CLOUD, DEVOPS, AND CONTAINERS

Introduction to Docker and Containers

Level: Foundation • 2 days (expandable to 3) • Virtual, In-person

Overview

Containers changed how software ships: instead of "it works on my machine," an application and everything it needs travel together in one runnable, portable unit. Docker made that idea practical, and containers are now the default packaging for modern software, which means understanding them is no longer optional for anyone who builds, tests, or operates applications.

This is a hands-on, foundation course. It assumes no container experience and builds up in dependency order: what a container actually is and how it differs from a virtual machine, then running and managing containers, then building your own images well, then data and networking, and finally composing multi-container applications. Rather than survey the whole container ecosystem, it builds real fluency with Docker itself, the foundation every orchestration tool assumes. Every module includes a lab at the terminal, and each module builds on the one before.

Who Should Attend

- Developers who need to build and run their applications in containers
- Operations, QA, and support engineers who work with containerized systems
- Anyone preparing to learn Kubernetes or cloud-native development

Prerequisites

- Comfort at a command line: running commands and editing files
- A general idea of how applications run (processes, ports, files) helps
- No container or Docker experience required

What You Will Learn

- Explain what containers are, how they work, and how they differ from virtual machines
- Run, inspect, and manage containers confidently from the command line
- Build efficient images with well-written Dockerfiles
- Persist data with volumes and connect containers with networks
- Define and run multi-container applications with Docker Compose
- Push, pull, and tag images with registries, and judge image hygiene

Course Outline

Day one: running and building

- What a Container Is
  - The problem: "works on my machine," dependency conflicts, and slow environments
  - Containers versus virtual machines: isolation without the weight
  - Images, containers, and registries: the three nouns that matter
  - Lab: run your first containers and inspect what is actually happening

- Working with Containers
  - The container lifecycle: run, stop, restart, remove
  - Ports, environment variables, and logs
  - Getting inside: exec, and debugging a running container
  - Lab: run a real service in a container, configure it, and troubleshoot it
- Building Images
  - Dockerfiles: instructions, layers, and the build cache
  - Writing images that are small, fast, and rebuildable
  - Lab: write a Dockerfile for a sample application, then build and run it

Day two: data, networks, and real applications

- Data and Persistence
  - Why containers lose data, and when that is a feature
  - Volumes and bind mounts, and choosing between them
  - Lab: give a database container storage that survives it
- Container Networking
  - How containers reach each other and the outside world
  - User-defined networks and container DNS
  - Lab: connect an application container to its database by name
- Multi-Container Applications
  - Docker Compose: the whole application in one file
  - Registries: pushing, pulling, and tagging images properly
  - Where containers go next: orchestration and *Kubernetes Fundamentals*
  - Lab: define and run a complete multi-service application with Compose

Extended Version

The three-day version keeps the same gradient and adds:

- Multi-stage builds and image optimization in depth
- Container security basics: users, image scanning, and trusted sources
- Containers in CI: building and publishing images from a pipeline
- A capstone: containerize a realistic application end to end, from Dockerfile to a composed, persistent, networked stack