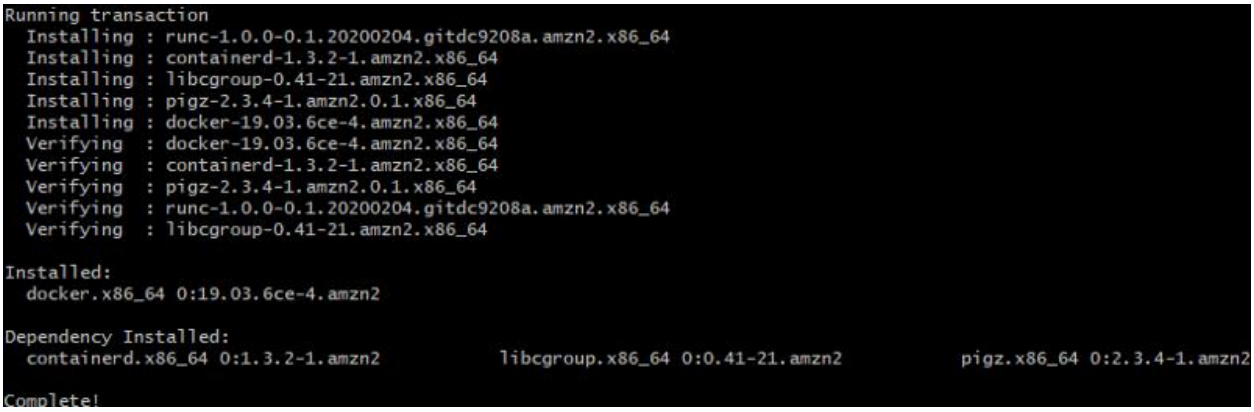
Docker Setup on AWS EC2 Instance

we will install Docker, setup the dockerhub account and run the test images in a container. Refer my docker article for the basic understanding

Install Docker on AWS EC2 instance

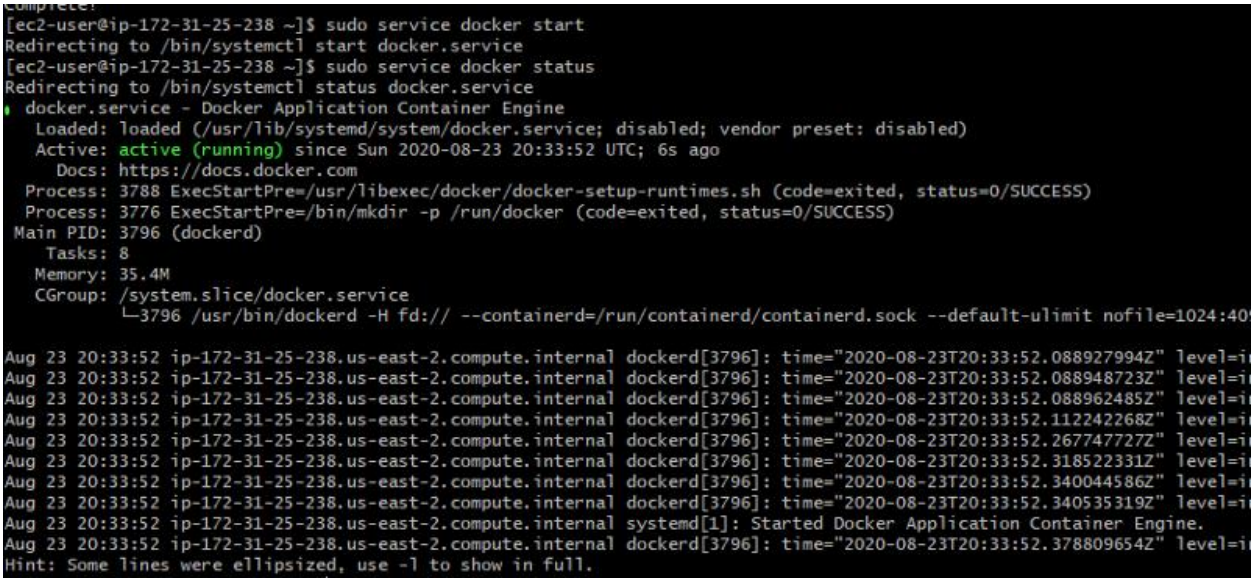
We will be using using `yum install` for the installation. If you have other OS, then please follow official documentation for the installation

```
sudo yum update -y

sudo yum install -y docker
```

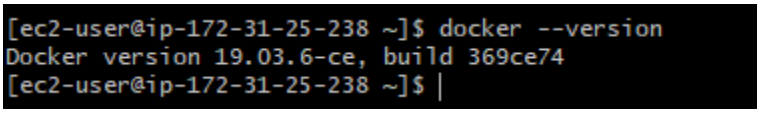


Start Docker

```
sudo service docker start

sudo service docker status
```



Check the docker version




```
docker images
```

You will see this error "**permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock**".

