**Docker**

**Introduction:**

**Containerization Tool:**

Docker



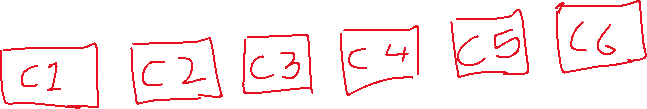
Rocket

Linux Containers (LXC)

**What docker helps you to achieve? (Use of Containers)**

We create container with the help of docker.

We can create multiple containers with the help of docker.



We run microservices inside the containers.

ex. For 10 microservices, will require 10 containers.



Containers are supposed to run Applications.

Container is a closed environment in that microservices are running.

Containers helps you to run your applications in isolation.

When we run any applications in the container, we add all the dependencies required by the application inside container.

Container=VM-OS

Containers are lightweight, Container is isolated computing capacity.

**Containerized Application**: Microservices running inside the containers called as containerized application.

**Docker Installation Commands:**

yum install docker -y

service docker status

service docker start

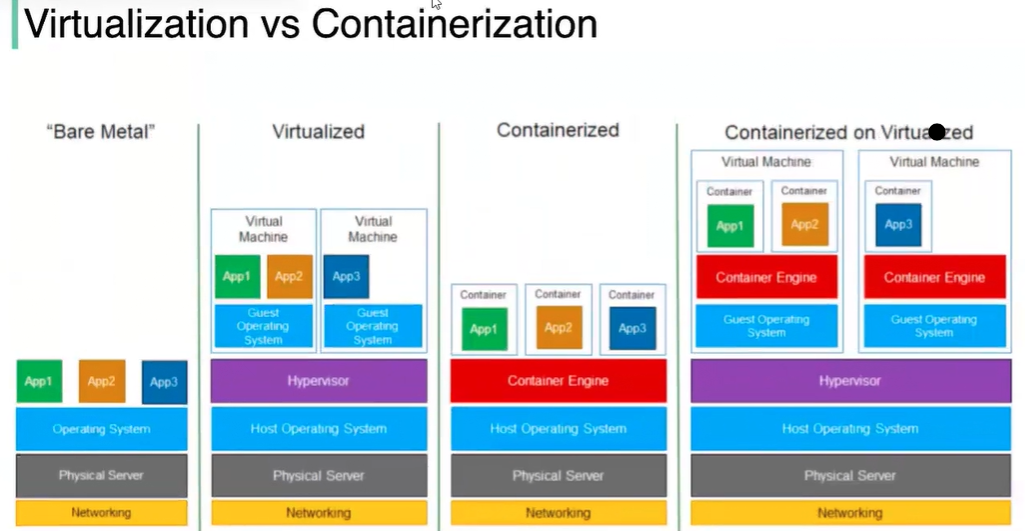
chkconfig docker on

usermod -aG docker ec2-user

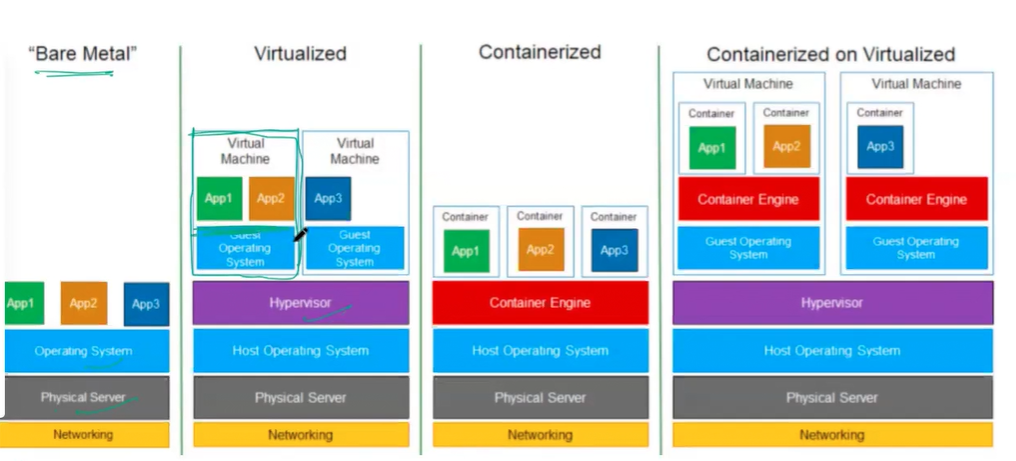
**Difference between Virtualisation and Containerization:**

Virtualization and containerization are both ways to run applications on a single server, but they differ in how they share resources and how secure they are.

