An IBM TechDoc from

IBM Z DevOps Acceleration Program

May, 2021 - Release 3.1

Building a Modern Pipeline on Mainframe

Proof-Of-Concept Cookbook

Nelson Lopez
Nelson.lopez1@ibm.com

Brice Small
brice@ibm.com

Tim Donnelly
donnellt@us.ibm.com

Abstract
Install, configure, and run a POC using Git, DBB, Jenkins and IDz

Table of Contents

1 Introduction

The purpose of this document is to outline the steps to install, configure and IVP[1] a zDevOps framework to run a basic Continuous Integration (CI) pipeline as a Proof of Concept (POC). While GitHub and Jenkins are used as examples any compliant Git Server and Orchestrator can be substituted.

1.1 Tools overview

The core tools used for the POC are listed below. Note that this guide shows the current GA versions at the time it was published and may not match screen and options in newer releases.

Rocket Software's Git – This component is Rocket Software's port of the Git SCM for USS. It is required to support Git operations in the USS environment for DBB-based builds.

IBM DBB Toolkit – This component contains the files necessary to perform DBB-based builds. This component is installed in the Unix Systems Services (USS) environment of z/OS.

IBM DBB Samples – This component contains the files that make up DBB sample groovy build framework called zAppBuild. It is a sample generic build engine for any mainframe program. This component is installed in the Unix Systems Services (USS) environment of z/OS.

IBM DBB WebApp – This component is a WebSphere Liberty application installed on a Linux server and is used to store metadata information such as dependencies between source modules, i.e. programs to copybooks, and dependencies between load library members, i.e. static calls between programs.

Jenkins Server and Agent – This component is installed on a Linux server and is used to orchestrate the steps in an automated DevOps pipeline. To do a DBB build, the Jenkins server uses a separately installed agent in USS.

IBM Developer for z/OS (IDz) – This product has two separate components. First, a mainframe component to listen for connection requests and access z/OS-based resources. Second, a Windows/Mac client component. This is the primary interface for the application developers. The IDz client contains intelligent source code editors, access to DBB functionality for personal builds and an interface to Git repositories. If assistance is required for IDz installation for either the client or server, the IBM Deployment Project Office (DPO) can be engaged.

1.2 Administrator roles

A z/OS Systems Admin will download DBB Toolkit, Git and sample scripts onto USS.

A Linux Admin will download, install, and configure the DBB WebApp.

A DevOps Admin(s) with an OMVS enabled z/OS account and read/write access to the installation folders will complete the configuration and execute automated builds.

[1] Installation Verification Procedures - IVP

2 Mainframe Installs - Git Client, DBB Toolkit, Groovy Samples

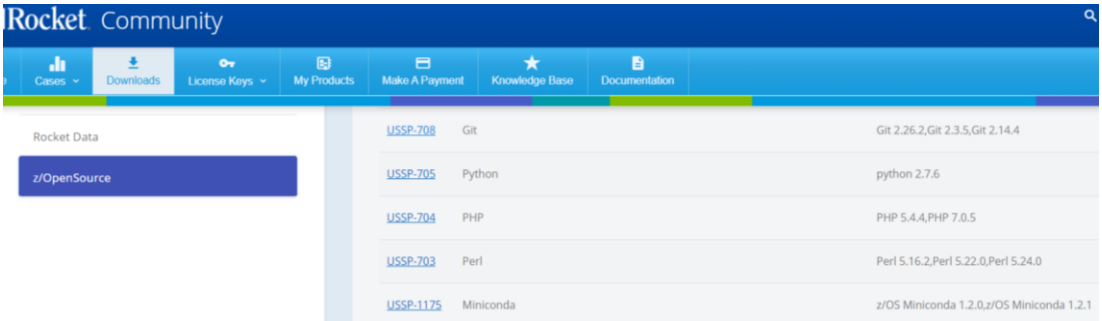
Required Role: z/OS Systems Admin (installation on USS)

Installs will be done on an LPAR where:

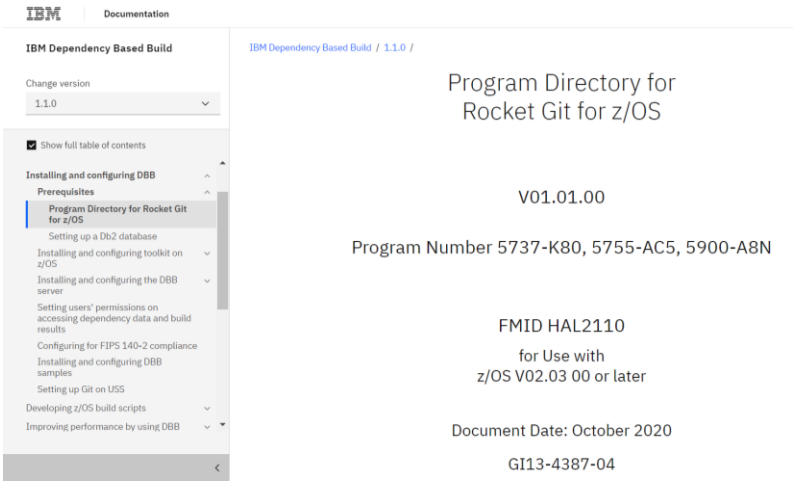
- An Admin can access the Unix Systems Services (USS) shell using putty or other tool.
- Developers can test DBB Builds (compile, links…)
- An Admin can clone from the public GitHub site (alternatives are available)
- DevOps Admins will migrate application source code from PDS(s) to Git
- SSH connectivity is established with Git Server, orchestrator like Jenkins and the DBB WebApp

2.1 Git Software for USS

Installation instructions on Rocket Software's Git client is available on their site https://www.rocketsoftware.com First time users must register to then navigate to the download page under "z/OpenSource". Start with the install of the latest version of Miniconda. Then follow the instructions on installing **Git, BASH and PERL.**



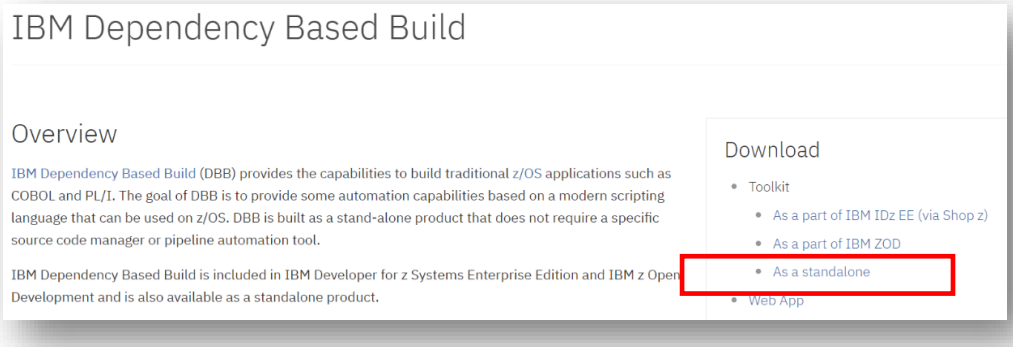
Alternatively, an SMP/E install of Git is available. Instructions are available in the latest DBB Documentation site https://www.ibm.com/docs/en/dbb



2.2 DBB Toolkit Download and Install

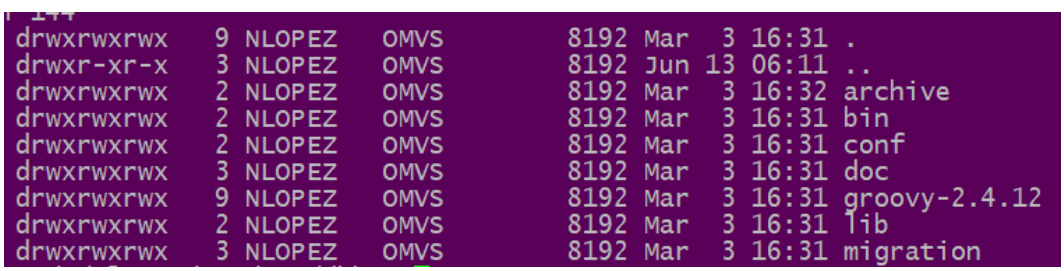
DBB's Toolkit is installed on z/OS's USS file system.
For DBB prerequisites visit https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.9/welcome.html[2]
A trial download is at https://ibm.github.io/mainframe-downloads/products/ibm-dependency-based-build.html
Download from the "As a standalone" link

