Containers and docker

Containers – Lightweight portable units that holds everything for running applications.

Infrastructure – Represents physical/virtual hardware where everything runs (e.g physical machine or cloud server)

Host operating system - Manages hardware and provides the kernel which container use. (e.g windows server, linux)

Docker engine - Provides environment to build, run and manage docker containers.

Benefits of container:

* Portability – Bundles app and dependencies in one package which can be run anywhere.
* Lightweight and fast – Shares host OS kernel, so no need for full OS per app.
* Efficient storage use - Can run many containers on one server without wasting resources.

Docker – Open-source platform for building, running and managing containers.

Docker hub – Public library of pre-built images. Cloud service for sharing applications and automating workflows.

Images – A read-only snapshot built from a docker file.

Docker file – A text file containing instructions for building an image.

Dockerfile → build → Image → run → Container

Benefits of Docker

1. Simplified deployment
2. Improved efficiency
3. Enhanced collaboration

Containers vs Virtual machines (VMs)

Hypervisor in VMs create and manage virtual machines by allocating resources like CPU, memory and storage.

Each VM needs to boot up a full guest OS, which can take minutes, much longer than containers. It also uses significantly more resources than containers do.

VM provides a strong isolation with each VM having its own OS. Containers offer process level isolation as they share the same host OS kernel, but they are isolated within the container itself.

VMs are less portable due to size and dependency on specific hypervisors.

docker ps = Shows running containers

docker ps -a = Shows all containers, including stopped ones

Dockerfile commands

FROM = Specifies the base image to use for the docker image.

RUN = Executes commands in the container.

COPY = Copies files from the host machine into the container.

WORKDIR = Sets the working directory for the subsequent instructions.

CMD = Specifies the command to run when the container starts.

Docker networking – How container communicate

Bridge network = Private internal network for containers on the same machine. Containers can communicate using their own IP addresses.

Host network = Container uses the host machines network directly without any isolation. Useful for applications that need to directly interact with the host system. Fast but less secure.

None network = Gives container no network interface at all. Used to ensure container has no network access at all. Hight security.

Linking containers together

1. Create a network
2. Run a database container
3. Run a web container

.

Docker compose – Tool that helps you define run multi-container applications together.

Benefits of docker compose:

* Makes development and testing easier as everything is in one file
* Ensures consistency between devs
* Enhances teamwork

docker-compose up = Command that reads the docker-compose.yml file, builds any necessary images and starts all the services you’ve defined.

docker-compose up -d = Runs containers in the background

docker-compose down = stops the containers.

docker-compose logs = View logs

Docker registries – Library for storing and sharing docker images.

Public registry – Open to everyone e.g docker hub

Private registry – Secure and restricted for yourself. E.g amazon ECR

Benefts of docker registries

* Streamline deployment – Can easily be accessed and deployed in different environments
* Enhance collaboration – Everyone on team can access it
* Ensure consistency – Eliminates ‘it works on my machine’ problem

Docker commands:

docker images = Lists all images

docker ps = Shows what containers are running

docker rm = removes containers

docker inspect <image-id> = Gives detail about image

docker rmi <image-id> = Removes images

docker system prune = Removes all unused docker images

docker exec = Runs a command inside a running container.

Kubernetes – Open-sourced platform designed to automate the deployment, scaling and management of containerised applications.

Docker swarm vs Kubernetes

Docker swarm = Easy to set up and use. No auto-scaling. Easy to start a cluster, limited to docker’s API capabilities.

Kubernetes = Complex, feature-rich. Auto scaling, difficult to start cluster, not limited to docker’s API capabilities.

Benefits of orchestration tools

* Manage large-scale deployment
* Ensure high availability – can recover from failures
* Automate scaling and recovery – Scale up or down depending on demand
* Simple
* Enhance reliability
* Resource utilisation