**Components of Docker**

1. Dockerfile:

* A text file that contains a set of instructions for building a Docker image.
* Dockerfile specifies the base image, adds dependencies, copies files, and runs commands necessary to create the image.
* Example instructions include `FROM`, `RUN`, `COPY`, `EXPOSE`, `CMD`, etc.
* Dockerfiles allow for reproducible builds and version-controlled configurations.

2. Docker Image:

* A lightweight, standalone, and executable software package that contains everything needed to run a particular application: code, runtime, libraries, dependencies, and configuration files.
* Docker images are built from Dockerfiles and are stored in a layered format.
* Images are immutable and can be shared, distributed, and reused across different environments.
* Docker images can be stored locally or pushed to a Docker registry for sharing with others.

3. Docker Container:

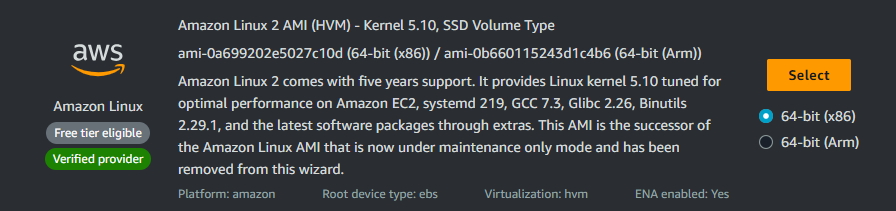
* An instance of a Docker image that runs as a process on the host machine's operating system.
* Containers are lightweight, portable, and isolated from each other and the host system.
* Containers encapsulate the application and its dependencies, ensuring consistency across different environments.
* Docker containers can be started, stopped, paused, and deleted using Docker commands.

4. Docker Registry (Docker Hub):

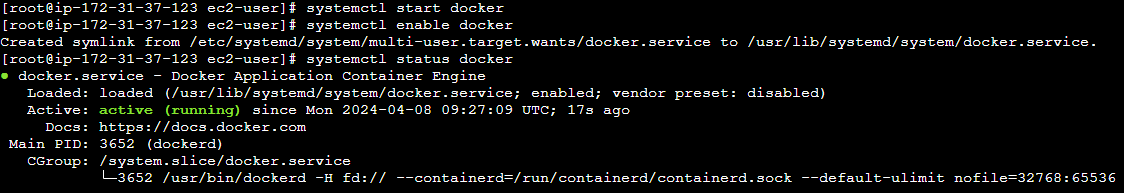
* A repository for storing and sharing Docker images.
* Docker Hub is the default public registry provided by Docker, where users can find official images for popular software packages.
* Users can also push their custom-built Docker images to Docker Hub for sharing with others.
* Docker registries allow users to organize, manage, and distribute Docker images across different teams and environments.
* Users can set up private Docker registries for storing proprietary or sensitive images within their organization.

Steps for creating a docker image and docker container.

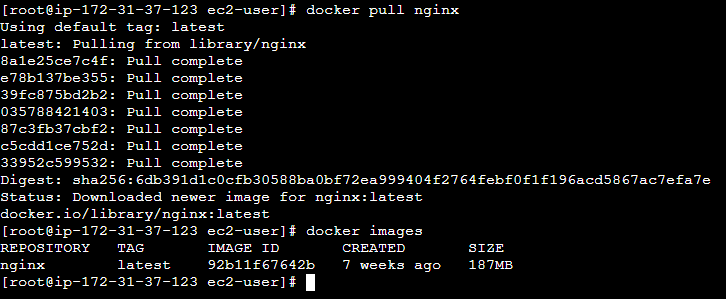
* First we need an Instance with the Amazon Linux 2 AMI.