- Under "Download" follow the link to "as a Standalone" for (dbb-ztoolkit-trial-1.x.x.tar)



- Copy the tar file with SCP or other tool to your USS home folder in binary format. You'll need 500mb of free space.
- Run the following commands:
  - `mkdir -p /usr/lpp/IBM/dbb`[3]
  - `tar -C /usr/lpp/IBM/dbb -xovf dbb-ztoolkit-trial-x.x.x.tar`
  - `chmod -R 755 /usr/lpp/IBM/dbb`
  - ***Note:** This folder is the DBB_HOME used in various configuration steps

Example DBB_HOME Dir.



```
drwxrwxrwx   9 NLOPEZ   OMVS        8192 Mar  3 16:31 .
drwxr-xr-x   3 NLOPEZ   OMVS        8192 Jun 13 06:11 ..
drwxrwxrwx   2 NLOPEZ   OMVS        8192 Mar  3 16:32 archive
drwxrwxrwx   2 NLOPEZ   OMVS        8192 Mar  3 16:31 bin
drwxrwxrwx   2 NLOPEZ   OMVS        8192 Mar  3 16:31 conf
drwxrwxrwx   3 NLOPEZ   OMVS        8192 Mar  3 16:31 doc
drwxrwxrwx   9 NLOPEZ   OMVS        8192 Mar  3 16:31 groovy-2.4.12
drwxrwxrwx   2 NLOPEZ   OMVS        8192 Mar  3 16:31 lib
drwxrwxrwx   3 NLOPEZ   OMVS        8192 Mar  3 16:31 migration
```

- Edit "**/usr/lpp/IBM/dbb/conf/gitenv.sh**" and replace all paths that start with "/rusr…" with "/var…" (i.e. the location where git binaries are installed).
- Ensure "/etc/profile" or the POC team's .profile has "export JAVA_HOME=/usr/lpp/java/J8.0_64 " (64 bit version of IBM's JAVA) see below
- Create or update a z/OS account(s) with an OMVS[4] segment, with at least a 1GB space for the Home directory and access to the installed DBB and Git folders. This account will be used by the DevOps(s) Admin to complete the configuration and conduct the POC.

[2] DBB 1.0.9 was the current GA release at the time this doc was published.
[3] This DBB_HOME should also be the default in the IDz Configuration
[4] Add OMVS to a user profile https://www.ibm.com/docs/en/zos/2.2.0?topic=racf-steps-defining-zos-unix-users

2.3 zOS USS .profile

Use IDz or TSO ISHELL to update or create a **.profile** file in your USS home directory as shown. Update the the paths used in the install. Note that DBB requires the 64bit version of Java. This .profile is used to initialize DBB, Git and Groovy environment for testing at the command line. Your DevOps Admin will copy this .profile into their USS home directory to complete the DBB Configuration and testing.

```
#! /bin/sh
export JAVA_HOME=/usr/lpp/java/J8.0_64
export IBM_JAVA_ENABLE_ASCII_FILETAG=ON
export DBB_HOME=$HOME/IBM/dbb
export GROOVY_HOME=$DBB_HOME/groovy-2.4.12
export DBB_CONF=$DBB_HOME/conf
export CLASSPATH=$CLASSPATH:$DBB_CONF   # added to pass DBB log4j props
. $DBB_HOME/conf/gitenv.sh

export PATH=$GROOVY_HOME/bin:$DBB_HOME/bin:$JAVA_HOME/bin:$PATH

## Skip 'SSL Cert Check' on USS when cloning with HTTPS
git config --global http.sslVerify false
```

2.4 Verifying z/OS installation (IVP)

Logon to USS and issue these commands to verify access and versions shown on the right of each command. Your version may be higher

- **`java -version`** - JRE 1.8.0 z/OS s390x-64-Bit
- **`git --version`** - 2.14.4_zos_b08
- **`groovyz -v`** - Groovy version: 2.4.12
- **`cat $DBB_HOME/bin/version.properties`** - version=1.0.9

If any command fails, review the installation instructions or contact your IBM representative.

2.5 Clone DBB Sample Scripts (zAppBuild)

You will need a github.com account for the next steps and access to the public GitHub site from USS. The appendix provides alternative ways on obtaining the samples.

Enter the following commands in USS:

- **`cd`**
- **`git clone git://github.com/IBM/dbb-zappbuild.git`**[5]
- **`cd dbb-zappbuild`**
- **`ls -lasT`**

dbb-zappbuild folder

```
NLOPEZ:/u/nlopez #>git clone git://github.com/IBM/dbb-zappbuild.git
NLOPEZ:/u/nlopez #>
NLOPEZ:/u/nlopez #>cd dbb-zappbuild/
NLOPEZ:/u/nlopez/dbb-zappbuild #>ls -lasT
total 304
  16                    drwxr-xr-x   7 NLOPEZ   OMVS        8192 Jun 13 09:18 .
  16                    drwxr-xr-x  57 NLOPEZ   OMVS        8192 Jun 13 09:18 ..
  16                    drwxr-xr-x   5 NLOPEZ   OMVS        8192 Jun 13 09:18 .git
  16 t ISO8859-1   T=on  -rw-r--r--   1 NLOPEZ   OMVS         896 Jun 13 09:18 .gitattributes
  16 t ISO8859-1   T=on  -rw-r--r--   1 NLOPEZ   OMVS         332 Jun 13 09:18 .project
  16 t ISO8859-1   T=on  -rw-r--r--   1 NLOPEZ   OMVS        4439 Jun 13 09:18 BUILD.md
  16 t ISO8859-1   T=on  -rw-r--r--   1 NLOPEZ   OMVS        1144 Jun 13 09:18 CONTRIBUTIONS.md
  16 t IBM-1047    T=on  -rw-r--r--   1 NLOPEZ   OMVS        1553 Jun 13 09:18 DCO1.1.txt
  16 t ISO8859-1   T=on  -rw-r--r--   1 NLOPEZ   OMVS        1236 Jun 13 09:18 INSTALL.md
  32 t ISO8859-1   T=on  -rw-r--r--   1 NLOPEZ   OMVS       11355 Jun 13 09:18 LICENSE
  16 t ISO8859-1   T=on  -rw-r--r--   1 NLOPEZ   OMVS        5370 Jun 13 09:18 README.md
  16                    drwxr-xr-x   2 NLOPEZ   OMVS        8192 Jun 13 09:18 build-conf
  48 t IBM-1047    T=on  -rw-r--r--   1 NLOPEZ   OMVS       17964 Jun 13 09:18 build.groovy
  16                    drwxr-xr-x   2 NLOPEZ   OMVS        8192 Jun 13 09:18 languages
  16                    drwxr-xr-x   4 NLOPEZ   OMVS        8192 Jun 13 09:18 samples
  16                    drwxr-xr-x   2 NLOPEZ   OMVS        8192 Jun 13 09:18 utilities
NLOPEZ:/u/nlopez/tmp/zapp/dbb-zappbuild #>git status
On branch development
Your branch is up-to-date with 'origin/development'.
```

[5] The default branch should be development

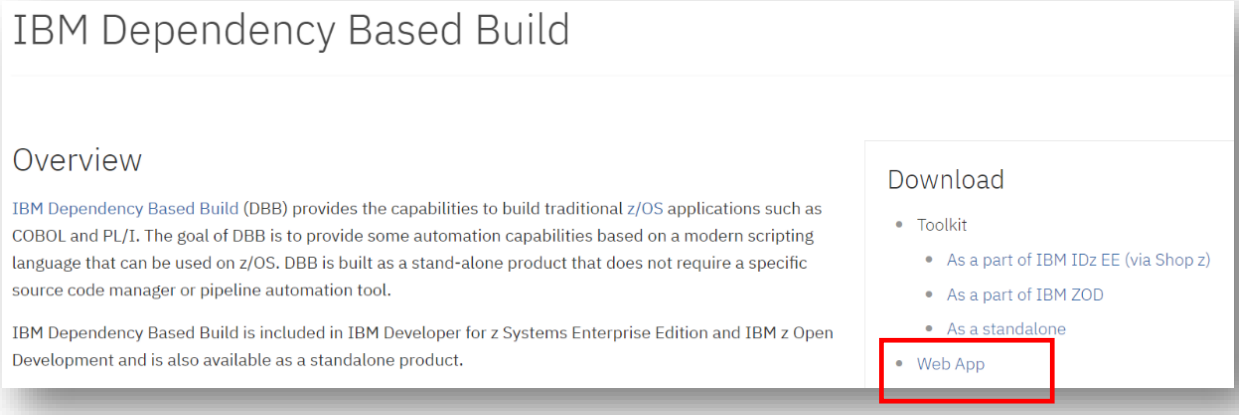
3 Distributed Install – DBB Server on Unix