|  |  |  |
| --- | --- | --- |
|  | **Virtualization** | **Containerization** |
| How it works | Creates virtual machines (VMs) with their own operating systems | Packages applications into containers that share the host operating system |
| Security | Stronger isolation between VMs | Less secure than VMs, but can be enhanced with features like Linux Security Modules |
| Resource usage | Uses dedicated hardware resources for each VM | Shares hardware resources across containers |
| Suitability | Good for running legacy applications | Good for microservices architectures, CI/CD, and PaaS  We don’t have OS, We have OS in Virtualisation  Start Time is more in VM, Start Time is Less in Containers  Resource Reservation or blocking is much better as compare to Virtualisation. |



**Architecture:**

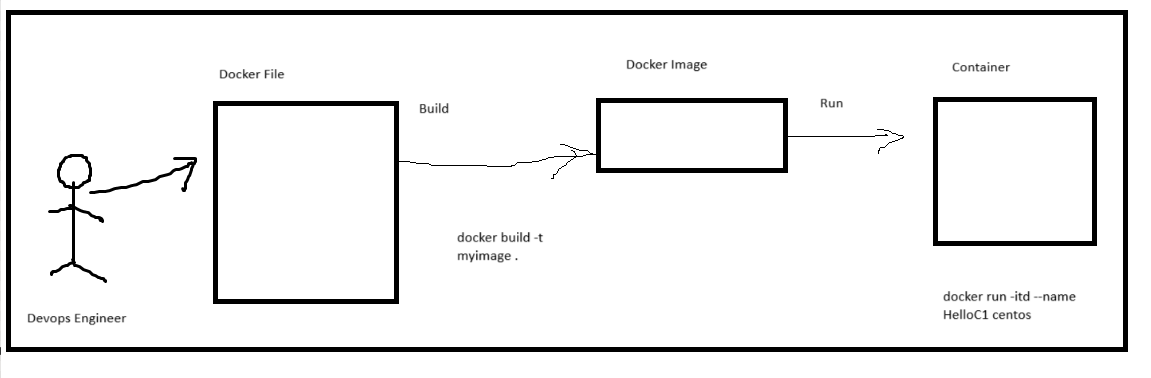


**Docker**: Devops Engineer is responsible for creating a docker file, Inside the docker file, he will write what he wants inside the container.

He will write os name application etc.

Once he writes a docker file, he will build docker file, he will get docker image. then he will run a docker image and he will create a container from docker image.

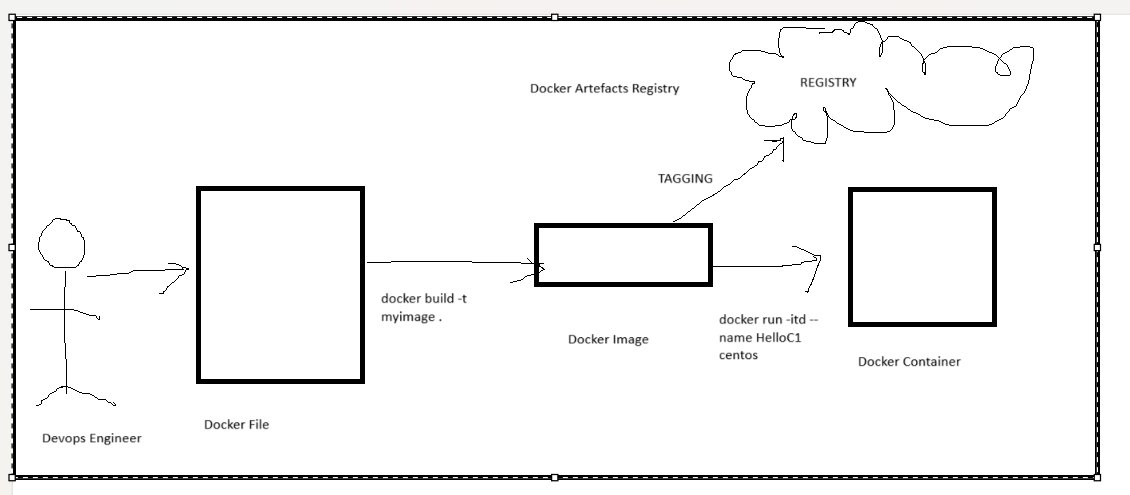
What we need inside the container has to be written inside the docker file.



We can create N no of containers using same Docker Image.

Containers are created with same image are identical to each other.

Docker File >>>>> Docker Image >>>>>>> Docker Container



**Docker File**: It’s a text file contain all the information to create Containers.

It’s a step-by-step instruction to create a Container.

A Dockerfile is a script that contains a series of instructions used to create a Docker image. These instructions define the base image, dependencies, configuration, and commands for setting up a containerized environment.

Docker file contains Keyword’s. Docker file runs in a sequential manner.

Docker file is ReadOnly.

Docker files are made of layers. No Of Layers are depend on the No of lines written in docker file.

