**Course Description:**

This is an intensive hands-on course aimed at experienced Spring developers. Learn how to quickly get started with Spring applications using Spring Boot. Learn how to arrange, configure and package Spring Boot applications. Learn all about the motivations and benefits of Microservices architecture. You will learn to deploy applications on to Pivotal Cloud Foundry and to bind to services. Learn all about Spring Cloud components like Zuul, Hysterix, Ribbon and Feign.

**Duration:** 4 Days

**Pre-requisites:**

* This training assumes no prior knowledge of Spring Boot and Spring Cloud, but assumes strong working knowledge of the following:
* Spring Core - Dependency Injection, XML, Annotation and Java Config, ApplicationContext
* Resource Configurations , Datasources etc
* Spring MVC and RESTful Services

**Note :** If these pre-requisites are not being met, it is strongly recommended to go through Core Spring training first or the duration needs to increase

**Recommended Learning Resources:**

* Spring Reference Documentation.

**Software& Hardware requirements:**

* Java JDK 1.8
* Eclipse for JEE [2019]
* Pivotal CF account (Optional)
* MySql database
* A reliable and unrestricted internet connection
* Openshift (red hat login)
* Rabbitmq
* Kafka
* Zookeeper
* Zipkin server
* Docker Tool box (docker quick start)
* Docker hub account

**High Level Course Contents**

* Spring Boot
* Microservices Architecture
* Spring Cloud
* Deploying on Pivotal Cloud (Optional, time permitting)
* Spring Cloud Components - Eureka, Zuul, Ribbon, Hysterix

Day 1 and Day 2 Microservice Concepts, Docker and Kuberente and Demo

Day 3 and Day 4 Hands on

**MicroServices Course Outline**

**Day 1**

**Existing Scenarios and Why MicroServices**

* System and service architecture, PaaS, and cloud-native design
* Overview of Monolithic System Architecture
* Running an entire monolithic applications functionality in a single process
* Overview of Webservices
* Service Oriented Architecture (SOA) and Microservices Architecture
* Service-Oriented Architecture (SOA)
* Benefit and Costs of SOA
* Successes and failures in implementing SOA
* Infrastructure and tools supporting SOA
* How Microservices Implement Web Service Concepts
* Continuous Deployment and Delivery (Continuous Integration, Continuous Build Process, etc.)
* Lightweight protocols
* Microservice Protocols and Standards
* HTTP, JMS, AMQP, Websockets, JSON, etc.
* Development Frameworks for Building Microservices
* Java-based frameworks (Spring Cloud)
* Javascript-based frameworks (Seneca)
* Decomposing a Monolith Application[+Practical]
* Developing independently deployable applications
* Organizing microservice applications around business capabilities
* Case study: Migrating a monolith application to three core microservices

**Microservices Architecture Core Concepts :- Advanced**

* Overall topology and architecture components
* Remote access protocols
* Protocol-aware heterogeneous interoperability
* Service components
* Service granularity
* Bounded context
* API layer
* Inter-service communication (choreography)
* Service orchestration
* Contract decoupling
* Accessing third-party systems
* Advantages and disadvantages
* Overall topology
* Service granularity
* Database scope
* Service deployment
* Advantages and disadvantages over MicroServices
* Primary benefits
* Case studies and example architectures

Day 2

Docker and Kubernetes

Introduction

• What is Docker

• Alternatives to Docker

• The evolution of containers

• How containers work

• Containers and Micro Service Architecture

Getting started

• Installing Docker

• Configuring Docker

• Running your first container

Docker Components

• Docker Client

• Docker Daemon

Docker Images

• What is an image

• What does an image contain

• Repositories

• Versioning & Tags

• docker images

Docker Hub

• What is the Docker Hub

• Creating an account

Docker run

• Overview

• Terminal Access

• Detached mode

Processes

• Process Ids

• Within a container

• PID

• docker ps

Logging

• docker logs

• ofollow

Port mapping

• Overview

• Simple port mapping

Building Images with Container Commits

• Read Only and Write layers, Copy on write

• docker commit

Dockerfile

• Introduction

• Instructions and images

• FROM

• RUN

• docker build

• obuild contexts

• CMD

• EXEC

• ENTRYPOINT

Starting and Stopping Containers

• docker ps

• docker start

• docker stop

Getting terminal access

• docker exec

Removing Images

• docker rmi

Docker Hub Repositories

• docker tag

• docker push

• docker pull

Volumes

• docker run /v

• VOLUME

• Do’s and Don’ts

Port Mapping

• docker run -P

• docker run -p

• EXPOSE

Linking

• Introduction

• docker --link

Continuous Integration

• Building Images

• Docker Hub Auto Build

Debugging Containers

• Using a volume for logging

• Docker inspect

Docker Daemon

• Starting and Stopping the Daemon

• Configuration

• Logging Level

Optional Topics for on-site / custom Docker Training

These topics span in total 2 days of additional course content and can be included in extended / custom on-site Docker courses.

• Security

• Private Registry

• Docker Machine

• Docker Compose

• Docker Swarm

• Kitematic

• Putting it all together - Microservices End-To-End

Kubernetes Architecture

**Learning Objectives:**Upon completion of this module, one will get to know about the key components that build a Kubernetes cluster – Master components, Node components and AddOns. We will also see how to install Kubernetes from scratch on Ubuntu VMs.