Required Role: Linux Admin (installation of the DBB Web Application)

3.1 DBB Sever Download (Linux)

DBB has a WebApp that is installed on a Linux Server.
Trial download is at https://ibm.github.io/mainframe-downloads/products/ibm-dependency-based-build.html

Download from the "Web App" link



3.2 DBB Server Install and IVP - Linux

- The host Linux server will need connectivity to the target z/OS LPAR over port 9443.
- Under the "Download" section follow the link to "Web App" (dbb-server-**<version>**.tar.gz)
- Copy the tar file with SCP or another tool in binary format to the **Linux Server** root folder. You will need 500mb free space.
- Untar the file `tar -xzvf dbb-server-<version>.tar.gz`
- The DBB Server will be installed under "**../wlp**"
- Start the DBB Server `sh ../wlp/bin/server start dbb`
- Verify the installation using your Linux server's IP:
  - From your browser enter https://***your-srv-ip***:9443/dbb/rest/collection/setup
  - Ignore the message "Your connection is not private" and continue
  - Enter the default DBB user id and password ADMIN/ADMIN
  - The message "OK – Setup completed."
- Log files are in `../wlp/usr/server/dbb/logs`
- To stop DBB `sh ../wlp/bin/server stop dbb`
- https://**your-srv-ip**:9443/dbb is the URL to access DBB's Build Results going forward

4 DBB Build Configuration

Required skills: DevOps Admin

The DevOps Admin should perform the steps in this section. They will use an SSH terminal to access the USS shell with putty or another tool using their z/OS logon with an OMVS segment and home directory. They will also need privileges to clone an IBM repo from the public GitHub site.

4.1 Configure DBB's Sample Scripts (zAppBuild[6])

For this step you might need help from a z/OS Admin to identify system datasets like the cobol compiler.

- Logon to USS and copy the dbb-zappbuild folder cloned during the USS install steps using these commands.
  - cd
  - cp ?/u/sysprog/dbb-zappbuild .
- Use IDz or TSO ISHELL to access your copy of zAppBuild. Navigate to the build-conf subfolder (**dbb-zappbuild/build-conf)**.
- Edit the **build.properties** file and update the 'dbb.RepositoryClient.url' with the DBB Web Application's URL as provided by the Linux Admin.
- Also ensure lines 50 & 53 both have the value of "ADMIN". Remove the "#" comment from 53.

```
000043 # default DBB Repository web application authentication properties
000044 # can be overridden by build.groovy script options
000045
000046 # build.groovy option -url, --url
000047 dbb.RepositoryClient.url=https://dbbdev.rtp.raleigh.ibm.com:9443/dbb/
000048
000049 # build.groovy option -id, --id
000050 dbb.RepositoryClient.userId=ADMIN
000051
000052 # build.groovy option -pw, --pw
000053 #dbb.RepositoryClient.password=
```

[6] For the latest on zAppBuild see https://github.com/IBM/dbb-zappbuild

- Open **datasets.properties** and update the various system libraries. The z/OS Admin may be able to provide the DSNs. Line 20 can be left blank if the Cobol compiler V6 is used. Products not used at your site may be left blank.

```
000001 # Dataset references
000002 # Build properties for Partition Data Sets (PDS) used by zAppBuild build scripts.
000003 # Please provide a fully qualified DSN for each build property below.
000004 # Ex:
000005 # MACLIB=SYS1.MACLIB
000006
000007 # z/OS macro library. Example: SYS1.MACLIB
000008 MACLIB=
000009
000010 # Assembler macro library. Example: CEE.SCEEMAC
000011 SCEEMAC=
000012
000013 # LE (Language Environment) load library. Example: CEE.SCEELKED
000014 SCEELKED=
000015
000016 # High Level Assembler (HLASM) load library. Example: ASM.SASMMOD1
000017 SASMMOD1=
000018
000019 # Cobol Compiler Data Sets. Example: COBOL.V4R1M0.SIGYCOMP
000020 SIGYCOMP_V4=
000021 SIGYCOMP_V6=
000022
000023 # PL/I Compiler Data Sets.  Example: PLI.V5R2M0.SIBMZCMP
000024 IBMZPLI_V52=
000025 IBMZPLI_V51=
000026
000027 # CICS Macro Library. Example: CICSTS.V3R2M0.CICS.SDFHMAC
000028 SDFHMAC=
000029
000030 # CICS Load Library. Example: CICSTS.V3R2M0.CICS.SDFHLOAD
000031 SDFHLOAD=
000032
000033 # CICS COBOL Library. Example: CICSTS.V3R2M0.CICS.SDFHCOB
000034 SDFHCOB=
```

4.2 IVP DBB Build Environment

The following Installation Verification Procedure (IVP) will build a Sample Cobol App on USS using DBB.

- Use IDz or the TSO ISHELL command to create a **.profile** file in your USS home directory.
- Copy the **.profile** created and tested by the Systems programmer in the prior step.
- Logon to your USS shell and enter **`"groovyz -v".`** If the version is not displayed contact your Admin.
- Run the following commands on USS:
  cd
  mkdir dbb-logs
  groovyz dbb-zappbuild/build.groovy -w dbb-zappbuild/samples -a MortgageApplication -h $USER -o dbb-logs --fullBuild

Note - You can cut and paste the above onto putty. Use the right mouse click to paste.

Expected results – "Build State Clean"

```
NLOPEZ:/u/nlopez #>groovyz dbb-zappbuild/build.groovy -w dbb-zappbuild/samples -a MortgageApplication -h  $USER -o dbb-logs -fullBuild

** Build start at 20200613.110848.008
** Repository client created for https://dbbdev.rtp.raleigh.ibm.com:9443/dbb/
** Build output located at dbb-logs/build.20200613.110848.008
** Build result created for BuildGroup:MortgageApplication-development BuildLabel:build.20200613.110848.008 at https://dbbdev.rtp.raleigh.ibm.c
om:9443/dbb/rest/buildResult/79763
** --fullBuild option selected. Building all programs for application MortgageApplication
** Writing build list file to dbb-logs/build.20200613.110848.008/buildList.txt
** Invoking build scripts according to build order: BMS.groovy,Cobol.groovy,LinkEdit.groovy
** Building files mapped to BMS.groovy script
*** Building file MortgageApplication/bms/epsmort.bms
*** Building file MortgageApplication/bms/epsmlis.bms
** Building files mapped to Cobol.groovy script
*** Building file MortgageApplication/cobol/epsnbrvl.cbl
*** Building file MortgageApplication/cobol/epsmlist.cbl
*** Building file MortgageApplication/cobol/epsmpmt.cbl
*** Building file MortgageApplication/cobol/epscmort.cbl
*** Building file MortgageApplication/cobol/epscsmrd.cbl
** Building files mapped to LinkEdit.groovy script
*** Building file MortgageApplication/link/epsmlist.lnk
** Writing build report data to dbb-logs/build.20200613.110848.008/BuildReport.json
** Writing build report to dbb-logs/build.20200613.110848.008/BuildReport.html
** Build ended at Sat Jun 13 11:09:17 EDT 2020
** Build State : CLEAN
** Total files processed : 8
** Total build time  : 28.765 seconds
```

Description of command line arguments:

- **groovyz** – A DBB command to invoke build.groovy
- **dbb-zappbuild/build.groovy** – This is the main build script
- **-w dbb-zappbuild/samples** – The **workspace** root of the sample application
- **-a MortgageApplication** – A sample Cobol application's main folder
- **-h $USER** – An MVS HLQ for PDSs created by the build. PDSs are automatically allocated using defaults as in **dbb-zappbuild/build-conf/Cobol.properties**.
- **-o dbb-logs** – A build output folder
- **--fullBuild** – An argument to build all the files in the -a folder

View the build Logs:

