Docker Docs:

Problem statement: code works perfectly fine on developer’s system, but fails in dev/QA/prod env due to system misconfigurations.

Why?????

Problems with VM:

1. Os (cost, update, upgrade)
2. Hardware (cost, hardware )
3. Tool version

**What is Docker**: Docker is tool that can be installed on system (laptop/VM)

It helps us to run containerized/Dockerized environments.

Dockers are independent from OS (you can run containers anywhere, if container is working on dev system, then it will run on others environment/system )

Docker is a platform that lets you develop, ship, and run applications. You can use Docker to separate applications from the infrastructure to deliver software more quickly. It enables you to manage the infrastructure similarly to how you manage applications. You can leverage Docker’s methodologies for testing, shipping, and deploying code to reduce delays between writing and running code in production.

Docker helps you package and run applications in a container, a loosely isolated environment. This isolation ensures you can run many containers simultaneously on one host. A container is lightweight and contains everything required to run your application. There is no need to rely on components installed on the host.

Once you install docker on system, it creates a docker engine

As a user we can interact with docker using docker CLI (cmd/powershell/gitbash etc)

CLI passes instruction (docker command) to REST API (it acts as mediator between CLI and docker daemon or engine), REST API passes instruction to Docker engine/Daemon (it is responsible for actual operation-> create container/delete container/list container/expose etc)

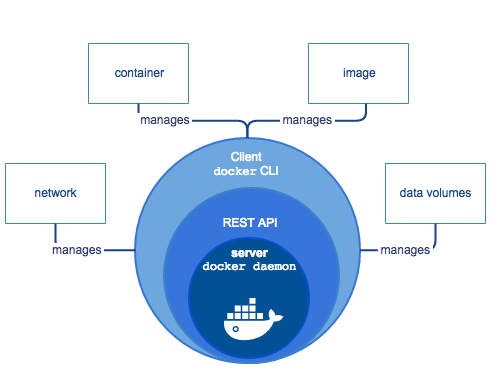
DOCKER CLI: Command line interface

DOCKER REST API: (it acts as mediator between CLI and docker daemon or engine

DOCKER DAEMON/Engine: it is responsible for actual operation

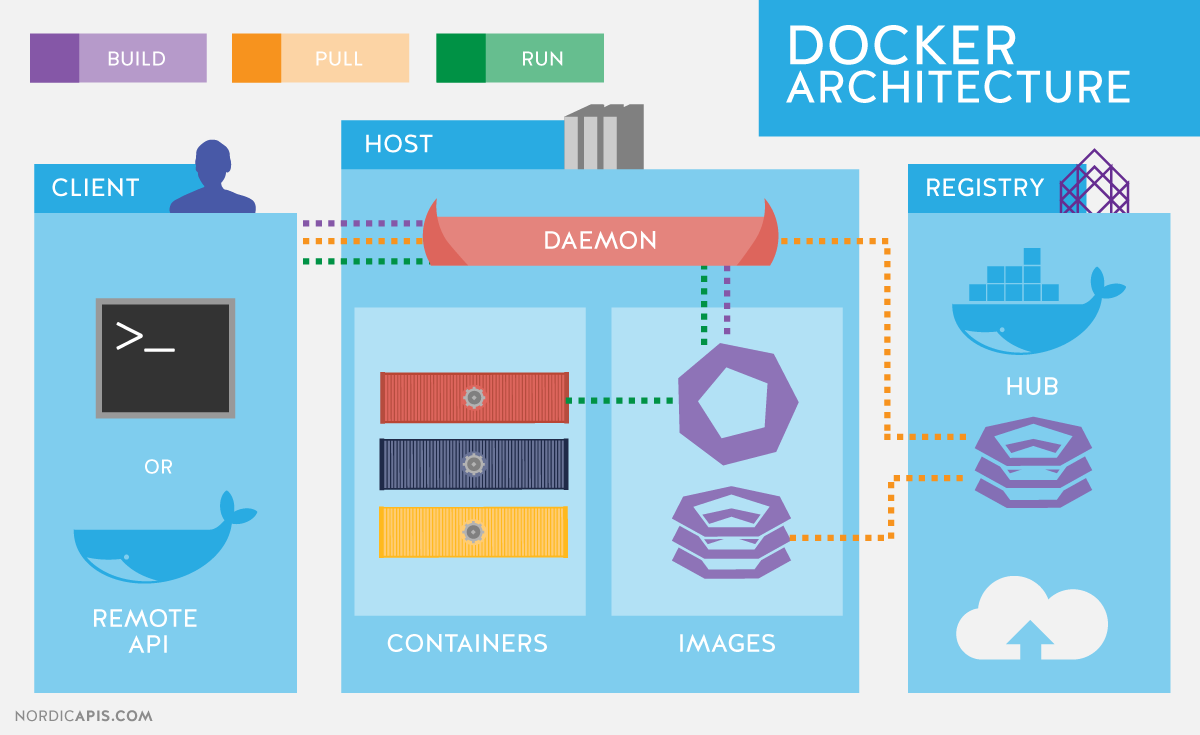
============**Docker Architecture:-**

<https://www.aquasec.com/cloud-native-academy/docker-container/docker-architecture/>



* **Docker Daemon**: A persistent background process that manages Docker images, containers, networks, and storage volumes. The Docker daemon constantly listens for Docker API requests and processes them.
* **Docker Engine REST API**: An API used by applications to interact with the Docker daemon; it can be accessed by an HTTP client.
* **Docker CLI**: A command line interface client for interacting with the Docker daemon. It greatly simplifies how you manage container instances and is one of the key reasons why developers love using Docker.

*The Docker architecture uses a client-server model and comprises of the Docker Client, Docker Host, Network and Storage components, and the* [*Docker Registry*](https://wiki.aquasec.com/display/containers/Docker+Registries+101)*/Hub. Let’s look at each of these in some detail.*

*****Docker Architecture***

====================Install docker on ec2 linux t2 micro/windows laptop

**Docker on windows**: <https://docs.docker.com/get-docker/>

**Docker on windows 64 bit home**: <https://docs.docker.com/desktop/install/windows-install/>

**Docker on aws ec2 linux:**

sudo su - (to become root user)

yum update -y

yum install docker -y

service docker start

service docker status

systemctl enable docker

docker --version

=================Docker Commands:

attach Attach local standard input, output, and error streams to a running container

build Build an image from a Dockerfile

commit Create a new image from a container's changes

cp Copy files/folders between a container and the local filesystem

create Create a new container

diff Inspect changes to files or directories on a container's filesystem

events Get real time events from the server

exec Run a command in a running container

export Export a container's filesystem as a tar archive

history Show the history of an image

images List images

import Import the contents from a tarball to create a filesystem image

info Display system-wide information

inspect Return low-level information on Docker objects

kill Kill one or more running containers

load Load an image from a tar archive or STDIN

login Log in to a Docker registry

logout Log out from a Docker registry

logs Fetch the logs of a container

pause Pause all processes within one or more containers

port List port mappings or a specific mapping for the container

ps List containers

pull Pull an image or a repository from a registry

push Push an image or a repository to a registry

rename Rename a container

restart Restart one or more containers

rm Remove one or more containers

rmi Remove one or more images

run Run a command in a new container

save Save one or more images to a tar archive (streamed to STDOUT by default)