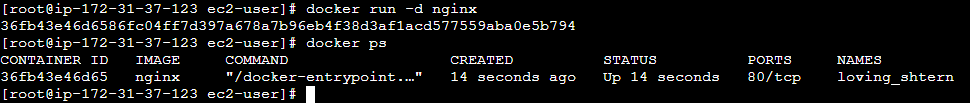
* After launching the Instance take the ssh of the instance.
* Now login as a root user.
* We need to install docker.
* Hit command “yum install docker -y”.
* Now we need to start and enable the service.
* Hit command “systemctl start docker”.
* Hit command “systemctl enable docker”.



* Now we need a image to create a container.
* We will pull an image.
* Hit command “docker pull nginx”
* Hit command “docker images” to view all the images.



* Now we have to run the image to create a container.
* “docker run -d image\_name” to run the image in background.
* Hit command “docker ps” to view the currently running container.



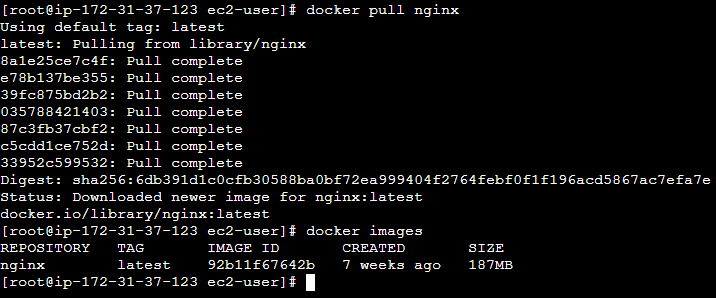
* Now we need to get access of the container.
* Hit command “docker run -it --name container\_name image\_name /bin/bash” to enter the container.

Docker Commands

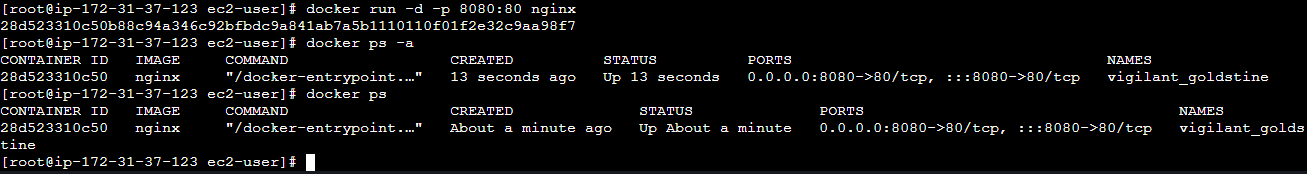
* # docker images : to view all the images available.
* # docker ps : to view all the running containers.
* # docker ps -a : to view all the containers including running and stopped.
* # docker pull image\_name : to pull the image.
* # docker rm container\_name : to remove the container.
* # docker start container\_name : to start the container.
* # docker rmi image\_name : to remove the image.
* # docker -v : to view the version of docker.
* # docker search name : to search the image.

Hosting a website in the Docker Container.

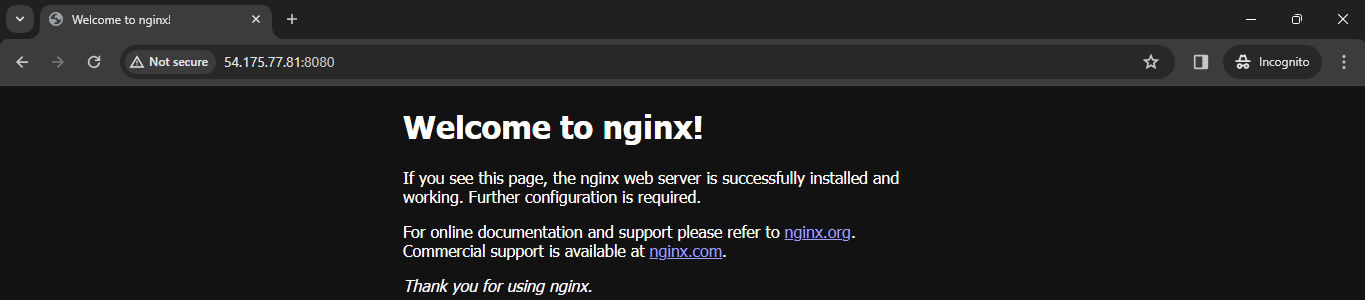
* First we need an image to create a container.
* Hit command “docker pull nginx”.



