# README FILE

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

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

### Package Details

The package contains a 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 *pythonquickstart application.*

#### Client Application

The Client application (socketClient.py) is a sample python Client application used to demonstrate Continuous Integration. The application that uses the socket module.

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

#### Server Application

The Server application (socketServer.py) is a sample python 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 <Host Name> and the <Port Number> 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 Python 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 |
| Python | Python is an interpreted high-level general-purpose programming language. | Python 2.7.18 or later version |
| SonarQube | Static Analysis Tools | 8.3.0.xx or later |
| Unit test or pytest | Unit Testing or Pytest Framework | Included in the python package (pip install pytest) |
| Nexus Repository Manager | Binary Artifact Repository | 3.17.0.01 or later |
| Python Coverage | Coverage, used to create coverage.xml | version 5.5 |
| Bandit | Source code security analysis tool for Python script | Version 1.5.1 |

Jenkins plug-in namely ( 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.

#### Uploading the project to SCM

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

Create a new repository in GITHUB (NSDevOpsPythonQuickStart)

Unzip the nsdevopspythonquickstart.zip file into a folder of your choice.

Using the GIT CLI upload the pythonquickstart 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/ nsdevopspythonquickstart.git.git

git push -u origin master

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

### Continuous Integration with PythonQuickStart

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>/<NSDevOpsPythonQuickStart>.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 changes to the repositories, node name appropriately.
* properties(
* [
* [
* $class: 'BuildDiscarderProperty',
* strategy: [$class: 'LogRotator', numToKeepStr: '10']
* ],
* ]
* )
* node ('master') {
* def remote = [:]
* remote.name = '<nsdev>'
* remote.host = '<XX.YYY.ZZ.AA>'
* remote.user = '<devops.user>'
* remote.password = '<XXXX>'
* remote.allowAnyHosts = true
* withEnv(["NEXUS\_VERSION=nexus3","NEXUS\_PROTOCOL=http","NEXUS\_URL= XX.YYY.ZZ.AA:8081","NEXUS\_REPOSITORY=python-nexus-repo","NEXUS\_CREDENTIAL\_ID=python-user","NEXUS\_CREDENTIAL\_PASSWORD=hpnonstop"]){
* stage('Code Checkout') { // for display purposes
* // Get some code from a GitHub repository
* git 'https://<devops-user>:<your-personal-token>@github.<yourcompanyname>.com/<devopsuser>/<NSDevOpsPythonQuickStart>.git/'
* }
* stage('Static Analysis') {
* // Run the pylint
* if (isUnix()) {
* sh "pylint --exit-zero --output-format=colorized ${env.WORKSPACE}/client/socketClient.py"
* sh "pylint --exit-zero --output-format=colorized ${env.WORKSPACE}/client/socketClient.py"
* // /usr/bin/bandit
* sh "/usr/bin/bandit -v -lll ${env.WORKSPACE}/server/socketServer.py"
* sh "/usr/bin/bandit -v -lll ${env.WORKSPACE}/client/socketClient.py"
* } else {
* bat("pylint --exit-zero --output-format=colorized ${env.WORKSPACE}\\client\\socketClient.py")
* bat("pylint --exit-zero --output-format=colorized ${env.WORKSPACE}\\client\\socketClient.py")
* }
* }
* stage("SonarQube Analysis") {
* // Run the maven build, not need for python application
* if (isUnix()) {
* sh "/opt/sonar-scanner/bin/sonar-scanner -Dsonar.projectName=\"PythonStartKit\" -Dsonar.projectKey=<project key> -Dsonar.projectBaseDir=/var/jenkins\_home/workspace/PythonKStart/ -Dsonar.sources=server,client -Dsonar.python.coverage.reportPATH=${env.WORKSPACE}/coverage.xml"
* } else {
* bat("C:\\Applications\\bin\\sonar-scanner -Dsonar.projectName=\"PythonStartKit\" -Dsonar.projectKey=<project key> -Dsonar.projectBaseDir=${env.WORKSPACE} -Dsonar.sources=server,client -Dsonar.python.coverage.reportPATH=${env.WORKSPACE}\\coverage.xml")
* }
* }
* stage('Binaries To NonStop') {
* sshCommand remote: remote, command: "mkdir -p /tmp/PythonKStart/server"
* sshCommand remote: remote, command: "mkdir -p /tmp/PythonKStart/client"
* sshPut remote: remote, from: 'server/socketServer.py', into: '/tmp/PythonKStart/server/socketServer.py'
* sshPut remote: remote, from: 'client/socketClient.py', into: '/tmp/PythonKStart/client/socketClient.py'
* sshCommand remote: remote, command: "cd /tmp/PythonKStart/;ls -lrt"
* }
* stage('Unit Test') {
* sshCommand remote: remote, command: "cd /tmp/PythonKStart/;/usr/bin/python server/socketServer.py & "
* sshCommand remote: remote, command: "cd /tmp/PythonKStart/client;/usr/bin/python socketClient.py"
* sshCommand remote: remote, command: "/bin/ps -eaf | /bin/grep socketServer | /bin/grep -v grep | /bin/awk '{ print \$2 }'| /bin/xargs /bin/kill -9"
* }
* stage('Regression Test') {
* //Add steps to copy regression test suite, setup test env and run the regression tests
* }
* stage('Artifact Repository Upload'){
* if (isUnix()) {
* echo "Create python package"
* ...
* ...
* sh "tar -cvf /tmp/PythonKStart-1.0.${BUILD\_NUMBER}.tar /tmp/PythonKStart > tar.log 2>&1"
* sh "gzip /tmp/PythonKStart-1.0.${BUILD\_NUMBER}.tar"
* sh "twine upload --verbose /tmp/PythonKStart-1.0.${BUILD\_NUMBER}.tar.gz --repository-url=http://XX.YYY.ZZ.AA:8081/repository/python-nexus-repo/ -u <user name> -p <password>"
* } else {
* bat("7z a -ttar -so dwt.tar C:\\PythonKStart | 7z a -si PythonKStart-1.0.${BUILD\_NUMBER}.tar.gz")
* bat("twine upload --verbose c:\\TEMP\\PythonKStart-1.0.${BUILD\_NUMBER}.tar.gz --repository-url=http://XX.YYY.ZZ.AA:8081/repository/python-nexus-repo/ -u <user name> -p <password>")
* }
* }
* }
* }//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

#### SonarCube Create Private Key

This section help us to create private key token from the sonarCube server and will be used from the Jenkins server while calling sonar-scanner with -Dsonar.projectKey=<> option.

Login to SonarCube user with username and password -> My Account (or http://<server ip>/account) -> security -> Enter Token Name (PythonStartKit) -> Generate

sonar-scanner -Dsonar.projectName=\"PythonStartKit\" -Dsonar.projectKey=<token> -Dsonar.projectBaseDir= ${env.WORKSPACE} /PythonKStart/ -Dsonar.sources=server,client -Dsonar.python.coverage.reportPATH=${env.WORKSPACE}/coverage.xml