**Docker Assignment 1**

1. What is Docker?

* Docker is a tool that promises to easily encapsulate the process of creating a distributable artifact for any application, deploying it at scale into any environment, and streamlining the workflow and responsiveness of agile software organizations.
* In layman term, it simply means with help of docker it becomes easier to work with different teams/platforms. Let me explain in simple terms.
* For eg. there are 2 teams developer and operations, dev team has build one application and set some rules regarding dependency and will share with operations team, now the operations teams will find it hard to scale up the application, to run the application. because it was developed by them to overcome some issues eg. OS issue or different libraries. So now Docker comes into picture if developer team add just docker then it can run on any platform, do developer team will share the docker image. so now the operations team can run this docker image on any platform since docker is platform independent.
* Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly.
* basic example whenever you develop any web site using PHP or maybe asp.net whenever you work on that project and move that project on to the web server there are definitely some uncertainties that occurs maybe your images are not being loaded properly maybe the path is a little bit different
* developer says it works absolutely fine on their machine but when same piece of code/application is moved to another machine then maybe they forgot to mention some of the dependencies that you might have installed while working on some other projects and that's why the problem is occurring so docker is designed to specifically to avoid this kind of issues.
* docker is just completely a different thing and it's compatible with almost any programming language or any project that you are
* docker allows you to have absolutely sealed airtight container and these containers are the absolute heart of the docker these container wrap up your entire code and these are absolutely portable the portability is one of the main feature of the docker now we can take this container and wherever you are going to put this container it's going to work absolutely and exactly like how it worked on your machine

1. What is a Container?

* A container is a same idea as a physical container think of it like a box within an application . Inside this box the application seems to have its own machine name and IP address and it also has its own disk drive
* All resources are virtual for eg. if there are 10 different containers running then it will use same OS kernel.
* Docker provides the ability to package and run an application in a loosely isolated environment called a container.
* We can run multiple containers at same time which are isolated and secure.
* Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host.
* If you are running an say ubuntu/Linux kernel and on top of that have 10 different applications container and those are linux other distro like(mint,Centos, Fedora,RHEL etc) then these containers kernel will use the same Ubuntu's Kernel as shared.
* If you running say Windows os and on top of that have 10 different windows app container can run easily, but if those 10 are now linux based then there will be a Linux VM in-between and inside that LinuxVM you can run those 10 Linux app based container on windows host through Linux VM.
* A docker container is a virtualized run time environment where users can isolate applications from an underlying systems.
* Container will have a container layer(read-write Mode ) and image layer which is read mode only. there can be multiple image layer

1. What are Docker Images?

* A Docker image is an immutable (which cannot be changed) file that contains the source code, all dependences, tool, libraries and other files which are basically needed for an application to run
* these are also called as snapshots
* it is just a template which has application or set of instructions in it
* without image you cannot build a container
* with the help of these templates you would be building a container
* from dockerfile -> Image -> Container
* dockerfile is set of instructions these file runs and execute commands which will create a image and then it is used a template which is used to run as a application inside a container
* An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.

1. What is Docker Hub?

* Docker Hub is docker registry where we save our images.
* Docker Hub is a public registry that anyone can use, and Docker is configured to look for images on Docker Hub by default. You can even run your own private registry.
* Docker Hub is a service provided by Docker for finding and sharing container images with anyone may be with your team. It is the world’s largest repository of container images with an array of content sources including container community developers, open source projects and independent software vendors (ISV) building and distributing their code in containers..

1. Explain Docker Architecture?

* Docker is a client-server architecture.
* The Docker client first communicates to the process called Docker daemon(dockerd), which builds and run and distributes you containers.
* The Docker client and daemon can be on same server or you can connect a Docker client to a remote Docker daemon which runs on some different server.
* The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface.
* One more Docker client is Docker Compose, allows you to work with applications which consist multiple/set of containers.
* Let us see some subcomponents
* client - for eg. docker build, pull and run -> this all talks with dameon which is on Docker host and it can be on same client machine or on remote machine.
* Docker Host consist of Docker daemon and container and images
* **Docker daemon** (dockerd) it basically waits and listens for Docker API requests and manages Docker objects such as images, containers, networks, and volumes.
* **Docker client** (docker) talks to main component that is Docker. When user executed commands like docker run, the client sends these commands to dockerd, which takes care for these command. The docker command uses the Docker API. The Docker client can communicate with more than one daemon.
* **Docker Desktop** is tool or application for your system(Mac or Windows) environment that enables you to build and share containerized applications and micro services. Docker Desktop has multiple things in it like Docker daemon (dockerd), Docker client (docker), Docker Compose, Docker Content and many more.
* **Docker registry** stores Docker images. Docker Hub is a public registry that anyone can use and Docker is configured to look for images on Docker Hub. We can also customize it by writing our own registry

1. What is a Dockerfile?

* dockerfile is set of instructions these file runs and execute commands which will create a image and then it is used a template which is used to run as a application inside a container
* It is a file which has some set of instructions to follow to assemble an image.