search Search the Docker Hub for images

start Start one or more stopped containers

stats Display a live stream of container(s) resource usage statistics

stop Stop one or more running containers

tag Create a tag TARGET\_IMAGE that refers to SOURCE\_IMAGE

top Display the running processes of a container

unpause Unpause all processes within one or more containers

update Update configuration of one or more containers

version Show the Docker version information

wait Block until one or more containers stop, then print their exit codes

==================

Q. What do you understand about the boot up process/ boot time ??

<https://www.baeldung.com/linux/boot-process>

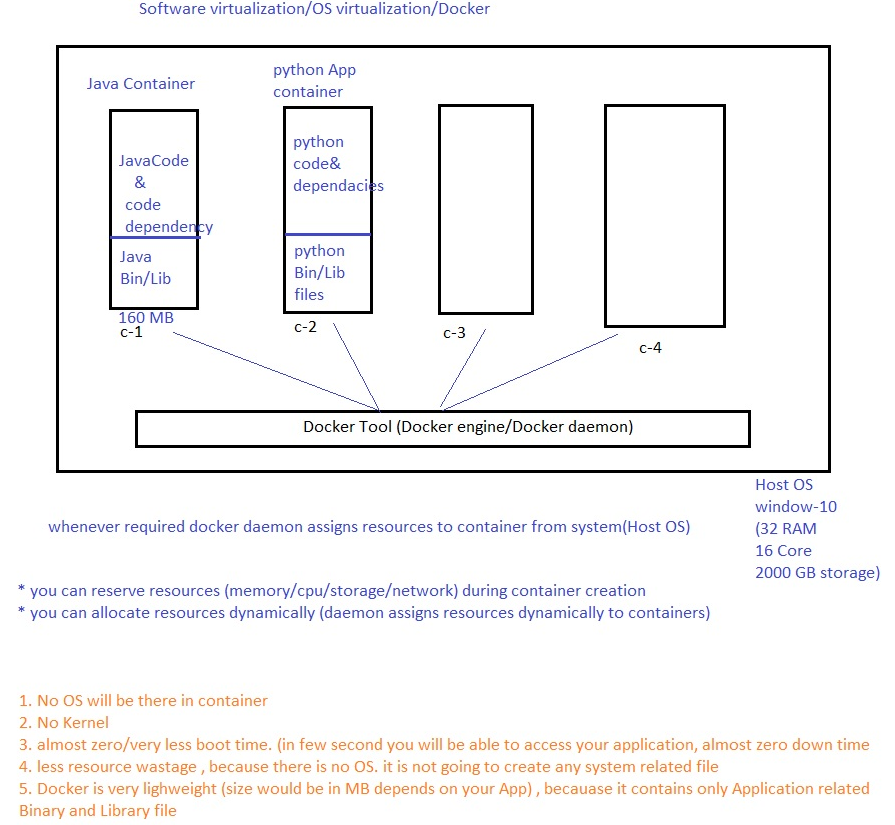
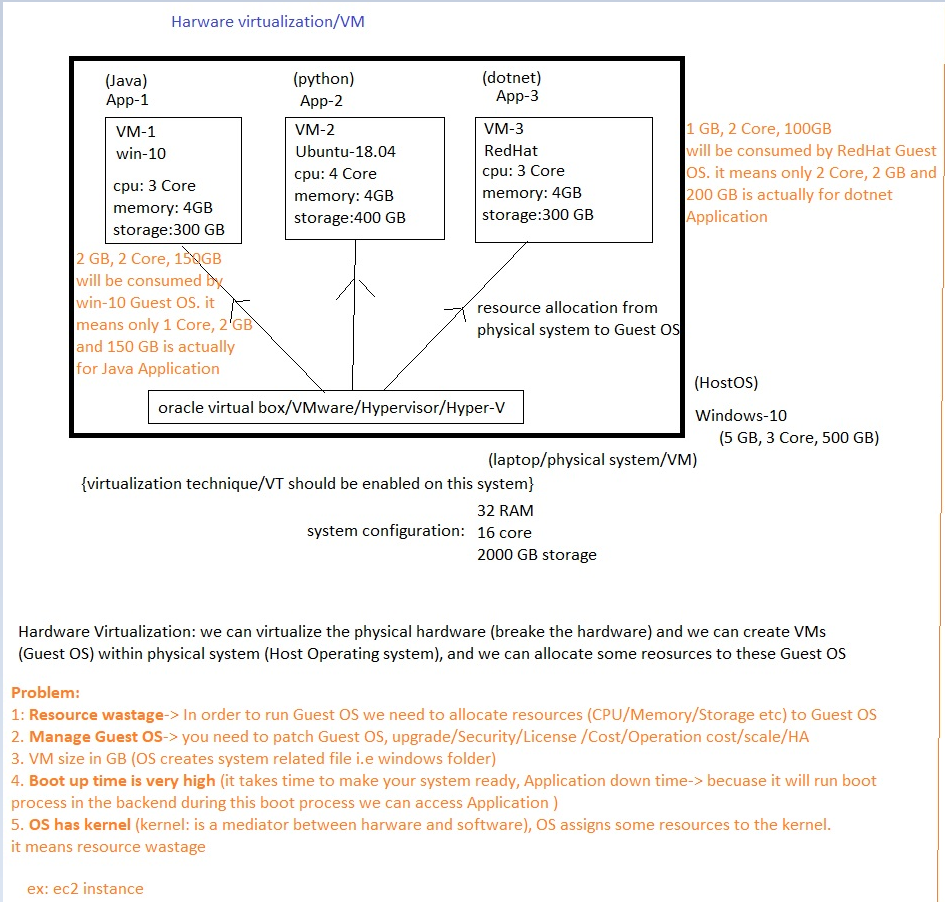
Q. Difference between Docker and VM ?

Or

Q. Difference between hardware virtualization and Software/OS virtualization ??

**Q. What is Docker image??**

**Q. What is a Docker container??**



Q. Benefits of Docker over VM??

<https://www.aquasec.com/cloud-native-academy/docker-container/docker-containers-vs-virtual-machines/#:~:text=This%20means%20Docker%20containers%20are,container%20from%20a%20container%20image>.

Q. What is Docker?

<https://www.ibm.com/in-en/topics/docker>

<https://www.aquasec.com/cloud-native-academy/docker-container/>

Q. What is Container technology??

<https://www.aquasec.com/cloud-native-academy/container-platforms/what-is-a-container-complete-guide-2022/>

Q. **What Are the Benefits of Containers?**

Containers provide a highly effective way to deploy applications and services at scale on any hardware. Applications or services running as containers use a small fraction of the resources on the host (enabling a large number of containers to run on one host). They are well isolated, so they don’t interfere with each other or directly affect the host’s operations.

Here are the main benefits of containers compared to other ways of running software on host infrastructure:

* **Lightweight**—because containers share the system’s operating system kernel, there is no need to run a complete operating system instance for each application, reducing the size of container files and resources needed. Containers can start quickly, are torn down easily, and are easy to scale horizontally, meaning they can better support cloud-native applications.
* **Portability and platform independence**—containers have all their dependencies inside. This means that the same software can be created once and run consistently on laptops, on-premise hardware, or in the cloud, with no reconfiguration required.
* **Support for modern architectures—**containers can be constructed from a simple configuration file and have a high level of portability and consistency. This makes them highly suitable for DevOps, microservices architectures, and serverless computing, in which software is built from small components that are iteratively developed.
* **Increased utilization—**containers allow developers and operators to increase CPU and memory utilization on physical machines. Containers allow granular deployment and scaling of application components, which can support microservices design patterns.

Q. what is Docker image??

<https://www.ibm.com/in-en/topics/docker>

<https://www.aquasec.com/cloud-native-academy/docker-container/>

Q. What is a Docker container??

Docker Container: Docker container is a runtime docker image.

