# **CI & CD – High level Documentation**

**Flow**

PROD Env

Master Branch

Merge Release branch into Master with unique Tag

Higher Environments

CI/CD Pipeline

Create Release branch off Master

QA Env

DEV Env

CI/CD Integration

GIT CHECK OUT

This process will be followed for every release

SONAR

Release Branch

CHECKMARX

Merge feature/hotfix into Release

JUNIT

Create feature/hotfix branch off Release

Feature Branch - 1

DEPLOYMENT

NEXUS

Feature Branch - 2

SELENIUM TEST

Hot Fix/Bug Fix

BUILD PROMOTION

# **Branch Details:**

# **Master Branch:**

This branch will contain the current **PROD** code changes.

# **Release Branch:**

This branch will be checked out from **MASTER branch** by the developer which will contain the code changes for the current release.

~~This is a single branch and will not be created for every release~~.

We can have release branch for each release (like, release-1.0, release-2.0). In this way, we can maintain repository for every release and have the track of repository and changes made for each release.

# **Feature Branch:**

This branch will be checked out from **DEVELOP branch** by the developer for developing the feature/story for the current release and it will be merged to release branch of the current release and moved sequentially from DEV to PROD.

**Naming convention**: feature/{task name}

**Example**: feature/flex-okta-edge-service.

# **BugFix or HotFix Branch:**

This branch will be checked out from **DEVELOP branch** by the developer for bugs/issues identified in DEV or QA Environment and it will be merged to release branch of current release and moved sequentially from DEV to PROD

**Naming convention**: hotfix/{bug name}

**Example**: hotfix/flex-okta-login-auth-issue.

# **CI/CD Process Details:**

1. Current projects from SVN should be moved to Git. This will be the Master branch.
2. Release branch (*may be 1.0*) will be created off from Master.
3. Feature branch for every feature will be created off from Release branch.
4. Once code changes are done for the feature, feature branch should be merged to Release branch.
5. CI/CD Engine will checkout the code changes from Release branch and does the automated build and deployment with below process.
   1. SONAR will be executed, and it will check for code quality issues/warnings.
   2. Checkmarx will be executed, and it will run a security scan for checking any vulnerability in the code.
   3. Junit will be executed, and it will run the unit tests.
   4. Once all the above process is successful, changes will be deployed into the DEV Environment.
6. Once Unit Testing is completed in DEV Environment, Nexus Repository Manager should be triggered, and it does the below process.
   1. Automated Selenium scripts should be executed.
   2. Once the above step is successful, war will be copied from DEV to QA Environment by Nexus.
   3. Once QA Validation is completed, Nexus script should be triggered again which will copy the war from QA and deploy it in PROD Environment.
7. Once PROD Deployment is completed, below steps needs to be done.
   1. Changes from Release should be Up merged to Master with a unique Tag.
   2. Version in the pom.xml should be incremented.
   3. New Release branch (*may be 2.0 or 3.0*) should be created from old release branch (*may be 1.0*) and that new release branch should be used in the current release cycle.

**Building a Deployment Pipeline using Git, Maven, Jenkins and GlassFish**

Develop

Manage

Build & Test

Host

Eclipse IDE

Git Repository

Jenkins CI Server

JUnit

Glassfish Dev Environment

Glassfish Testing Environment

Glassfish Production Environment

Commit

Update

Clone

Fetch

Deploy

Deploy

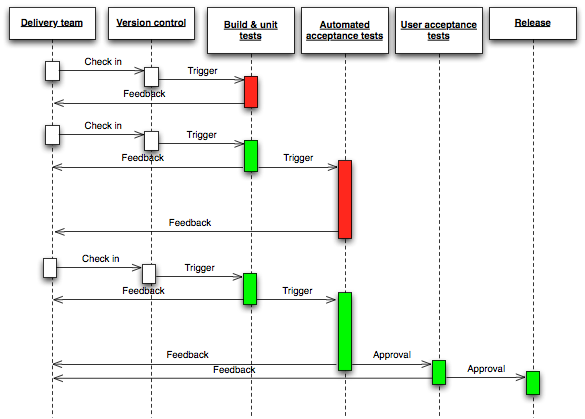
Deploy

Test

Maven

Manage Build & Dependency

**Continuous Integration and Continuous Delivery**



**Continuous Integration and Continuous Delivery consists of the following open source components:**

* Git for source code repository
* Jenkins for Building, Continuous Integration and creating Continuous Delivery pipelines.
* Maven Script to define the dependency artifacts and build script.
* Nexus as the artifact repository for caching public artifacts and hosting canary and real release artifacts.
* Checkmarks & SonarQube provides a platform to maintain code quality
* Jenkins Pipeline Library to help reuse Jenkins Pipeline functions and steps across projects.

**Git Bash Commands (Basics)**:

1. For cloning a project from Git to Local machine.

**git clone *<clone\_url>*.**

This will always clone and install the **master** branch.

1. For cloning a specific branch from Git to Local machine.

**git clone *<clone\_url>* -b *<branch\_name>***

1. To view the list of new or modified files that you have made in your local repository.

**git status**

1. Git Staging - To stage/add the files for the next commit,

**git add**

1. Git commit with message

**git commit -m ‘<*commit\_message*>’**

This will commit the changes to your local machine repository

1. To pull the latest changes from remote repository

**git pull**

1. Git Push

**git push**

Alternatively, we can make use of Tortoise Git tool for managing all the above commands.

**Git Cheat Sheet:**