- `cd dbb-logs`
- `ls` - list the build folders
- `cd build.`*`xxx`* - where *xxx* is the folder name (see log)
- `cat EPSMLIST.cobol.log` - show the sysprint of a compile

sample compiler output

```
NLOPEZ:/u/nlopez #>cd dbb-logs
NLOPEZ:/u/nlopez/dbb-logs #>ls
build.20200613.110848.008
NLOPEZ:/u/nlopez/dbb-logs #>cd build.20200613.110848.008/
NLOPEZ:/u/nlopez/dbb-logs/build.20200613.110848.008 #>ls
BuildReport.html       EPSCMORT.cobol.log     EPSMLIS.bms.log        EPSMLIST.linkedit.log  EPSMPMT.cobol.log      buildList.txt
BuildReport.json       EPSCSMRD.cobol.log     EPSMLIST.cobol.log     EPSMORT.bms.log        EPSNBRVL.cobol.log
NLOPEZ:/u/nlopez/dbb-logs/build.20200613.110848.008 #>cat EPSMLIST.cobol.log
PP 5655-EC6 IBM Enterprise COBOL for z/OS  6.1.0 P190905                Date 06/13/2020  Time 11:09:07   Page     1

Invocation parameters:
 LIB,CICS

        IGYOS4090-I   The "LIB" option specification is no longer required.  COBOL library processing is always in effect.
```

5 Jenkins, Git and DBB Integration and IVP

Required skills: DevOps/Jenkins/Git Admin

In this step you will configure Jenkins and a Git servers to run a DBB build on zOS. Refer to the Jenkins site for details on how to download and install Jenkins at https://www.jenkins.io/download/ and https://www.jenkins.io/doc/book/installing/.

Note: The following navigation applies to Jenkins version 2.235.2 and above. In most cases, common version agnostic navigation directions have been provided. However, some differences may be encountered for older or newer versions of Jenkins.

5.1 Jenkins Server Plugins

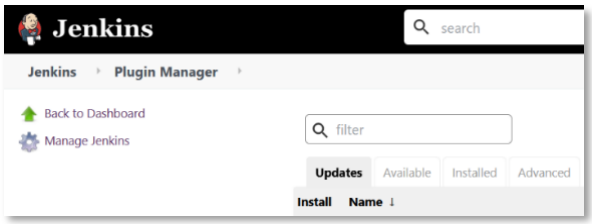
z/OS agents use standard SSH[7], OpenSSH_6.4p1, OpenSSL 1.0.2h and Rocket Git client using these required plugins:

- Credentials (2.3.13 and above)
- Git Client (version 3.5.0 and above is required for HTTPS-based access from USS)
- SSH Credentials Plugin (1.18.1 and above)
- SSH Build Agents Plugin (1.31.2 and above)
- Durable Task Plugin (1.35 and above)

Follow these steps to install or review plugins:

- From the Jenkins home page, using the MenuBar drop down select “Manage Jenkins” followed by the “Manage Plugins”.

- The “Plugin Manager” page should appear.



- In the "Available” tab, scroll and check off the required plugins. If these plugins are not “Available”, they may be on the “Installed” tab. In which case, review the minimum require version and update as needed.

[7] Most modern orchestrators provide similar SSH interfaces. Most Git servers also allow the use of SSH keys for remote access. For more on OpenSSH see https://www-01.ibm.com/servers/resourcelink/svc00100.nsf/pages/zOSV2R3SC276806/$file/foto100_v2r3.pdf

5.2 Jenkins Credentials

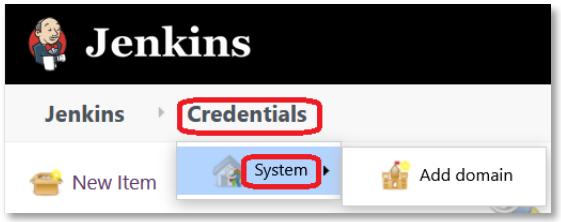
This POC will use the DevOps person's credentials. However, in a production environment the best practice is to create service accounts to access Jenkins, Git and zOS resources

Note: The following navigation applies to Jenkins version 2.235.2 and above. In most cases, common version agnostic navigation directions have been provided. However, some differences may be encountered for older version of Jenkins.

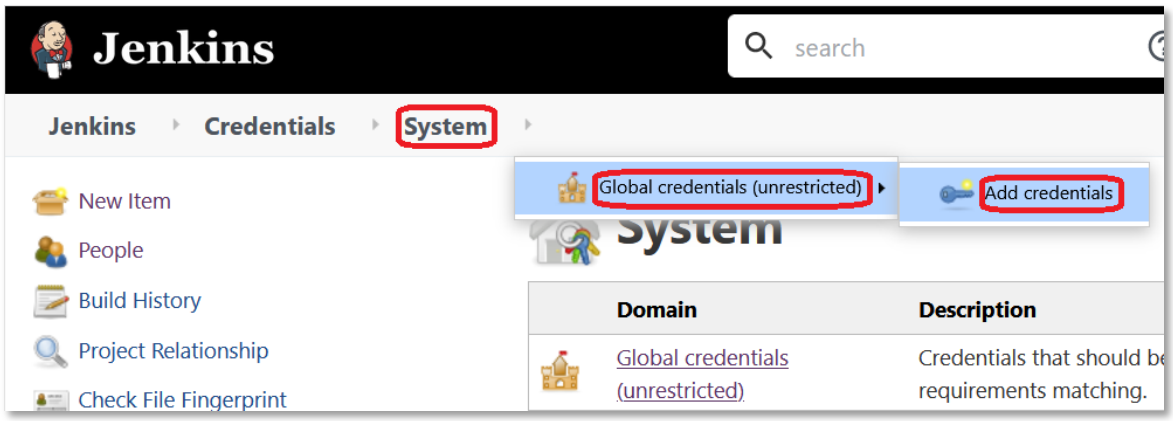
5.2.1 Add your z/OS ID to Jenkins

- From the Jenkins home page, using the MenuBar drop down select "Manage Jenkins" followed by the "Manage Credentials".

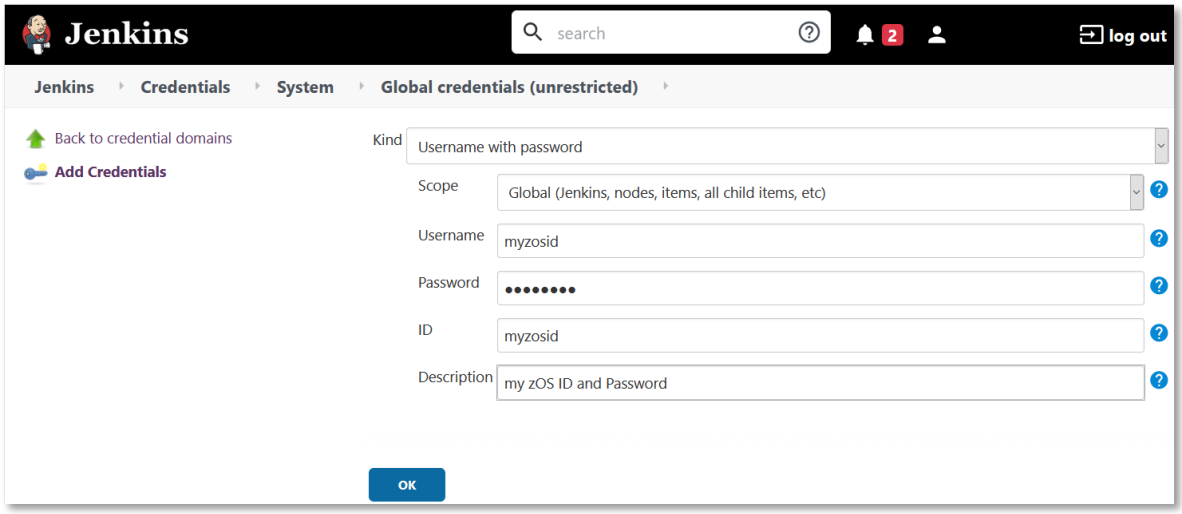
- Using the MenuBar, select "Credentials" followed by the "System" selection in the drop down.



- Using the MenuBar, select "System" followed by the "Global credentials" followed by the "Add credentials" selections in the drop downs.



- The “Global credentials – Add Credentials” page should appear



- Provide the following credentials
  - Kind - Set to “Username with password”[8]
  - Scope - “Global”
  - Username - z/OS RACF/ACF2 user id.
  - Password - z/OS RACF/ACF2 user password.
  - ID - A freeform ID used by Jenkins to reference this account.
  - Description - An optional description.
- Click “Ok” to save.