**Simple Docker File:**

# Set the base image

FROM node:16

# Set the working directory in the container

WORKDIR /app

# Copy package.json and package-lock.json

COPY package\*.json ./

# Install dependencies

RUN npm install

# Copy the rest of the application files

COPY . .

# Expose port (optional, depends on your app)

EXPOSE 3000

# Command to run the application

CMD ["npm", "start"]

FROM: Specifies the base image to use (in this case, node:16 is used for a Node.js application).

WORKDIR: Sets the working directory where subsequent commands will be run.

COPY: Copies files from the host machine into the container.

RUN: Executes a command inside the container (like installing dependencies).

EXPOSE: Informs Docker about the port the container will listen on.

CMD: Defines the default command to run when the container starts (usually the app itself).

**Linux Kernel Features used by Docker:**

**Namespace (Isolation):**

**Control Groups (Resource Management):**

**Union Filesystem (): Docker used union filesystem**

**Groups and Capabilities (Security):**

**Iptables(Traffic Routing):**

**Chroot(Filesystem Isolation):**

**Docker Ecosystem Component:**

**Docker File:** It’s a file of instructions used to create the docker image.

**Docker Image:** Its used to create containers

**Docker Registry:** Docker base images are stored in docker registry also we can distribute the images.

Manage and unmanaged docker registry.

**Docker Client:**

Docker Engine is a client-server application.

The client used the Docker REST API to control or interact with the Docker Daemon though CLI commands.

The Daemon creates and manages Docker Objects such as images, Containers, networks and data volumes.

**Docker Engine:** It Build, ship and runs the applications.

**Docker Demon:** It checks the client request and communicates the docker component in order to perform the service.

**Docker Registry:**

**Managed Docker Repository:** ECR. Azure Repo, Docker Hub, Google Container Registry

90% company used managed registry

UBS is using ECR.

**Unmanaged Docker Repository:** Nexus Sonatype, JFrog, Harbor

Banks used Unmanaged docker registry.

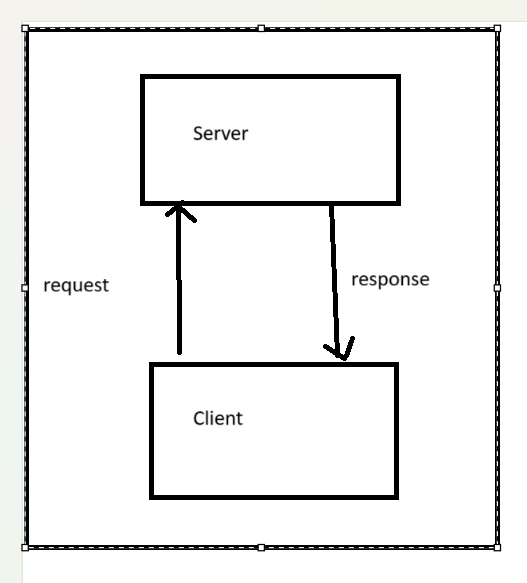
Ex. City is using Jfrog

**2 Tier Architecture and 3 Tier Architecture:**

**2 Tier Architecture:**

It has only 2 components. Client and Server.

Client sends a request to server. Server receive the request and send response to client.

****

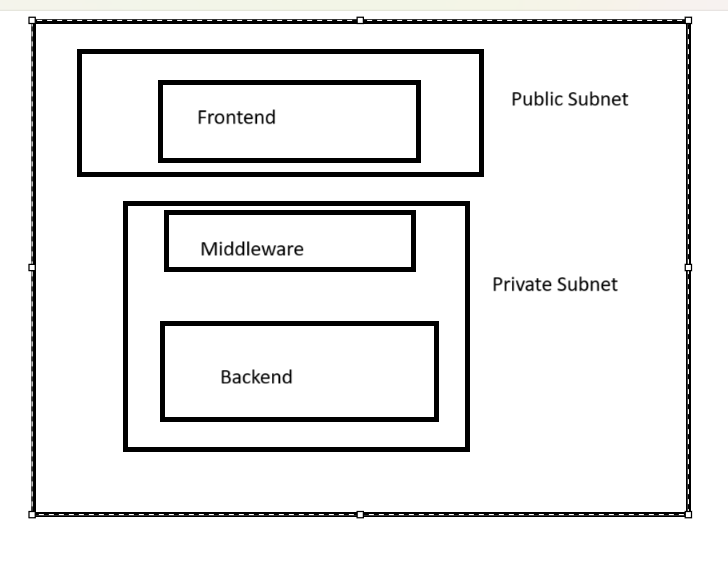
**3 Tier Architecture:**

It has 3 components. Frontend, Middleware and Backend.

