Introduction to Continuous Integration and Continuous Delivery with Jenkins

The **Jenkins** is continuous Integration server for complete build automation of software projects. The Jenkins supports the different type of job build configurations, triggering the build processes and monitoring executions of build jobs, such as building a software project or jobs run by schedulers and updating the people involved by notifications configured via email.
This course provides the complete learning and hands-on experience required to jump start the team's adoption of continuous integration and delivery management with Jenkins.

Prerequisite Knowledge
Participants should have sound programming skills and exposed to application environments and having worked with containerized applications in Docker.

Objectives
To introduce the continuous Integration and delivery with Jenkins for automated software project build and deployment process along with Docker and Jenkins integration.

Training Methodology
This is complete practical oriented training session with more focus on practical work.
The theoretical topics are discussed interactively and technical details are demonstrated with practical examples. The participants work on the hands on exercises which strengthen the concepts learned**.**
Each topic is supplemented with practical demonstrations and hands on exercises.

Target Population
Maximum 20 participants matching the prerequisites with every one working on separate System having the preconfigured set up as follows.

Class Room Setup
Participants Machine: The Intel core compatible CPU with Windows 10 OR Ubuntu Linux 18.04 (64 bit) System with minimum 8GB RAM and 500GB HDD. Adobe PDF Reader, WinZip latest, JDK1.8 64 bit, GIT SCM Repository, The GIT Client, Docker for Windows or Docker on Linux and Mozilla Firefox/Google chrome latest browser to be installed and configured. The zip distributions for build automation tools and Jenkins ver.2.249.1.war to be made available.

Links for downloading
Jenkins
https://mirrors.tuna.tsinghua.edu.cn/jenkins/war-stable/2.249.1/jenkins.war

Apache Ant
https://mirrors.estointernet.in/apache//ant/binaries/apache-ant-1.9.15-bin.zip

Apache Maven
https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.zip

Git
https://github.com/git-for-windows/git/releases/download/v2.30.1.windows.1/Git-2.30.1-64-bit.exe

Docker for Windows
https://docs.docker.com/docker-for-windows/install/

Docker for Linux
https://docs.docker.com/engine/install/ubuntu/

The participants must have admin rights on their machines.
Live internet connection with reasonable speed and download permissions has to be made available in the class room.

Training Duration: - 5 days

Instructor: Prakash Badhe.

Course outline

The course outline is based on the approval from the client team.

The applications and automation tools used in the demonstrations and case studies will vary based on the environment, application programming platform and tools preferred in automation pipeline by the client team.

1. Participants Role: --Tester/Admin
2. Web application platform: -- Java
3. Automation tool used in demonstrations: Apache Ant
4. The Web application deployment server: Apache Tomcat on local machine and nginx with Docker

Day1

The phases of application life cycle

- The code repository
- Unit and integration testing
- Test-Driven Development Process
- Code Refactoring
- BDD Testing

- Regression testing
- Code quality checks
- Integration and Delivery
- Monitoring, feedback and corrective actions

Applications build process

- Development environment
- Unit and integration testing environment
- Code quality analysis tools
- Code repositories
- Package and Deployment
- Automate the build process
- Build automation tools like Apache Ant, Maven etc.
- Integration and delivery tools

The Testing Phases

- The unit testing.
- The test plans, Test cases and test suits
- Test execution and monitoring
- Integrate units tests Into the build System
- Integration testing
- Acceptance testing

Test-Driven Development Process

- The Unit Testing Framework
- The Test First Approach
- The TDD process.
- Why Adopt TDD
- Where to Begin TDD
- The TDD cycle : Red-Green-Refactor
- Application development with TDD process

Code Refactoring Process

- Why refactoring?
    - Change the code structure without affecting external behavior
    - Make the code more readable
    - Make the code more reusable and flexible.
    - Apply Design Patterns in refactoring process
    - Avoid code duplication
    - Make the code more Loose coupling and code abstractions
- Testing before and after re-factoring
- Identify and Implement Refactoring
- The Refactoring techniques and patterns

- SOLID Principles overview
- The continuous refactoring process

The Test-Driven Development Impact

- Developer confidence
- Iterative and incremental development
- Application delivery management
- Easier and flexible change management.
- Customer involvement
- Max code coverage with lesser dead code
- Easier trouble shooting
- Continuous delivery
- Continuous improvements

The TDD Challenges

- The TDD Infrastructure
- The process and people mindset
- Repetitive Testing
- Manage dependencies
- Continuous Testing
- Acceptance Testing
- Regression Testing

Mock Testing

- Testing the code dependencies
- Testing in absence of dependencies
- Isolate the dependencies
- Test to isolate and verify the failures
- Mock testing requirements
    - Mock the absent code
    - Verify the code invocation
    - Verify the code interactions
    - Isolate the problem in case of complex data flow
- Mocking frameworks usage
    - What Are Mock Objects?
    - When and How to Use Mock Objects
    - Stubs and Mocks
    - Mocking the database and services
    - Test with mock objects
    - Mock object life cycle
- Working with Mock objects in Mockito framework

