DevOps using Git, Jenkins, Artifactory and ELK stack for J2EE applications

What is DevOps ?

DevOps is a new Philosophy that can help organizations innovate faster and be more productive and responsive to business needs. It promotes collaboration between developers and operations by automating infrastructure, automating workflows and continuously measuring application performance.

DevOps Goals and Benefits

- Increase rate of software delivery.
- Improves company’s time to market potentially from months and weeks to days and hours.
- Provide huge competitive advantage due to shorter release cycles.
- Help them automating the releases and their infrastructure so that they can focus on increasing business value
- Reduce manual processes leading to happier employees and customers.
- Lower failure rates of releases.
- Faster mean time to recovery.
- Shortened lead time between fixes.

DevOps Adoption Interest

- Use of Agile Development processes.
- Demand for frequent releases to production.
- Availability of cloud infrastructure and tools.
- Increased usage of data center automation and configuration management tools.
- Increased focus on test automation and continuous integration methods.
- A critical mass of publicly available best practices in the industry.

DevOps Toolchain

DevOps is a set of tool chain that will help organizations achieve their goals. The tools fit into several categories that are:

Code	-	SCM, workflow, development, review and merge.
Build	-	Continuous integration and builds.
Test	-	Continuous testing that provide feedback.
Package	-	Packaging code and release to artifact repository.
Release	-	Release, promotion and change management.
Configure	-	Automated Infrastructure and configuration management.
Monitor	-	Application performance and log monitoring.

DevOps Course Goals

In this course we will cover the tool chain that will cover the **code, build, test, package**, part of release and part of application monitoring.

The course will enforce the fundamentals around DevOps and help you understand in depth of how these tools fit together will help you transform your organization and teams and also help you set expectations across the business and technology teams.

DevOps At End of Course

At the end of course you will be able to understand the benefits of DevOps model and be able to setup and use the tools that will facilitate in source control management, continuous integration, testing, release management, efficient deployment workflows and application monitoring . We will be using some of the best open source tools available in the industry.

In addition to learning the tools you will also have a complete DevOps environment setup on your computer that you can use for further learning and provide you options to explore further.

Assumptions

- Basic understanding of the SDLC lifecycle.
- Basic understanding of the source code management.
- Ability to install tools as part of the course.
- Ability to configure tools as part of the course.
- Basic understanding of Java programming language and build tools like Maven.
- Ability to hands on development in Java using Spring Boot.
- Please note that though the focus here is using Java programming the tool chain can be applied to any technology.

Course Prerequisites

- Access to Mac or Windows computer.
- Access to Internet for downloads.
- Admin rights to the computer.
- JDK 1.8.x installed.

Acronyms:

SDLC	-	Software Development Lifecycle.
SCM	-	Software Configuration Management.
ELK	-	Elasticsearch, Logstash and Kibana.
GitFlow	-	Git Workflow.
SSH	-	Secure Shell.
GIT	-	Version Control System.
CI	-	Continuous Integration.
Repository	-	Place where artifacts are stored.

Section 2 - Required Tools

- GitHub Account: https://github.com/join
- Install GIT: https://www.git-scm.com/downloads
- Install Atlassian SourceTree: https://www.sourcetreeapp.com
- *Optional: Install GIT FLOW for executing via Command Line*

  https://github.com/nvie/gitflow/wiki/Installation

Section 2 - Download of Tools

- Download and Install STS: https://spring.io/tools/sts/all
- Download Editor like ATOM and Install: https://atom.io
- Download Maven: http://maven.apache.org
- Download Tomcat: https://tomcat.apache.org/download-80.cgi
- Download Artifactory: https://www.jfrog.com/open-source/
- Download Jenkins: https://jenkins.io/download/
- Download ElasticSearch: https://www.elastic.co/downloads/elasticsearch
- Download Logstash: https://www.elastic.co/downloads/logstash
- Download Kibana: https://www.elastic.co/downloads/kibana

Section 3 - Install of Tools

1. Install Tomcat
   a. Install to Development.
   b. Start tomcat.
   c. Validate setup.
2. Install Artifactory
   a. Install to Development.
   b. Startup artifactory.
3. Install Maven.
   a. Install to Development.
   b. Setup Maven Home.
   c. Validate setup.
4. Install Jenkins
   a. Install.
   b. Start Jenkins.
   c. Validate setup.

