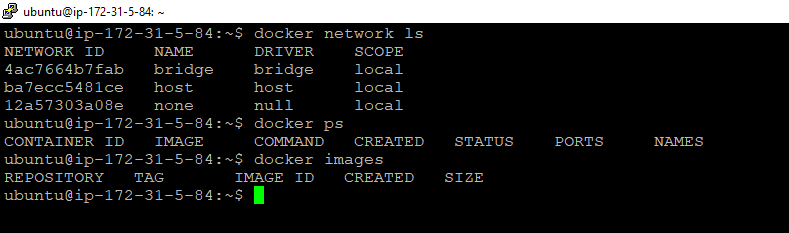
**Default Bridge Network:**

* When Docker is installed, it creates a default bridge network named bridge. Containers connected to this network can communicate with each other using internal IP addresses.
* Containers on the default bridge network can access the external network, but incoming connections from the external network are usually not allowed.

Let’s do practical project on this:-

First check the networks after installing docker on you machine.

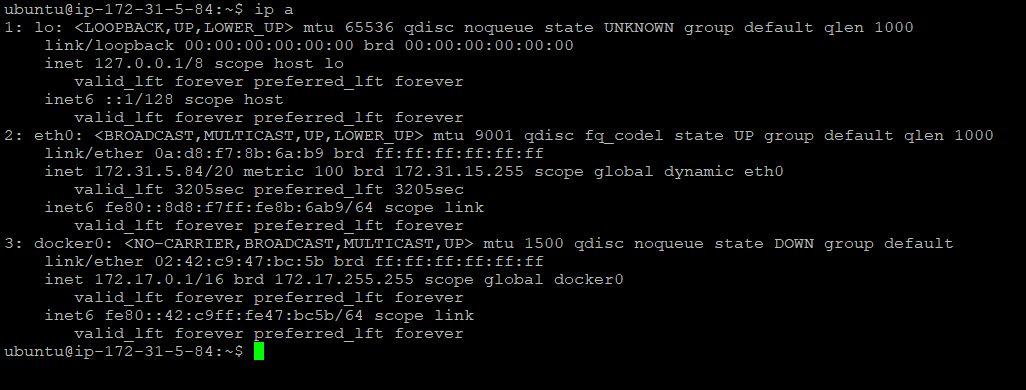
Command: **$ docker network ls**



You can see there are 3 different networks that have been created automatically post docker installation. No container is running, this is just a fresh new vm after docker installation.

The very first network you see, it is default bridge network. The driver is bridge. If we create a docker container now it will use the default bridge network. Until and unless we are specifying any network bridge while creating a container it will use the default bridge network.

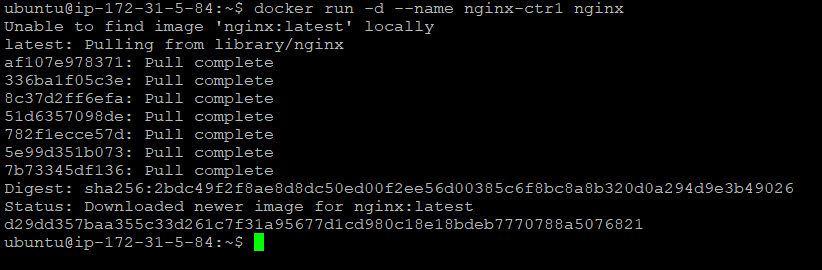
Excited to deep dive into it? Let’s move forward…..



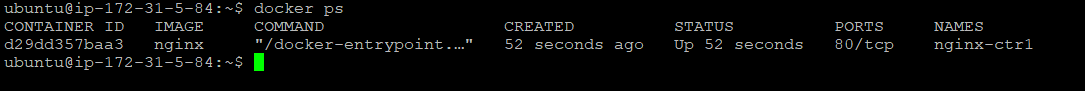
Do you see docker0, this is your docker Default bridge network. You can see the IP 172.17.0.1/16 and the subnet range is 172.17.0.0/16. Now if you create a container using this default bridge network, the container IP will be 172.17.0.2/16

Let’s create a container and see which network it is going to use and the IP. To create the containers I will use simple NGINX image which is available in DockerHub.