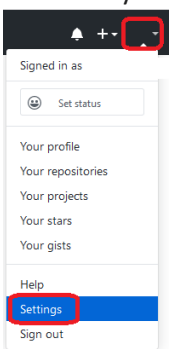
[8] Alternatively, generate SSH keys for a RACF/ACF2 OMVS acct and create a Jenkins credential using the KIND type of "SSH Username with private key". Username is the OMVS ID. Paste the generated **PRIVATE** key into the Jenkin's "Private Key" field. Copy the generated public key to the OMVS acct's authorized keys file "cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys".

5.2.2 Generate and add your z/OS SSH Public Key to Git

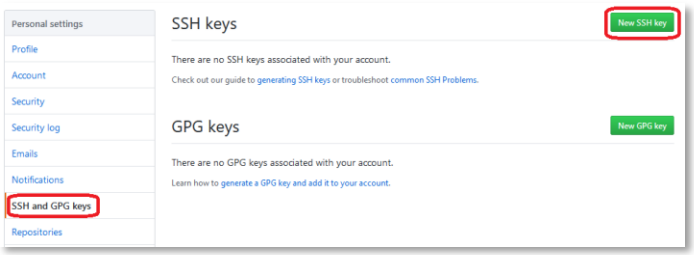
1. From the USS command line:
   - **cd**
   - **ssh-keygen -t rsa -b 4096 -C "?user@myOrganization.com"** - where ?user@... is your Git acct[9]
   - press enter and accept all defaults
   - **cat .ssh/id_rsa.pub**
   - copy the output as shown (no trailing blanks)

USS – generate SSH Keys

2. From your GitHub Account, select account "setting"



3. From the left pane, select the "SSH and GPG keys" option.



4. Select "New SSH key" option
   - Provide a Title (Ex: **user@myOrganization.com**)
   - Paste your **Public** key

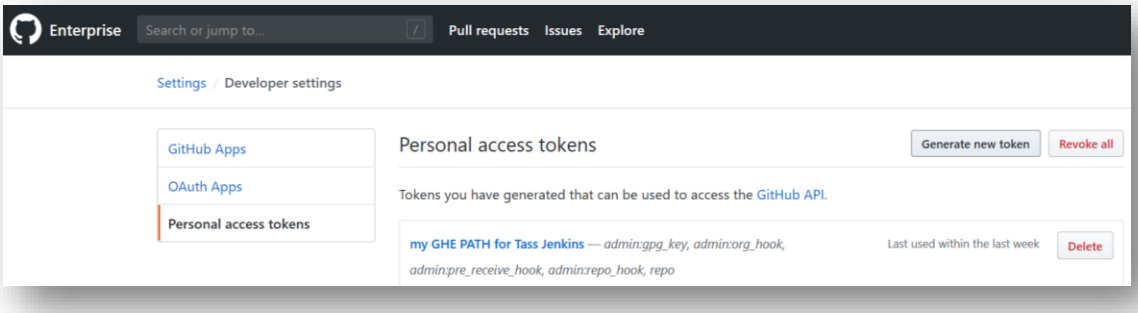
[9] Some sites require passing a user id in the comment field as part of the SSH key authentication. Check with your local Admin for details.

5.2.3 Add a Git Personal Access Token (PAT) to Jenkins

A GitHub PAT is used for HTTPS-based[10] Git requests from both Jenkins and USS.

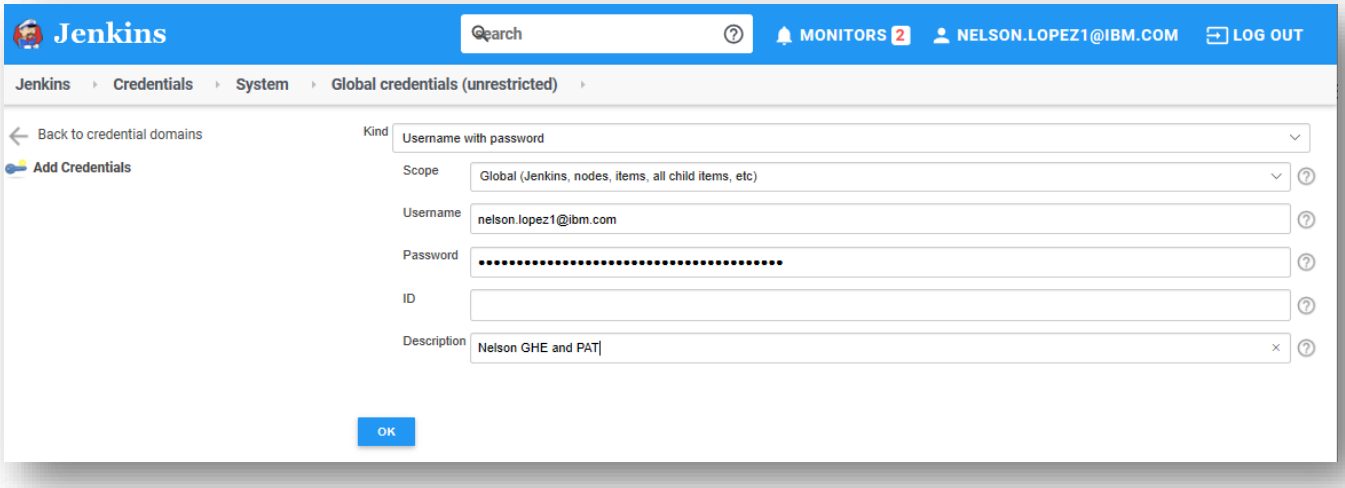
In GitHub

- Under your profile, select 'Settings', 'Developer settings' to access the 'Personal access tokens' page.
- Press 'Generate new Token' and give it a name.
- Enable all the options under "select scope".
- Press Generate token at the bottom of the page.
- Copy your new token for pasting in Jenkins.



In Jenkins

- Add a new Jenkins Credential of Kind "username with password".
- A Username like your Git Account ID but can be any value
- The password is your **PAT** (paste it in)
- Leave the ID field blank
- Provide a description to easily find this account when creating pipeline jobs.



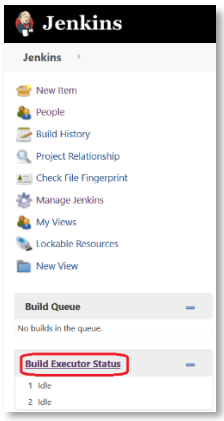
[10] This assumes unsigned HTTPS cert access to the Git server. If your server requires signed certs, work with your Security Admin. The default TrustStore is a .pem file in USS's '/tmp' dir using Microsoft's CA.

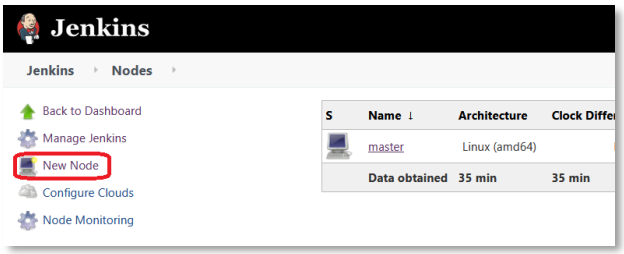
5.3 Configure a Jenkins Agent for USS

A Jenkins agent is a Java application that is deployed (via SFTP) from a Jenkins Server to a remote z/OS Host to run DBB build jobs.

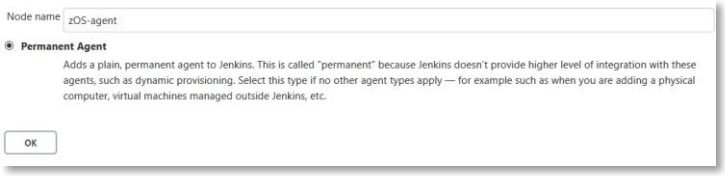
Note: The values on the screen below may not reflect the actual values of your installation. Please ensure the version and paths match your installed values.

- From the Jenkins home page, select “Build Executor Status” followed by the “New Node” selection in the left pane.

