Index

JEE FULL STACK 2 WITH DEVOPS & CLOUD(AWS). - (8 WEEKS)

JEE with DevOps & Cloud(AWS) variant provides exposure to the entire spectrum of Java technologies starting from Core Java to Spring. It focuses on Web Application development using DevOps & AWS and Spring Technology. The following table lists the course structure.

Sr. No.	Course	Duration	Immersive approach Remarks
1	L&D Orientation	**1**	
2	Soft Skills Foundation – Part 1	1	
3	Core Java 8 + Database & PostGreSQL with DevOps (Git, SonarQube, Gradle)	12.5	**Project kick off-Individual project use cases has to be implemented**
4	**Core Java 8 Test**	**0.5**	Coding and MCQ Test
5	Soft Skills Foundation – Part 2	1	
6	JPA with Hibernate with PostgreSQL	**2**	
7	Spring 5.0 (Core + MVC + REST + Data JPA + Data REST+ +H2+Rest Template) with Jenkin	**7**	
8	Swagger	2	
9	Soft Skills Foundation – Part 3	1	
10	Sprint 1 (API Development using Spring Boot application and Spring Data JPA and test API Using Swagger) + MCQ	5	Backend implementation using Spring REST and Spring Data JPA+Swagger+Spring Rest Template+ With DevOps-Jenkin,Git,Gradle & SonarQube
11	**Sprint 1 Evaluation**	**1**	
12	Soft Skills Foundation – Part 4	1	
13	AWS	1	**Cloud Computing +Cloud Native+AWS Introduction and EC2**

14	Docker	2	
15	Kubernates	3.5	
16	**Container Deployment (EKS)**	1.5	
17	**Sprint2 (AWS+Docker+kubernates + MCQ Test**	**3**	**Spring boot application deployment in Docker and Kubernetes.**
18	Sprint 2 Evaluation	1	
19	L1 Preparation	**1**	
20	L1 Test	**1**	
	Total Training Duration	**49**	

Agile SCRUM

Execution:

- **Week 1 – Participants to complete Agile coursera course to understand Agile methodology before Project Kick off to understand and use daily scrum meeting, Project Backlog, Sprint Backlog, Sprint review.**
- **Week 1 Project Kick off – Expectation setting by BU Mentor**
    - **Sprint Planning , Group formed , Case study shared . (BU Support)Declarations and**
- **Week 2 - Requirement review ,Understanding and Design artifacts , Use case , class , sequence diagram to prepared**
- **Week 4 - Artifacts Reviewed and Functional requirement design through interfaces , Test Cases.**
- **Sprint 1 implementation with code reviews of L&D and BU trainer**
    - **Test case reviews**
    - **Code reviews**
    - **Performance monitoring during the sprint implementation and sharing the feedback**
    - **Sprint - 1 Evaluation 30mins/participant**

- **Sprint 2 implementation with code reviews of L&D and BU trainer**
    - **Test case reviews**
    - **Code reviews**
    - **Performance monitoring during the sprint implementation and sharing the feedback**
    - **Sprint - 2 Evaluation 30min/participant**

Core Java 8

Program Duration: 10.5 days

Contents:

- **Declarations and Access Control**
  - Identifiers & JavaBeans
  - Legal Identifiers
  - Sun's Java Code Conventions
  - JavaBeans Standards
  - Declare Classes
  - Source File Declaration Rules
  - Class Declarations and Modifiers
  - Concrete Subclass
  - Declaring an Interface
  - Declaring Interface Constants
  - Declare Class Members
  - Access Modifiers
  - Nonaccess Member Modifiers
  - Constructor Declarations
  - Variable Declarations
  - Declaring Enums

- **Object Orientation**
  - Encapsulation
  - Inheritance, Is-A, Has-A
  - Polymorphism
  - Overridden Methods
  - Overloaded Methods
  - Reference Variable Casting
  - Implementing an Interface
  - Legal Return Types
  - Return Type Declarations
  - Returning a Value
  - Constructors and Instantiation
  - Default Constructor
  - Overloaded Constructors
  - Statics
  - Static Variables and Methods
  - Coupling and Cohesion