Section 3 - Continued

1. Setup Artifactory:
    a. Startup Artifactory
    b. Setup Admin account.
    c. Port: 8081
2. Setup Tomcat:
    a. Startup tomcat.
    b. Update tomcat-users.xml.
    c. Port: 8080
3. Setup Jenkins:
    a. Setup Jenkins.
    b. Port: 8090
4. Setup Maven:
    a. Startup maven home.
    b. Generate maven security.xml and settings-security.xml.

Section 4 - Development

1. Setup SSH for GitHub.
2. Create a GitHub repository.
3. Create a Spring Boot Project.
4. Import Project to STS.
5. Configure for SCM and Artifactory.
6. Commit changes to the feature branch and push to GitHub.
7. Create a pull request to develop branch and merge.

Section 5 - Continuous Integration

1. Setup SSH and Maven Settings for Jenkins User.
2. Install and Configure Jenkins Plugins.
   a. Conditional Buildstep Plugin
   b. Deploy to container Plugin
   c. Environment Injector Plugin
   d. Git Parameter Plugin
   e. GitHub Branch Source Plugin
3. Build SNAPSHOT version using Jenkins.
4. Configure Jenkins to Deploy to Tomcat.

Section 6 - GitFlow Release (1.0.0)

1. Start a Release Flow and Update version on develop.
2. Tag the code for Release.
3. Release code to QA from Tag.
4. Release code to PROD from Artifactory.
5. Finish Current Release.

Section 7 - GitFlow Release (1.2.0)

1. Code changes on the feature branch and commit code.
2. Merge the code from feature to develop.
3. Code Deployed to DEV.
4. Start a Release Flow and Update version on develop.
5. Tag the code for Release.
6. Release code to QA from Tag.
7. Release code to PROD from Artifactory.
8. Finish Current Release.

Section 8 - GitFlow Release (1.2.1)

1. Create a bugfix release.
2. Commit and merge changes from bugfix feature to bugfix release.
3. Create a tag from bugfix release and deploy to DEV.
4. Release code to QA from Tag.
5. Release code to PROD from Artifactory.
6. Finish BugFix Release.

Section 9: Application Monitoring

1. Start a Release for Logging enhancements.
2. FInish Release 1.3.x.
3. Install and Configure ElasticSearch and Kibana.
4. Install and Configure Logstash.
5. Configure Kibana Dashboard.
6. Review and Monitor logs.

Section 10 - Wrap Up

1. We have learned the tool chain to support several phases of the DevOps model.
2. We have hands-on experience in configuring the tools and understanding how they fit in.
3. We have gained beyond basic understanding of the tools and phases.
4. We could use the learning from this course and expand on customizing as per organizations need.
5. We have gained more knowledge about DevOps.

Section 11 - Integrate SonarQube (Bonus)

1. Install and Configure SonarQube.
2. Update Maven settings.
3. Code changes on the develop branch and commit changes.
4. Configure Jenkins for Sonar Integration and run the Sonar Build.
5. Review the Sonar Report for the project.
6. Start the release and merge to master.

Section 11 - Jenkins Pipeline (Bonus)

1. Review and understand the Jenkins Pipeline.
2. Code Changes on develop branch for JenkinsPipeline configuration.
3. Create a Jenkins Job for Pipeline Integration on the default develop branch.
4. Run the pipeline job and review the results.
5. Start the release and merge to master.

Section 12 - GitHub JIRA Integration Part I (Bonus)

1. JIRA Confluence Overview.
2. JIRA Workflow Overview.
3. GitHub Configuration for JIRA.
4. JIRA configuration for GitHub.

Section 13 - GitHub JIRA Jenkins NGROK Integration NGROK Part II (Bonus)

1. NGROK Overview.
2. GitHub webhook for NGROK changes.
3. Code changes per the story and push the feature branch to GIT.
4. Review the webhook trigger and results in Jenkins, GitHub and JIRA.