Docker

What is a Virtual machine?

A virtual machine (VM) is an emulation of a computer system that runs on another computer system.Essentially, a virtual machine allows you to run one operating system on top of another operating system.

Virtual Machine	Docker
Virtual machines provide a complete virtualized environment, including a virtualized operating system, virtualized hardware, and virtualized network devices.	Docker, on the other hand, uses a different approach. Instead of creating a full virtualized environment, Docker containers provide isolated environments for applications that share the host operating system kernel.
Each virtual machine is an isolated environment that runs its own operating system, and applications running in one virtual machine are separate from applications running in another virtual machine.	Docker containers are more lightweight than virtual machines, as they don't require a full copy of the operating system for each container.
Virtual machines typically allocate a fixed amount of resources (such as CPU, memory, and storage) to each virtual machine, which can lead to resource waste if the virtual machine isn't using all of its allocated resources.	Docker containers, on the other hand, can share resources more efficiently, as they are designed to be lightweight and can run in the same host operating system as other containers.

What is docker container?

- They are the lightweight alternatives of VM
- no need to allocate resources, docker will auto allocate based on need
- They are the runtime instances of docker image

what is docker image?

It is a read only template to create a container built by docker user and stored in docker hub.

What is docker hub?

Repository of docker images (public and private),

What is docker registry?

Storage component of docker images

This is same as dockerhub

here, one container may need multiple images.

What is Docker's History?

Indiana university >> they started docker. >> came up with concept of microservices

What is Docker tool?

It is designed to create, deploy and run the container easily

Port mapping

- It is used to access the services running inside a docker container
- we open a host port to give an access to corresponding container host inside a container
- all the request made to host port can be directed to container

What is Docker volume?

- Docker volume is used to retain the data on the host across the lifetime of a container.

Volume maping

Volume mapping in Docker allows you to persist data generated by a Docker container, even after the container is deleted. This is useful when you want to store data generated by a container, such as log files or database files, which can be useful for backup and restore operations.so that the data is available even if the container is recreated.

Volume mapping is achieved by specifying a mapping between a directory on the host file system and a directory in the container when you start a Docker container. The host directory is known as the "mount point", and the container directory is known as the "mount target".

What is Micro services?

It is a software development architectural style that structures application as a collection of loosely coupled service.

Ex: Flipkart

What is Monolithic style of development?

It is a software development architectural style where single component change affects whole application

Ex: SBI

Docker architecture:

Docker uses a client-server architecture to manage containers. The Docker client communicates with the Docker daemon, which does the heavy lifting of building, running, and managing containers.

The main components of the Docker architecture are:

1. **Docker Client:** This is the command-line tool that interacts with the Docker daemon and allows you to build, run, and manage containers.
2. **Docker Daemon:** This is the background process that runs on the host machine and manages containers, images, networks, and volumes.
3. **Docker Registries:** These are the centralized locations where Docker images are stored and distributed to clients. The default registry is Docker Hub, but you can also use private or local registries.
4. **Docker Images:** These are the blueprint for containers, and they are created using a Dockerfile.
5. **Docker Containers**: These are instances of Docker images that run as isolated processes on the host machine. Containers can share the host machine's resources, such as CPU, memory, and storage, but they are isolated from each other and from the host operating system.
6. **Docker Networks:** These allow containers to communicate with each other and with the host machine.
7. **Docker Volumes:** These are persistent storage areas that can be mounted into containers to persist data even if the container is deleted.

In summary, the Docker architecture is based on the client-server model, where the Docker client communicates with the Docker daemon to manage containers and their associated resources, such as images, networks, and volumes.

Docker commands

What is the command to start/stop/ know the status of docker?

`sudo docker service start`

`sudo docker service stop`

`sudo docker service stop`

What is the command to see the list of images we have pulled?