- **Assignments**
  - Stack and Heap—Quick Review
  - Literals, Assignments, and Variables
  - Literal Values for All Primitive Types
  - Assignment Operators
  - Casting Primitives
  - Using a Variable or Array Element That Is Uninitialized and Unassigned
  - Local (Stack, Automatic) Primitives and Objects

Capgemini Sensitive

- Passing Variables into Methods
- Passing Object Reference Variables
- Does Java Use Pass-By-Value Semantics?
- Passing Primitive Variables
- Array Declaration, Construction, and Initialization
- Declaring an Array
- Constructing an Array
- Initializing an Array
- Initialization Blocks
- Using Wrapper Classes and Boxing
- An Overview of the Wrapper Classes
- Creating Wrapper Objects
- Using Wrapper Conversion Utilities
- Autoboxing
- Overloading
- Garbage Collection
- Overview of Memory Management and Garbage Collection
- Overview of Java's Garbage Collector
- Writing Code That Explicitly Makes Objects Eligible for Garbage Collection

- **Operators**
  - Java Operators
  - Assignment Operators
  - Relational Operators
  - instanceof Comparison
  - Arithmetic Operators
  - Conditional Operator
  - Logical Operators

- **Flow Control, Exceptions**
  - if and switch Statements
  - if-else Branching
  - switch Statements
  - Loops and Iterators
  - Using while Loops
  - Using do Loops
  - Using for Loops
  - Using break and continue
  - Unlabeled Statements
  - Labeled Statements
  - Handling Exceptions
  - Catching an Exception Using try and catch
  - Using finally
  - Propagating Uncaught Exceptions
  - Defining Exceptions
  - Exception Hierarchy
  - Handling an Entire Class Hierarchy of Exceptions
  - Exception Matching

Capgemini Sensitive

    - Exception Declaration and the Public Interface
    - Rethrowing the Same Exception
    - Common Exceptions and Errors

- **Gradle Fundamentals**
    - Introduction
    - Folder Structure
    - Install and Setup Gradle on Windows

    - Dependencies in Build Scripts
    - Gradle Wrapper
    - Lifecycle Tasks: The Base Plug In
    - Using Project Info and the check command
    - Creating Variables and external properties
    - Creating a Build Scan
    - Dependencies

- **TDD with Junit 5**
    - Types of Tests
    - Why Unit Tests Are Important
    - What's JUnit?
    - JUnit 5 Architecture
    - IDEs and Build Tool Support
    - Setting up JUnit with Maven
    - Lifecycle Methods
    - Test Hierarchies
    - Assertions
    - Disabling Tests
    - Assumptions
    - Test Interfaces and Default Methods
    - Repeating Tests
    - Dynamic Tests
    - Parameterized Tests
    - Argument Sources
    - Argument Conversion
    - What Is TDD?
    - History of TDD
    - Why Practice TDD?
    - Types of Testing
    - Testing Frameworks and Tools
    - Testing Concepts
    - Insights from Testing
    - Mocking Concepts
    - Mockito Overview
    - Mockito Demo
    - Creating Mock Instances
    - Stubbing Method Calls

Capgemini Sensitive

- **Strings, I/O, Formatting, and Parsing**
    - String, StringBuilder, and StringBuffer
    - The String Class
    - Important Facts About Strings and Memory
    - Important Methods in the String Class
    - The StringBuffer and StringBuilder Classes
    - Important Methods in the StringBuffer and StringBuilder Classes
    - File Navigation and I/O
    - Types of Streams
    - The Byte-stream I/O hierarchy
    - Character Stream Hierarchy
    - RandomAccessFile class
    - The java.io.Console Class
    - Serialization
    - Dates, Numbers, and Currency
    - Working with Dates, Numbers, and Currencies
    - Parsing, Tokenizing, and Formatting
    - Locating Data via Pattern Matching
    - Tokenizing

