Linux Foundation - Kubernetes for App Developers

Code:	LFD459
Length:	3 days
URL:	View Online

This course will teach you how to containerize, host, deploy, and configure an application in a multi-node cluster. Starting with a simple Python script, you will define application resources and use core primitives to build, monitor and troubleshoot scalable applications in Kubernetes. Working with network plugins, security and cloud storage, you will be exposed to many of the features needed to deploy an application in a production environment.

Skills Gained

In this course you'll learn how to:

- Containerize and deploy a new Python script
- Configure the deployment with ConfigMaps, Secrets and SecurityContexts
- Understand multi-container pod design
- Configure probes for pod health
- Update and roll back an application
- Implement services and NetworkPolicies
- Use PersistentVolumeClaims for state persistence
- And more

Prerequisites

To get the most out of this course, you should have basic Linux command line and file editing skills and be familiar with using a programming language (such as Python, Node.js, Go). A knowledge of Cloud Native application concepts and architectures (such as is taught in our free Introduction to Kubernetes edX MOOC) is helpful for this course.

Course Details

Course Outline

Introduction

- Objectives
- Who You Are
- The Linux Foundation
- Linux Foundation Training
- Preparing Your System
- Course Registration

- Labs

Kubernetes Architecture

- What Is Kubernetes?
- Components of Kubernetes
- Challenges
- The Borg Heritage
- Kubernetes Architecture
- Terminology
- Master Node
- Minion (Worker) Nodes
- Pods
- Services
- Controllers
- Single IP per Pod
- Networking Setup
- CNI Network Configuration File
- Pod-to-Pod Communication
- Cloud Native Computing Foundation
- Resource Recommendations
- Labs

Build

- Container Options
- Containerizing an Application
- Hosting a Local Repository
- Creating a Deployment
- Running Commands in a Container
- Multi-Container Pod
- readinessProbe
- livenessProbe
- Testing
- Labs

Design

- Traditional Applications: Considerations
- Decoupled Resources
- Transience
- Flexible Framework
- Managing Resource Usage

- Multi-Container Pods
- Sidecar Container
- Adapter Container
- Ambassador
- Points to Ponder
- Labs

Deployment Configuration

- Volumes Overview
- Introducing Volumes
- Volume Spec
- Volume Types
- Shared Volume Example
- Persistent Volumes and Claims
- Persistent Volume
- Persistent Volume Claim
- Dynamic Provisioning
- Secrets
- Using Secrets via Environment Variables
- Mounting Secrets as Volumes
- Portable Data with ConfigMaps
- Using ConfigMaps
- Deployment Configuration Status
- Scaling and Rolling Updates
- Deployment Rollbacks
- Jobs
- Labs

Security

- Security Overview
- Accessing the API
- Authentication
- Authorization
- ABAC
- RBAC
- RBAC Process Overview
- Admission Controller
- Security Contexts
- Pod Security Policies
- Network Security Policies
- Network Security Policy Example

- Default Policy Example
- Labs

Exposing Applications

- Service Types
- Services Diagram
- Service Update Pattern
- Accessing an Application with a Service
- Service without a Selector
- ClusterIP
- NodePort
- LoadBalancer
- ExternalName
- Ingress Resource
- Ingress Controller
- Labs

Troubleshooting

- Troubleshotting Overview
- Basic Troubleshooting Steps
- Ongoing (Constant) Change
- Basic Troubleshooting Flow: Pods
- Basic Troubleshooting Flow: Node and Security
- Basic Troubleshooting Flow: Agents
- Monitoring
- Logging Tools
- Monitoring Applications
- System and Agent Logs
- Conformance Testing
- More Resource
- Labs

Closing and Evaluation Survey