Q. Docker CICD flow??

<https://medium.com/alterway/adding-image-security-scanning-to-a-ci-cd-pipeline-2faff072d>

**=================Docker Commands:**

docker images ---> list all the images locally docker VM

docker ps ---> list all the running container

docker ps -a ---> to list all the running + stopped container

docker pull <image name> ---> download docker image from DockerHub/Docker Registry(company managed docker registry)/AWS ECR (elastic container registry)

ex: docker pull python:3.10

or syntax: docker pull <registry id>/<imagename>:<tag value> - This is for organisation level

docker pull ethas/python:3.10 where ethas is company docker registry name

docker pull tomcat:jdk11

Create Docker container:-

syntax: docker run **-i -t** --name <container name> -p <VM port>:<app Port> <docker image name>:tag name

or

docker run -**it** --name <container name> -p <VM port>:<app Port> <docker image name>:tag name

-i: interactive mode

-t: terminal

-p: port (port mapping/port forwarding) -> container can not accessed outside from DockerHost VM directory

ex: docker run -it --name mytomcat -p 10000:8080 tomcat:jdk11 (VM port should be unique)

Q. How to access Containerized Apps from outside the DockerHost VM??

http://<public IP of Docker VM>:<VM port>

ex: http://18.194.209.95:10000

docker run -it --name myapache -p 10001:80 httpd

ex: <http://18.194.209.95:10001>

(Please press ctrl+c to exit from container’s terminal)

How to start a container??

syntax: docker start <container ID Or Name>

How to stop a running container??

syntax: docker stop <container ID/Name>

How to restart a running container??

syntax: docker restart <container ID/Name>

How to login to a running container/How to get into the container's terminal??

If you want troubleshoot, you can login to the terminal & check background processess

syntax: docker exec -it <container name/id> /bin/bash

/bin/bash : login to container's bash shell

login to apache container:

docker exec -it 8e1796be0f88 /bin/bash

ls

cd htdocs/

cat index.html

Q. Why are dockers lightweight??

vi index.html # by default you will not get vi package,install vi package separately

**dockers are lightweight because it contains binary and Library files related to Application only**

apt-get update -y # yum will not work, use "apt-get" as package manager when we have "debian" as base image

{How to identify the base image -> Go to docker hub of image-> check for github link -> Go to github -> Go to code -> You will find Dockerfile in root directory/some other folder in the code repo}

apt-get install vim -y # download vim package manager to modify index.html

vi index.html # now you can modify contents

Q. Difference between background mode & foreground mode??

docker run -it --name myapache1 -p 10002:80 httpd #running container in foreground mode

docker run -itd --name myapache2 -p 10003:80 httpd #running container in background mode (detached mode)

-d detached mode

You can run containers in attached mode (in the foreground) or in detached mode (in the background).

By default, Docker runs the container in attached mode. In the attached mode, Docker can start the process in the container and attach the console to the process’s standard input, standard output, and standard error.

Detached mode, started by the option --detach or –d flag in docker run command, means that a Docker container runs in the background of your terminal. It does not receive input or display output. Using detached mode also allows you to close the opened terminal session without stopping the container.

------------------How to create Custom docker image from Docker container??

ans: for testing purpose we can create docker images from docker container(once your done with testing , for backup purpose you can create custom docker images from docker container)

syntax: docker commit <container ID> <new docker image>

ex: docker commit 8e1796be0f88 customapche

docker images

------------How to upload/publish docker image to docker hub/AWS ECR/Company managed Docker Registry

first create your account on DockerHub: <https://hub.docker.com/>

Assign a tag (rename) to your local docker image (always it should starts with your ID)

syntax: docker tag <current docker image name>:<current tag value> <DockerHubID>/<New image name>:<New Tag Value>

OR

syntax: docker tag <current docker ID> <DockerHubID>/<New image name>:<New Tag Value>

ex: docker tag customapche:latest pkw0301/webui:jan0823 #dateformat 080123 , 01/02 , or release number

docker images # once your image is ready, you can upload to the dockerHub

docker login # Please provide your DockerHub id & password

syntax: docker push <DockerHubID>/<New image name>:<New Tag Value>

ex: docker push pkw0301/webui:jan0823

Now you can pull this newly created image to anywhere:

docker pull pkw0301/webui:jan0823

& you also can create the container:

docker run -itd -p 10004:80 pkw0301/webui:jan0823

docker ps

you can access App from browser: **http://<public IP of Docker Image>:10004**

verify: Login to DockerHub

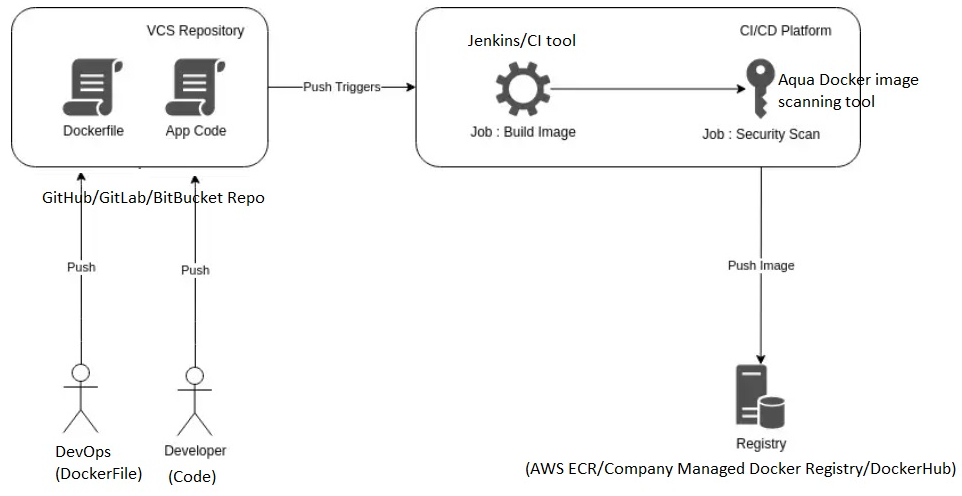
<https://hub.docker.com/>

click on your profile -> you will be able to see all the docker images

Assignment: How to upload custom docker image to aws ecr ??

create aws ecr service -> assign tag according to AWS -> upload docker image to AWS ECR

Docker CI/CD flow: <https://medium.com/alterway/adding-image-security-scanning-to-a-ci-cd-pipeline-2faff072d>



**====================Docker ECR:(Elastic Container Registry)**

-> Similar to DockerHub

-> we can use ECR as **private** Docker Registry

(from security , control, traceability etc we prefer AWS ECR over DockerHub )

Security -You can implement all the AWS IAM best practices (MFA, Password etc)

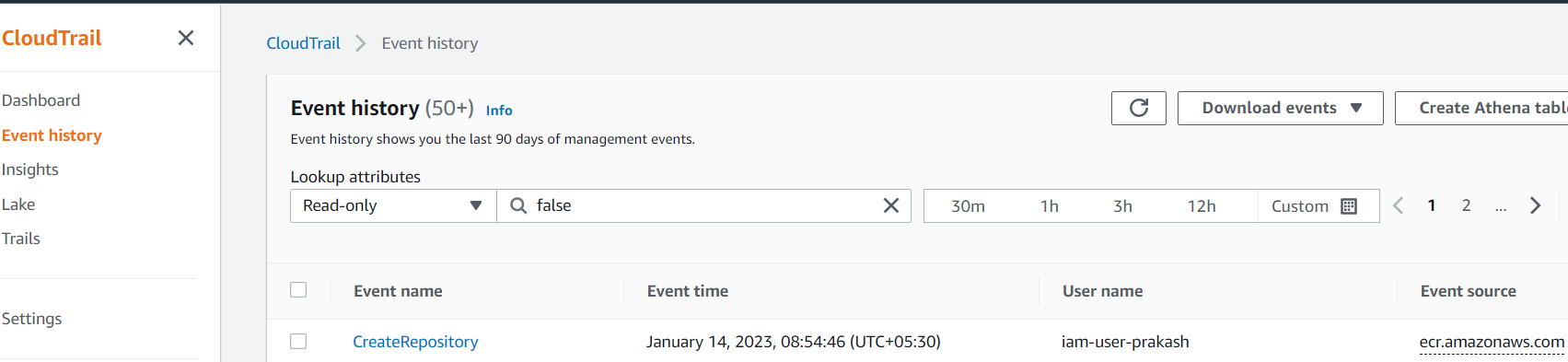
Control: AWS IAM policies

(ECR full access -upload docker image to ECR/Delete Docker image from ECR/Rename AWS ECR/Docker pull/Docker Tag/ etc by policies creating you can control the access)