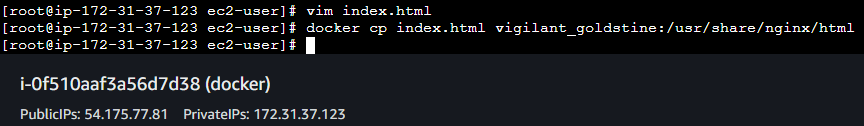
* Now we have to host a website inside the container.
* The container is isolated and the IP is different than our instance.
* So we need to expose the port number of the container.
* Hit command “docker run -d -p 8080:80 nginx”.

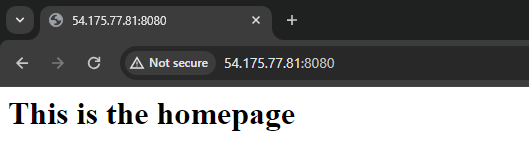


* Not the container is running.
* Hit the IP of your instance with the port number to check.

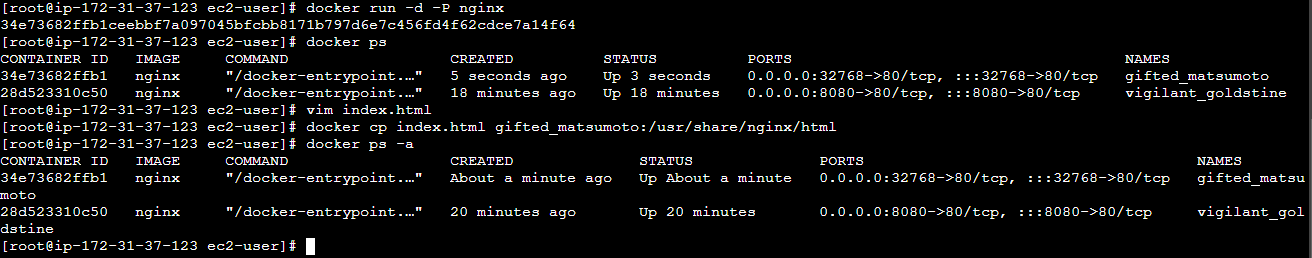


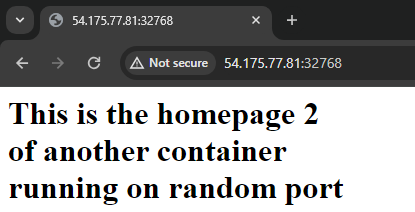
* Now we want our page instead of nginx.
* Create a file named index.html.
* Now we need to copy the index.html file in the container.
* Hit command “docker cp index.html container\_name:/usr/share/nginx/html/” to copy the index.html file to the container.
* now try to hit the IP of your instance with the port number 8080.





* We are forwarding all the requests coming to port 8080 on the host machine to port 80 inside the Docker container.
* # docker run -d -p 8080:80 nginx : This command explicitly specifies a port mapping. It maps port 80 inside the container to port 8080 on the host machine. In other words, it binds port 8080 on the host to port 80 in the container, allowing you to access the web server running inside the container via http://localhost:8080.
* # docker run -d -P nginx : This command uses automatic port mapping. It tells Docker to choose a random port on the host machine and map it to the default port exposed by the container. So, if the container exposes port 80, Docker will map it to a random port on the host machine. You can use “docker ps” to see which port has been assigned.
* Now we don’t have to give the port everytime we can use random ports.
* Now we will create another container and host another webpage in that container.
* Hit command “docker run -d -P nginx”. This command will create another container.





* We can also create a image with our container.
* Hit command “docker commit container\_name image\_name”. Here you can give the name to you image.

