# README FILE

This package is designed to demonstrate the Continuous Integration of a C Application profile of NonStop DevOps through cquickstart 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 simple C application, JenkinsFile that can be used to automate the Continuous Integration Workflow and this CStarterKitUsageInstructions.docx file that provides instructions to use the cquickstart *application.*

#### Server Application

The C application (helloC.c ) is a prints the sysinfo and sutversion details to the console

### 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 |
| C-C++ Cross Compilers | NonStop C-C++ Cross Compilers | L17.08 or later |
| Maven | Standard Build Tool | 3.6.3 or later |

Jenkins plug-in namely (Publish Over SSH, Pipeline)

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

## Using the C-C++ Cross Compilers on Windows

The C /C++ Cross Compilers for NonStop can be used in multiple ways on Windows. Given below are three common mechanisms. One or more of these is demonstrated using the starter kit

### Usage with NSDEE

NSDEE is a IDE environment on NonStop based on the Eclipse IDE. Follow the instructions in the NSDEE manual to setup NSDEE Development environment. The Cross Compilers are internally referenced by the NSDEE and the you can either use a Managed Make or a if you product has a Makefile, the same can be used. NSDEE is helpful in editing and updating the code based and when used with GIT plug-in, the code check-in into your organization GIT and a Jenkins GIT Hook or Poll SCM trigger can kick start the build process.

### Usage with Cygwin

Cygwin is a Unix/Linux like environment on Windows. It is a large collection on tools that provide functionality that is similar to Linux distribution on Windows. The C-C++ cross compilers for NonStop can be used with a Cygwin environment. Follow the instructions in the NSDEE Manual to setup the the Cygwin environment. Make is a build tool and is available in Cygwin. Ensure Make is installed.

Make can be used to build C-C++ applications for NonStop using the Cross Compilers. This is demonstrated in the C Starter kit (Makefile).

Note – Makefile in the starter kit assumes that the C-C++ Cross compilers version for L22.09 is installed and it is available at C:\Program Files (x86)\ HPE NonStop\L22.09. Hence the compiler rot env variable is set to C:\\PROGRA~2\HPENON~2\L22.09 (which is a short path version of the above location).

COMP\_ROOT=C:\\PROGRA~2\HPENON~1\L22.09

Please execute a dir /x command to find the short path version for where the cross compilers are installed and modify the variable appropriately.

### Usage with Windows Command line, Batch and PowerShell scripts

The C-C++ cross compilers can be executed directly on the windows command prompt and through batch and powershell scripts. For this it is important to find the path version of the location where the cross compilers are installed. This can be done using the dir /x command. Once done set the COMP\_ROOT to that path using the set command and then the compiler is ready to be used. A sample script is included in the compile.bat in the starter kit.

set COMP\_ROOT=C:\PROGRA~2\HPENON~1\L22.09

%COMP\_ROOT%\usr\bin\c89.exe helloC.c -o helloC.exe

## Using the Starter Kit

The C 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 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 c for the C 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 (NSDevOpsCQuickStart)

Copy the c contents into a folder of your choice.

Using the GIT CLI upload the c folder 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>/<NSDevOpsCQuickStart>.git

git push -u origin master

where

NSDevOpsCQuickStart 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 C Starter Kit

### C 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 nsdevopscquickstart .

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 CDeveloperPipeline
* 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>/<NSDevOpsCQuickStart>.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 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>/<NSDevOpsCQuickStart>.git/'

}

stage('Build & Test') {

make;

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

sshPut remote: remote, from: 'C:/Applications/Jenkins/workspace/ CDeveloperPipeline/helloC.exe', into: '/home/devopsuser/cquickstart'

}

}//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 CDeveloperPipeline

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.

# Continuous Integration with C Starter Kit and AzureDevOps

### AzureDevOps from Microsoft Azure is a DevOps for complete application lifecycle automation.

Seamless integration of NonStop Application Release and Lifecycle Management using Azure DevOps is possible today.

Azure DevOps supports both Microsoft hosted agent and a self-hosted agent. NonStop recommends using the Self-hosted agent for now as NonStop cross compilers can be hosted on the agent.

Follow the instructions given in Microsoft Documentation to setup the Self-hosted agent.

<https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/v2-windows?view=azure-devops>

Once this is setup, ensure that the cross compilers are setup in system where the agent will run. The starter kit can be used with AzureDevOps.

#### Azure DevOps project and pipeline

Login to Azure DevOps

Create a new project

Specify project name (cstarter) and description

Choose GIT as version control and work item process as BASIC & create the project

#### 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 c for the C Starter Kit.

#### Uploading the project to SCM

Copy C folder into a folder of your choice.

### Changes based on NonStop Build Environment setup

Based on the NonStop build environment setup, do the following changes

Cygwin – If Cygwin is installed, make command will work. Open the make file, check the COMP\_ROOT location.

Use dir /x to find the short path version of the location where the compilers are located. Update the following line in the make file

export COMP\_ROOT=C:\\PROGRA~2\HPENON~1\L22.09

Windows batch or PowerShell – If Cygwin is not installed, use the windows batch or PowerShell script for the build. Update following line in the batch script (compile.bat) based on the compiler location.

set COMP\_ROOT=C:\PROGRA~2\HPENON~1\L22.09

Using the GIT CLI upload the C starter kit to the github using the following commands

git init

git add \*

git commit \* -m "Message"

git remote add origin https://<azuredevopsuid>@dev.azure.com/<azuredevopsuid>/javastarter/\_git/ctartergit push -u origin master

git push -u origin all

#### Developer Pipeline

In the Azure DevOps portal, create a new pipeline

Select the cstarterproject as the project for which pipeline will be created

Choose AzureReposGit as the location where code resides.

Pipeline can either be created using maven option or using existing azure-pipelines.yml files

#### Using Azure to create and run pipelines

Once the project is select and repository location is given, the Azure DevOps service identifies the language type and provides suggestions for build pipeline creation.

Since the build has to be done on a Self Hosted agent, ensure the agent is running. Also, ensure the agent has Python and PIP installed. In the pipeline script generated update the following

pool:

  vmImage: ubuntu-latest

to

pool: Default

Save and Run the pipeline.

#### Using Existing Azure YAML Pipeline files

The NSDevOps python starter kit packages an azure-pipelines.yml file. Choose the Existing Azure YAML Pipeline files option and select the azure-pipelines.yml from the repository. Save and Run.