Traceability: Cloud trail -> user activity

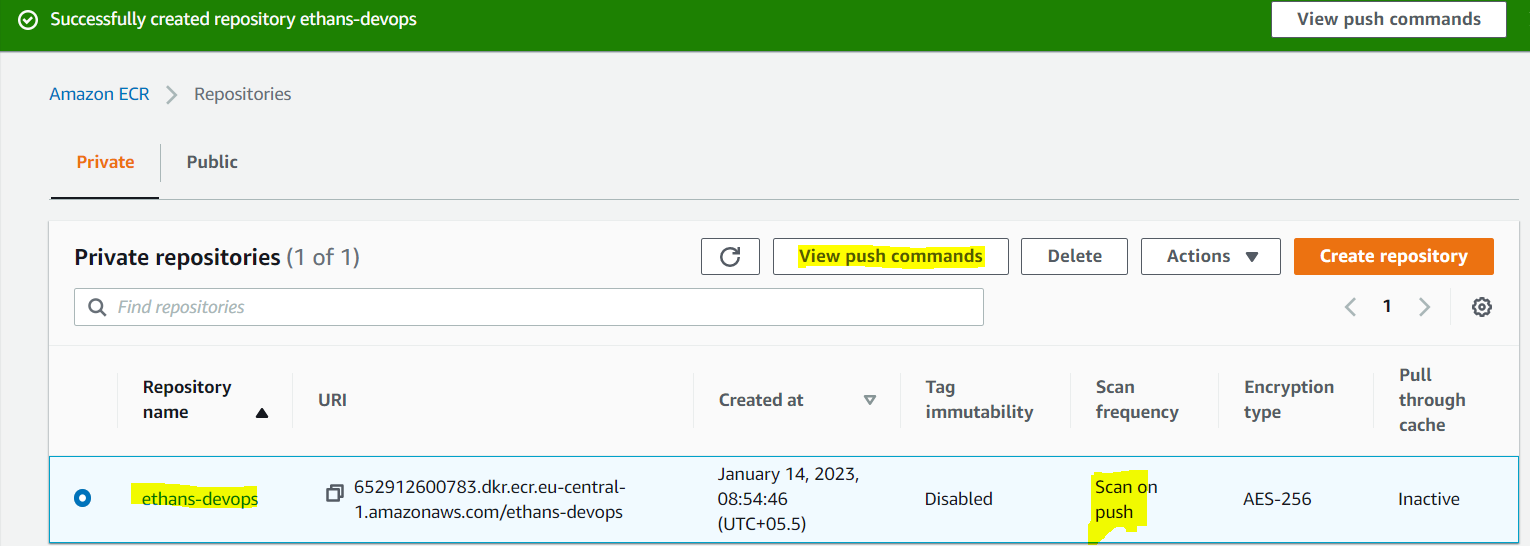
Use CloudTrail to meet your governance, compliance, and auditing needs for your AWS accounts.

Create a cloud trail to store logs/user activity logs

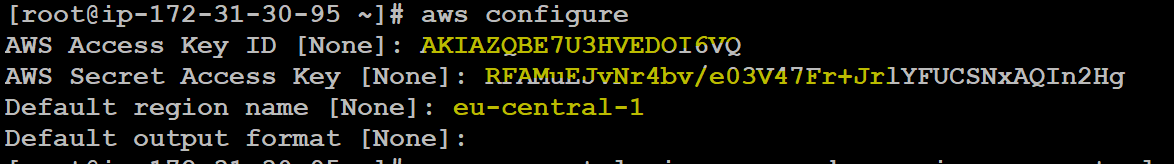


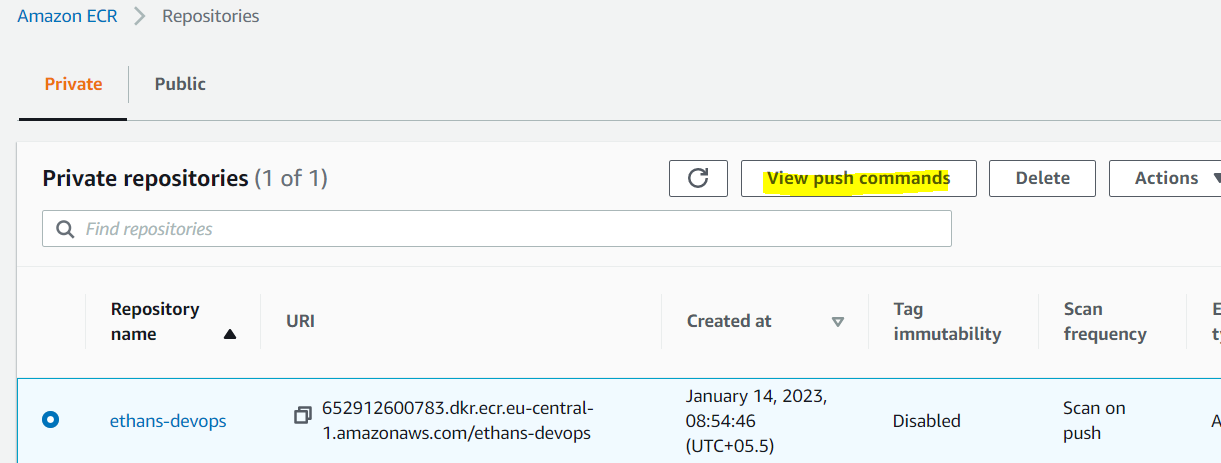
Login as IAM user (IAM login user) -> create AWS ECR (make sure IAM user should have Create ECR permission)

You can select scan on push button during ECR creation -> once you ECR is ready -> you can upload docker images



Once you are connected to the Docker VM -> type aws configure -> provide access key and secret key





