Designing CI/CD around Microservices and Containers

- Understand what containers are and why they are critical to microservices
- Explore container technologies and orchestration tools such as Docker and Kubernetes
- Understand how microservices and containers impact CI/CD pipelines
- Design pipelines to build, test, deploy, monitor microservice applications
- Build security and quality into CI/CD pipelines with automation
- Adapt team structure to succeed with microservices

As organizations switch from monolithic software architecture to microservices and infrastructure-as-code software architecture, they discover their dependence on their continuous integration and continuous deployment pipelines tends to dramatically increase. The use of container technologies such as Docker and Kubernetes enables creating decoupled services that can be independently built and deployed, but requires a high level of automation to properly build, test, and orchestrate their deployment.

In this course, we explore the impact microservices have on the CI/CD delivery processes. This includes a brief overview of microservices, what we need to be successful when building them, and some of their unique needs related to development, testing, and deployment. We also look at why microservices and containers are often associated with each other, how containerization enables smaller, independent services, and the importance of container orchestration systems such as Kubernetes in managing the deployment and configuration of these multi-service software systems in various test and production environments.

Succeeding with large, complicated microservice-based systems requires a high degree of automation to keep up with the constant rate of change. We will talk about CI/CD success patterns in designing and implementing the right build, test, and deployment steps to meet quality and security requirements. Microservices, by nature, create multiple independent pipelines that trigger at different rates but eventually come together into shared testing environments. They have unique requirements for testing that often involve testing with upstream and downstream dependencies to ensure that your entire enterprise system functions properly. All of this has to be managed in persistent test environments but can greatly be aided by the creation of on-demand, ephemeral test environments that can be created and destroyed when needed.

We tie up the course with a discussion of how proper team and organizational structure can help you to succeed with microservices by fostering a DevOps culture. We discuss different team structures that can enhance communication and improve success and prevent silos from forming between development teams, as well as between development, QA, and operations teams.

Who Should Attend

- Technical managers leading development teams
- Developers implementing microservices
- Test automation professionals testing microservices
- DevOps practitioners building CI/CD pipelines

Participants should be familiar with Continuous Integration and Continuous Delivery and have a conceptual knowledge of virtualization, containers, and microservices. Coveros recommends **Foundations of DevOps—ICAgile Certification** [1]for those outside of the above roles.

Laptops Required

This is a hands-on course. With their laptops, participants will have the opportunity to browse, construct, experiment

with, and orchestrate their own containers and CI/CD pipelines. Participants will get an AWS server instance running Kubernetes, Docker, and Jenkins to gain valuable experience on the hows and whys of containers and pipeline automation.

Course Outline

Intro to microservices
What they are
Impact on various aspects of dev, qa, ops processes

Containers as a platform
What are containers?
What are containers useful for?
Container orchestration systems

DevOps tooling and platform

Building CI/CD pipelines
Intro to pipelines CI, CD, C-monitoring
Tools and automation
Patterns of success (sprinkled throughout)

Testing is critical
You will fail without a lot of testing
Impact of microservices on testing
Test containers

Team and organization structure
Dev vs. QA vs. Ops
Prevent silos
Integrating DevOps across teams