**Docker , Openshift Kubernetes Swagger and LightSpeed API Training**

**Duration 64 hours**

**Day 1**

**1. Docker and Linux Container Technology: Introduction and Use-Cases**

* Modern Infrastructure Terminology
* Virtualization
* Hypervisors
* Hypervisor Types
* Type 1 Hypervisors
* Type 2 Hypervisors
* Type 1 vs Type 2 Processing
* Paravirtualization
* Virtualization Qualities (1/2)
* Virtualization Qualities (2/2)
* Disadvantages of Virtualization
* Containerization
* Virtualization vs Containerization
* Where to Use Virtualization and Containerization
* Containerization: High-Level
* Popular Containerization Systems
* What are Linux Containers
* Docker
* OpenVZ
* Solaris Zones (Containers)
* Container Orchestration Tools
* Docker Swarm
* Kubernetes
* Mesos and Marathon
* Mesos and Marathon (contd.)
* Docker Use-Cases
* Microservices
* Microservices and Containers / Clusters
* Summary

**2. Docker in Action**

* Docker Basics
* Where Can I Run Docker?
* Installing Docker Container Engine
* Docker Toolbox
* What is Docker?
* Docker Architecture
* Docker Architecture Diagram
* Docker Images
* Docker Containers
* Docker Integration
* Docker Services
* Docker Application Container Public Repository
* Docker Run Command
* Starting, Inspecting, and Stopping Docker Containers
* Docker Volume
* Dockerfile
* Docker Compose
* Using Docker Compose
* Dissecting docker-compose.yml
* Specifying services
* Dependencies between containers
* Injecting Environment Variables
* Summary

Day 2

**3. Managing Docker State**

* State and Data in Docker
* Volumes
* More About Volumes
* Uses for Volumes
* Working With Volumes
* Create Volume
* Use Volumes with Containers
* Bind Mounts
* Using Bind Mounts
* tmpfs Mounts
* Storing Data in the Container
* Storage Drivers
* Remote Data Storage
* Networking
* The Default Bridge Network
* User-Defined Bridge Networks
* Docker Network Commands
* Creating a User-Defined Bridge Network
* Summary

**4. Kubernetes Core Concepts**

* Kubernetes Basics
* What is Kubernetes?
* Container Orchestration
* Kubernetes Architecture
* Kubernetes Concepts
* Cluster and Namespace
* Nodes
* Master
* Pod
* Using Pods to Group Containers
* Label
* Label Syntax
* Annotation
* Label Selector
* Replication Controller and Replica Set
* Service
* Storage Volume
* Secret
* Resource Quota
* Authentication and Authorization
* Routing
* Docker Registry
* Summary

Day 3

**5. Kubernetes Architecture(Openshift)**

* Architecture Diagram
* Components
* Kubernetes Cluster
* Master Node
* Kube-Control-Manager
* Nodes
* Other Components
* Interacting with Kubernetes
* Summary

**6. Working with Kubernetes**

* Installation
* Startup
* Kubernetes Tools
* kubectl Command Line Interface
* API Proxy
* Dashboard
* Kubernetes Component Hierarchy
* Deployments
* Deployment Commands
* Updating Deployments
* Network Considerations
* Services
* Namespaces
* Other Useful Commands
* Summary

**7. Scheduling and Node Management**

* Kubernetes Scheduler Overview
* Trusting the Kubernetes Scheduler
* Scheduling Process
* Scheduling Process – Predicates
* Scheduling Process – Priorities
* Scheduling Algorithm
* Kubernetes Scheduling Algorithm
* Scheduling Conflicts
* Controlling Scheduling
* Label Selectors
* Label Selectors (Contd.)
* Node Affinity and Anti-affinity
* Node Affinity Example
* Node Antiaffinity Example
* Taints and Tolerations
* Taints and Tolerations (Contd.)
* Taints and Tolerations – Example
* Summary

Day 4

**8. Managing Networking**

* Kubernetes Networking Components
* The Kubernetes Network Model
* Networking Scenarios
* Container-Container Communication
* Pod-Pod Communication
* 1.3 Pod-Service Communication
* External-Service Communication
* Accessing Applications
* Useful Commands
* Summary