Frontend: UI

Middleware: Business logic

Backend: Database

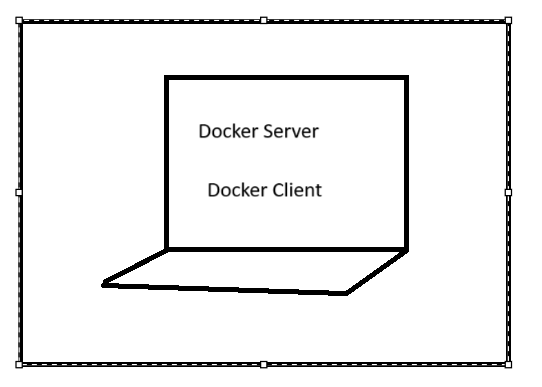
****

**Docker Architecture:**

Docker Support 2 tier Architecture. There are 2 conditions.

**Condition 1:**

You can have Docker Client and Docker Server both on one machine.

****

**Condition2:**

You can have Docker Client and Docker Server on separate machine.

Docker client will install on separate machine.

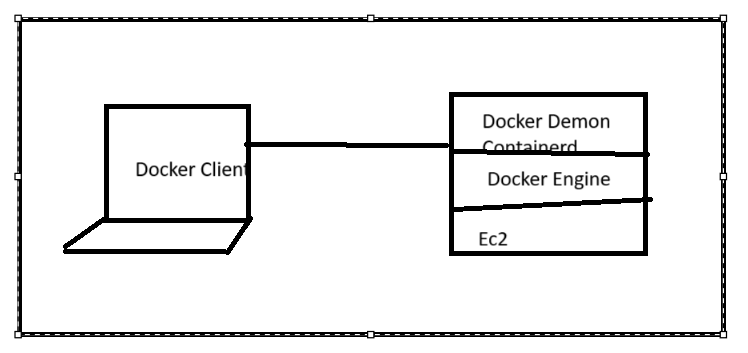
Docker Server will install on separate machine. We will have Docker engine in Ec2 instance and on top of that we will have Docker demon (Containerd)

If you written a docker commands, if you run any commands, each and every command will be intercepted by docker demon running on docker server.

Its ideally responsible for intercepting the commands sent by docker client and performing the task.

Ex. When you run docker pull centos, command will be intercepted by docker demon, it will check if the image store local or not, if not, it will be going to download centos image from docker hub.

By default, all images will be downloaded from docker hub.

****

**Docker Commands:**

**docker info:** To check Docker information. Containers details

**docker --version:** To check docker version

**docker pull centos :** To pull the images from docker hub (centos is the image name which we want to pull)

**docker pull jenkins/Jenkins:** To pull/download the Jenkins image

**docker build -t myimage . :** To build image from docker file

**docker run -itd --name HelloC1 centos:** Command to create container

**docker exec -it 75cad9207061 /bin/bash:** To Login into container (75cad9207061 : Container ID)

**docker images:** To list the available images

**docker ps :** used to list the containers healty containers

**docket stats container id:** To check cpu memory

**docker restart container id:** To restart container

**docker pause container id:** To pause container

**docker unpasue container id:** To unpause the container

**docker rename container id new name :** To rename container

**docket ps-a:** list all the container (healthy and Unhealthy)

**docket stop container id:** To stop container

**docker start container:** To start the container

**docker rm container id:** To delete the container

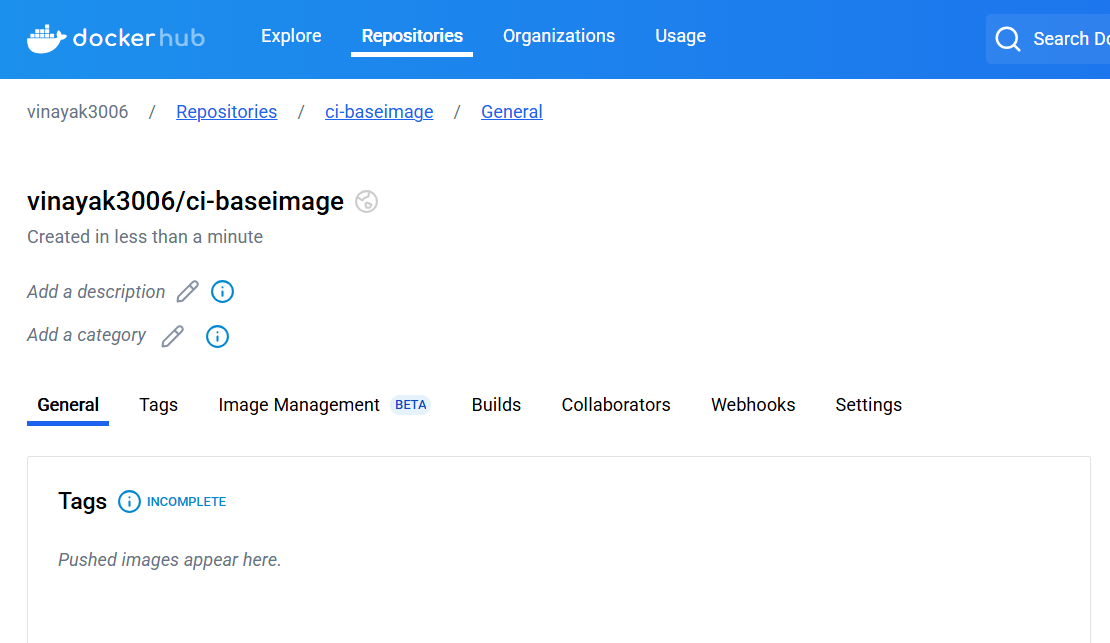
**docker rmi image name:** To delete image

