**Docker**

* **Introduction of Docker:**
* Docker is a Containerization software.
* Docker is a virtualization software.
* It is written in GO language.
* It was released in 2013.
* Developed by Solomon Hykes and Sebastian phal.
* Makes developing and deploying applications much easier.
* Docker is a free and open source.
* It is one of the Devops tool.
* Docker is a set of platforms as a service product that use OS level virtualization to deliver software in packages called containers.
* Docker is an open-source platform that automates the deployment, scaling, and management of applications with lightweight, portable containers.
* **Why docker?**
* Using docker lets you ship code faster, standardize application operations, seamlessly move code, and save money by improving resource utilization.
* Docker is an open platform for developing, shipping, and running application.
* Handover our application safely to the end user.
* **What is container?**
* Container is a way to package application with all the necessary dependencies and configuration.
* It is a virtual machine which does not have any operating system.
* **What is containerization?**
* Containerization in Docker is a technology that allows developers to package applications and their dependencies into a standardized unit called a container.
* **Key features:**
* Portable
* Lightweight
* They do not have complete operating system.
* Docker image: helps us to container to create multiple containers.
* **Docker lifecycle:**
* Write a docker file and execute this docker file create a docker image and execute this image create a docker container.
* Execute docker file to docker image using by “build” command
* Execute docker image to docker container by using “run” command
* All these inputs stored in docker engine.
* **Docker workflow:**
* **Monolithic:**
* **Micro service:**
* **Virtualization:** create number of virtual servers inside our machine or physical server. Hypervisor is used to create the virtualization.
* **Drawbacks of virtualization:**
* It is old method.
* If we use multiple OS, the system performance is low.
* **Docker objects:**
* When you use Docker, you are creating and using images, containers, networks, volumes, plugins, and other objects.
* Docker client and server architecture:
* Docker uses a client-server architecture.
* The docker client talks to the docker daemon, which does the heavy lifting of building, running, and distributing your docker containers.
* The docker client and daemon can run on same system.
* Docker client server architecture consists of:

1. Docker client:

* A CLI that allows users to interact with the docker daemon.
* Sends commands to the docker daemon(server) to build, run and manage containers.
* Can communicate with multiple docker daemons(servers) over a network.

1. Docker daemon:

* It is the server-side component of docker.
* Listens from API requests from the docker client and manages docker objects such as images, containers, networks, and volumes.
* Handles the creation, execution, and management of containers.

1. Rest API:

* The communication between docker client and docker daemon is facilitated by a REST API.
* The API allows clients to send HTTP requests to perform operations on docker objects.

1. Docker registry/hub:

* A repository for storing and distributing docker images.
* Docker hub is the default public registry, but private registries can also be set up.
* The docker client can push images to and pull images from the registry.
* Difference between docker file and docker image and docker container?

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | Docker file | Docker image | Docker container |
| Definition | A docker file is a text file that contains a set of instructions for building a docker image. | A docker image is a read only template that contains the application code, libraries, dependencies, and runtime. | A docker container is a runnable instance of a docker image. it includes everything needed to run an application. |
| Purpose | Defines the environment and dependencies needed for an application. | Serves as blueprint for creating containers. | Provides an isolated environment for running applications. |
| Immutability | Mutable (can be modified) | Immutable (once created it does not change) | Mutable (can be started, stopped, and modified) |
| Creation | Written manually by developers. | Built from a docker file. | Created from a docker image. |
| Storage | Not stored just a script. | Can be stored in a registry (docker hub). | Runs in memory and is ephemeral |
| Execution | Not executed directly. | Not executed directly. | Executed to run applications. |
| Layers | NA | Composed of multiple layers from docker file. | NA |

* **How to create a docker image:**
* To create a docker image we need to create a docker file.
* Docker file name should be unique (first letter should be capital ‘D’).
* To write a docker file, we should know the set of instructions.
* **Docker file instructions:**
* File name should be capital letter.
* This file is usually present in project root directory.
* Some important commands:
* From: used to specify the base image (from where we are pulling the image)
* Label: to add labels, labels are like meta information of the image. (detail information of the image).
* Run: used to run the commands on docker images.
* Workdir: it makes the directory as current working directory, if the folder is not available then it will automatically create the folder.
* Expose: to open a port on the container. (Maps the container to a port.)
* Cmd: used for starting the application.
* Entry point: like the CMD command.
* Copy: to copy file from local host to docker image and source can only be local file system.
* Add: adding a file to the docker image and source can be local file system or remote URL.
* **Docker home Directory:**
* /var/lib /docker.
* **Docker commands:**
* Docker –-version – to see the version of docker.
* If config –a –configure and display network interfaces on Linux systems.
* Docker –o – it is bridge network.
* Docker info- to see complete information if docker.
* Docker images- to see how many docker images are there.
* Docker image ls- same a docker image command.
* Docker ps – to see number of docker containers.
* Note: display only the current running containers.
* Docker ps –a – it shows all running/ not running docker containers.
* Docker pull <image name>: the command is used to download a docker image from a docker registry like docker hub to your local machine.
* Docker inspect image ID: it will display the pulled image.
* Docker run –itd –name bhavya –p 40:80: image id- used to create docker container.
* Docker ps – to see the number of containers.
* **We take backup we need to create tag first and the wee need to push it.**
* How to create a tag:
* Docker tag<image name>: latest docker hub <username/dev:jenkins>
* **Docker installation**
* Login into AWS account.
* Launch one new ec2 instance with name as Docker\_instance.
* Open git bash and connect the bash and ec2 instance.
* Execute “Sudo su” command.
* Install docker on Linux server “yum install –y docker”
* To see the version of docker “docker --version”
* Linux systems to display network interface configuration “ifconfig -a”
* To start the service of docker “service docker start”
* Too see the information of docker “docker info”
* Move to the docker directory “cd /var/lib/docker”
* Give “ls”
* Move to the ec2-user “cd/home/ec2-user”
* To see the docker images “docker images”
* To see all the containers “docker ps –a".
* **How to pull image from the official account.**
* **First, we create repository in docker hub.**
* Open docker hub account
* create one repository with name dev
* Click on Create
* repository created.
* **Steps**: **connecting instance with gitbash and execute some comands before pulling started.**
* create one ec2 instance
* connect with gitbash by using .pem key.
* Execute sudo su command
* cd *var/*lib/docker. (To move the docker directory.)
* Ls
* docker images (it shows all docker images)
* docker ps (it shows all the containers)
* **steps: pull docker images from official website:**
* docker pull nginx (nginx image downloaded)
* docker images (it shows all information about the image, repository,tag,image id and size).
* docker inspect <imageid> (it shows commands about image)
* docker run -itd –name <container name> -p 40:80 <image id> (create container>
* docker ps (it shows container information containerid,imageid, containername,port number)
* docker login (give docker hub username and password)
* docker tag nginx:latest <docker hub username/repositoryname:jen (it creates the tag)
* docker push dockerhubusername/repositoryname:jen.