**Topics:**

* Introduction to Kubernetes Master
* kube-apiserver
* etcd key-value store
* kube-scheduler
* kube-controller-manager
* cloud-controller-manager
* Components of Cloud Controller Manager - Node Controller, Volume Controller, Route Controller, Service Controller
* Introduction to Node Components of Kubernetes
* Docker
* kubelet
* kube-proxy
* kubectl
* Add-ons in Kubernetes: Cluster DNS, Kubernetes Dashboard, Container Resource Monitoring, Cluster level logging

**Hands On/Demo:**

* Create 2 Ubuntu(17.10) VM’s in Virtual box
* Create a single-node cluster using VMs
* Install KubeCtl, a command line tool to manage clusters
* Start a single-node cluster locally
* Get cluster details
* List all nodes associated with the cluster
* Stopping a cluster
* Deleting a cluster
* Installing & Accessing the Kubernetes dashboard

### Deploy an app to Kubernetes Cluster

**Learning Objectives:**In this module, you will understand how to deploy an app using Kubectl to the local Kubernetes cluster and why we need a Pod.

**Topics:**

* Introduction to Pods
* Why do we need a Pod?
* Pod Lifecycle
* Working with Pods to manage multiple containers
* Pod Preset
* What is a Node?
* kubectl basic commands
* Containerized app deployment on local kubernetes cluster

**Hands On/Demo:**

* Deploy a containerized app image in the locally setup kubernetes cluster
* List all local deployments
* Create a kubectl proxy for forwarding communication to cluster-wide private network
* Curl to verify that the app is running
* List all existing pods
* Get description of a specific pod
* View logs of the container
* Execute commands directly on the container

### Expose App, Scale App And Update App in Kubernetes

**Learning Objectives:**In this module, you will learn what a service is, how to expose the deployed app outside the Kubernetes cluster, how to scale up/down the replicas of the app and how to provide updates to the app.

**Topics:**

* What is a Service?
* Labels and Selectors
* Deployment Controller
* Replica Set
* Replication Controller
* Scaling out a deployment using replicas
* Horizontal pod autoscaler
* Load balancing
* Rolling Update
* Ingress and its types

**Hands On/Demo:**

* Create a new service
* Add ha-proxy to configuration file as proxy to expose the application
* Expose the service outside the cluster using ha-proxy
* List all services
* Get more details of a particular service
* Get more information about a label
* Use labels to query required pods
* Create a new label to the pod
* Scale up the above deployment to 4 replicas
* Scale down the above deployment to 2 replicas
* Update the image of the application
* Check the rollout status
* Rollback an update
* Delete the service created

**Day 3**

**Spring Boot - Basics**

* Why Spring Boot
* Spring Boot Set Up
* Introduction to Spring Initializer
* Custom changes in spring Boot configuration
* Profiles
* Maven Build

**Spring Boot – Annotations**

* @SpringBootApplication
* @EnableAutoConfiguration
* @ComponentScan
* @ConditionalOnClass
* @ConditionalOnMissingClass
* Quick Recap on Spring Web Annotations
* Quick Recap on Spring Data Annotations

**Spring Boot – Starters**

* Spring Boot –Web Starter
* Spring Boot – Data Starter

**Spring Boot – Migration the existing Spring Application to the Spring Boot**

* Migrate theSample Spring Application – Banking Use Case or as the participants may use their own scenario with at least 3 different use cases in the single application - To the Spring Boot Framework

**Spring Boot – Actuators**

* Introduction to Actuators
* Configuration of different Endpoints
* Customization of different Endpoints
* Creation of new Endpoints

**Spring Boot – Filters**

* Introduction to Filters
* Filters Ordering
* Custom Filters

**Spring Boot – View Integration**

* Thymeleaf View

**Overview of Spring Cloud**

* Spring Cloud sub-projects: Config Server & Bus, Eureka, Ribbon, Feign, and Hystrix

**Day 4**

**Spring Cloud Eureka**

* Introduction
* Eureka Vs. AWS ELB
* Eureka Vs. Route 53
* Eureka integration in Netflix
* Eureka architecture
* Configuring Eureka
* Scaleability, robustness, reliability and availability factors
* Using memcached caching services
* Cache configuration
* Load-balancing caches
* Non-java services
* clients
* Configuring Regions
* Monitoring clients
* Monitoring servers
* Advanced monitoring using Servo

**Centralized, versioned configuration management with Spring Cloud Config**

**Spring Cloud Hystrix**

* What is Hystrix ?
* Hystrix purpose
* Features
* Latency and failure control
* Monitor cascading failures
* Rapid recovery
* Graceful degradation
* Real-time monitoring, Alerts and Operational control
* Application isolation
* Hystrix architecture
* Configuring app containers
* Dependency management
* Managing and monitoring network traffic
* Call wrapping
* Timeouts
* Thread pools
* Success measurement

**Deployments**

* Setting up a Spring Cloud Development Environment and do the deployment of single MicroServices on Pivotal Cloud , Pivotal Cloud Commands and CLI interface

**Spring Cloud Ribbon**

* Introduction
* Service discovery integration
* Fault tolerance
* Configurable load-balancing rules
* Dependency management
* Ribbon configuration
* application.yml file
* Ribbon architecture
* Ribbon infrastructure
* Failure resiliency
* Ribbon client development
* Client configuration
* Ribbon client api
* Customizing the defaults
* Client property configuration
* Ribbon standards
* Ribbon integration with Eureka
* Caching ribbon configuration
* Configuring Hystrix thread pools
* Ribbon key management
* Load balancing best practices
* Ribbon best practices
* Fault tolerance configuration

**ZULL : working with API Gateway**

* **Declarative REST clients with Feign**

**Deployments**

* Setting up Docker and Docker Compose : and do the deployment of single MicroServices on Docker via Pivotal
* **Spring Cloud – Zookeeper**
* **Spring Cloud - Security**
* **Spring Cloud –Sleuth**
* **Spring Cloud – Zipkin**
* **Final Considerations for Building Production-ready Systems**

Making system easy for beginners

Making system complete so that it can serve as the foundation for enterprise applications