**hostname:** To see the name of container

**exit:** To logout from container

**Upload Docker Image to Docker Hub (Manually)**

1. Create Account on Docker Hub.
2. Create Repository.

****

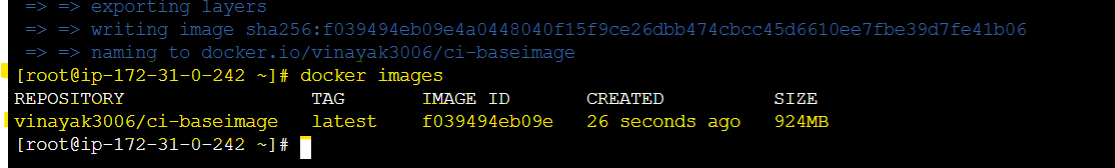
1. Write a docker file on ec2 instance and save it.
2. Build the image from Docker file using command below.

Docker build -t vinayak3006/ci-baseimage .

1. It will start to create image.

****

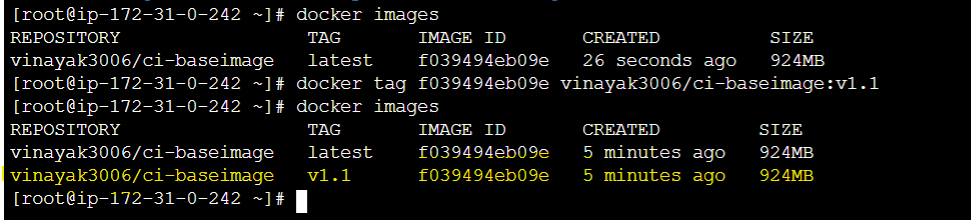
1. Once done, verify the image using **docker images** command.



1. Once the docker image is created, do the tagging like below.

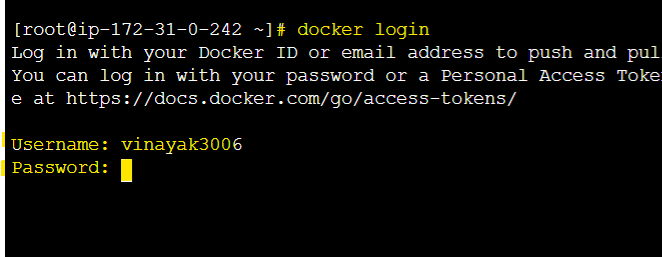
docker tag image id image name:v1.1

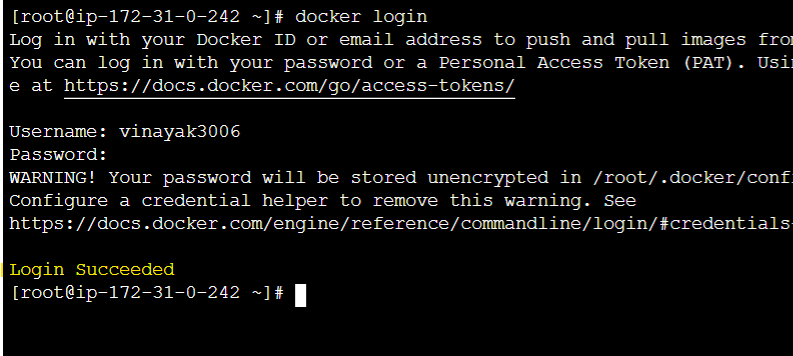
docker tag f039494eb09e vinayak3006/ci-baseimage:v1.1