Day2

The Continuous Integration Process

- Challenges in manual integration
- Make it continuously Integrate
- The automated process of integration - CI
- The continuous Integration (CI) Work flow
- The CI Practices and principles
    - Commit Code Frequently
    - Don’t Commit Broken Code
    - Iterative builds
    - Fix Broken Builds immediately
    - Avoid Getting Broken Builds
    - Write Automated tests
    - All Tests Must Pass
    - Automate As Much As Possible
    - Shared Ownership of the code
    - Build Notifications
    - Teamwork

The Continuous Integration Implementation

- How to Implement CI
    - Code repository and code base
    - Automated Testing
    - Automated Builds
    - External and build dependencies
    - Build automation tools
    - **Continuous Integration** Tools
- Continuous Testing
    - Automated Unit, Integration, System and Functional Tests
    - Write Tests for Defects
    - Integration of unit, integration and acceptance testing With CI
    - Code coverage measurement
- Continuous Deployment/Delivery
    - Philosophy of CD
    - Release and versioning Strategies
    - Automated acceptance testing
    - Test reports generation
    - Continuous monitoring and feedback generation
- Continuous Quality check

    - Code Inspection vs. Code Testing
    - Static Code Analysis
    - Code quality analysis Tools
    - Code linter
    - Code quality logs
    - Quality gates
- Continuous Improvements
    - The Feedback Loop
    - The Improvement cycle

The Jenkins CI Server: Getting Started

- Jenkins Features
- Jenkins installation and configuration
- Installing required Jenkins plug-ins
- The build jobs in Jenkins
- Components of a build job pipeline
- Create Freestyle Build job
- The project build configuration
- IDE and GIT repository integration
- Jenkins GIT repository integration
- The build process startup trigger
- The job schedulers
- Pre build activities
    a. Fetch the code from repository
    b. Execute the unit test cases/suites
    c. Execute native shell/batch scripts
    d. Integrate with code quality tools Sonar, FindBugs, PMD, Checkstyle etc.
- The build stage
- Post build activities
    a. Deployment
    b. Acceptance testing
    c. The code quality metrics.
    d. The code coverage measurement
    e. Generate test and analysis reports.
    f. Notify the build reports via email
    g. Monitor the jobs.

Continuous Delivery process

- Elements of delivery process
- Integrate development and deployment environments.
- Configure delivery pipeline.
- Customize Jenkins plug-ins.

Day3

More jobs with Jenkins

- The maven jobs
- The parameterized jobs
- Upstream and downstream jobs with dependencies
- Manage multiple build jobs in sequence and parallel
- Conditional steps
- Monitoring jobs in Jenkins.
- Troubleshooting tips for Job failures

Jenkins pipeline jobs

- Create and manage pipeline jobs with pipeline plug-in
- Jenkins build job with pipeline
- The building blocks of pipeline job
- Manage the pipeline with code repository
- The pipeline and jenkinsfile syntax
- Infrastructure as Code (IaC) with Pipeline

Jenkins Nodes

- Manage Distributed Builds
- Configure Master/Slave distributed nodes
- High availability/Clustering configuration overview
- Manage scalability with Jenkins cluster

Day4

Jenkins REST API and Cli

- The REST API exposed by Jenkins
- Execute the actions remotely with REST API calls
- Monitor the jobs with REST API
- Manage remote execution from command line with CLI

Jenkins with Groovy

Groovy Language Introduction

- What Is Groovy
- Need of the groovy
- The Groovy Interpreter
- Groovy keywords
- The Groovy Compiler
- Groovy as Java
- The groovy class and script
- Groovy dynamics
- Groovy data types
- Groovy conventions
- Groovy and Java integration
- String manipulation with groovy

Groovy applications and usage

- Groovy shorthand and defaults
    - Assumed imports
    - Default visibility, optional semicolon
    - Optional parentheses and types
    - Optional return keyword
- Strings in groovy
- Groovy Object Oriented Programming.
- Control structures
- Groovy Ranges
- Groovy Collections
- Working with Lists and Maps
- Assertions with groovy
- Regular expressions
- The overloaded Operators with groovy

Jenkins with Groovy Scripting

- Manage Jenkins and jobs with groovy scripting in Jenkins groovy console
- Manage Environment Variables with groovy script
- Execute the build jobs with groovy script
- Execute the groovy script from shell
- Execute the groovy script remotely with Http Post method.

Day5

Docker integration with Jenkins

- Overview on Docker container applications
- The Docker plug-in
- Build the image from Dockerfile in pipeline stage

- Manage containers in pipeline job
- The Docker-Pipeline plug-in
- Run sidecar containers in parallel
- Build the image from repository
- Publish the Docker image
- Configure web application integration with Jenkins
- Configure continuous delivery in Docker with Jenkins

****