JARDINE LLOYD THOMPSON INDIA

**January 2017**



JARDINE LLOYD THOMPSON INDIA

**Table of Contents**

[1. Executive Summary 4](#_Toc498094785)

[2. Overview of Continuous Integration 4](#_Toc498094786)

[3. Continuous Integration with GitHub using Jenkins 5](#_Toc498094787)

[3.1. Pre-requisites: 5](#_Toc498094788)

[3.2. Setup Jenkins and GitHub on the build server 5](#_Toc498094789)

[3.3. Configure project for continuous build 5](#_Toc498094790)

[3.4. Notifications between GitHub and Jenkins 6](#_Toc498094791)

[3.4.1. Jenkins changes. 6](#_Toc498094792)

[3.4.2. GitHub changes. 6](#_Toc498094793)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Created By & Date | Reviewed By & Date | Comments |
| 0.1 | Ronak Kapadia  08 November 2017 |  |  |
|  |  |  |  |

# Executive Summary

This document outlines detailed overview about continuous integration using Jenkins. This document covers following areas:

1. Continuous integration overview and Jenkins as continuous integration server
2. Prerequisites for continuous integration using Jenkins
3. Plug-ins required for CI using Jenkins and GitHub
4. Technical implementation (Configuration changes) in Jenkins and GitHub

# Overview of Continuous Integration

Continuous Integration (CI) is a development practice of automating the build and testing of code every time a team member commits changes to version control. The continuous integration (CI) enables rapid development with superior quality of the product. There are major three steps in Continuous integration:

* + - Continuous Build: Whenever there is a check-in from developer into the source control, an automated build is triggered. Based on the result of the build, we can identify if there are any compilation issues with the latest checked-in code.
    - Automated Test case execution: If successful, the code is deployed in SIT environment, where automated test cases (Unit and API) are executed. This will ensure that there are no logical issues with the latest checked-in code. Also, it will help to determine if the checked-in code is not impacting any existing functionality.
    - Continuous Deployment: Once the automated test case execution is complete, based on the results, the code is merged from development branch to the master branch. (Typically once in a week). This code is then deployed into the UAT environment for regression testing.

There are many Continuous Integration tools available in the market such as TFS, TeamCity, CircleCI, Jenkins, etc. **One of the most popular and flexible Continuous Integration Tool is Jenkins.**

Jenkins is an open-source CI tool written in Java. It originated as the [fork of Hudson](https://jenkins-ci.org/blog/2011/01/11/hudsons-future/) when the Oracle bought the Sun Microsystems. The advantages of Jenkins are:

* Open source.
* Cross platform
* Flexible – with plugins.

# Continuous Integration with GitHub using Jenkins

# Pre-requisites:

* Jenkins setup (https://jenkins-ci.org/)
* Github setup
* .Net SDK (if not present)
* Nuget.exe
* Internet connectivity on the Jenkins server
* A service account in GitHub.

# Setup Jenkins and GitHub on the build server

1. Install Jenkins on the build server using Jenkins setup.
2. On successful installation, a windows service with name Jenkins will be created. The default installation location will be Program files(X86)\Jenkins.
3. Go to services. Open Jenkins service. Change Log on as credentials to service account credentials.
4. Open URL: <http://localhost:8080/> (Jenkins is by default installed on 8080 port. If you want to change the port number, change it from <installation Dir>\Jenkins.xml file)
5. Enter Initial administrator password. It will be available at: <installation dir>\secret\initialAdminPassword.
6. Install required plug-ins. Some of the important plug-ins are:

MS Build Plugin

Git Plugin

GitHub Plugin

We can install plug-ins at a later stage also from: Manage Jenkins 🡪 Manage Plugins. Once plugins are installed, manually restart the Jenkins service.

1. Go to: Manage Jenkins 🡪 Global tool configuration. Configure Git and MSBuild Path.
2. Install Git on the build server. Generate Public-private key combination for Service account. Add public key (SSH) in GitHub settings of service account.
3. Add Credentials into Jenkins. Go to Jenkins 🡪 Credentials 🡪 Domain 🡪 Click on Add Credentials.

* For public repositories: Create one credential with Username (Github Username) and password (Github Password).
* For private repositories: Create credential with SSH Username (Github Username) and private key.

# Configure project for continuous build

1. Go to Jenkins 🡪 New Item 🡪 Enter Build name 🡪 Select Freestyle project 🡪 Click on OK
2. Go to Source code management 🡪 Select Git Radio button
3. Enter Repository URL (GitHub URL) 🡪 Credentials

* For public repository use username and password
* For private repository use username and SSH Key.

1. Select Branches to Build:

Branch Specifier: \*/<Branch name>

1. Build Environment:

Select Delete Workspace before build starts

1. Execute windows batch command:

Step 1: Remove published websites before build starts.

RD /S /Q "C:\ClientPortal\Published"

Step 2: Restore nuget packages for the solution.

"C:\Jenkins Setup\NuGet.exe" restore "C:\ClientPortal\Workspace\ClientPortal\Interactive.Portal.sln"

Step 3: Build projects (API and web) using MSBuild.

"C:\Program Files (x86)\MSBuild\14.0\Bin\msbuild.exe" "Interactive.Portal.Api\Interactive.Portal.Api.csproj" /T:Build /p:Configuration=RELEASE /p:VisualStudioVersion=12.0 /p:OutputPath="C:\ClientPortal\Published\API" /clp:NoSummary /p:DebugSymbols=false /p:DebugType=None

"C:\Program Files (x86)\MSBuild\14.0\Bin\msbuild.exe" "Interactive.Portal.Web\Interactive.Portal.Web.csproj" /T:Build /p:Configuration=release /p:VisualStudioVersion=12.0 /p:OutputPath="C:\ClientPortal\published\Web" /clp:NoSummary /p:DebugSymbols=false /p:DebugType=None

Step 4: Copy the published websites to the deployment location.

XCOPY /s /y "C:\ClientPortal\Published\Web\\_PublishedWebsites\Interactive.Portal.Web\\*.\*" "C:\inetpub\wwwroot\Interactive.Portal\Interactive.Protal.Web\"

XCOPY /s /y "C:\ClientPortal\Published\API\\_PublishedWebsites\Interactive.Portal.API\\*.\*" "\\10.1.3.4\c$\inetpub\wwwroot\Interactive.Portal\Interactive.Portal.API\"

XCOPY /s /y "C:\ClientPortal\web.configs\web\web.config" "C:\inetpub\wwwroot\Interactive.Portal\Interactive.Protal.Web\web.config"

XCOPY /s /y "C:\ClientPortal\web.configs\API\web.config" "\\10.1.3.4\c$\inetpub\wwwroot\Interactive.Portal\Interactive.Portal.API\web.config"

# Notifications between GitHub and Jenkins

# Jenkins changes.

1. Install Plug-in in Jenkins: embeddable- Build-plugin
2. Go to Jenkins🡪 Build 🡪 Configure 🡪

Select Build Triggers:

Select GitHub hook trigger for GITScm polling

# GitHub changes.

1. Open GitHub. Select the project 🡪 Go to settings
2. Click on ‘Integrations and Services’🡪Click on Add Services

Enter Jenkins hook URL as: <Jenkins Url>/github-webhook/