To fix this error,you need to add your user to the docker group as shown below:

```
sudo usermod -a -G docker ec2-user
```

Now disconnect from the server and connect again

Run the same command and you should be able to see all the images which are present on your system

```
docker images
```

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-25-238 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
[ec2-user@ip-172-31-25-238 ~]$
```

Docker has installed successfully.

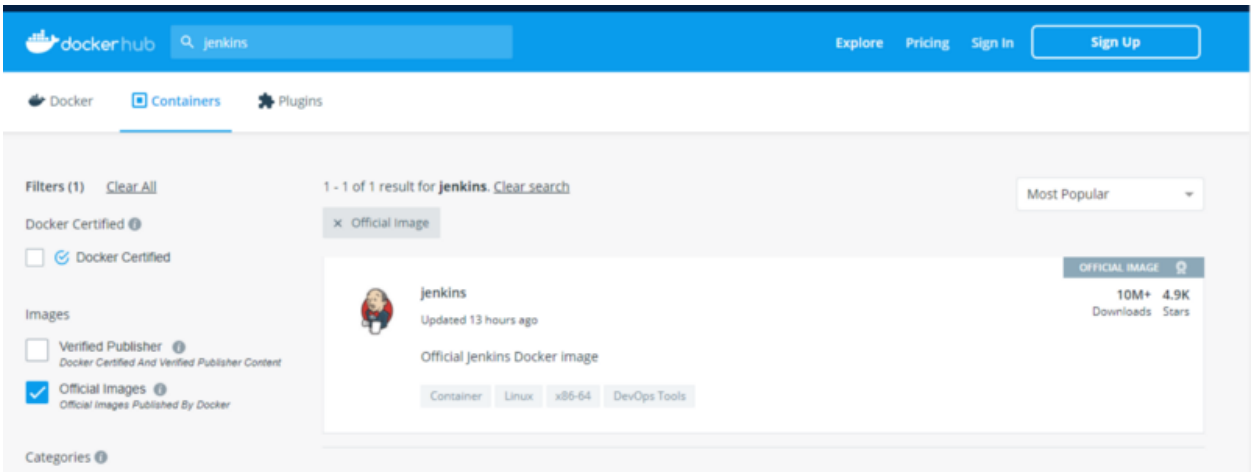
Docker Images

A docker image is a template with specific instructions for running your application in a container. It is an immutable file which contains instructions to get the source code from repo, get all the dependencies,tools and other files which are required to build your application For ex- You need a run a java container then you can directly use the images which are provided by vendor.

DockerHub

DockerHub is a place where you can find all the images for any tool which is provided by the vendor directly.

Let's say you want to run a jenkins inside a container then you can go to Dockerhub and find the official image which is directly provided by the Jenkins and use that.



Signup for dockerhub ?

- When you are building an application then you need a place to store your images so that it can use be used across your teams and can integrated with your CI/CD tools.
- Dockerhub is a repository which is used to store lot of images(with tags)
- Sign up and start using dockerhub to store your own images

Test Docker installation

We will use hello-world image which is already present in the dockerhub to test our installation

```
docker run hello-world
```

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-25-238 ~]$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest: sha256:7f0a9f93b4aa3022c3a4c147a449bf11e0941a1fd0bf4a8e6c9408b2600777c5
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

[ec2-user@ip-172-31-25-238 ~]$
```

If you see the message "**Hello from Docker**" that means docker is installed successfully.

Now if you will again run `docker images` then you should be able to see hello-world as shown below

```
[ec2-user@ip-172-31-25-238 ~]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              bf756fb1ae65        7 months ago        13.3kB
[ec2-user@ip-172-31-25-238 ~]$
```

We have seen how to use the existing images which are present in Dockerhub. Now we will explore how we can create our own image and push it in Dockerhub

Build Docker Image

To build a docker image,you need a `Dockerfile`with set of instructions as per your application. Here, we are using a sample Dockerfile for running a nginx container

- Clone the docker sample git repo

```
FROM nginx
COPY html /usr/share/nginx/html
CMD ["./wrapper.sh"]
```

