my-linux

* **docker rm**: Removes (deletes) a container.
* **my-linux**: That’s the name of the container you created earlier.

Since your container was already stopped (because it exited automatically), Docker let you remove it right away.

docker run -itd --name my-linux ubuntu



It runs an Ubuntu container in the background with an interactive terminal and names it my- linux.

docker ps

listed running containers

docker exec -it my-linux /bin/bash

accessed the running container interactively (bash shell)

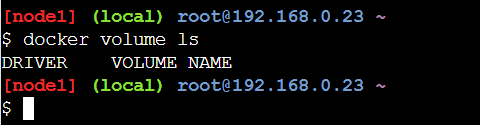
ps –a

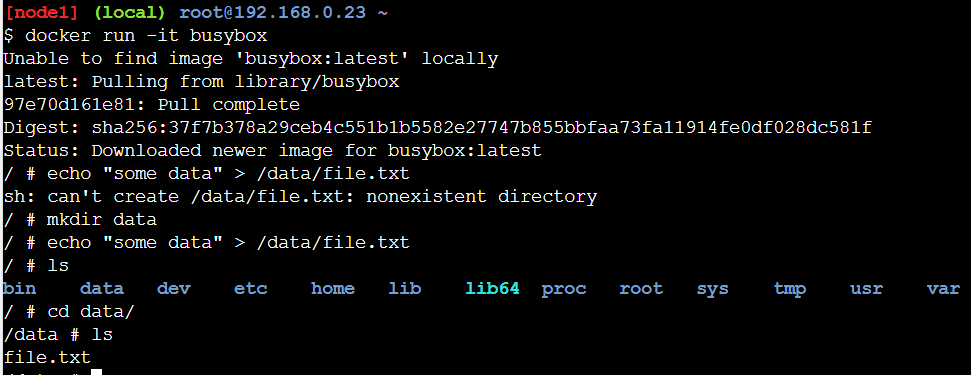
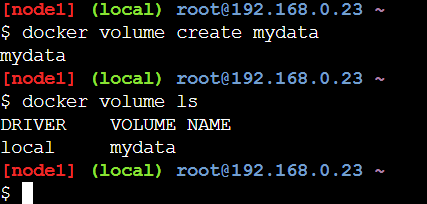
ps -a lists running processes inside the container.

docker ps command shows the list of running containers

docker network ls command on your Docker host docker network ls

docker volume ls command lists all Docker volumes available on your system.





docker volume create mydata creates a new Docker volume named mydata

**docker run -it busybox**

Starts an interactive container using the busybox image.

**echo "some data" > /data/file.txt**

Initially failed because the /data directory didn’t exist.

**mkdir data**

You created a new directory named data.

**echo "some data" > /data/file.txt**

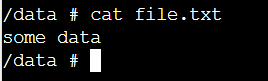
This time, it succeeded since /data now exists.

**ls and cd data/ && ls**

Confirmed that file.txt was created inside /data.

You created a /data directory inside a **busybox** container. Wrote "some data" into /data/file.txt.

Used cat file.txt to display the contents of the file.



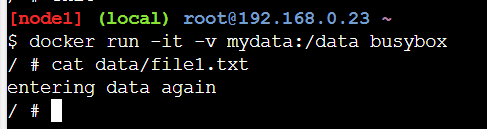
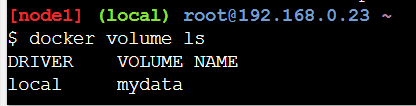
docker run -it -v mydata:/data busybox

This mounts a **named volume** mydata to /data inside the container. echo "entering data again" > data/file1.txt

cd data/ ls –lrt

file1.txt was created with contents: entering data again

ls -lrt confirms the file was created with proper permissions and timestamp



now you're exploring the results of running different containers using

## docker volume ls:

Lists existing Docker volumes.

docker run -it -v mydata:/data busybox

**docker run**: Runs a new container.

**-it**: Interactive terminal mode.

**-v mydata:/data**: Mounts the Docker volume named mydata to the /data directory inside the container.

**busybox**: Lightweight Linux container image used here. # cat data/file1.txt

Displays the content of /data/file1.txt

The file file1.txt exists inside the mydata volume and has the content "entering data again". This confirms that:

* The Docker volume mydata is being persistently used across containers.
* Data written to mydata is accessible when mounted again.