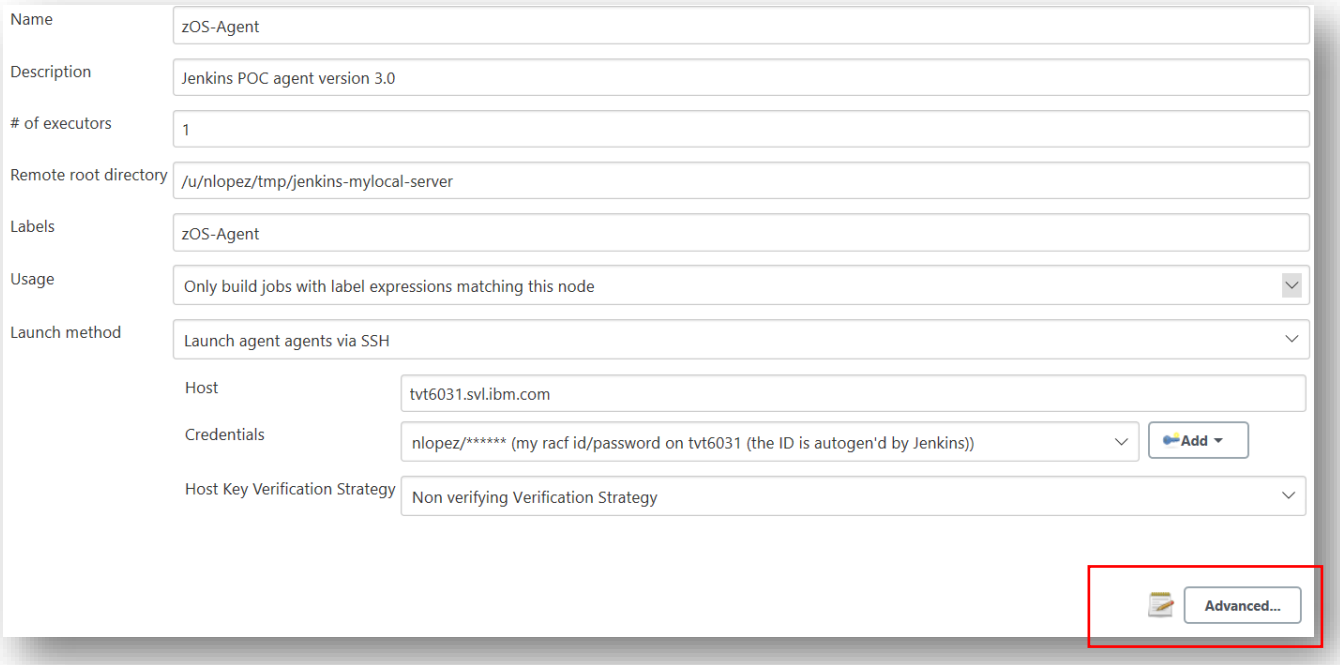




- Provide a name for your new agent “Node Name”, enable "Permanent Agent" and click "OK".

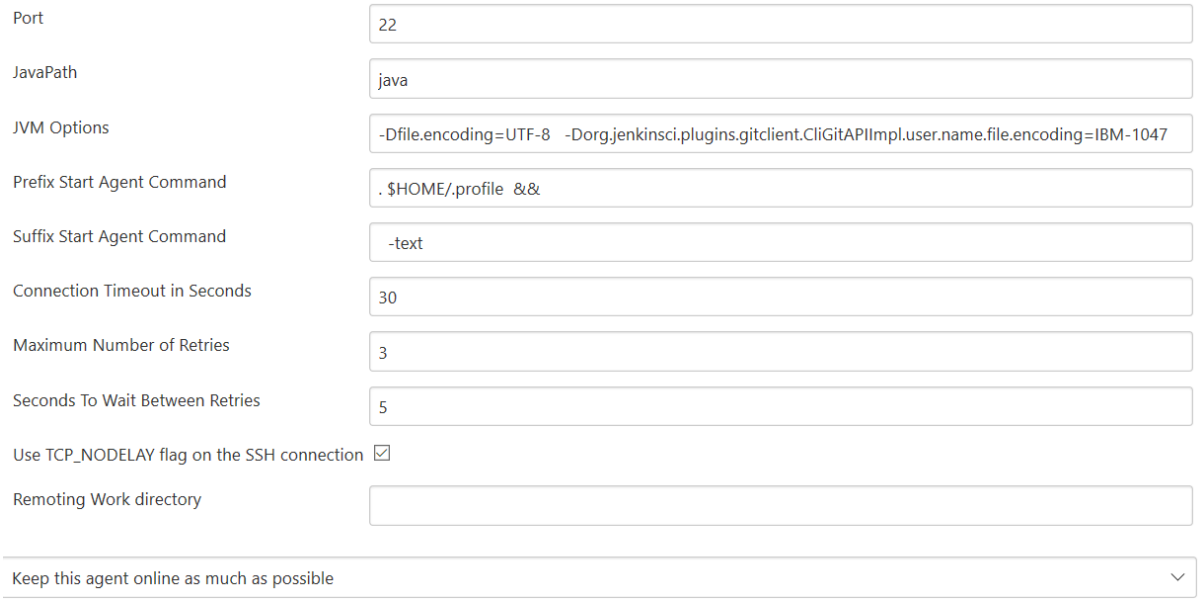


- Supply the following information



Field	Value	Notes
Name	zOS-Agent	Or any descriptive name for the agent.
# of executors	1	
Remote root directory	*?/jenkins-remote*	Where "?" is an *absolute* path to a folder in the SSH User's home directory. Jenkins-remote can be any name.
Labels	zOS-Agent	A label used in pipeline job(s) to point to this agent
Usage	Only build Jobs with label ...	
Launch Method	Launch agents via SSH	
Host	*?Your z/OS IP or DNS*	
Credentials	*?Your z/OS SSH Account*	From the dropdown select the previously defined z/OS SSH account.
Host Key Verification...	Non verifying ...	

- Click the “Advanced” button and supply the following information

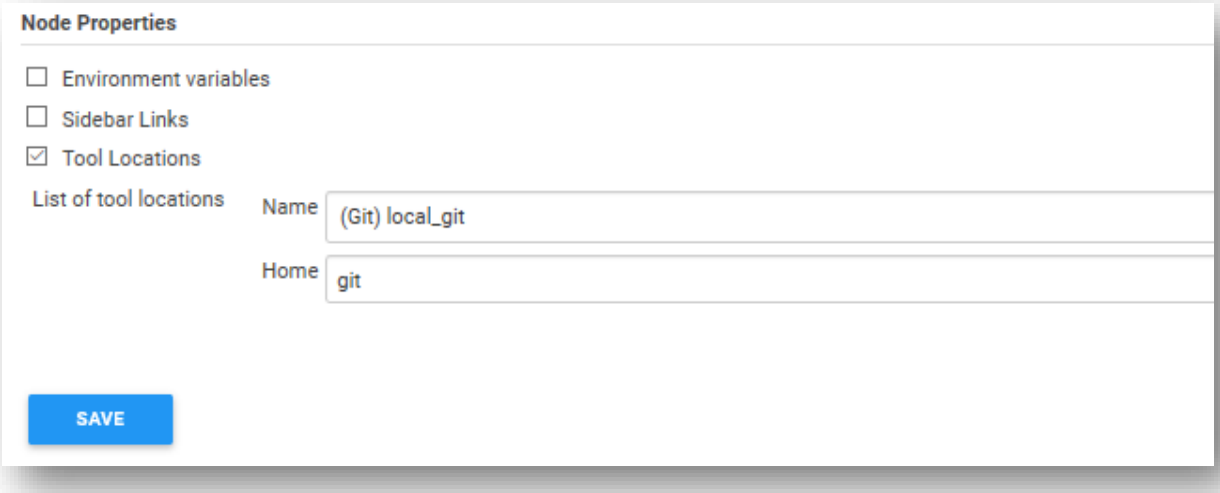


Field	Value	Notes
Port	22	Ensure firewall rules allow access between the Jenkins Server and USS on the port.
Java path	java	Note: Java and other system environment variable will be defined by the “Prefix Agent Start” option below. You don’t need to give the full path name.
JVM Options[11] ***(all one line)***	-Dfile.encoding=UTF-8 -Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.name.file.encoding=IBM-1047 -Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.password.file.encoding=IBM-1047	
Prefix Agent Start Command	. $HOME/.profile &&	Point to SSH user’s the .profile previous defined. Add a trailing space.
Suffix Start Agent Command	-text	Add a leading space.
Maximum Number of Retries		Take default value
Seconds To Wait Between Retries		Take default value
Use TCP_NODELAY flag on the SSH connection		Take default value
Availability	Keep agent online as much as possible	

Note: for Rocket Git 2.26.x, you may need to change the 'file.encoding' to IS08859-1

[11] Git Client Plugin 3.5.0 options for support of HTTPS username and password in EBCDIC

- Scroll to the “Node Properties” and check off the “Tools Locations” and enter the values shown.