- Build an image using `docker build` .
- `-t` is used to provide the name of your image and you can provide the tag in the 'name:tag' format

```
docker build -t nginxtest .

docker build -t nginxtest:v1 .

docker images
```

```
[ec2-user@ip-172-31-25-238 DockerSampleApp]$ docker build -t nginxtest:v1 .
Sending build context to Docker daemon  101.4kB
Step 1/4 : FROM nginx
 ---> 4bb46517cac3
Step 2/4 : COPY wrapper.sh /
 ---> Using cache
 ---> c4647cad88a4
Step 3/4 : COPY html /usr/share/nginx/html
 ---> Using cache
 ---> 1ba8a07422e5
Step 4/4 : CMD ["./wrapper.sh"]
 ---> Using cache
 ---> 3b4955441f5e
Successfully built 3b4955441f5e
Successfully tagged nginxtest:v1
[ec2-user@ip-172-31-25-238 DockerSampleApp]$
```

```
[ec2-user@ip-172-31-25-238 DockerSampleApp]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginxtest           latest              3b4955441f5e        10 minutes ago      133MB
nginxtest           v1                  3b4955441f5e        10 minutes ago      133MB
```

- You can see two images are created one with the version as `latest`, and another with the tag as `v1`

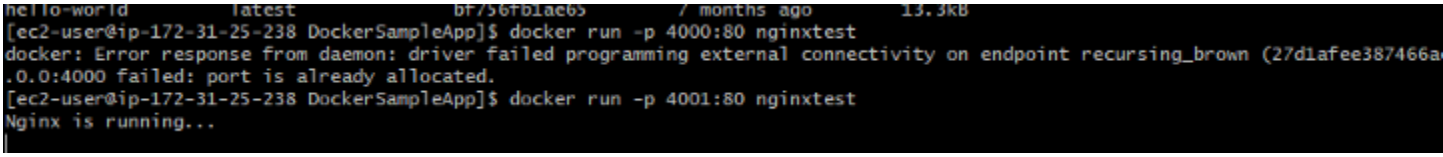
Run the container

Now we will run the container using port forwarding.

Port-forwarding is used to expose the container's port 80 (standard http port) on the host's port 4001

```
docker run -p 4001:80 nginxtest
```

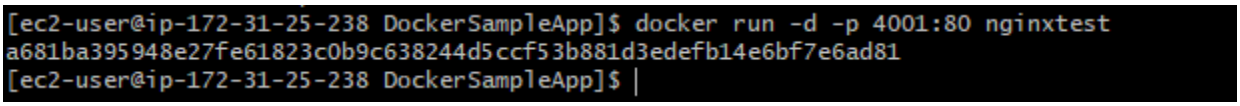
- This will run the container in attached mode that means container will exited as soon your exit from your window.
- If you use the port which is already being used by another container then you will see the "**bind error: port is already allocated**" as shown below
- Make sure that `4001` port is not being used by any other container



Run the container in detached mode

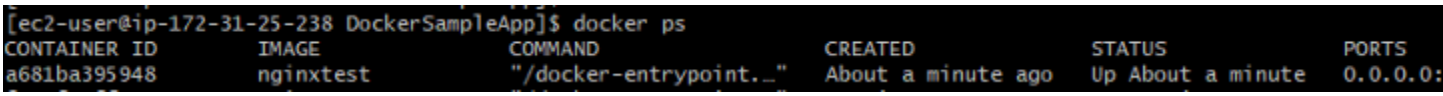
This will run the container in the background. Even if you disconnect from the machine container will be running without any issues.

```
docker run -d -p 4001:80 nginxtest
```



Now you can see that container is up and running using below command:

```
docker ps
```

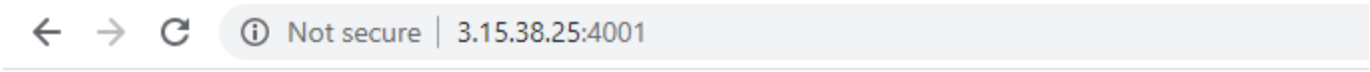


Finally, you can browse the url using curl directly on the server and you should be see the response in html format as shown below:

```
curl http://localhost:4001
```

```
[ec2-user@ip-172-31-25-238 DockerSampleApp]$ curl http://localhost:4001
<!DOCTYPE html>
<html lang="en">
<head>

  <!-- Basic Page Needs
  ------------------------------------------------- -->
  <meta charset="utf-8">
  <title>Docker :)</title>
  <meta name="description" content="">
  <meta name="author" content="">

  <!-- Mobile Specific Metas
  ------------------------------------------------- -->
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- FONT
  ------------------------------------------------- -->
  <link href="//fonts.googleapis.com/css?family=Raleway:400,300,600" rel="stylesheet" type="text/css">

  <!-- CSS
  ------------------------------------------------- -->
  <link rel="stylesheet" href="css/normalize.css">
  <link rel="stylesheet" href="css/skeleton.css">

  <!-- Favicon
  ------------------------------------------------- -->
  <link rel="icon" type="image/png" href="images/favicon.png">

</head>
<body>

  <!-- Primary Page Layout
  ------------------------------------------------- -->
```

You can also browse the url on the webpage, make sure that AWS EC2 instance has an TCP inbound rule open for this port



Hello Docker!

This is being served from a **docker** container running Nginx.