`docker image ls`

What is the command to pull a particular image? ex. ngnix (this ofcourse should be referred to the dockerhub website)

`docker pull nginx`

What is the command to run a command in a docker container? ex. centos

`docker run -it centos`

▼ **More information**

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

d, --detach : Run container in background and print container ID

e, --env list Set environment variables

i, --interactive Keep STDIN open even if not attached

> 💡 stdin – It stands for standard input, and is used for **taking text as an input.**
>
> stdout – It stands for standard output, and is used to text output of any command you type in the terminal, and then that output is stored in the stdout stream

t, --tty Allocate a pseudo-TTY

💡 What is a TTY in docker? **The -t (or --tty) flag tells Docker to allocate a virtual terminal session within the container**
. This is commonly used with the -i (or --interactive) option, which keeps STDIN open even if running in detached mode

v, --volume list Bind mount a volume

```
sanjay@sanjay-HP-Pavilion-Notebook:~$ docker run -it centos
[root@e8c537ee971c /]#
```

What is the command to display list of running containers?

`docker ps`

What is the command to display list of all containers?

`docker ps -a`

What is the command to start a container? ex: id : 0b4002a108dd

`docker start 0b4002a108dd`

Can also use to start multiple containers : docker start id1 id2 id3

What is the command to run a command in a running container? (ex. id “0b4002a108dd” command: open the bash terminal)

`docker exec -it 0b4002a108dd /bin/bash`

▼ **More information**

Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

Run a command in a running container

Options:

-d, --detach	Detached mode: run command in the background
--detach-keys string	Override the key sequence for detaching a container
-e, --env list	**Set environment variables**
--env-file list	Read in a file of environment variables
-i, --interactive	**Keep STDIN open even if not attached**
--privileged	Give extended privileges to the command

-t, --tty	**Allocate a pseudo-TTY**
-u, --user string	Username or UID (format: <name|uid>[:<group|gid>])
-w, --workdir string	Working directory inside the container

What is the command to create a docker volume? ex. docker_volume_file

`docker volume create docker_volume_file`

This is not a file which we can find using ls / ls -a

▼ More information on what is volume mounting.

Mounting a Volume Inside Docker Container - GeeksforGeeks

When you are working on a micro-service architecture using Docker Containers, you create multiple Docker Containers to create and test different components of your application. Now, some of those

https://www.geeksforgeeks.org/mounting-a-volume-inside-docker-container/?ref=rp

what is the command to display all the existing Docker Volumes?

`docker volume ls`

what is the command to volume map a container (image: centos) with host path /home/ubuntu and continer path /apps and also give a name to the container as cloudtechnology

`sudo docker run -itd --name cloud technology -v /home/ubuntu:/apps centos`

What is the command to enter into a bash shell of a container with id 7c248c5c67

`sudo docker exec -it 7c248c5c67 /bin/bash`

What is the command to Remove unused data of docker images?

`sudo docker system prune`

What is the command to *Build an image from a Dockerfile?*

`sudo docker build . -it dockerfile`

-t, --tag list Name and optionally a tag in the 'name:tag' format

💡 The dockerfile must be named exactly as dockerfile. Nothing else.

A Dockerfile is a text file that contains instructions for building a Docker image. The file is used by the `docker build` command to create an image that can be run as a container.

A Dockerfile typically starts with a base image, such as an operating system image or a language runtime, and then adds additional files, configures the environment, and sets up the application. The instructions in a Dockerfile are executed in the order they appear, and each instruction creates a new layer in the image.

```
# Use an official Node.js runtime as the base image
FROM node:14-alpine

# Set the working directory in the container
WORKDIR /app

# Copy the application code to the container
COPY . .

# Install the required packages
RUN npm install

# Expose the application port
EXPOSE 3000

# Define the command to run when the container starts
CMD ["npm", "start"]
```

In this example, the Dockerfile starts with a base image of Node.js 14 and sets the working directory in the container to `/app`. It then copies the application code into the container, installs the required npm packages, exposes the application port, and defines the command to run when the container starts.