**9. Managing Persistent Storage**

* Storage Methods
* Container OS file system storage
* Docker Volumes
* Kubernetes Volumes
* K8S Volume Types
* Cloud Resource Types
* configMaps
* Creating configMaps from Literals
* Creating configMaps from files
* Using configMaps
* emptyDir
* Using an emptyDir Volume
* Other Volume Types
* Persistent Volumes
* Creating a Volume
* Persistent Volume Claim
* Persistent Volume
* Pod that uses Persistent Volume
* Secrets
* Creating Secrets from Files
* Creating Secrets from Literals
* Using Secrets
* Security Context
* Security Context Usage
* Summary

Day 5

**10. Working with Helm**

* What is Helm?
* Installing Helm
* Helm and KUBECONFIG
* Helm Features
* Helm Terminology
* Searching for Charts with helm CLI
* Adding Repositories
* Helm Hub – Search
* Helm Hub – Chart Page
* Installing a Chart
* Upgrading a Release
* Rolling Back a Release
* Creating Custom Charts
* Common Chart Files
* Helm Templates
* Installing A Custom Chart
* Packaging Custom Charts
* Summary

**11. Continuous Integration Fundamentals**

* Jenkins Continuous Integration
* Jenkins Features
* Running Jenkins
* Downloading and Installing Jenkins
* Running Jenkins as a Stand-Alone Application
* Running Jenkins on an Application Server
* Installing Jenkins as a Windows Service
* Different types of Jenkins job
* Configuring Source Code Management(SCM)
* Working with Subversion
* Working with Subversion (cont’d)
* Working with Git
* Build Triggers
* Schedule Build Jobs
* Polling the SCM
* Maven Build Steps
* Configuring Jenkins to Access Kubernetes
* Jenkins Pipeline
* Jenkins Pipeline Output
* Installing Jenkins Plugins
* Summary

**12. Lab Exercises**

*Lab 1. Managing Containers  
Lab 2. Building Images  
Lab 3. Dockerfiles  
Lab 4. Deploying Stateful Services in Docker  
Lab 5. Custom Network Management  
Lab 6. Docker Volumes  
Lab 7. Accessing the Kubernetes API  
Lab 8. Working with Kubernetes Workloads  
Lab 9. Scheduling and Node Management  
Lab 10. Accessing Applications  
Lab 11. Using Persistent Storage  
Lab 12. Getting Started with Helm  
Lab 13. Build CI Pipeline with Jenkins*

