# README FILE

This package is designed to demonstrate the Continuous Integration of a Purely Platform Agnostic Application profile of NonStop DevOps through a *javaquickstart* application.

The Pre-requesite setup, tooling, gating criteria and artifacts like the JenkinsFile can be used to automate the Continuous Integration Phase of any Java Project. This package acts as a starter kit for users.

### Package Details

The package contains a java net client application, a server application JenkinsFile that can be used to automate the Continuous Integration Workflow and this README file that provides instructions to use the *javaquickstart application.*

#### Client Application

The Client application (JavaHelloClient) is a sample Java Client application used to demonstrate Continuous Integration. The application that uses the java.net package.

The application takes two inputs <IPAddress> of the server to connect to and the <PortNumber> on which the server is listening at.

The client can be run on-platform (NonStop) or off-platform(windows).

#### Server Application

The Server application (JavaHelloServer) is a sample Java Server application to demonstrate Continuous Integration. The Server application waits for requests from client application. Once a connection request is made by the client, the server connects to the client and exchanges greeting message. The Server application takes two inputs, namely the <HostName> and the <PortNumber> on which the server will listen. The server can be run on-platform (NonStop) for this demo. It can be run as an OSS process or in Pathway Environment.

### Setup

#### Pre-requisites software

The following are the pre-requisites on the system identified for Build Management.

The following are the pre-requisites on the system identified for the Java Starter Kit.

|  |  |  |
| --- | --- | --- |
| Software | Description | Version |
| Jenkins | Build Management System | 2.222.3 or later |
| GITHub | Source Code Control |  |
| GIT | Source Code Control | 2.26.2-64-bit or later |
| Java | Java Development and Runtime | 1.8.0\_181 or later version of JDK 8. |
| Maven | Standard Build Tool | 3.6.3 or later |
| SonarQube | Static Analysis Tools | 8.3.0.xx or later |
| Junit | Unit Testing Framework | 1.5.1 (included in the package) |
| Nexus Repository Manager | Binary Artifact Repository | 3.17.0.01 or later |

Jenkins plug-in namely (maven, git, static analysis tools, Nexus Artifact Uploader, Static Analysis Utilities, Static Analysis Collector Plug-in, Publish Over SSH, Junit, Pipeline)

Perform the setup as per the “HPE NonStop and Modern DevOps - Instructions Set for CI/CD” document accompanied with this package.

#### Downloading the project from HPE NonStop github

Log on to GITHUB and browse to <https://github.com/HewlettPackard/NonStop>

Click on Clone and download the HPE NonStop Samples as ZIP file using Download Zip option

Unzip the nsdevops folder that contains the NonStop DevOps starter kits.

Browse to the nsdevopsjavaquickstart for the Java Starter Kit.

#### Uploading the project to SCM

Make sure that you can log on to https://github.<youcompanyname>.com/

Create a new repository in GITHUB (NSDevOpsJavaQuickStart)

Copt nsdevopsjavaquickstart folder into a folder of your choice.

Using the GIT CLI upload the javaquickstart to the github using the following commands

git init

git add \*

git commit \* -m "Message"

git remote add origin https://devops-user:432c909d9851092fadb9895625fb926ad89616d6@github. <youcompanyname>.com/devops-user/ NSDevOpsJavaQuickStart.git

git push -u origin master

Update the items marked in red with values appropriate to your setup.

### Continuous Integration with JavaQuickStart

In DevOps, CI Phase is broadly addressed into two workflows. The usage of quick starts for each workflow is demonstrated in the following sections.

#### Developer Pipeline

This workflow triggers when a developer checks-in the code, it performs scm checkout, build, checkstyle analysis and unit testing,

Since many developers may be involved in development and for each the remote NonStop Node and environments vary, this is workflow is given as a **pipeline script** job.

A template could be created for the project and individual developers can use the template and update their specific environment details.

#### Jenkins Job Setup

* Create a Jenkins Pipeline Job SampleDeveloperPipeline
* In the General tab
  + Provide the Description

Example - Automated build triggered (Jenkins PollSCM build trigger), automated code check-out (by Jenkins from GITHUB), Automated Build (Jenkins Pipeline Script and build automation tools), Binaries copied to NonStop

Static Analysis of code (CheckStyle),Unit Tests run on NonStop (Junit)

* + Select GitHub Project and provide the SCM Link

https://<devops-user>:<your-personal-token>@github.<yourcompanyname>.com/<devops-user>/<NSDevOpsJavaQuickStart>.git

* In the BuildTriggers Tab,
  + Enable the PollSCM option and provide the schedule how often the SCM has to poll. Providing \* \* \* \* \* will poll every minute.
  + Note the spaces in the above pattern.
* In the Advanced Project Options,
  + Under Pipeline Definition, choose the Pipeline Script option.
  + Copy paste the script below after making appropriate changes to the repositories, node name, Jenkins workspace location \* everything marked in red in the script below

properties(

[

[

$class: 'BuildDiscarderProperty',

strategy: [$class: 'LogRotator', numToKeepStr: '10']

],

]

)

node ('MASTER') {

def mvnHome

def remote = [:]

remote.name = 'kronos'

remote.host = 'XX.YYY.ZZ.AA'

remote.user = 'devops.user'

remote.password = 'XXXXX'

remote.allowAnyHosts = true

stage('Code Checkout') { // for display purposes

// Get some code from a GitHub repository

git https://<devops-user>:<your-personal-token>@github.<yourcompanyname>.com/<devops-user>/<NSDevOpsJavaQuickStart>.git/'

// Get the Maven tool.