1. Push the docker image to Docker Hub.
2. Login to docker hub using cli, enter your username and password. Use command below,

docker login



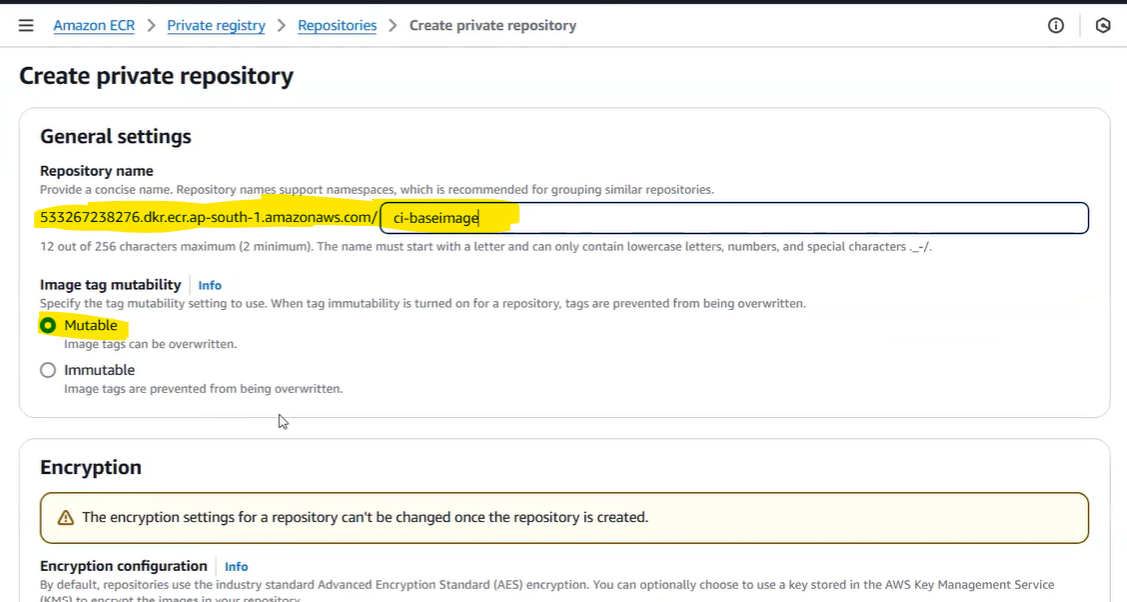


1. Push the image to docker Hub using command below.

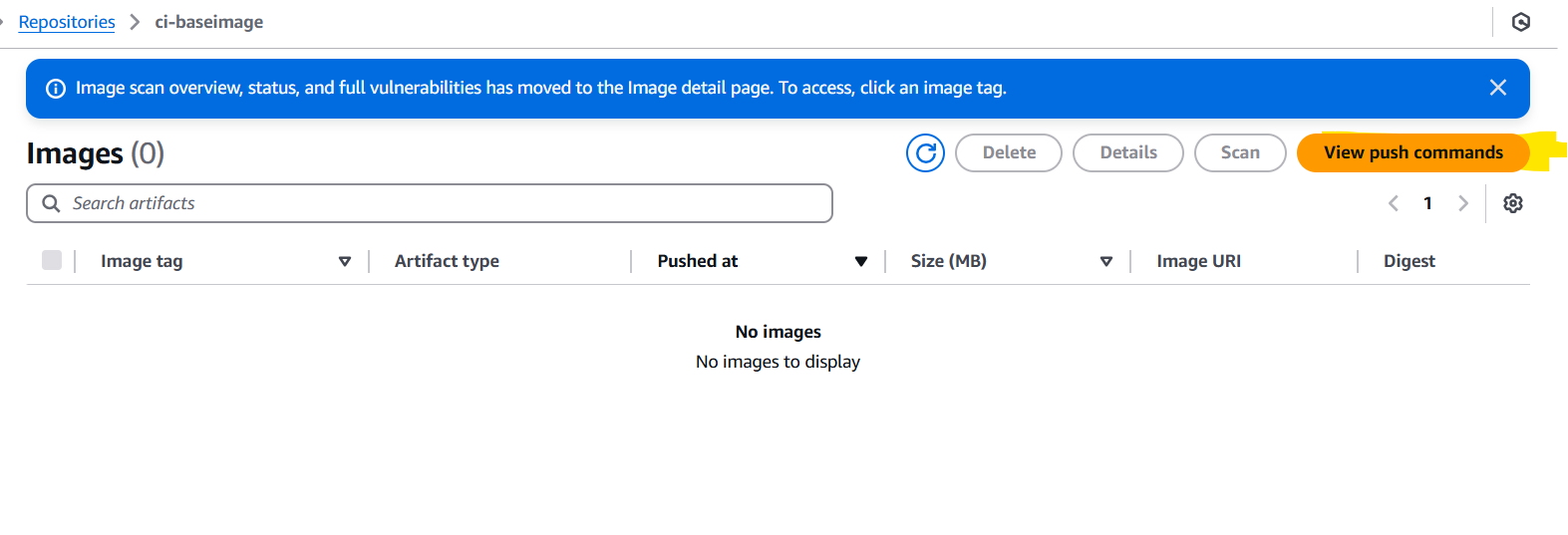
docker push vinayak3006/ci-baseimage:v1.1