Day 6

**Swagger API Design First**

**Swagger**

**Introducing APIs and OpenAPI**

**What is an API Ecosystem?**

**What is OpenAPI?**

**Where do OpenAPI definitions fit in ?**

**And what is Swagger?**

**What about REST?**

**When to use OpenAPI**

**Getting set up to make API Requests**

**Introducing the FarmStall API and some of its business logic**

**First two operations of FarmStall API**

**Introducing a tool to make HTTP requests — Postman**

**Executing API requests and inspecting the responses**

**First OpenAPI definitions**

**Introducing the OpenAPI specification**

**Learning about YAML.**

**Describing our first GET operation.**

**Using SwaggerEditor to write OpenAPI definitions**

**Introducing SwaggerEditor**

**Writing the smallest OpenAPI definition in SwaggerEditor**

**Adding GET /reviews into our definition**

**Interacting with our API**

**Describing API responses**

**JSON Schema**

**Status Codes**

**Media Types (aka MIME)**

**Describing GET /reviews response**

**Creating resources**

**Describing POST /reviews with a request body**

**Executing operations with request bodies**

**Describing GET /reviews/{reviewId} with a path parameter**

**Verifying our reviews are getting created**

**Adding Authentication and Authorization**

**Identifying the difference between authentication and authorization.**

**Adding operations for creating users.**

**Adding an operation for getting a user’s token (authentication).**

**Adding the Authorization header to POST /reviews operation (authorization).**

**Preparing and hosting API documentation**

**Adding metadata to our API definition**

**Writing a description in Markdown**

**Grouping operations together using tags**

**Hosting our API documentation online using SwaggerUI and Netlify.com**

**Designing a web application**

**Idea Definition of a Web Application**

**Designing a web application**

**Domain Modeling and APIs**

**Adding functionality to the domain model with user stories**

Day 7

**Creating an API design using OpenAPI**

**Creating reusable schemas in OpenAPI**

**Converting the Web Application domain model into schemas**

**Designing an API following the CRUD approach**

**Creating paths and operations for the Our API**

**Building a change workflow around API Design First**

**Identifying the critical issues around an API Design First approach**

**Communicating and reacting to change**

**Setting up a workflow to solve those critical issues using GitHub**

**Tying the GitHub workflow together.**

**Walking through an example change to the API Definition**

**Implementing frontend code and reacting to changes**

**Building the frontend against a mock server (Prism) based on OpenAPI**

**Identifying a missing API operation**

**Choosing which mock data response to get from prism**

**Using OpenAPI examples to verify that API changes make sense**

**Building a Backend with Swagger Codegen**

**Generating backend code with Swagger Codegen**

**Optimizing an OpenAPI definition for code generation**

**Updating OpenAPI for the backend**

**Designing a Mongoose/MongoDB database based on the domain model**

**Implementing a basic API operation**

**Integrating and releasing the web application**

**Adding minimum viable authentication**

**Managing repositories**

**Setting up an integrated webserver**

**The API Design First approach**

**Choosing the constraints**

**Creating the first design**

**Creating the OpenAPI definition**

**Keeping the code and design in sync**

**Highlighting why certain decisions were made in José’s API Design First approach**

**Touching up when a design is complete**

**Building RESTful Web Services Using JAX-RS**

**Introduction to REST**

**REST and the Rebirth of HTTP**

**RESTful Architectural Principles**

**Designing RESTful Service**

**The Object Model**

**Model the URIs**

**Defining the Data Format**

**Assigning HTTP Methods**

**Your First JAX-RS Service**

**Developing a JAX-RS RESTful Service**

**Deploying Our Service**

**HTTP Method and URI Matching**

**Binding HTTP Methods**

**@Path**

**Subresource Locators**

**JAX-RS Injection**

**The Basics**

**@PathParam**

**@MatrixParam**

**@QueryParam**

**@FormParam**

**@HeaderParam**

**@CookieParam**

**Common Functionality**

**JAX-RS Content Handlers**

**Built-in Content Marshalling**

**JAXB**

**Custom Marshalling**

**Response Codes, Complex Responses, and Exception Handlin**

**Default Response Codes**

**Complex Responses**

**Exception Handling**

**HTTP Content Negotiatio**

**Conneg Explained**

**Language Negotiation**

**Encoding Negotiation**

**JAX-RS and Conneg**

**Leveraging Content Negotiation**

**HATEOAS**

**HATEOAS and Web Services**

**HATEOAS and JAX-RS**

**Scaling JAX-RS Applications**

**Caching**

**Concurrency**

**Deployment and Integration**

**Deployment**

**Configuration**

**EJB Integration**

**Spring Integration**

**Securing JAX-RS**

**Authentication**

**Authorization**

**Authentication and Authorization in JAX-RS**

**RESTful Java Clients**

**java.net.URL**

**Apache HttpClient**

**RESTEasy Client Framework**

**RESTEasy Client Proxies**

**JAX-RS Implementations**

**Jersey**

**Apache CXF**

**API Testing with Postman**

**Getting Started with API Testing**

**Understanding Web Services and APIs**

**Mapping Your APIs**

**Simple Authentication with Postman**

**Risks of Using Services and APIs**

**Hands-On API Testing with Postman**

**Install and Set Up API Testing Challenges**

**Testing GET Requests**

**Using Parameters in API Calls**

**Find the Secret: Testing Challenge**

**Find the Secret: Testing Challenge – Solution**

**Testing Your API Calls**

**Testing POST Calls**

**Testing PUT Calls**

**Testing DELETE Calls**

**Automating API Tests in Postman**

**Getting Started with Collections in Postman**

**Validating APIs with Postman Requests**

**Sharing Code Between Tests in Postman**

**Mocking with Postman**

**Running Tests Using Newman**

**Building Quality APIs Using Postman**

**Debugging Your API**

**Improving API Documentation**

**API Monitoring**

**Using the Postman API**

**Advanced Techniques in Postman**

**Using Postman for Data Driven Testing**

**Capture API Requests with the Postman Proxy**

**Importing Existing API Definitions**

**Using tv4 to Validate API Schema**

**Okta: API Access Management using OAuth2/Secure API using Okta**

Day 8

Light Speed API