Field	Value	Notes
Name	(Git) local_git	Accept the default name.
Home	git	Enter 'git'. The path to git will be resolved in the .profile

- Press “Save” to complete and save the Node information
- Use the “Launch agent” button to launch the agent.

  - A successful connection message is displayed at the bottom of the log.
  - If the agent does not start, carefully review all configuration values. Use “Disconnect” and then relaunch to apply changes (partial log output shown below). Also review/disable any 'node Monitoring' thresholds under the 'Build Executor Status' page.

```
SSHLauncher{host='tvt6012.svl.ibm.com', port=22, credentialsId='nlopez', jvmOptions='-Dfile.encoding=UTF-8  -Xquickstart  -
Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.name.file.encoding=IBM-1047   -Dorg.jenkinsci.plugins.gitclient.CliGitAPIImpl.user.password.file.encoding=IBM-
javaPath='java ', prefixStartSlaveCmd='.  /u/nlopez/Jenkins-Agent-Prefix.sh  && ', suffixStartSlaveCmd=' -text ', launchTimeoutSeconds=30, maxNumRetries=3, retryW
sshHostKeyVerificationStrategy=hudson.plugins.sshslaves.verifiers.NonVerifyingKeyVerificationStrategy, tcpNoDelay=false, trackCredentials=true}
[10/03/20 14:58:21] [SSH] Opening SSH connection to tvt6012.svl.ibm.com:22.
[10/03/20 14:58:22] [SSH] WARNING: SSH Host Keys are not being verified. Man-in-the-middle attacks may be possible against this connection.
[10/03/20 14:58:22] [SSH] Authentication successful.
[10/03/20 14:58:22] [SSH] The remote user's environment is:
@="sh"
ERRNO="0"
HOME="/u/nlopez"
IFS="
"
LINENO="0"
LOGNAME="NLOPEZ"
MAIL="/usr/mail/NLOPEZ"
MAILCHECK="600"
OPTIND="1"
```

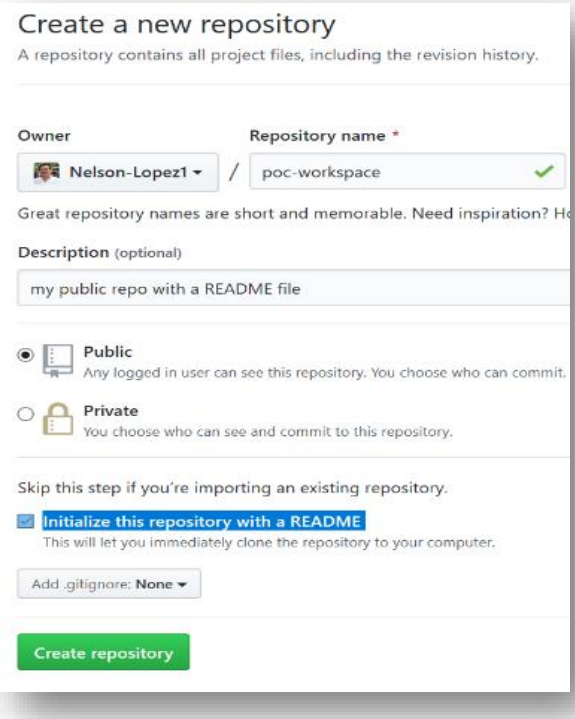
```
Agent successfully connected and online
```

Note: The values in the above log may not reflect the value in your configuration.

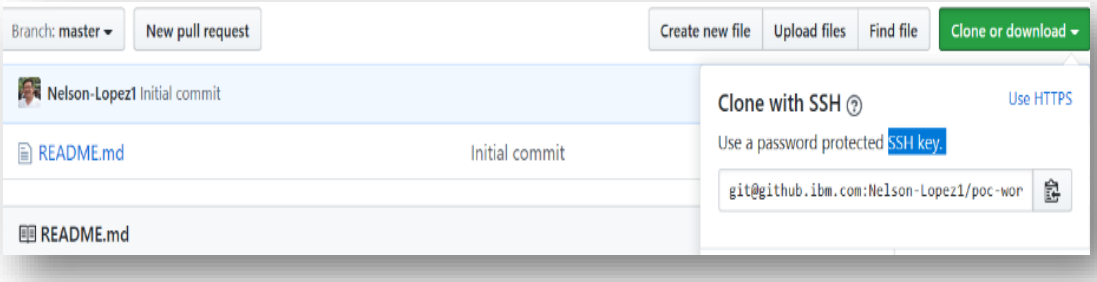
5.4 Create a GitHub repository

Follow these steps to initialize a new Git Repo as part of the POC. Other non-GitHub environments can be used.

1. In GitHub, create a **PUBLIC** repo called "poc-workspace" and **enable** *"initialize this repository with a README'* and press "Create …".

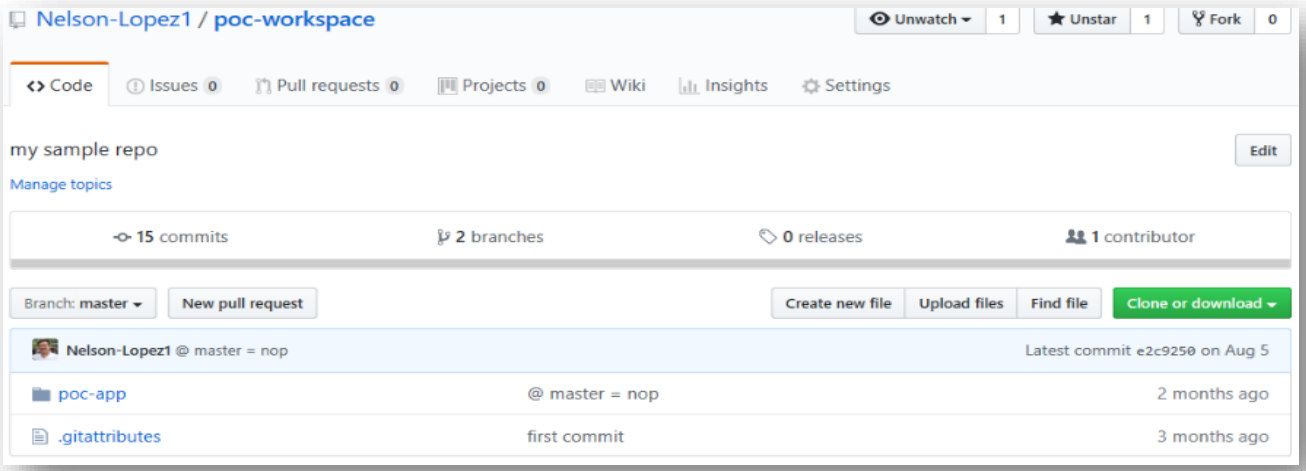


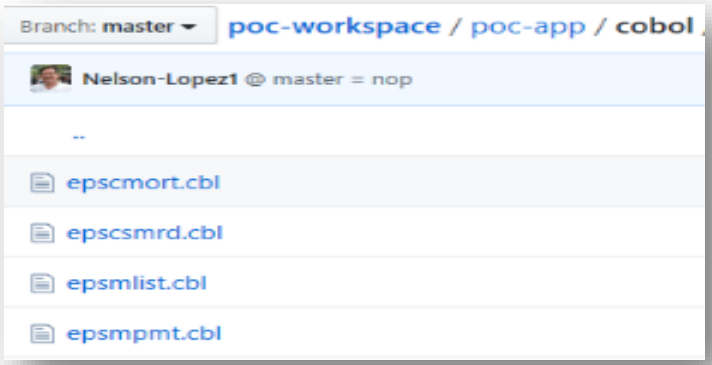
2. On the next screen, under the clone button, cut the new repo's **SSH** URL for pasting in USS.