Click on view push command and Run **docker login** then **tag**

Syntax: docker tag <current image> <ecr repo id>

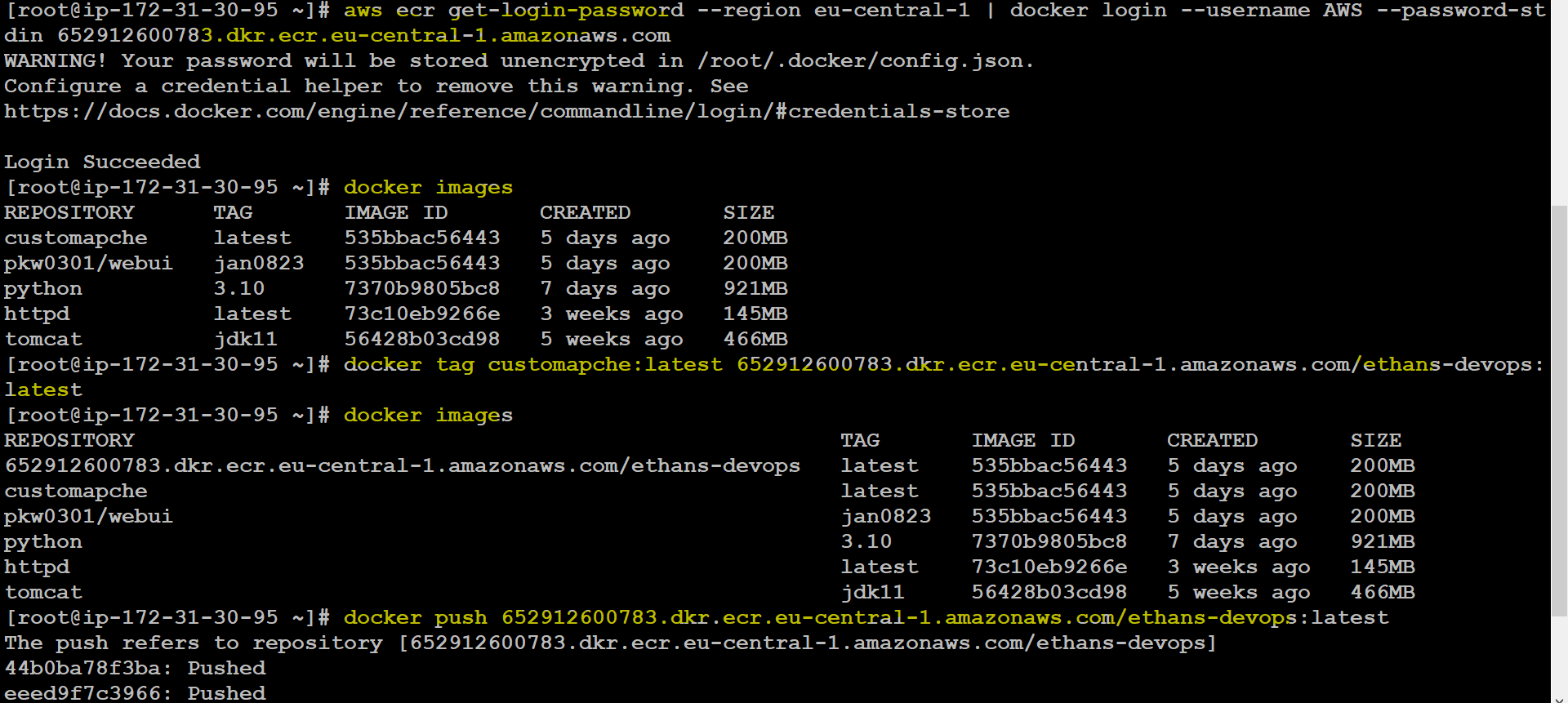
Ex: docker tag **customapche:latest** 652912600783.dkr.ecr.eu-central-1.amazonaws.com/ethans-devops:latest

docker push 652912600783.dkr.ecr.eu-central-1.amazonaws.com**/ethans-devops:latest**

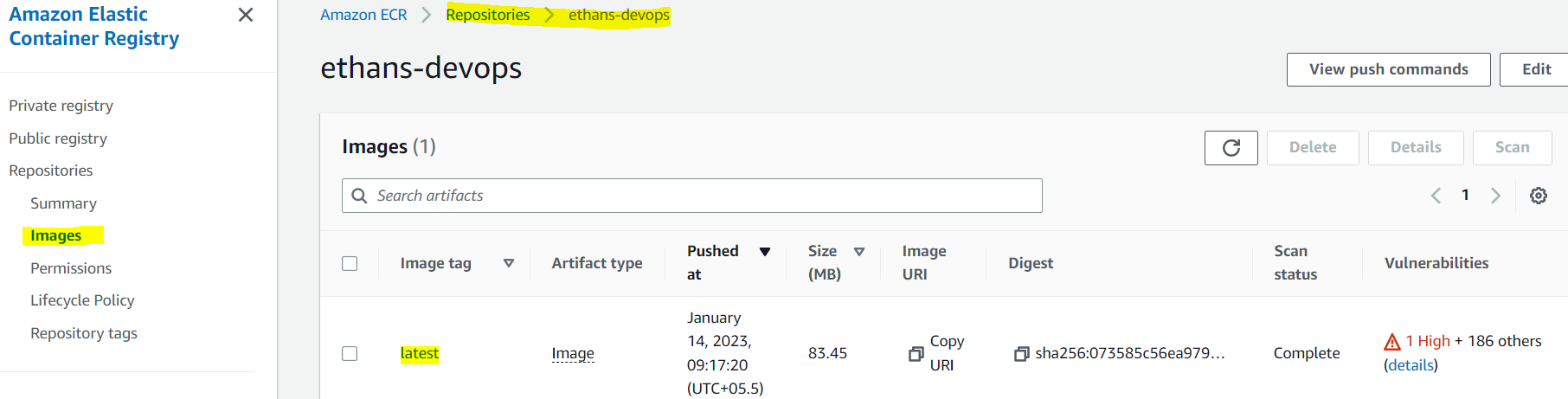
**syntax: docker tag <current image name>:<tag> <DockerRegistry ID>/<Repo name>:<tag Value>**

**docker push <DockerRegistry ID>/<repo name>:<tag Value>**

**DRe**



You can list the images



Remove Docker image (delete):

docker rmi <image name or id>

Q. Can we remove running or in use docker images?

Ans: We can not remove running or in use docker images , we can forcefully delete docker image

docker rmi <image name or id > --force —>this will remove the tag, image will be present on system if container is in running mode

Then you can delete container

Remove Docker Container (delete)

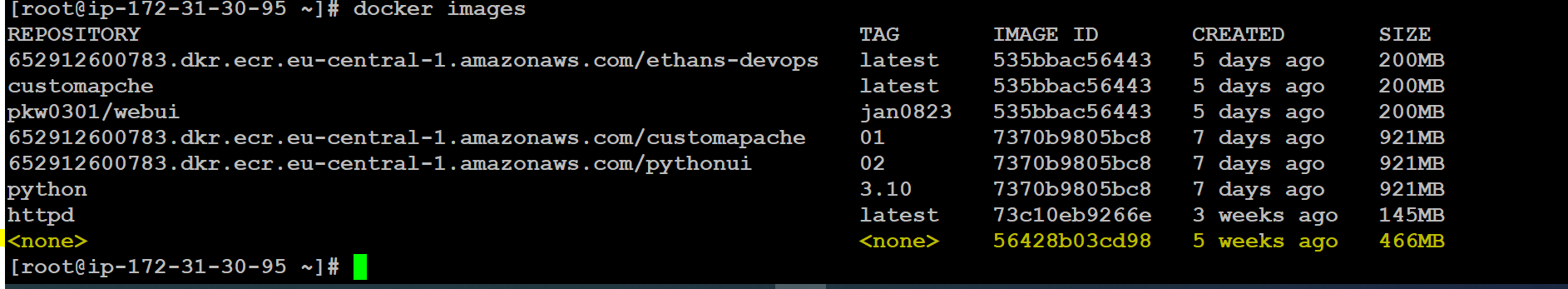
docker rm <container name or id>

\*\* you can not delete a running container, first you need to stop the container

docker stop <container Id>

Then you can delete a container

\*\* the image which we were trying to delete will be there on system



Q. What is nginx??

Nginx is similar to apache, this can be used to deployed web Application

Q. What do you understand by **dangling** images ???

Docker creates helping images during container creation, the image will be there on your system but in an unassigned state called dangling image.

dangling image consumes the space on your system (unused)

docker images -f dangling=true

Or

Q. have you written any script to clean unused image (the image which is not attached to a container)

<https://gist.github.com/shhider/97d94daf5f02a3c153abdc535520c7cc>

docker rmi $(docker images --filter "dangling=true" -q --no-trunc)

Or

docker image prune

Or

What do you understand by **docker system prune** /**docker image prune**/**docker container prune** ?