- **Generics and Collections**
    - Overriding hashCode() and equals()
    - Overriding equals()
    - Overriding hashCode()
    - Collections
    - So What Do You Do with a Collection?
    - List Interface
    - Set Interface
    - Map Interface
    - Queue Interface
    - Using the Collections Framework
    - ArrayList Basics
    - Autoboxing with Collections
    - Sorting Collections and Arrays
    - Navigating (Searching) TreeSets and TreeMaps
    - Other Navigation Methods
    - Backed Collections
    - Generic Types
    - Generics and Legacy Code
    - Mixing Generic and Non-generic Collections
    - Polymorphism and Generics

- **Threads**
    - Defining, Instantiating, and Starting Threads
    - Defining a Thread
    - Instantiating a Thread
    - Starting a Thread

- Thread States and Transitions
- Thread States
- Preventing Thread Execution
- Sleeping
- Thread Priorities and yield( )
- Synchronizing Code
- Synchronization and Locks
- Thread Deadlock
- Thread Interaction
- Using notifyAll( ) When Many Threads May Be Waiting

- **Lambda Expressions**
  - Introduction
  - Writing Lambda Expressions
  - Functional Interfaces
  - Types of Functional Interfaces
  - Method reference
- **Stream API**
  - Introduction
  - Stream API with Collections
  - Stream Operations

Database Using PostgreSQL

Duration : 2 days

Contents:

- **Introduction**
  - The Relational Model
  - What is PostgreSQL?
  - PostgreSQL – Data Types
  - Arrays Functions and Operators
- **Understanding Basic** PostgreSQL **Syntax**
  - The Relational Model
  - Basic SQL Commands - SELECT
  - Basic SQL Commands - INSERT
  - Basic SQL Commands - UPDATE
  - Basic SQL Commands – DELETE

- **Querying Data with the SELECT Statement**
  - Wildcards (%, _)
  - The SELECT List
  - SELECT List Wildcard (*)
  - The FROM Clause
  - How to Constrain the Result Set
  - DISTINCT and NOT DISTINCT

Capgemini Sensitive

- **Arrays Functions and Operators**
    - array_append
    - array_cat
    - array_lower
    - array_to_string
    - array_agg
    - every,Count,sum,avg
    - Array Operators
- **Filtering Results with the Where Clause**
    - WHERE Clause
    - Boolean Operators
    - The AND Keyword
    - The OR Keyword
    - Other Boolean Operators BETWEEN, LIKE, IN, IS, IS NOT

- **Shaping Results with ORDER BY and GROUP BY**
    - ORDER BY
    - Set Functions
    - Set Function And Qualifiers
    - GROUP BY
    - HAVING clause

- **Matching Different Data Tables with JOINs**
    - Table Aliases
    - CROSS JOIN
    - INNER JOIN
    - OUTER JOINs
    - LEFT OUTER JOIN
    - RIGHT OUTER JOIN
    - FULL OUTER JOIN
    - SELF JOIN
    - Natural Join

- **Creating Database Tables**
    - CREATE DATABASE
    - CREATE TABLE
    - NULL Values
    - PRIMARY KEY
    - CONSTRAINT
    - ALTER TABLE
    - DROP TABLE
- **PostgreSQL Transactions**
    - BEGIN, COMMIT, ROLLBACK

- **PostgreSQL Constraints**
    - CHECK, UNIQUE, NOT NULL

Capgemini Sensitive

DevOps/ CI CD concepts (GitHub/Nexus, CI Jenkins, Sonar)

Contents:

- Introduction to DevOps :
    - What is DevOps
    - Evolution of DevOps
    - Agile Methodology
    - Why DevOps
    - Agile vs DevOps
    - DevOps Principles
    - DevOps Lifecycle
    - DevOps Tools
    - Benefits of DevOps
    - Continuous Integration and Delivery pipeline
    - Use-case walkthrough

- GitHub
    - What is DevOps
    - Introduction to Git
    - Version control
    - Repositories and Branches
    - Working Locally with GIT
    - Working Remotely with GIT

