



# Containerization

## What is Containerization?

- Containerization = a lightweight form of virtualization that packages an application and its dependencies together.
- Containerization simplifies deployment, scaling, and management of applications.
- Use cases: microservices, CI/CD pipelines, and development environment consistency.

## Introduction to Docker

### What is Docker?

- A tool designed to make it easier to create, deploy, and run applications by using containers.
- Key components: Docker Engine, Dockerfile, Docker Compose, and Docker Hub.

### Docker Engine

- Docker Engine is the heart of Docker. It's a lightweight runtime and toolkit that allows you to build and run containers.
  - Client-server architecture with the Docker daemon and client CLI.
  - REST API for interaction and how containers are run on the host operating system.
- It handles the lifecycles of containers, from building to running and finally stopping them.

### Dockerfile

- A Dockerfile is a script containing a series of instructions for Docker to build an image. (i.e. a blueprint for Docker images)
- Dockerfiles automate the process of creating Docker images and ensure consistency.

- The main instructions you'll use are `FROM`, `RUN`, `COPY`, and `CMD` which define the container environment.

## Docker Compose

- Docker Compose allows you to define and run multi-container Docker applications. It's mainly useful for running an app locally.
- Uses a YAML file to configure application services and how it simplifies deployment.
- With a single command, you can create and start all services from your configuration.

## Docker Hub

- Docker Hub is a cloud-based registry to distribute and share Docker images.
- Teams can share container images, automate workflows, and more.
- Vast library of community and official images available for use.

# Using Docker in a production environment

- A common issue is that code behaves differently in different environments.
- The major cloud providers like AWS, Azure, and Google Cloud offer managed services for Docker containers, e.g., ECS, AKS, and GKE.
- Docker tags and registries help in maintaining versions.
- Use environment variables to manage config settings across local and cloud.

## Summary and Takeaways

- Docker is a critical tool for modern software architectures, offering a solution for environment discrepancies and facilitating cloud integrations.
- Understanding the basics of Docker can significantly improve your DevOps capabilities.
- Consistency between local and cloud environments is crucial and achievable with Docker.