Introduction to Agile Practices in DevOps

Agile software development is the process of software development where the application requirements and desired solutions evolve through the collaboration of small and cross functional teams involving the customer as well. This technique advocates adaptive planning, evolving solutions in iterative way, frequent delivery of usable application in increments and continuous improvement with testing and refactoring. This leads to rapid and flexible response by the development team to the change requirements at any stage.
Agile Methodology involves continuous iteration of development and testing in the SDLC process. This software development method emphasizes on iterative, incremental, and evolutionary development.
DevOps is a **practice** of bringing development and operations teams together. **DevOps** is a software engineering practice and culture that aims at bridging the gap between the Development and Operations team by collaborative efforts so that the customers get quick releases, increased deployment frequency with more useful releases of the product in close alignment with business objectives.
The central concept in **DevOps** is to manage end-to-end engineering processes.
The DevOps practice strongly recommends automation and monitoring at all steps of software development and release cycles from unit testing,
build, integration, acceptance testing, infrastructure management and releasing to deployment.
With DevOps deploying code to production becomes faster and in an automated manner. It helps to increases an organization's speed to deliver application and services.
This course introduces agile practices and applying the agile practices technically from development to deployment and monitoring and quick actions with the usage of tools and techniques.

Prerequisites
The participants with any of the following prerequisites

- Working on Java web application development
- Working on Testing(QA)
- Participants from build/release management
- Project Manager/Architect
- Operation team

Training Methodology
The theoretical topics are discussed interactively and practical coverage is done by practical demonstrations for each topic and hands on practical exercises which strengthens the concepts learned.
The course material includes pdf presentations, documents and sample programs with case study exercises.

Session Machine
Intel dual core compatible CPU with minimum 4GB RAM and 500GB HDD with Windows 7/8/10 64 bit OR Ubuntu Linux 16.04 64 bit. Adobe pdf reader, JDK 1.8 64 bit, Eclipse-Jee-Photon 64 bit, Apache Ant, Apache Tomcat 8.0,Jenkins 2.250 and Firefox browser to be installed.
The participants have to install this on their session machines for practical hands on work.

Live internet connection with reasonable speed and download permissions should be available in the training room.
The participants must have admin rights on their systems.

Course Duration: Three days.

Instructor: Prakash Badhe.

Course Plan

Day1
SDLC Overview

- Waterfall development model and its disadvantages
- Manage the requirement change in Waterfall model
- Delayed delivery to the customer
- Poor fault detection and rectification
- Poor management of the requirements change

The next topic is the theoretical discussion about agile methodologies and practices.

Bird's Eye View about Agile Methodologies

- Why Agile?
- What is Agile?
- When Agile?
- Being Agile.
- Extreme programming practices
- Small team size
- Pair programming
- Develop and Build in smaller increments
- Bridge the gap between customer and developer communications.
- Getting the quick feedback and correcting
- Iterative development with smaller increments
- Methodologies overview scrum, kabana, scrum, XP.

Agile Practices

- Agile techniques
- Forming ‘The Agile Team’
- Breaking the big monolith product into smaller pieces
- Test Driven development - Test First approach.
- Refactoring the code and regression testing
- Continuous Testing
- Integrates the small product pieces with final testing
- Continuous Integration
- Acceptance Testing
- Continuous Delivery and deployment
- Build automation
- Collaboration between the Dev and Ops teams
- Bridge the gaps in Developer and IT Operations communications
- Unused/dead code?
- Code coverage
- Build and test automation

The next topics are the practical oriented discussion and demonstration about following Agile practices in technical environment.

Test-Driven Development

- Why TDD
- How to Adopt TDD
- Where to Begin TDD
- The Test First Approach
- The TDD cycle : Red-Green-Refactor
- Implement a case study in TDD way
- BDD and ATDD overview.

Application development in TDD way

- Case study introduction and overview
- Understand the requirements
- Start with lowest minimum component.
- Write first test case and let it fail first time.
- Write minimum code to make the first test success.
- Write more test cases and let them fail first time one by one.
- Write minimum code to satisfy the test cases one by one.
- Verify success with regression tests
- Evolve the application in TDD way.
- Apply mock testing
- The stub and proxy behavior
- Testing the Servlet outside the container

- Using the mock objects.

