# README FILE

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

The Pre-requesite setup, tooling, gating criteria and artifacts like the JenkinsFile can be used to automate the Continuous Integration Phase of any JavaJNI 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 JavaJNIStarterKitUsageInstructions.docx file that provides instructions to use the *javajniquickstart 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, calls a native method to construct a greeting message and exchanges greeting message with the client. 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 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.

## Using the Starter Kit

The Java Starter Kit requires SCM, Jenkins. Perform the setup using the instructions set provided in the last section. Please note, when SCM is setup it is important to select

to select Unix-style line endings as mentioned earlier. This is because, if this option is not selected, the Makefile when moved to NonStop does not have ^M characters which will cause errors. Hence, make sure to have Unix-style line endings.

### 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 javajni for the Java JNI 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 (NSDevOpsJavaJNIQuickStart)

Copy the javajni contents into a folder of your choice.

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

git init

git add \*

git commit \* -m "<Message>"

git remote add origin https://<devops-user>:<your-personal-token>@github.<yourcompanyname>.com/<devops-user>/<NSDevOpsJavaJNIQuickStart>.git

git push -u origin master

where

NSDevOpsJavaJNIQuickStart is the name of your GITHUB repository

devops-user is the username

<your-personal-token> is the SCM Personal Token

github. <yourcompanyname>.com is the GitHub repository

Replace the items mentioned within <> and marked in red font with values appropriate to your setup.

### Continuous Integration Pipelines using Java JNI Starter Kit

### JNIDeveloper Pipeline

This workflow triggers when a developer checks-in the code. The pipeline performs scm checkout, builds on NonStop Node and unit testing using Junit & Boost Framework

The required software are packaged into nsdevopsjavajniquickstart .

Since many developers may be involved in development and each of their remote NonStop Node and environments vary, this is workflow is given as a scripted pipeline 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 JNIDeveloperPipeline
* 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>/<NSDevOpsJavaJNIQuickStart>.git

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

* 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 changes to the repositories, node name appropriately.
  + Please note the node name is case sensitive. Use the name as per the Jenkins configuration

properties(

[

[

$class: 'BuildDiscarderProperty',

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

],

]

)

node ('master') {

def mvnHome

def remote = [:]

remote.name = '<nsdev>'

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

remote.user = '<devops.user>'

remote.password = '<XXXX>'

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>/<NSDevOpsJavaJNIQuickStart>.git/'

mvnHome = tool 'Maven'

}

stage('Copy To NonStop, Build & Test') {

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

sshPut remote: remote, from: 'C:/Applications/Jenkins/workspace/ JNIDeveloperPipeline/', into: '/home/devopsuser/javajniquickstart'

sshCommand remote: remote, command: "cd /home/devopsuser/javajniquickstart; make"

}

}//node

Note : Update the items marked in red in the above syntax with values appropriate to your setup

* Apply the changes and Save the Job.

#### Jenkins Job Setup

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 JNIDeveloperPipeline

Each stage can be monitored by looking at the stage and logs per stage can be viewed.

Note – In this sample, the master branch has been used. However, in a typical application development scenario, the developers might work on specific branches. These can be integrated with the main/master branch if the build succeeds. This step can be automated in the developer pipeline. So that Nightly Build can be triggered.