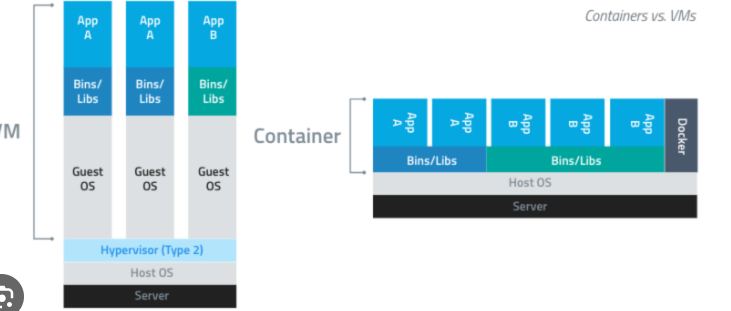
What is docker :



Docker is a conterization platform for developing ,packaging,shipping ,and running applications .

It provides the ability to run an application in an isolated environment called container .

Makes deployment and development efficient .

What is a container :

A way to package an application with all the necessary dependencies and configurations

It can be easily shared

Makes deployment and development efficient

|  |
| --- |
| Applications run |
| Container with libereies ,dependencies,tools |
| Docker engine |
| Operating system {windows} {linux} {mac} |
| Hardware aws |

Docker vs virtual machines

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Low impact on os ,very fast ,low disk space usage . | High impact on os ,slower,high disk space usage |  |  |  |
| Sharing ,rebuilding and distribution is easy | Sharing ,rebuilding,and distribution is challenging |  |  |  |
| Encapsulated apps instead of whole machine | Encapsulated whole machine |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Main components of docker :

Docker file

Docker image

Docker container

Docker registry : A docker registry is a central repository for storing and distributing docker images

What is Docker, and what problem does it solve in software development?

Docker is a open source platform that allows developers to automate the deployment ,scaling , and management of applications using containerization technology .

It provides a method to package an application and its dependencies into standardized unit called containers .

Consistency across environments

Dependency management

Isolation

Portability

Scalability

Rapid deployment

Versioning and rollback

Continuous Integration and Continuous Deployment (CI/CD): Docker is widely used in CI/CD pipelines to automate the build, test, and deployment process. It enables developers to release code faster and more frequently, promoting a more agile development workflow.

Explain the difference between Docker containers and virtual machines.

Docker containers and virtual machines (VMs) are both technologies used for virtualization, but they differ in their approach, architecture, and resource usage. Here are the main differences between Docker containers and virtual machines:

1. \*\*Architecture:\*\*

- Docker Containers: Containers run on top of a single host operating system (OS) and share the host OS's kernel. Each container contains the application, its dependencies, and the necessary runtime libraries. Containers isolate the application processes from the host and other containers using Linux namespaces and control groups (cgroups).

- Virtual Machines: VMs, on the other hand, require a hypervisor (like VMware, VirtualBox, or Hyper-V) to create and manage multiple virtualized environments on a physical server. Each VM includes a separate OS kernel, full OS, application, and its dependencies, making VMs more resource-intensive than containers.

2. \*\*Resource Utilization:\*\*

- Docker Containers: Containers are lightweight and use a minimal amount of resources because they share the host OS's kernel. They only require the resources needed to run the application and its dependencies, which makes them more efficient in terms of resource utilization.

- Virtual Machines: VMs are heavier since they run multiple complete OS instances, each with its kernel and associated overhead. This can result in higher resource consumption, leading to a more significant overhead compared to containers.

3. \*\*Boot Time:\*\*

- Docker Containers: Containers can start almost instantly, usually in milliseconds, because they don't need to boot an entire OS. They use the host OS's kernel directly, which speeds up the initialization process.

- Virtual Machines: VMs generally take longer to start since they need to boot up a full OS, which can take several seconds or even minutes depending on the OS and hypervisor.

4. \*\*Isolation:\*\*

- Docker Containers: Containers provide process-level isolation, meaning each container runs in its isolated environment, separate from other containers on the same host. However, containers share the host OS's kernel, which may result in less isolation compared to VMs.

- Virtual Machines: VMs offer stronger isolation because each VM runs on its own separate OS and has its kernel. This isolation makes VMs more secure but comes with a performance trade-off.