- Jenkins
    - Introduction to CI
    - Jenkins Introduction
    - Creating Job in Jenkins
    - Adding plugin in Jenkins
    - Creating Job with Gradle & Git

- Jenkins With TDD(Junit testing)
    - Integration of jUnit testing with Jenkins

- Sonar

JPA Using PostgreSQL

Program Duration: 2 days

Contents:

- **Introduction**
  - Introduction & overview of data persistence
  - Overview of ORM tools
  - Understanding JPA
  - JPA Specifications
- **Entities**
  - Requirements for Entity Classes
  - Persistent Fields and Properties in Entity Classes
  - Persistent Fields
  - Persistent Properties
  - Using Collections in Entity Fields and Properties
  - Validating Persistent Fields and Properties
  - Primary Keys in Entities
- **Managing Entities**
  - The EntityManager Interface
  - Container-Managed Entity Managers
  - Application-Managed Entity Managers
  - Finding Entities Using the EntityManager
  - Managing an Entity Instance's Lifecycle
  - Persisting Entity Instances
  - Removing Entity Instances
  - Synchronizing Entity Data to the Database
  - Persistence Units
- **Querying Entities**
  - Java Persistence query language (JPQL)
  - Criteria API
- **Entity Relationships**
  - Direction in Entity Relationships
  - Bidirectional Relationships
  - Unidirectional Relationships
  - Queries and Relationship Direction
  - Cascade Operations and Relationships

Spring 5.0

Program Duration: 13 days

Contents:

1. **Spring Core**

   Spring Core Introduction / Overview
   - Shortcomings of Java EE and the Need for Loose Coupling
   - Managing Beans, The Spring Container, Inversion of Control
   - The Factory Pattern
   - Configuration Metadata - XML, @Component, Auto-Detecting Beans
   - Dependencies and Dependency Injection (DI) with the BeanFactory
   - Setter Injection

Spring Container

- The Spring Managed Bean Lifecycle
- Autowiring Dependencies

Dependency Injection

- Using the Application Context
- Constructor Injection
- Factory Methods
- Crucial Namespaces ‘p’ and ’c’
- Configuring Collections

Metadata / Configuration

- Annotation Configuration @Autowired, @Required, @Resource
- @Component, Component Scans. Component Filters
- Life Cycle Annotations
- Java Configuration, @Configuration, XML free configuration
- The Annotation Config Application Context

2. **Spring MVC**

Introduction / Developing Web applications with Spring MVC

- The WebApplicationContext and the ContextLoaderListener
- Model View Controller
- Front Controller Pattern
- DispatcherServlet Configuration
- Controllers, RequestMapping
- Working with Forms
- Getting at the Request, @RequestParam, @RequestHeader
- ModelAndView

Advanced Techniques

- Spring form tags and Model Binding, @ModelAttribute

Spring Controllers

- Using @ResponseBody
- JSON and XML data exchange

RESTful Web Services

- Core REST concepts
- REST support in Spring 5.x
- Use Spring MVC to create RESTful Web services
- REST specific Annotations in Spring
- Working with RestTemplate
- URITemplates, @PathVariable, @RequestParam
- JSON and XML data exchange
- @RequestMapping

3. **Spring Boot**

Capgemini Sensitive

SPRING BOOT Introduction

- Spring Boot starters, CLI, Gradle plugin
- Application class
- @SpringBootApplication
- Dependency injection, component scans, Configuration
- Externalize your configuration using application.properties
- Context Root and Management ports
- Logging

Using Spring Boot

- Build Systems, Structuring Your Code, Configuration, Spring Beans and Dependency Injection, and more.