Command: **$ docker run -d --name nginx-ctr1 nginx**



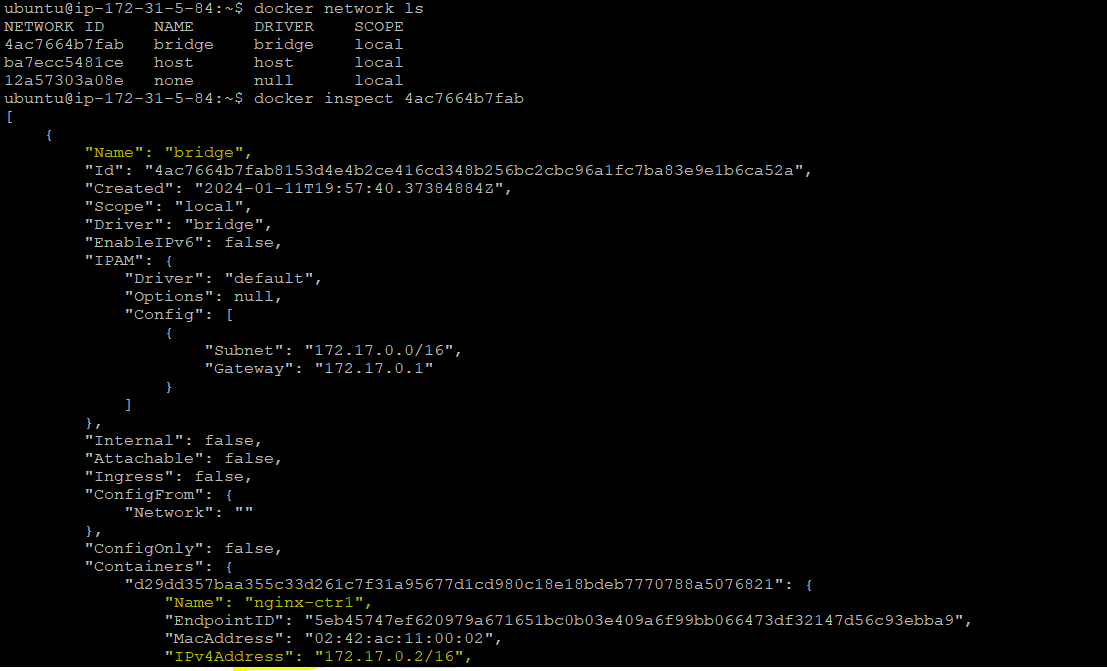
We have created a container, nginx-ctr1 which is running with nginx image.



Now check the network bridge to see the container IP and network details.

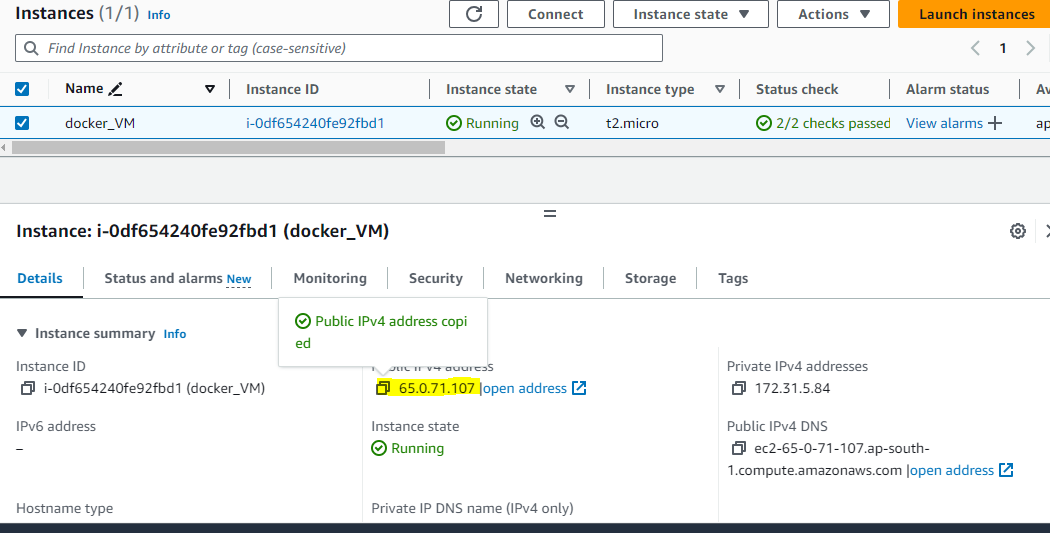
Commands: **$ docker network ls (copy the network ID)**