5. \*\*Portability:\*\*

- Docker Containers: Containers are highly portable due to their lightweight nature. Docker containers can be easily moved and executed on different environments that support Docker, providing consistency and eliminating the "it works on my machine" problem.

- Virtual Machines: While VMs can be moved between different hypervisors and cloud platforms, they are generally less portable than Docker containers due to the dependencies on the hypervisor and the larger size of VM images.

In summary, Docker containers are more lightweight, efficient, and faster to start compared to virtual machines. They excel at deploying microservices and applications that require fast scaling, while virtual machines are well-suited for scenarios requiring stronger isolation and support for different operating systems within the same host. The choice between containers and VMs depends on the specific use case and the balance between isolation and resource efficiency needed for the application.

Docker file > docker image > docker container > access the app

Step 1 : create a new directory

Mkdir myapp

Cd myapp

Step 2 :create a file called index.html echo “hello world” > index.html

Step 3 : create a file called docker file **touch dockerfile----docker file contains instructions to create a docker image**

**When we run docker file we will get docker image**

**When we run the docker image docker containers are created.**

Step 4 : open the docker file in text editor and add the fallowing lines

From ngnix

Copy indexhtml /usr/share/nginx/html

Step 5 : start docker and build docker image from the docker file

Docker build –t myapp

This command builds a new docker image with the tag myapp using the dockerfile in the current directory

Step 6 :run docker container from the image

Docker run -p 8080:80 myapp

This command tells docker to run myapp container and map port 8080 on your local machine to port 80 inside the container

Access the app :open the browser <https://localhost:8080to> see the hello world message displayed on your browser .

**How do you share docker image with others**

Inoreder to download package in docker

Docker pull nginx

To check docler images created are not

Docker images

Inorder to create container from the image

Docker create –name demo nginx

**How do you create a container from the docker image**

**docker pull nginx** ----command to download the ngnix applications

**docker run -d -p 80:80 --name my\_nginx nginx**------to start a new container from the image

**docker run -it --name my\_container ubuntu:latest /bin/bash**

n this example, the container will be created from the **ubuntu** image, and it will run an interactive shell (**/bin/bash**) inside the container. The **-it** flags allocate a pseudo-TTY and allow interactive access to the shell.

Top commands used in docker :

Docker –version

Docker login

Docker search

Docker pull

Docker inspect

Docker create

1. What is Docker, and what problem does it solve in software development?
2. Explain the difference between Docker containers and virtual machines.
3. How do you install Docker on different operating systems?
4. What is a Docker image, and how is it different from a container?
5. How do you create a Docker image?
6. How do you share Docker images with others?
7. How do you run a Docker container from an image?
8. Explain the role of a Dockerfile in Docker.
9. What are Docker volumes, and why are they important?
10. How do you manage networking in Docker?
11. What is the purpose of Docker Compose, and how does it work?
12. How do you scale Docker containers?
13. Can you explain the concept of Docker swarm mode and its benefits?
14. What is the difference between Kubernetes and Docker?
15. How do you update a Docker container?
16. How can you ensure security in Docker containers?
17. What are the best practices for optimizing Docker images and containers?
18. How do you debug issues in Docker containers?
19. What is Docker Hub, and how do you use it?
20. Can you explain Docker's layering and how it impacts performance?

What is docker image or container image

Docker image is a read only template that consists a set of instructions for creating a container that can run on the docker platform .

It provides a convieneient way to package up applications and preconfigured server environments .

Hub.docker.com

Inorer to add data to container we will use writable layer but this writable layer when you stop and start the container the data may loss

On top of the image during run time docker add the writable layer

To download the image we can use docker pull mysql

To check the images docker images

Docker create imageID

Docker ps –a ----docker details running or stopped

Docker start containerid

Inorder to interact with the container docker exec it -- /bin/bash

Docker networking

Bridge networking :default network used to communicate between the containers .we can use the same ip address but ip address

Host networking :

None networking :disables the network for the container

# Use the official Node.js base image

FROM node:14

# Set a custom environment variable

ENV MY\_APP\_ENV=production

# Set the working directory inside the container

WORKDIR /app

# Copy the application code into the container

COPY . /app

# Install dependencies (if you have any)

RUN npm install

# Set the command to start your Node.js application

CMD ["node", "app.js"]