Spring Boot Essentials

- Application Development, Configuration, Embedded Servers, Data Access, and many more
- Common application properties
- Auto-configuration classes
- Spring Boot Dependencies

4. Spring Data JPA

- Spring Data JPA Intro & Overview
- Core Concepts, @RepositoryRestResource
- Defining Query methods
- Query Creation
- Using JPA Named Queries
- Defining Repository Interfaces
- Creating Repository instances
- JPA Repositories
- Persisting Entities
- Transactions

5. Spring Data REST

- Introduction & Overview
- Adding Spring Data REST to a Spring Boot Project
- Configuring Spring Data REST
- Repository resources, Default Status Codes, Http methods
- Spring Data REST Associations
- Define Query methods
- Work with H2 Database

6. Swagger : 2 days

- **Rest Architectural Design pattern**
- Anatomy of API request
- API Definition file
- Open API initialtive
- Open API Specification (OAS 3) basics
- Schemas

AWS

Program Duration: 1 day

Contents:

- Cloud Basics
    - What is and Why Cloud?
    - Why Cloud Computing
    - Key characteristics of Cloud
    - Cloud Computing Architecture
    - Cloud Deployment and Service Model Selection criteria
    - Cloud APIs
    - Cloud benefits and Challenges
    - Different Cloud implementer
    - Latest trend
- Cloud Native Concepts
    - Cloud technology
    - Cloud Native Approach
    - Purpose of Cloud Native
    - What are Cloud Native companies doing differently to improve IT agility Benefits of Cloud native
    - Hybrid cloud
- AWS Basics of different services
    - AWS history
    - Cloud Computing and Amazon Web Services
    - Functionality offered by AWS
    - The Differences that Distinguish AWS
    - Features of AWS service
    - Different AWS web services in Cloud
    - AWS global infrastructure
- Compute services
    - Amazon EC2

Docker

Program Duration: 2 days

Contents

- Introduction to Docker
    - Limitation of VM
    - Introduction to Container
    - Container Vs VM
    - What is Docker
    - Docker Community
    - Docker Architecture
    - Docker Installation

- Docker Platform overview
    - Docker Platform
    - Docker Engine
    - Docker Images
    - Docker containers
    - Registry
    - Repositories
    - Docker Hub
  - **Deploying a Containerized App**

    Module Overview
    Warp Speed Run-through
    Containerizing an App
    Hosting on a Registry
    Running a Containerized App
    Managing a Containerized App
    Multi-container Apps with Docker Compose
    Taking Things to the Next Level with Docker Swarm

Kubernetes

Program Duration: 3 .5 days

Contents

- **Introduction of Kubernetes**
    - What Is Kubernetes?
    - Kubernetes What and Why

- **Kubernetes Architecture**
    - Module Overview
    - Kubernetes Big Picture View
    - Kubernetes Masters
    - Kubernetes Nodes

Capgemini Sensitive

- The Declarative Model and Desired State
- Kubernetes Pods
- Stable Networking with Kubernetes Services
- Game Changing Deployments
- The Kubernetes API and API Server
- Api Server
- Scheduler
- Controller Manager
- etcd - the cluster brain

- **Getting Kubernetes**
  - Module Overview
  - Getting kubectl
  - Getting K8s in the Cloud

- **Working with Pods**
  - Module Overview
  - App Deployment Workflow
  - Creating a Pod Manifest
  - Deploying a Pod
  - Deployment vs StatefulSet
  - Pod Identity
  - Scaling database applications: Master and Worker Pods
  - Pod state, Pod Identifier
  - 2 Pod endpoints

- **Kubernetes Deployments**
  - Module Overview
  - Kubernetes Deployment Theory
  - Creating a Deployment YAML
  - Deploying a Deployment
  - Self-healing and Scaling
  - Rolling Updates and Rollbacks

- **ClusterIP Services**
  - Service Communication
  - Multi-Port Services
  - Headless Services
  - NodePort Services
  - LoadBalancer Services

- Helm - Package Manager
  - Package Manager and Helm Charts
  - Templating Engine
  - Use Cases for Helm
  - Helm Chart Structure
  - Values injection into template files

Capgemini Sensitive

Container Deployment Service EKS – 1.5 Days

- Creation of an EKS cluster
- Configure kubectl using AWS CLI
- Serverless pods
- Scaling the cluster

Capgemini Sensitive