DOCKER BASICS

STANDARDIZED UNITS OF SOFTWARE FOR RELIABLE, SCALABLE DEVELOPMENT.

**DOCKER ARCHITECTURE**

- Docker virtualizes the host operating system, allowing it to be shared between the containers; contrast this to virtual machines (VMs), which are based on hardware virtualization.
- Apps are isolated within their own environments to ensure portability & security.

**WHY CONTAINERS?**

- **APPLICATION ISOLATION**
  Encapsulate an entire runtime environment for easy deployment.
- **FASTER + LIGHTER**
  Unlike VMs, containers do not require one OS per application. Boot time is quick.
- **RESOURCE EFFICIENCY**
  Reduced cloud spending because a single server can host many containers.
- **MODULARITY**
  Applications can be split into microservices for independent management & development.

**WHY DOCKER?**

Docker has become synonymous with containerization, but the technology has actually been around for years. The early versions of Docker leveraged XLC ("for Linux Containers"); nowadays it uses custom functionality for improved portability & container management.

**WORKING WITH DOCKER**

Developers issue commands via a client CLI, which uses a REST API to interact with the Docker service daemon.

- Code is assembled into an image by the Docker Engine based on instructions contained in the Dockerfile.
- Images are blueprints for containers.
- Containers are live running instances of images.
- Docker Hub is a public repository that allows developers to access and share Docker images.