Testing with Mock objects

- Code Dependencies
- Isolate dependencies during testing
- Testing in isolation with mock objects
- The Mocking process
- Assert and Verify test Invocations
- Record-Play-Verify
- The stub and proxy behavior
- Testing the Servlet outside the container

Code Refactoring

- Why refactor?
- Change the code structure without affecting external/functional behavior
- Make it maintainable.
- Make it more reusable and flexible.
- When to refactor?
- Re-test the re-factored Code
- Regression testing
- Apply Design Patterns in refactoring process
- Loose coupling with code abstractions
- Design Principles overview

The participants work on the implementation of shared Case study in TDD way.

Day2

Agile Continuous Integration Process

- Manual Integration
- Code integration in teams
- Role of CI in Agile Practice
- Agile techniques for CI
- CI Philosophies
- CI automation
- Quick delivery and updates
- Agile advantage.

CI Role in DevOps

- Manual build and test execution
- Manual integration challenges and limitations

- Automate the build with the automation Tool
- Build automation tools : Ant,Maven,Gradle overview
- Configure build tasks such as compile, package and invoke test cases
- Test and execute the build script

Agile Build and practices

- Source code repository (GIT/SVN)
- Automate the build and integration process
- Continuously Integrate and build the application releases
- The Continuous Integration Server : Jenkins
- Jenkins architecture overview
- Configure Jenkins and related tools
- Type of jobs
- Configure a new build job
- Configure Trigger for the build
    - Build on demand
    - Periodic build
    - Trigger from repository for change in the code base
    - Dependent jobs
- Pre-build Stages
    - Compile
    - Unit Test
    - Integration Test
    - Package for the deployable artifact
- Monitor the build job execution
- Generate unit test reports
- Add more Post build activities in the build job
    - Code coverage check
    - Code quality analysis
    - Package the build
    - Execute batch files/shell scripts
    - Notify the collaborators by email about the result of build
    - Trigger another build job
- Parameterized build jobs
- Distributed build with master-slave
- Jenkins clustering overview
- Overview on groovy scripting in Jenkins

The participants work on Jenkins to define new build job and execute.

The participants incorporate a change into the user story to Understand the continuous integration process.

Day3

Production Environment

- Continuous delivery of the application releases.
- Apache Tomcat 8/Oracle Weblogic server
- The Docker container environment
- Cluster with Load balancer
- Centralized log management

Application Delivery in Manual mode

- Production deployment environment
  - Remote Web Server
  - Docker Container environment
  - Cloud
- Manual delivery tasks
  - Manage the production deployment environment
  - Deploy the artifact to production
  - Invoke the acceptance test cases
  - Generate the acceptance test reports

The participants commit a small change into the user story implementation and build and deploy it from the CI server to understand the continuous automatic delivery process.

Application Delivery by Ops Team

- Overview on the provisioning tool (Configuration Management) to create and manage the production environment. E.g. Puppet, Chef, Ansible etc.
- Deliver the application in smaller increments frequently to the production server
- Deployment scenarios

The participants write set of acceptance test scenarios for the application deployed and invoke it as part of post delivery process.

BDD Introduction

- What is Behavior Driven Development
- Cucumber and Gherkin Introduction
- Cucumber with Selenium for Java environment

Acceptance Testing

- Define the acceptance test specifications for the application in BDD way
- Configure Cucumber with Selenium to invoke the acceptance test specs.
- Define the build script to run the acceptance test scenarios
- Ops team invokes the acceptance test scenarios as part of post deployment process.
- Generate acceptance test reports. Observer with Cucumber

The Ops team will be monitoring the application in production environment for performance, functioning issues, metrics etc. and convey to the proper team for action.

Scaling and updates by Ops team

- Monitor and analyze for optimum performance
- Scale up/down the nodes in the cluster
- Deploy Rolling updates
- Blue-Green deployment

Ops team on monitoring

System tools

- Monitor network services
- Monitoring of host resources processor load, disk usage, system logs etc.
- Remote monitoring
- Detect the down or unreachable hosts
- Notify the collaborators for services/host problems by email/SMS etc.
- Monitor performance data

****