Once you have created the Dockerfile, you can build the image using the `docker build` command:

```
docker build -t myapp .
```

This command builds an image named `myapp` using the current directory as the build context. You can then run a container from the image using the `docker run` command:

```
docker run -p 3000:3000 myapp
```

This command maps port 3000 on the host to port 3000 in the container and runs the `myapp` image as a container. The application should now be accessible at `http://localhost:3000`.

What is the command to remove one or more images from Docker? ex. container id 5d0da3dc9764

`sudo docker rmi 5d0da3dc9764`

```
Usage:  docker rmi [OPTIONS] IMAGE [IMAGE...]

Remove one or more images

Options:
  -f, --force      Force removal of the image
      --no-prune   Do not delete untagged parents
```

What is docker compose?

Docker Compose is a tool for defining and running multi-container Docker applications. It allows you to define the services, networks, and volumes needed to run your application in a single file called a `docker-compose.yml` file. With Docker Compose, you can start and stop all the services defined in your `docker-compose.yml` file with a single command

What should be the name of a docker compose file?

docker-compose.yml

A `docker-compose.yml` file is a YAML file that defines your application's services, networks, and volumes. Here's an example `docker-compose.yml` file for a simple web application:

```
version: '3'
services:
  web:
    image: nginx
    ports:
      - "80:80"
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: password
```

In this example, the `docker-compose.yml` file defines two services: a `web` service that runs the `nginx` image, and a `db` service that runs the `postgres` image. The `web` service maps port 80 on the host to port 80 in the container, and the `db` service sets the `POSTGRES_PASSWORD` environment variable.

What is the command to start a docker-compose file?

```
docker-compose -up -d
```

-d : to start in a detached mode

Starts the services defined in the `docker-compose.yml` file.

What is the command to scale docker container in docker-compose?

```
docker-compose scale SERVICE=3
```

What is the command to stop the docker-compose?

```
docker compose down
```

Stops and removes the containers, networks, and volumes created by `docker-compose up`

What is the command to stop the docker-compose and erase the data?

```
docker compose down --volume
```

Other commands:

`docker-compose logs` : Shows the logs for the containers created by the `docker-compose.yml` file.

`docker-compose exec`: Runs a command in a running container created by the `docker-compose.yml` file.

`docker-compose build`: Builds or rebuilds the images defined in the `docker-compose.yml` file.

`docker-compose ps`: lists the containers created by the `docker-compose.yml` file, along with their status and ports.

`docker-compose config`: validates the `docker-compose.yml` file and displays the configuration that would be used.

`docker-compose stop`: stops the services defined in the `docker-compose.yml` file.

`docker-compose restart`: restarts the services defined in the `docker-compose.yml` file.

What is docker swarm

Docker Swarm is a native clustering and orchestration solution for Docker containers. It allows you to create and manage a swarm of Docker nodes, and deploy services to the swarm that are made up of one or more containers.

Uses of docker swarm:

- High availability: By distributing services across multiple nodes, Docker Swarm provides high availability and automatically replaces failed containers with healthy ones.
- Load balancing: Docker Swarm automatically balances the load of incoming requests across the containers in a service.
- Scalability: Docker Swarm makes it easy to scale a service up or down by adding or removing containers.
- Management: Docker Swarm provides a unified management interface for all the nodes in a swarm, making it easier to manage a large-scale deployment.

Commands:

- `docker swarm init`: Initializes a Docker node as a swarm manager.
- `docker swarm join`: Joins a Docker node to a swarm as a worker.
- `docker service create`: Creates a new service in the swarm.
- `docker service ls`: Lists all services in the swarm.
- `docker node ls`: Lists all nodes in the swarm.
- `docker service inspect`: Provides detailed information about a service in the swarm.
- `docker service update`: Updates the configuration of a service in the swarm.
- `docker service rm`: Removes a service from the swarm.
- `docker service scale`: Scales a service in the swarm up or down.
- `docker service update`: Updates a service in the swarm with new configuration parameters.