Docker prune deletes unnecessary/unused resources (image/container/storage ct)

docker system prune

WARNING! This will remove:

- all stopped containers

- all networks not used by at least one container

- all dangling images

- all dangling build cache

<https://dirask.com/posts/docker-and-docker-compose-list-of-most-useful-commands-common-cmd-1yN6QD>

**Assignment**: Install docker on your jenkins VM, Create any container/pull any image from docker hub/remove tag from image (image with <none> tag on jenkins)

Create a freestyle job -> execute shell -> then provide below command

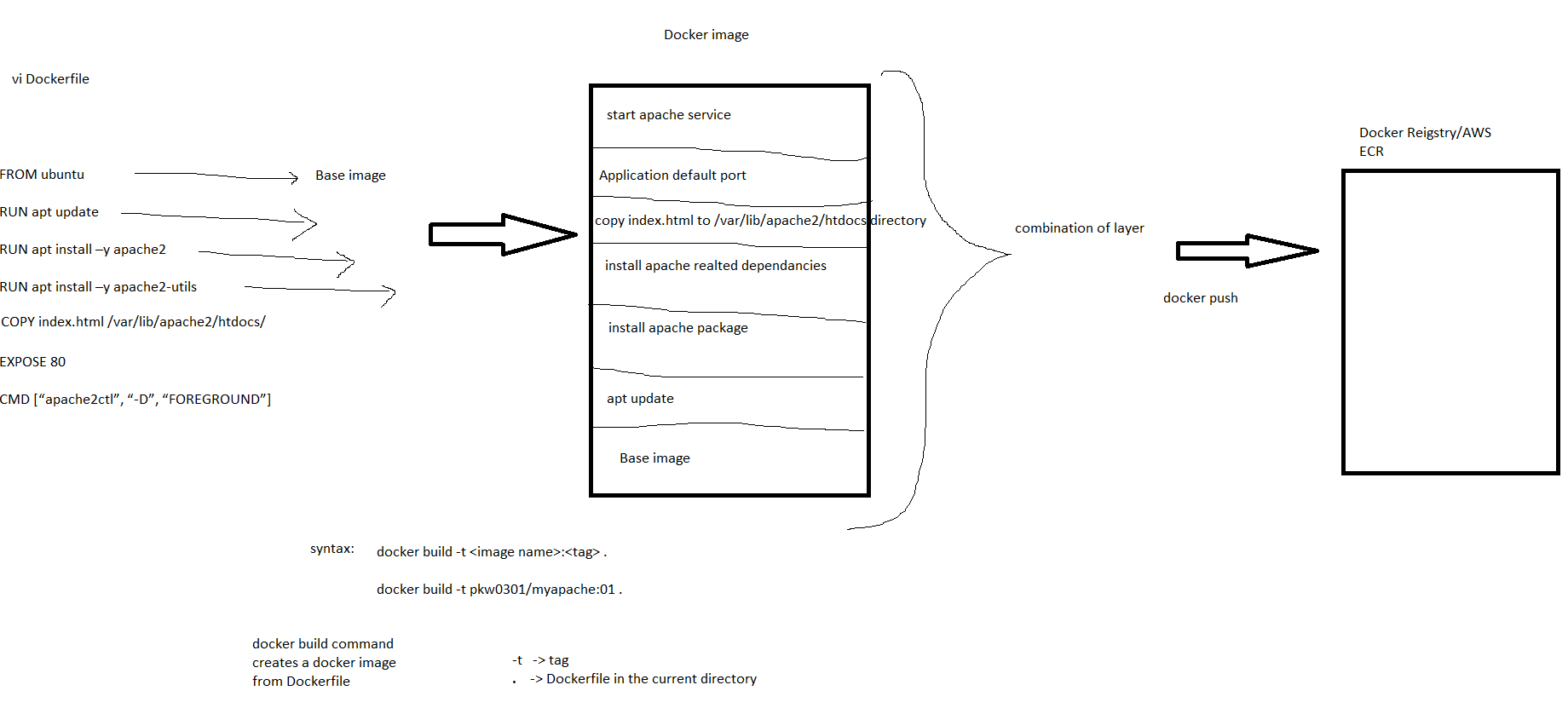
<https://gist.github.com/shhider/97d94daf5f02a3c153abdc535520c7cc>

And run your jenkins job -> it should delete dangling from jenkins VM

=================Dockerfile

It contains Docker instructions, (docker related commands), this file can be uploaded to Code repository along with the code

<https://www.liatrio.com/blog/building-with-docker-using-jenkins-pipelines>



**========================Dockerfile best practices:** <https://docs.docker.com/develop/develop-images/dockerfile_best-practices/>

**Before you write Dockerfile**

1. information gathering (as we know that code activity will be taken by development team)

a: what type code they have worked (java/python/dotnet/nodejs/infrastructure related etc) -> based on code we can decide the base image for Docker

b: identify code version (jdk7, jdk8,jdk10, jdk11 , python3.6, python3.7 , python3.10 and so on...)

2. ASk for code dependencies, how to run the code, what are things required to run code, do we need war/jar/ear/zip/xyz file to run App, do we need declare any variable

3. Application port number (default port number)

**Dockerfile instruction**

vi Dockerfile

FROM openjdk:8-jdk-alpine #search base image in Docker REgistry/DockerHub/AWS ECR

**FROM: Dockerfile always starts with "FROM" instruction, here you can define your base image**

Q1. shortest/smallest dockerfile instruction?

ans: in the smallest Dockerfile we can have only one docker instruction "FROM"

Q2. Can we have multiple "FROM" in a Dockerfile??

ans: yes, you can many FROM in dockerfile

**LABEL: Docker instruction**

FROM ubuntu:18.04 #Ubuntu, Redhat, Alpine, CentOS -> they are infrastructure related Docker images -> to run script,

LABEL author=prakash

LABEL description="this is for testing purpose"

LABEL env=dev

Syntax: LABEL <key>=<value>

LABEL can be used to define description, author information, tags, docker related information

How to create Docker image from Dockerfile????

vi Dockerfile

FROM ubuntu:18.04

LABEL author=prakash

LABEL description="this is for testing purpose"

LABEL env=dev

LABEL vm=linux

docker build -t pkw0301/myubuntu:01 . # pkw0301 = is my DockerHub id, or AWS ECR ID , t=tag

. (dot) = we are asking DockerEngine to create docker image from Dockerfile, DockerEngine creates docker image from current directory, make sure you put dockerfile in the current directory

**RUN : Docker instrcution**

can be used to execute commands **during docker image creation**, you can install required packages/dependencies/binary/library file

can be used to run any linux/shell command

can be used to download packages from internet/specific URL/company URLs

vi Dockerfile

FROM ubuntu:18.04

LABEL author=prakash

LABEL description="this is for testing purpose"

LABEL env=dev

LABEL vm=linux

RUN apt-get update -y && \

apt-get install git -y && \

apt-get install wget -y

docker build -t pkw0301/mycustomimage:05 .

docker inspect <docker image/container name/> ----> you can inspect, you will get complete information about docker image/ docker container

docker run -itd --name mycontainer-5 pkw0301/mycustomimage:05 /bin/bash ----> create container and verify .

{/bin/bash -> use bin/bash when you are working with infrastructure related container, you DON't Need to add /bin/bash during Application related container}

download apache on ubuntu base image:-

FROM ubuntu:18.04

LABEL author=prakash

LABEL descritption="this is for testing purpose"

LABEL env=dev

LABEL vm=linux

RUN apt-get update -y && \

apt-get install wget -y && \

apt-get install apache2 -y ----> you can download any application package on docker image

docker build -t pkw0301/myapache:06 .

docker run -itd --name myapche06 -p 16000:80 pkw0301/myapache:06

docker exec -it myapche06 /bin/bash

service apache2 start ----> start apache service in a container

go to the browser: http://<public IP of Doker VM>:16000

OR

vi Dockerfile

FROM httpd:2.4.54

LABEL author=prakash

LABEL description="this is for testing purpose"

LABEL env=dev

LABEL vm=linux

docker build -t pkw0301/myapache:07 .

docker run -itd --name myapche07 -p 16001:80 pkw0301/myapache:07

go to the browser: http://<public IP of Docker VM>:16001

**EXPOSE : Docker instruction**

we can declare application default port number in Dockerfile

vi Dockerfile

FROM httpd:2.4.54

LABEL author=prakash

LABEL description="this is for testing purpose"

LABEL env=dev

LABEL vm=linux

EXPOSE 80 #this will help you to run container on same port, so that Application can be accessible from outside the DockerHost VM

docker build -t pkw0301/myapache:08 .