Command - **docker build** will be used to build an image from a docker file

* In simple terms dockerfile is set of instructions these file runs and execute commands which will create a image and then it is used a template which is used to run as a application inside a container

1. What is the purpose of the EXPOSE command in Dockerfile?

* EXPOSE command in dockerfile tells Docker to get all its information required during the runtime from a specified Port. This can be TCP/UDP
* It just tells that add metadata to the image to describe that the container is listening on port number.. this can be in TCP/UDP by default it's TCP

1. Why is docker monitoring necessary?

* Monitoring is the most important for any application mainly in PRODUCTION environment or somewhere in lower environment like dev as well.
* Real time monitoring is must to ensure applications in containers peak performance. Also for resources management which can hamper the physical system/server on which container is running though in some case virtually
* To Detect and solve issues at an early stage and work proactively to avoid similar issues production
* If any new container or app is deployed in docker/container then it is highly recommend to get that monitoring on for smooth ongoing production traffic which may use these apps in one way or other.
* Implement changes safely as the entire environment is monitored
* Based on the reports from monitoring usage we can tune applications to deliver improved performance and better experience for users
* If monitoring is enabled then we can check and optimize resource allocation

1. Explain the implementation method of continuous integration (CI) and continuous deployment (CD) in Docker.

* You can run **Jenkins** on Docker
* Using **docker-compose** you can run integration tests in Jenkins
* Let us understand below trminology in detail
* CI it is a coding practice that allows the development team to make and implement small changes in the code and version control methods very often.
* Continuous integration is usually done in the form where all developers push the code onto a shared repository mostly multiple times a day.
* Nowadays it has become very important to have such a mechanism to integrate and validate the changes made to the code in a parallel way.
* Jenkins is an automation tool with built in plugins for continuous integration tasks. It is used to continuously build and test projects making it easier to integrate the changing codes to it.
* Continuous delivery is an extension of CI in a way that it starts in the process process where CI stops.
* CD allowed us automate all releases to the infrastructure defined. It basically ensures an automated way to push the code changes.
* Let's assume that your source code is in github repository and then you clone it locally.
* usually this will be in development and let's see the steps to move it to production alone with CI and CD
* Using a source code you will write a docker file which will create an image for it.
* with your apps and the run time required for it let say you build a version 1 of it then you can test run it locally either you can use docker run commands or if you apps needs a backend service such as DB or you want to automatically setup your application create those environment such as docker compose. Docker compose is basically to run multiple containers parallel and is extremely used to test run locally .Once you test it and find it ok you can publish the image to registry. Let's say you published the version 1 of your app/image. Now how do you take it to your production live env staging to prod. What we saw so far is part of continuous integration env or the dev env. how does it works into production like environment is where we would use system like kubernetes. which will help to do CD and deployment and so on.

For eg.We can create a docker file with following

FROM ubuntu

RUN apt-get update

RUN apt-get install-u python3 pythin3-pip

RUN pip3 install flask

RUN mkdir /pot/app

WORKDIR /opt/app

COPY . /opt/app

ENTRYPOINT FLASK\_APP=/opt/app/app.py flash run --host=0.0.0.0

Now we need to build with below commands open terminal and execute below commands

docker build . -t calc -> to build a image of above commands on top of ubuntu image and tag

docker images -> to check the image and there should be image with name as calc

docker run -p 8080:5000 cal -> to do a port mapping to allow access external to container and run

-> from this step output we get the IP which we can check in browser with port 8080

If you need to push it somewhere then first login

docker login -provide user id and password

docker push calc - Now you can sign into your docker hub and check it repository from there

docker ps -> to check the existing active containers

1. What is a Docker Engine?

* Docker Engine is an open source containerization technology/process for building and containerizing various applications. Docker Engine you can image like client-server application with:
* A server which has a long-running daemon process dockerd.
* APIs which specify interfaces that programs can use to talk to and instruct the Docker daemon.
* A command line interface client docker.