**Docker**

**What is Docker ?**

Docker is an open source platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker’s methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

**What is a container?**

Containers are software that wrap up all the parts of a code and all its dependencies into a single deployable unit that can be used on different systems and servers.

Multiple isolated containers can be launched together to form microservices which can be easily managed using any orchestration tool e.g., docker swarm, Kubernetes etc.

|  |  |
| --- | --- |
| **Docker** | **Virtual machine** |
| Contains only the bare minimum parts of the O.S required to run the software updates are easy and simple to do. | Contains the complete O.S that is normally used on systems for general purpose updates are time consuming and tough. |
| The isolation provided by a container is not complete as of a VM but is adequate. | VM provides complete isolation from concerning host system and is also more secure. |
| Containers are way more efficient as they only utilize the most necessary parts of O.s they act like any other software on the host system. | VM are less efficient as they have to manage full blown guest O.S VM have to access host resources through hypervisor. |
| Containers are self-contained environments that can easily be used on different O. S | Vm are not that easily ported with the same setting from one operating system to another |
| Containers are very easy to scale they can be easily added and remove based on requirements due to their light weight. | Vm are not very easily scalable as they are very heavy in nature. |
| Containers can be easily deployed easily using the docker CLI or making use of cloud services provided by aws or azure. | VMs can be deployed by using the powershellor by using the VMM or using cloud services such as aws or azure. |

**Use for:**

* Microservices
* Consistent development environment

**Microservices:**

Microservice architecture – a variant of the service-oriented architecture structural style – arranges an application as a collection of loosely-coupled services. In a microservices architecture, services are fine-grained and the protocols are lightweight.

**Docker environment :**

* Docker engine
* docker objects
* docker registry
* docker compose
* docker swarm

1. **Docker engine:**

Docker engine is as the name suggests , its technology that allows for the creation and management of all the docker process. It has three major parts to it.

* Docker CLI
* Docker Api
* Docker demon

1. **Docker Objects:**

* Docker images
* Docker container
* Docker volume
* Docker network
* Docker swarm nodes & services.
  + - * 1. **Docker images**

Docker images are set of instruction that are used to create containers and execute code inside it**.**

* + - * 1. **Docker volume**

Docker volume are basically persistent storage locations for the containers. They can be easily &safely attached and remove from different container. And they are also portable from system to another.

**Volume drivers**

Docker volumes drivers allow you to perform unique abilities such as creating persistent storage on other host, cloud , encrypt volumes. They basically enhance the abilities of a volume.

* + - * 1. **Docker network**

A docker network is basically a connection between one or more containers.one of the more powerful things about the docker containers is that they can be easily connected to one other and even other software, this makes it very easy to isolate and manage the containers.

1. **Docker registry:**

You can think of registries as storage location for docker images. These images can be versioned in the registry as well.

**Docker hub:**

You have many options for a docker registry, you can go with docker Hub as your main docker registry as there is already a docker command to pull and push images to it. if you don’t want to use docker hub there are many alternatives to it.

1. **docker compose:**

docker compose is just a service within docker that lets us launch multiple containers at the same time.

1. **docker swarm:**

docker swarm is a service within docker that allows us to manage multiple containers.

**Docker File:**

Docker files basically scripts that you can write and then build into an image. The image can then can be run to create the container. It’s like a shell script.

**Docker file format:**

Syntax: From <base image>

Syntax: ADD <source > <destination>

Syntax: Copy <source > <destination>

Syntax: Run <Command>

Syntax: workdir <directory>

Syntax: CMD <Commands>

Syntax: Volume <path>

Syntax: Expose <ports>

Syntax: Entry point <command> <parameter 1 > <parameter 2>

Syntax: Label <key>=<value>

**Commands:**

* docker -v , docker --version // for check version
* docker , docker help , docker --help // for help docker commands , list command
* docker info // detailed about docker version system
* docker pull <image name> // for pull the image
* docker images , docker image ls // show the list of docker images
* docker run -it -d -- name -p (port) <image name> // run image in docker container
* docker ps // shows the container // + -a // for all
* docker exec -ti <container name> bash // for going inside container
* docker stop <container id> // for stop container
* docker kill <container id> // for forcefully stop
* docker restart // restart the container
* docker rm // remove the container // -f //for forcefully
* docker commit <C ID > custom-image // for commit the container setting
* docker tag <old image name > <dockerhub username/new image name> // rename the image
* docker login // login docker --- //connect docker hub And engine
* docker push <username/image name > // push the images in docker hub
* docker rmi <images id or name > // remove the images
* docker build // creating images
* docker network create --driver <driver name> <network name> // creating new network
* docker network inspect <network id> // for detail network
* docker network ls // list of network
* docker stat // show the stats of the containers
* docker log // for info
* docker inspect // for info
* docker image ls // show all images
* Docker container ls // show the running container
* docker container ls -a // show all also stop container

|  |  |
| --- | --- |
| * docker volume create my-vol // | Create your Volume for Persistent Data |
| * docker volumes ls // | List down the volumes |
| * docker volume inspect my-vol // | inspect the volumes |
| * docker volumes rm my-vol // | remove volume |

* docker system df // check sizes
* docker volumes prune // remove all unused local volume

* docker rm $(docker ps -aq) // remove all stoped container