docker run -itd --name myapche07 -p 16002:80 pkw0301/myapache:08

go to the browser: http://<public IP of Docker VM>:16001

**WORKDIR: Docker instruction**

You can set the working directory, when you connect to the container's terminal you will get WORKDIR as default directory

vi Dockerfile

FROM ubuntu:18.04

LABEL author=prakash

LABEL description="this is for testing purpose"

LABEL env=dev

LABEL vm=linux

RUN apt-get update -y && \

apt-get install wget -y && \

apt-get install apache2 -y

WORKDIR /usr/local/apache2

OR (this is NOT correct because you can not set WORKDIR before installing apache,

/usr/local/apache2 will not be there so if you set WORKDIR before installing apache you will get unexpected output )

FROM ubuntu:18.04

LABEL author=prakash

LABEL description="this is for testing purpose"

LABEL env=dev

LABEL vm=linux

WORKDIR /usr/local/apache2

RUN apt-get update -y && \

apt-get install wget -y && \

apt-get install apache2 -y

**COPY: Docker instruction**

using this instruction we can copy files from Docker Host VM to Docker image

vi Dockerfile

FROM ubuntu:18.04

LABEL author=prakash

LABEL description="this is for testing purpose"

LABEL env=dev

LABEL vm=linux

RUN apt-get update -y && \

apt-get install wget -y && \

apt-get install apache2 -y

EXPOSE 80

WORKDIR /usr/local/apache2

COPY index.html /usr/local/apache2/htdocs/ # /usr/local/apache2/htdocs this is similar to /var/www/html

COPY <Docker VMsource file location> <docker image destination >

docker engine will copy file from Docker VM to docker image, first I need to create index.html on Docker VM

docker build -t pkw0301/myapache:09 .

docker run -itd --name myapche09 -p 16003:80 pkw0301/myapache:09

docker exec -it myapche09 /bin/bash

service apache2 start ----> start apache service in a container

go to the browser: http://<public IP of Docker VM>:16003

**ADD: Docker instruction**

ADD is similar to COPY. using this instruction we can copy files **from Docker Host VM** as well as from **URL (outside VM it could be internet/company specific URLs)** to Docker image

vi Dockerfile

*FROM ubuntu:18.04*

*LABEL author=prakash*

*LABEL description="this is for testing purpose"*

*RUN apt-get update -y && \*

*apt-get install wget -y && \*

*apt-get install apache2 -y*

*EXPOSE 80*

*WORKDIR /usr/local/apache2*

*ADD index.html /usr/local/apache2/htdocs/*

*ADD* [*https://prakash-static-web.s3.eu-central-1.amazonaws.com/sample.war*](https://prakash-static-web.s3.eu-central-1.amazonaws.com/sample.war)***./htdocs/***

**(. =** /usr/local/apache2 **)**

OR ADD <https://prakash-static-web.s3.eu-central-1.amazonaws.com/sample.war> **/usr/local/apache2/htdocs/**

docker build -t pkw0301/jandevops:01 **.**

**docker run -itd --name jan21 -p 17000:80 pkw0301/jandevops:01**

**docker exec -it jan21 /bin/bash**

Q. What is the difference between ADD and COPY??

ADD: we can copy files from local docker VM & URLs

COPY: we can copy files from local docker VM only

Q. Why should we avoid ADD over COPY??

Ans: ADD copies file to docker layer directly, Now that file is part of the docker image.

So you can not delete files from the docker image layer. It unnecessarily increases the size of docker image, though ADD copies file to docker layer so chances are very high that it may copy some binaries from the internet. That we avoid ADD docker instruction to download files from the internet. We can use **RUN wget <URL>** to download files.

*FROM ubuntu:18.04*

*LABEL author=prakash*

*LABEL description="this is for testing purpose"*

*RUN apt-get update -y && \*

*apt-get install wget -y && \*

*apt-get install apache2 -y*

*EXPOSE 80*

*WORKDIR /usr/local/apache2/htdocs/*

*COPY index.html /usr/local/apache2/htdocs/*

***RUN wget***[*https://prakash-static-web.s3.eu-central-1.amazonaws.com/sample.war*](https://prakash-static-web.s3.eu-central-1.amazonaws.com/sample.war)

*(destination would be your WORDIR by default)*

Q. ADD is not recommended to download files from URL then how to download files from internet??

**CMD: Docker instruction**

CMD is similar to RUN, We can run Linux commands to start the process, execute scripts, download dependencies **,during container creation**

**(RUN -> during image creation**

**CMD-> during container creation)**

**ENTRYPOINT: Docker instruction**

This is similar to CMD, you can either use CMD or ENTRYPOINT, the only difference is ENTRYPOINT has higher precedence over CMD , if CMD and ENTRYPOINT both are there in the Dockerfile then sequence will start from ENTRYPOINT

ref link: <https://docs.docker.com/engine/reference/builder/>

========================

**Q. deploy web application (java related war file) tomcat 8 docker container based on ubuntu:18.04**

ans: base image as ubuntu:18.04

download dependencies jdk,

download a tool to extract a file (gzip, unzip)

download tomcat 8 (https://archive.apache.org/dist/tomcat/), download tomcat 8 tar.gz file , untar tomcat file (tar xvzf fileName , xvzf=extract, verbose,zip, file)

copy war file, extract war file, delete war file afer extract (https://www.middlewareinventory.com/blog/sample-web-application-war-file-download/)

expose port number

CMD: start tomcat service

FROM ubuntu:18.04

LABEL author=prakash

LABEL app=webapp

RUN apt-get update -y && \

apt-get install wget -y && \

apt-get install gzip -y && \

apt-get install openjdk-8-jdk -y && \

wget <https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.85/bin/apache-tomcat-8.5.85.tar.gz> && \

tar xvzf apache-tomcat-8.5.85.tar.gz

ADD <https://github.com/AKSarav/SampleWebApp/raw/master/dist/SampleWebApp.war>

/apache-tomcat-8.5.85/webapps/

or

***FROM*** *ubuntu:18.04*

***LABEL*** *author=prakash*

***LABEL*** *app=webapp*

***RUN*** *apt-get update -y && \*

*apt-get install wget -y && \*

*apt-get install gzip -y && \*

*apt-get install openjdk-8-jdk -y && \*

*wget https://archive.apache.org/dist/tomcat/tomcat-8/v8.5.85/bin/apache-tomcat-8.5.85.tar.gz && \*

*tar xvzf apache-tomcat-8.5.85.tar.gz && \*

*rm apache-tomcat-8.5.85.tar.gz*

***WORKDIR*** */apache-tomcat-8.5.85/webapps/*

***RUN*** *wget* [*https://github.com/AKSarav/SampleWebApp/raw/master/dist/SampleWebApp.war*](https://github.com/AKSarav/SampleWebApp/raw/master/dist/SampleWebApp.war)

***EXPOSE*** *8080*

***CMD*** *["/apache-tomcat-8.5.85/bin/catalina.sh" , "run"]*

{to start tomcat service simply run catalina.sh file}

or

CMD ["service tomcat start" , "-y"]

docker build -t pkw0301/webapptomcat:05 .

docker images

docker run -itd --name webapptomcat -p 18050:8080 pkw0301/webapptomcat:05

docker ps

docker exec -it 3286318ad5c4 /bin/bash

pwd (it should point to /apache-tomcat-8.5.85/webapps/)

ls (SampleWebApp.war & SampleWebApp file)

{tomcat is very advance tool, automatically it extracts the war file}

How to access:-

http://3.70.100.221:18050/SampleWebApp/

**==============Q. deploy web application (java related war file) tomcat 8 docker container**

FROM tomcat:8.5.85-jdk8-temurin-focal

LABEL author=prakash

RUN apt-get update -y && \

apt-get install wget -y

WORKDIR /usr/local/tomcat/webapps/

RUN wget https://github.com/AKSarav/SampleWebApp/raw/master/dist/SampleWebApp.war

CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]

docker build -t pkw0301/webapptomcat:10 .

docker images

docker run -itd --name webapptomcat10 -p 18051:8080 pkw0301/webapptomcat:10

docker ps

docker exec -it webapptomcat10 /bin/bash

pwd (it should point to /usr/local/tomcat/webapps/)

ls (SampleWebApp.war & SampleWebApp file)

{tomcat is very advanced tool, automatically it extracts the war file}

cd /usr/local/tomcat/bin/

ls (you will be able to see catalina.sh file)

How to access:-

http://3.70.100.221:18051/SampleWebApp/

**==================Dockerfile for python**

ref link: -> <https://www.docker.com/blog/how-to-dockerize-your-python-applications/>

**==================Dockerfile for nodeJs**

<https://www.digitalocean.com/community/tutorials/how-to-build-a-node-js-application-with-docker>

**pip** => pip is a package manager for python code, using pip you can download dependencies (requirements.txt, or check with the dev team)

**npm**(node package manager) => npm is a package manager for node code, using npm you can download dependencies (packages.json, or check with the dev team)

**===========How to integrate Docker with Jenkins**

1-start jenkins server (if needed you can increase the size from t2micro to t2medium..only for this lab)

2-install docker on jenkins -> start & enable docker on jenkins

3-connect to jenkins dashboard-> download docker plugins

4-integrate docker & jenkins -> manage jenkins -> configure system -> at botton you will get cloud/docker -> click on cloud option -> search docker from drop down

5-click on question mark , click on help page, you will get a docker.sock path (unix:///var/run/docker.sock ), copy path & paste docker.soak path that's it

so that jenkins can call/run docker command (just to grant permission to jenkins)

6. create a declarative pipeline -> syntax generate -> execute shell -> then pass docker command

https://octopus.com/blog/jenkins-docker-ecr

<https://www.liatrio.com/blog/building-with-docker-using-jenkins-pipelines>

=======================================Docker Jenkins Integration

RUN wget https://github.com/AKSarav/SampleWebApp/raw/master/dist/SampleWebApp.war

CICD best practices: install Docker on Jenkins VM

COPY /usr/local/jenkins/workspace/<JOBNAME>/webapp/target/webapp.war /usr/local/tomcat/webapps/

source directory: /usr/local/jenkins/workspace/JOBNAME/target/webapp.war ---> Jenkins Docker Host VM

Destination Directory: /usr/local/tomcat/webapps/ ---> Docker image

FINAL Dockerfile

FROM tomcat:8.5.85-jdk8-temurin-focal

LABEL author=prakash

RUN apt-get update -y && \

apt-get install wget -y

WORKDIR /usr/local/tomcat/webapps/

COPY **webapp/target/webapp.war** /usr/local/tomcat/webapps/

CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]

1. install docker on jenkins

2. install plugin docker related plugins

3. manage jenkins ->manage credential (DockerHub credentials)

4. manage jenkins -> manage node & cloud -> select cloud -> copy docker.sock path & change permission

chmod 777 /var/run/docker.sock

5. create declarative job

**-----vi Dockerfile**

FROM tomcat:8.5.85-jdk8-temurin-focal

LABEL author=prakash

RUN apt-get update -y && \

apt-get install wget -y

WORKDIR /usr/local/tomcat/webapps/

COPY webapp/target/webapp.war /usr/local/tomcat/webapps/

CMD ["/usr/local/tomcat/bin/catalina.sh", "run"]

**----vi JenkinsDockerpipeline-file**

pipeline

{

agent any

stages

{

stage('scm checkout')

{steps {git branch: 'master' , url: 'https://github.com/prakashk0301/maven-project' }}

stage('code build')

{steps { withMaven(globalMavenSettingsConfig: '6ae3dfc3-6ab7-4e7b-a9bd-418f757884b0', jdk: 'JAVA\_HOME', maven: 'MAVEN\_HOME', mavenSettingsConfig: '90c73295-0087-412a-9860-16baf9cf34f2')

{ sh 'mvn clean package' }

}}

stage ('docker build or create image')

{steps { sh 'docker build -t pkw0301/mavenwebapp:01 .' }} //pkw0301 is docker registry id

stage ('push docker image to docker registry')

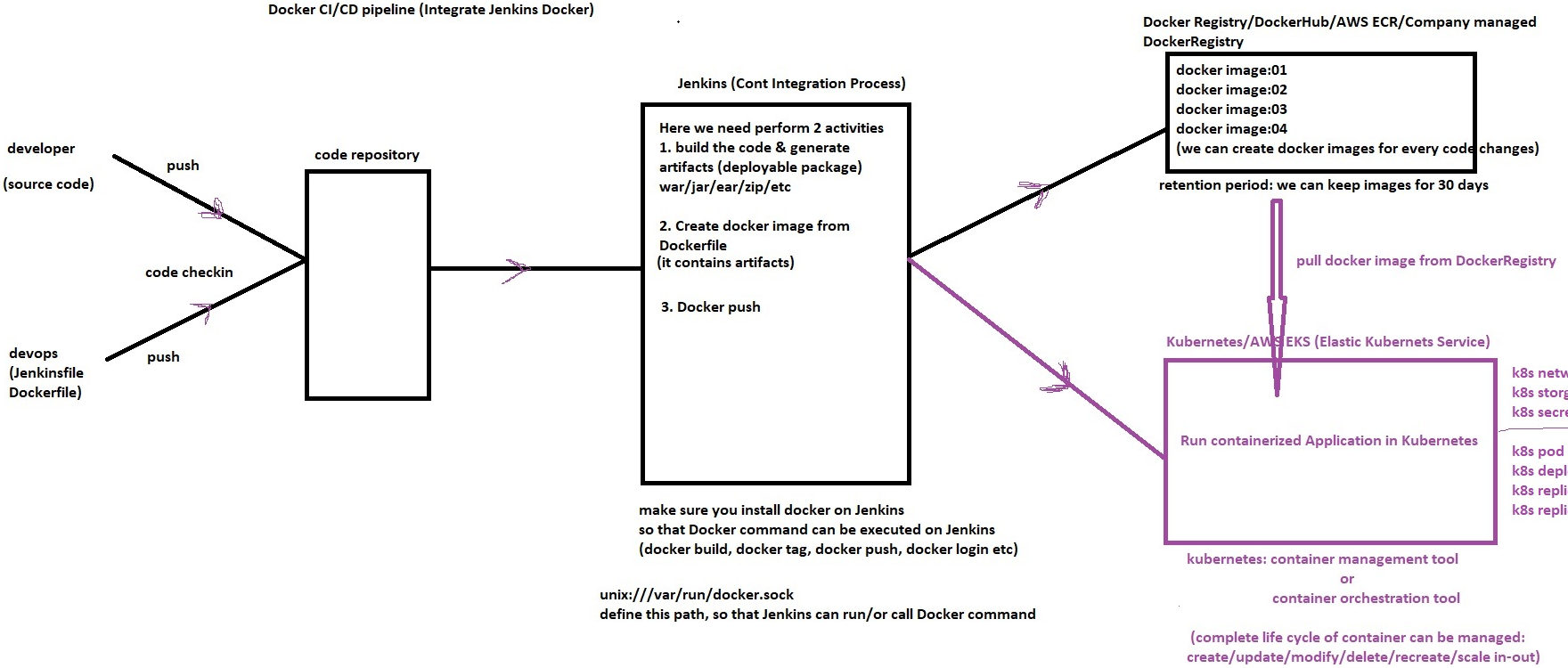
{steps { withDockerRegistry(credentialsId: 'dockerHubRegistry', url: 'https://index.docker.io/v1/')

{ sh 'docker push pkw0301/mavenwebapp:01' }

}}

}

}



assignment:

1. upload docker images to AWS ECR from Jenkins

hint:make sure install amazon ECR plugin