**Docker Points**

**1 . What is Docker?**

Docker is a virtualization S/w that make development nd deployment of applications easy.

* Docker does this by packaging application with all the necessary dependencies , configurations, tools and runtimwe environment
* (Standerdized unit that has everything the application needs to run)

**2. What are Docker Images?**

Docker images are light weight ,standardalone and executable package that includes everything needed to run a piece of s/w , including source code, runtime libraries , environment variables and configuration file.

* Basically docker images are blue-prient for creating Docker Containers.

**3. What are Docker Containers?**

Docker Conatiners are the run time instances of a Docker images.

* It encapsulates an application and its dependencies ,providing a consistent env to run the application.
* From one image we can run multipal containers

How to install Docker?

What is docker Engine?

What is Docker Client?

**List of Docker Commands**

**Basic Commands**

* **docker --version**: Show the Docker version installed.
* **docker info**: Display system-wide information about Docker.

**Image Commands**

* **docker images**: List all images on the local machine.
* **docker pull <image>**: Download an image from a registry.
* **docker build <path>**: Build an image from a Dockerfile.
* **docker tag <image> <new\_image>**: Tag an image with a new name.
* **docker rmi <image>**: Remove an image.
* **docker history <image>**: Show the history of an image.
* **docker save -o <file> <image>**: Save an image to a tar archive.
* **docker load -i <file>**: Load an image from a tar archive.
* **docker commit <container> <new\_image>**: Create a new image from a container’s changes.

**Container Commands**

* **docker ps**: List running containers.
* **docker ps -a**: List all containers.
* **docker run <image>**: Create and start a new container from an image.
* **docker start <container>**: Start a stopped container.
* **docker stop <container>**: Stop a running container.
* **docker restart <container>**: Restart a running container.
* **docker pause <container>**: Pause all processes in a container.
* **docker unpause <container>**: Unpause a paused container.
* **docker rm <container>**: Remove a container.
* **docker kill <container>**: Kill a running container.
* **docker exec -it <container> <command>**: Run a command in a running container.
* **docker attach <container>**: Attach to a running container.

**Volume Commands**

* **docker volume create <volume>**: Create a new volume.
* **docker volume ls**: List all volumes.
* **docker volume inspect <volume>**: Display detailed information about a volume.
* **docker volume rm <volume>**: Remove a volume.
* **docker volume prune**: Remove all unused volumes.

**Network Commands**

* **docker network create <network>**: Create a new network.
* **docker network ls**: List all networks.
* **docker network inspect <network>**: Display detailed information about a network.
* **docker network rm <network>**: Remove a network.
* **docker network connect <network> <container>**: Connect a container to a network.
* **docker network disconnect <network> <container>**: Disconnect a container from a network.

**Docker Compose Commands**

* **docker-compose up**: Build, create, start, and attach to containers for a service.
* **docker-compose down**: Stop and remove containers, networks, images, and volumes.
* **docker-compose build**: Build or rebuild services.
* **docker-compose ps**: List containers.
* **docker-compose start**: Start existing containers.
* **docker-compose stop**: Stop running containers.
* **docker-compose restart**: Restart running containers.
* **docker-compose logs**: View output from containers.
* **docker-compose exec <service> <command>**: Execute a command in a running container.
* **docker-compose pull**: Pull service images.

**System Commands**

* **docker system df**: Show docker disk usage.
* **docker system prune**: Remove all unused data (containers, networks, images, volumes).
* **docker system events**: Get real-time events from the server.

**Docker Rigistries:** Location where ready docker images are available are called Docker rigistries.

* Docker hosts one of the biggest docker rigistries called “**Docker Hub”.**
* Docker tags are used to identify images by name**.**

**Port Binding:** Bind the container ports to the host port to make the service available to the outside world.

* **Container port v/s Host port:**Application inside container runs in an isolated Docker network.
* We need to expose the container port to Host port(the machine containers runs on)
* Only one service can run on specific port on host.
* Standard to use the same port on host as container is using.

Syntax for port binding:

**docker run –d –p <host port>:<container port> <image name**

**Public and private docker rigisteries:**

**Difference b/w registries and repositories**

**Dockerfile : Creating own Docker images**

Dockerfile build instructions:

run

build

Dockerfile Docker image Docker container

**Dockerfile**: we need to create a “definition” of how to build an image from our application is called Dockerfile.

* To create our awn images 1st we need to create Dockerfile .
* Dockerfile is a text document that contains commands to assemble an iamge.
* Docker can then build an image by reading those instructions.

**Cmd to build image from definition**

**docker build –t <name \_to\_image:version> .**

where (.) represent the location of Docker file ,i.e in this case dockerfile is present in current working directory that’s why .

Above cmd is to build images from dockerfile. Againif we want to create Docker containers from those images use cmd.

**docker run –d –p <host port>:<container port> <image name:version>**