**$ docker inspect 4ac7664b7fab (4ac7664b7fab ->my network ID)**

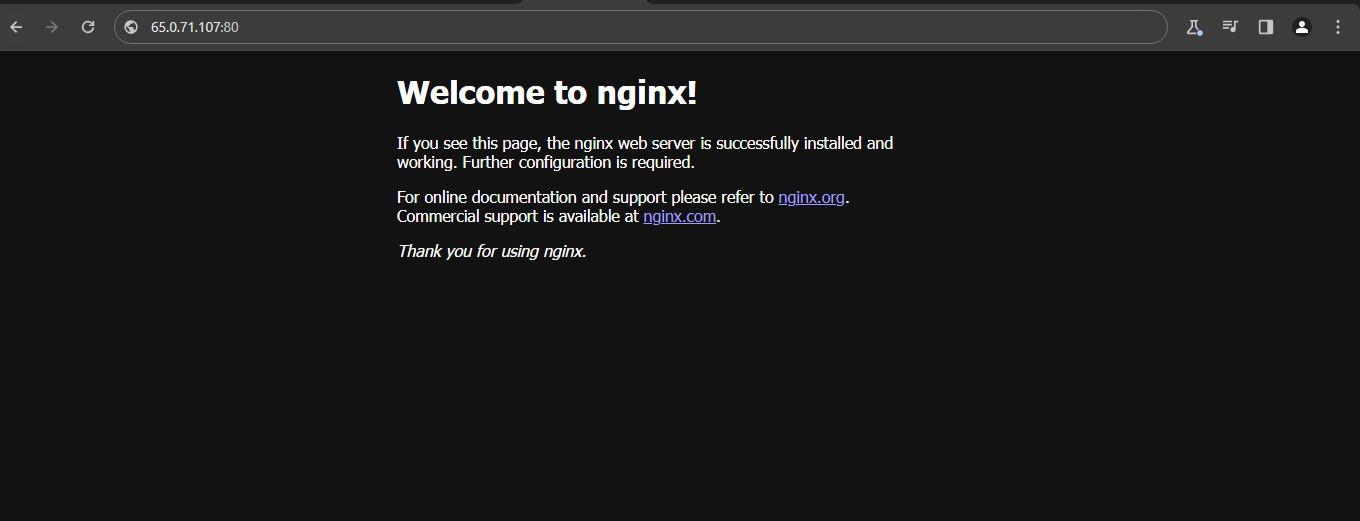


In the above screen shot you can see the highlighted ip, container and bridge details. We have discussed that the IP and subnet of Default bridge network. This is now proving that statement.

The nginx is running on port 80. We should check if our app is running. Copy the public IP of your EC2 instance and browse it in the browser in port 80. Make sure you have allowed port 80 allowed in your Security Group’s inbound rule.



Our nginx container is running successfully



You are done with one hands-on project on Docker -> Default bridge network.

The default bridge network in Docker, often named "bridge," comes with its own set of advantages and disadvantages.

**Advantages:**

* Ease of Use:

The default bridge network is created automatically when Docker is installed, making it convenient for users who just want to start using Docker without configuring custom networks.

* Out-of-the-Box Communication:

Containers attached to the default bridge network can communicate with each other using internal IP addresses without any additional configuration.

* Internet Access:

Containers on the default bridge network typically have access to the external network, allowing them to download packages, updates, or interact with external services.

* Name Resolution:

Docker automatically provides DNS resolution for container names within the default bridge network, making it easy to reference containers by their names.

**Disadvantages:**

* Isolation:

Containers on the default bridge network are not fully isolated from the host or other containers on the same network. This might not be desirable in certain scenarios where stricter isolation is required.

* Limited Routing Control:

The default bridge network has limited control over routing and network policies. For more advanced network configurations, such as defining specific routes or using custom network plugins, users may need to create their own user-defined bridge networks.

* No Multicast Support:

The default bridge network does not support multicast communication between containers. This can be a limitation for certain applications that rely on multicast for communication.

* Port Mapping Challenges:

When exposing ports from a container to the host using the default bridge network, there might be port conflicts if multiple containers attempt to use the same port on the host. Users may need to manage port mappings carefully.

* Limited Cross-Host Communication:

The default bridge network is designed for communication within a single host. If you have a multi-host setup or a Docker Swarm cluster, you may need to explore other networking options, such as overlay networks.

*In summary, while the default bridge network in Docker provides a quick and easy way to get started with container communication, it may not be suitable for all scenarios, especially those requiring more advanced networking features, strict isolation, or cross-host communication. Users often consider creating their own user-defined bridge networks or exploring other Docker networking options based on their specific requirements.*