// \*\* NOTE: This 'M3' Maven tool must be configured

// \*\* in the global configuration.

mvnHome = tool 'Maven'

}

stage('Automated Build') {

// Run the maven build

// If Jenkins master is behind a firewall use the

// -DproxySet=true -DproxyHost=<host> -DproxyPort=<port>

withEnv(["MVN\_HOME=$mvnHome"]) {

if (isUnix()) {

sh '"$MVN\_HOME/bin/mvn" clean package'

} else {

bat(/"%M2\_HOME%\bin\mvn" package /)

}

}

}

stage('Static Analysis') {

// If Jenkins master is behind a firewall use the

// -DproxySet=true -DproxyHost=<host> -DproxyPort=<port>

bat(/"%M2\_HOME%\bin\mvn" checkstyle:checkstyle /)

}

stage('Binaries To NonStop') {

//Replace C:/Applications/Jenkins/ to the location where

//Jenkins is installed in your system

sshCommand remote: remote, command: "mkdir -p /home/devopsuser/javaquickstart"

sshPut remote: remote, from: 'C:/Applications/Jenkins/workspace/ SampleDeveloperPipeline/server/target/javahelloserver-1.0.0-SNAPSHOT.jar', into: '/home/devopsuser/javaquickstart/javahelloserver-1.0.0-SNAPSHOT.jar'

sshPut remote: remote, from: 'C:/Applications/Jenkins/workspace/SampleDeveloperPipeline/junit-platform-console-standalone-1.5.1.jar' , into: '/home/devopsuser/javaquickstart/junit-platform-console-standalone-1.5.1.jar'

sshCommand remote: remote, command: "cd /home/devopsuser/javaquickstart;ls -lrt"

}

stage('Unit Test') {

sshCommand remote: remote, command: "cd /home/devopsuser/javaquickstart;/usr/tandem/nssjava/jdk180\_l80/bin/java -jar junit-platform-console-standalone-1.5.1.jar -cp javahelloserver-1.0.0-SNAPSHOT.jar -c com.hpe.nsk.nsdevops.server.JavaHelloServer"

}

}//node

* Apply the changes and Save the Job.

#### Jenkins Job Execution

Since the Poll SCM option is turned on, when the developer makes a change and commits the code, the build is triggered. The Developer can monitor the Job through the Jenkins Job Console.

Alternately, the job can be triggered manually by clicking the Build Now option in the Jenkins SampleDeveloperPipeline

### Nightly Build Pipeline

This workflow triggers at a given time of the day. The pipeline performs scm checkout, build, checkstyle analysis using sun\_checks.xml, static analysis using SonarQube, unit testing using Junit and uploads the binaries into artifact repository.

Ensure SonarQube and Nexus Repository Manager are up and accessible.

#### Jenkins Job Setup

* Create a Jenkins Pipeline Job SampleNightlyBuildPipeline
* In the General tab
  + Provide the Description
  + Select GitHub Project and provide the SCM Link

https://<devops-user>:<your-personal-token>@github.<yourcompanyname>.com/<devops-user>/<NSDevOpsJavaQuickStart>.git

Update the items marked in red with values appropriate to your setup.

* In the BuildTriggers Tab,
  + Select the Build periodically option and provide the schedule how often the Job be should run. Providing H 10 \* \* \* will run the job at 10 am every day.
  + Note the spaces in the above pattern.
* In the Advanced Project Options,
* Under Pipeline Definition, choose the Pipeline Script from SCM option.
  + Select the SCM as GIT

Provide the GIT repository URL (https://<devops-user>:<your-personal-token>@github.<yourcompanyname>.com/<devops-user>/<NSDevOpsJavaQuickStart>.git)

* + Since the Git Authentication is based on Personal Token, select “none” for Repository Credentials.
  + Select /master for the branches to build and Auto for repository browser
  + Select script path as Jenkinsfile
* Apply the changes and Save the Job.

#### Jenkins Job Setup

Since the Build Periodically option is turned on, when the build get triggered automatically at the specified time. The Developer can monitor the Job through the Jenkins Job Console.

Alternately, the job can be triggered manually by clicking the Build Now option in the Jenkins SampleNightlyBuildPipeline