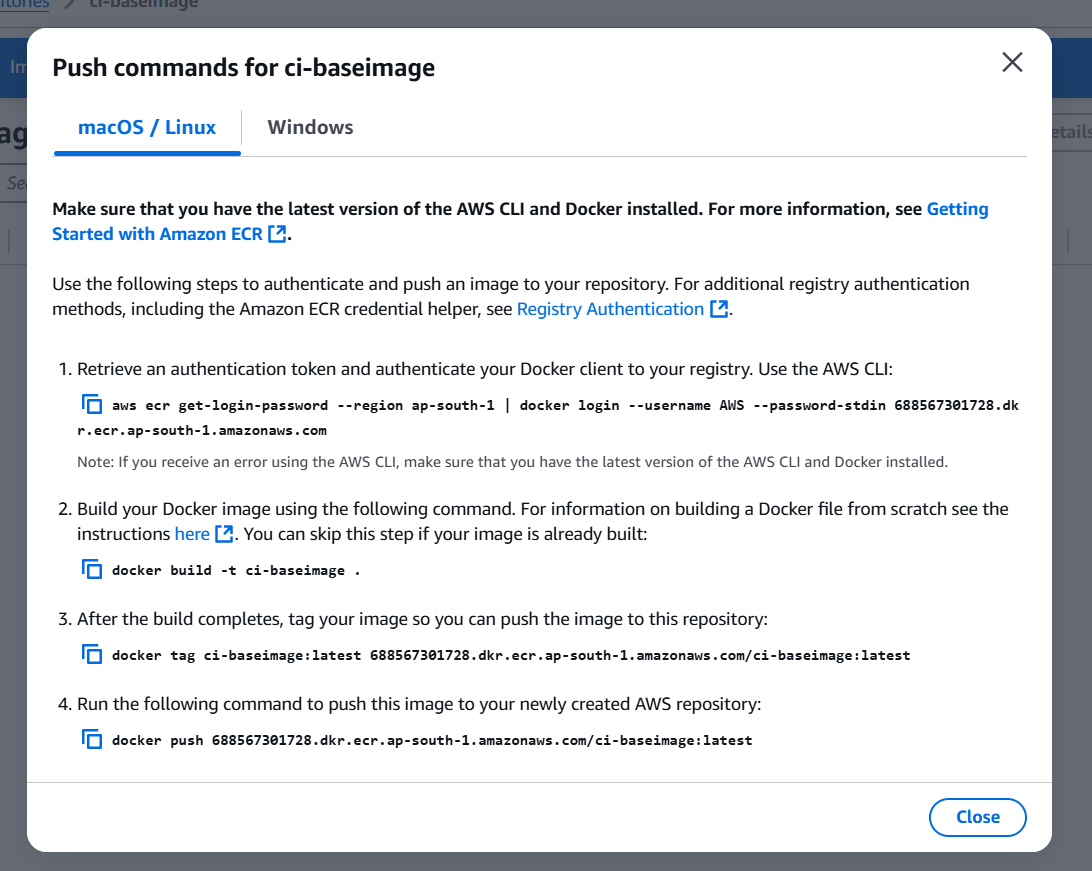
**Upload Docker Image to AWS ECR (Manually)**

