CLOUD, DEVOPS, AND CONTAINERS

Kubernetes Fundamentals

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

Overview

Kubernetes has become the operating system of the modern data center: the standard way to run containers at scale across every cloud and on premises. It is also famous for its learning curve, not because any one concept is hard, but because there are many concepts and most explanations present them in the wrong order.

This is a hands-on, practitioner course. It teaches Kubernetes in the order the concepts actually depend on each other: what problem the orchestrator solves, then pods, then the controllers that manage pods, then networking, then configuration and storage, and finally the health, scaling, and packaging practices that make workloads production-ready. Rather than survey every API resource, it builds deep fluency with the core set that covers most real work. Every module includes a lab against a live cluster, and each module builds on the one before.

Who Should Attend

- Developers who need to deploy and debug applications on Kubernetes
- Operations and DevOps engineers adopting container orchestration
- Architects who want hands-on grounding before designing for Kubernetes

  Learners who have never worked with containers should take *Introduction to Docker and Containers* first.

Prerequisites

- Working knowledge of containers and images, per *Introduction to Docker and Containers*
- Comfort at a command line
- Familiarity with YAML helps but is not required

What You Will Learn

- Explain what Kubernetes does and how the cluster's pieces fit together
- Deploy applications with pods, deployments, and rolling updates
- Expose applications inside and outside the cluster with services and ingress
- Configure applications with ConfigMaps and Secrets, and attach persistent storage
- Keep workloads healthy with probes, resource requests, and autoscaling
- Package and install applications with Helm

Course Outline

Day one: the core objects

- Why an Orchestrator
  - The problems containers create at scale, and how Kubernetes solves them
  - Cluster anatomy: control plane, nodes, and the API

    - Desired state: the one idea that explains everything else
    - Lab: explore a live cluster with kubectl and read what is running
- Pods and Deployments
    - Pods: the unit of scheduling, and why it is not just "a container"
    - Deployments and ReplicaSets: self-healing and rolling updates
    - Lab: deploy an application, scale it, break it, and watch Kubernetes heal it
- Services and Networking
    - Why pods need services: stable names for moving targets
    - ClusterIP, NodePort, LoadBalancer, and ingress
    - Lab: expose the application inside the cluster and to the outside world

Day two: production-ready workloads

- Configuration and Storage
    - ConfigMaps and Secrets: configuration out of the image
    - Volumes, persistent volumes, and claims
    - Lab: externalize the application's configuration and give it durable storage
- Health and Scaling
    - Liveness, readiness, and startup probes
    - Resource requests and limits, and what the scheduler does with them
    - Horizontal pod autoscaling
    - Lab: add probes and resource settings, then autoscale under load
- Packaging with Helm
    - Why raw YAML stops scaling, and what Helm adds
    - Charts, values, releases, and rollbacks
    - Where to go next: managed Kubernetes and *Deploying to Azure Kubernetes Service (AKS)*
    - Lab: package the application as a Helm chart and roll back a bad release

Extended Version

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

- Namespaces, RBAC, and the basics of securing a cluster
- Observability: logs, metrics, and debugging failing workloads systematically
- StatefulSets, jobs, and the workload types beyond deployments
- A capstone: deploy a complete multi-service application to Kubernetes with configuration, health checks, scaling, and Helm packaging