1. Go to AWS ECR Services.
2. Create repository. ci-baseimage is the name of repository.
3. Click on create.

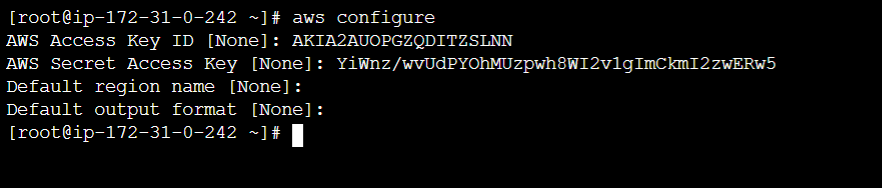


1. Go to the repository and click on view push commands.

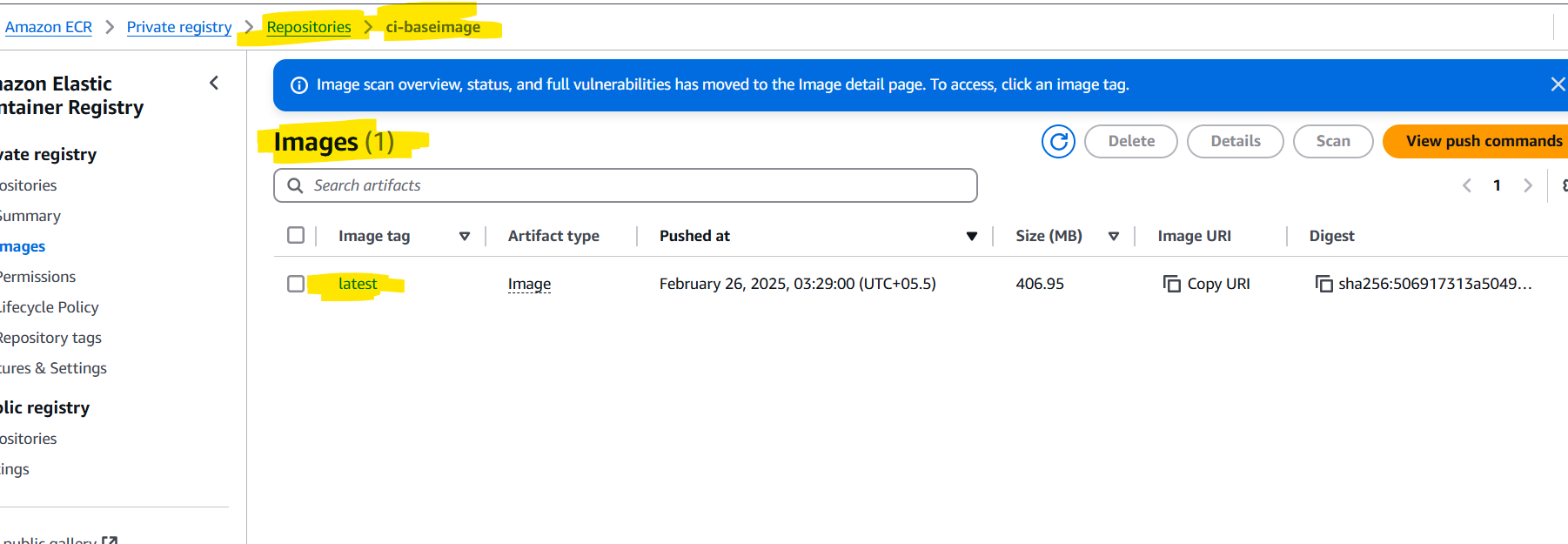




1. You need IAM user (Access and Secrete Access key) to run these commands.
2. Configure Access and Secrete Access key.



1. Run the push commands now.
2. Image will be uploaded to AWS ECR specified repository.



**Uplaod Docker Image (Automation)**

1. Add the Dockerfile in intellej idea code.
2. Add the Jenkins file in the same code.
3. Make the necessary changes in docker and Jenkins file.

**Errors while running pipeline:**

* **/var/lib/jenkins/workspace/netflix-ms@tmp/durable-4ab398a0/script.sh.copy: line 1: docker: command not found**

This error mentioned that docker is not install on Jenkins server to run docker command.

We need to install docker application on Jenkins server to resolve this error.

**Command to install docker:**

**yum install docker -y**

**service docker status**

**service docker start**

**chkconfig docker on**

**usermod -aG docker ec2-user**

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

* **http://%2Fvar%2Frun%2Fdocker.sock/\_ping": dial unix /var/run/docker.sock: connect: permission denied**

It saying that permission is not granted to mention path.

Need to grant permission below to resolve the error.

**chmod 777 /var/run/docker.sock**

* **> [2/4] RUN cd /opt && wget** [**https://download.oracle.com/java/21/latest/jdk-21\_linux-x64\_bin.tar.gz**](https://download.oracle.com/java/21/latest/jdk-21_linux-x64_bin.tar.gz) **&& tar xzf jdk-21\_linux-x64\_bin.tar.gz && rm jdk-21\_linux-x64\_bin.tar.gz && mv jdk-21\* /usr/local/java21 && echo "export JAVA\_HOME=/usr/local/java21" >> /etc/profile && echo "export PATH=$JAVA\_HOME/bin:$PATH" >> /etc/profile:**

**0.456 /bin/sh: 1: wget: not found**

**Need to check the download java syntax**

* **Could not find credentials entry with ID 'dockerhubCred'**

Need to create global credentials in Jenkins. It will allow docker hub to communicate with Jenkins.

**Add Credentials in Global credentials. Add Docker Hub username and password and the credentials id mentioned in error.**

**Check docker pipeline plugin is install or not in installed plugin, if not install it.**

**>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<**

* **Credentials 'dockerhubCred' is of type 'Username with password' where 'org.jenkinsci.plugins.plaincredentials.StringCredentials' was expected.**

Need to create credentials with string, Secrete value

Create new credentials.

Create secrete text credentials.

Give ID (dockerhubCred ) and secrete (Docker Hub Password).

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>><<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

* **ERROR: Could not find credentials matching ecr:ap-south-1:ecr-credentials**

**Need to create access key and secrete acces key credentials for ECR.**

**Also check if ECR plugin is install in installed plugins are not, if not install it.**