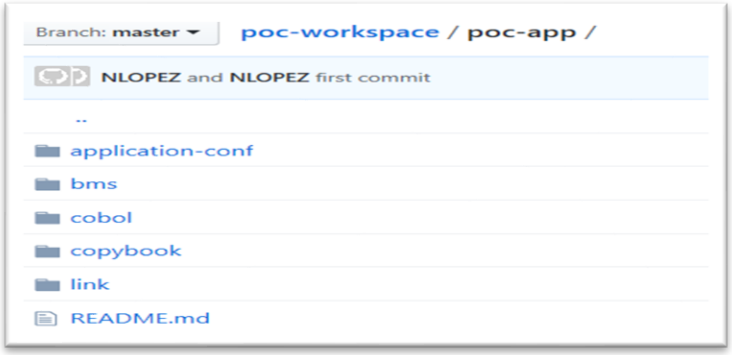


3. In USS:
   - **`cd`**
   - **`git clone paste-your-ssh-repo-url-here`**
   - **`cd poc-workspace`**
   - **`cp $HOME/dbb-zappbuild/.gitattributes .`**
   - **`mkdir poc-app`**
   - **`cp -r $HOME/dbb-zappbuild/samples/MortgageApplication/**  poc-app`**
   - **`git add .`**
   - **`git commit -m "first commit"`**
   - **`git push`**

4. In GitHub, refresh your repo's page to review the results.
   - Note the standard DBB folder structure, .gitattributes, a sample application sub-folder 'poc-app' containing the required 'application-conf' and related sample source code folders.
   - Also note the file extensions of the source files. Use this as a template for any new applications.

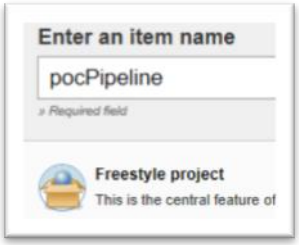




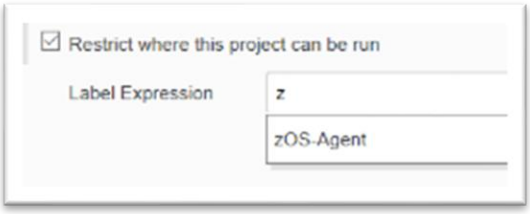


5.5 Create a Jenkins Freestyle DBB Build Job

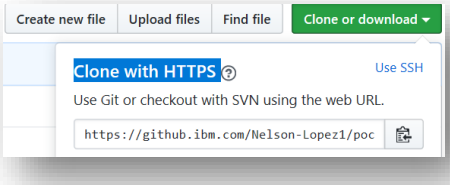
In Jenkins, select "New Item" from the home page and give it a name like “pocPipeline”. Choose "Freestyle" and then "OK".



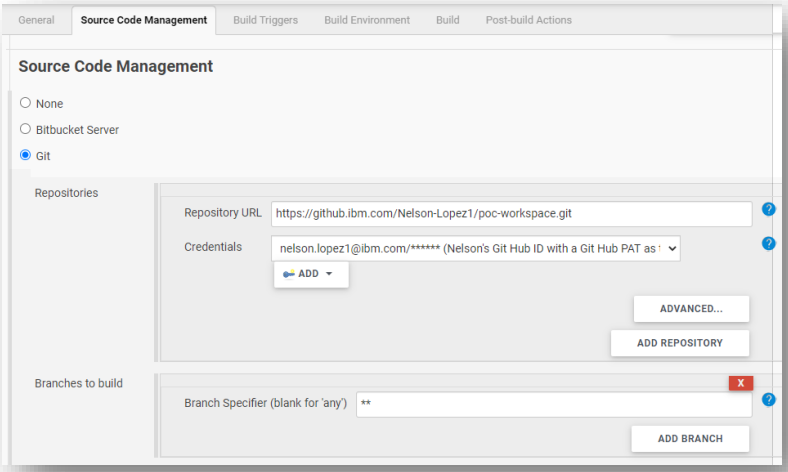
On the next page, enable "Restrict where …” and enter the new agent’s label “zOS-Agent”.



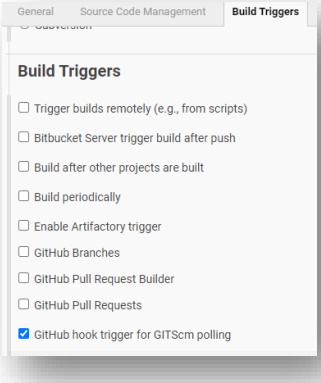
From the new Git repo’s page, press the green “Clone…” button and cut the **HTTPS**-based URL.



Back at the Jenkins job, paste the URL in the “Source Code Management” section of your job. From the dropdown, select your new PAT-Based credentials

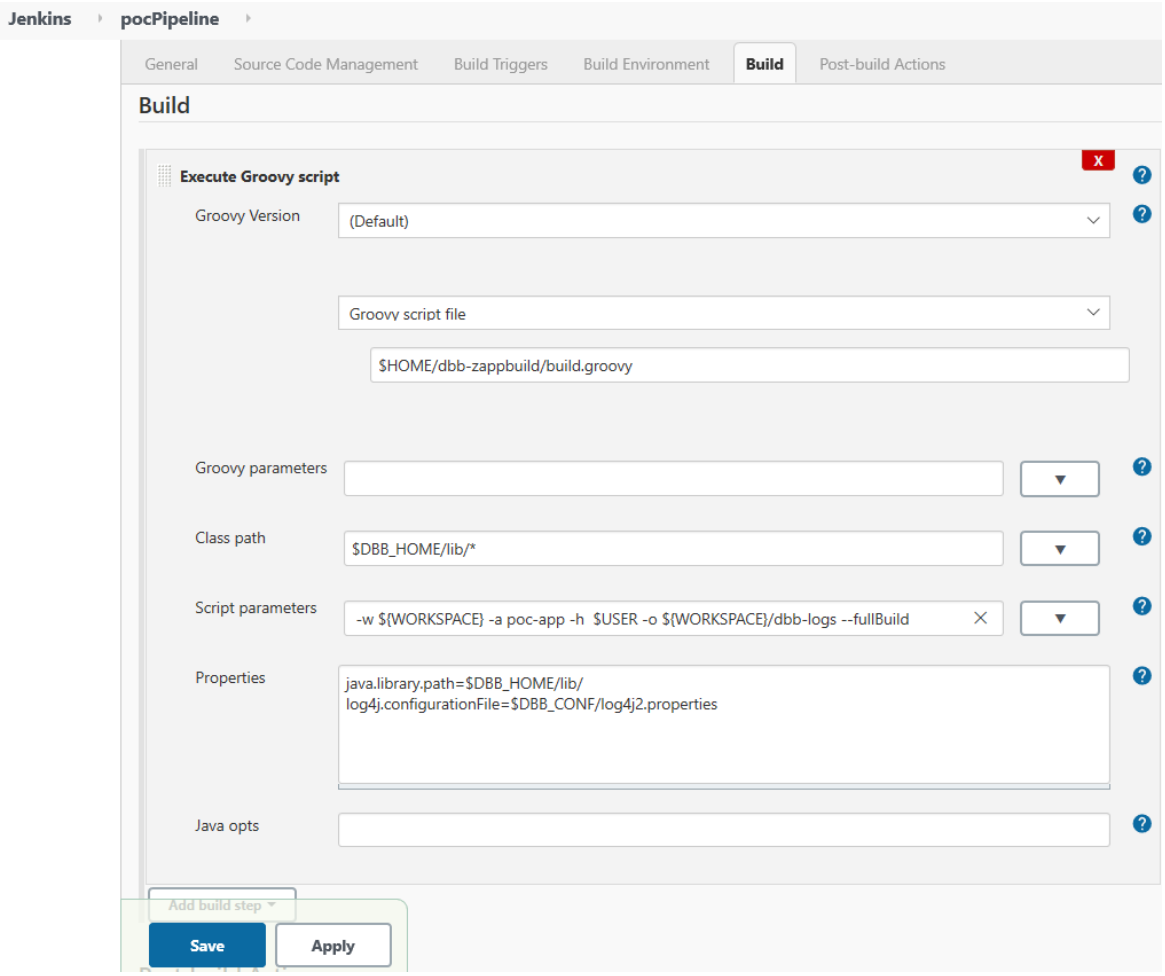


In the ‘Build Triggers’ section’, enable “GitHub hook trigger…” to auto-start this job on a git event (scope) like a commit or pull-request.



In the “Build” section, press “Add Build Step” and select "Execute Groovy Script". Add the full USS path of the installed groovy samples (zAppBuild) to run DBB’s main “**build.groovy**” script. Click "Advanced" and enter the values as shown below then press “Save”.

Field	Value
Groovy script file	$HOME/dbb-zappbuild/build.groovy
Class path	$DBB_HOME/lib/*
Script parameters[12]	-w ${WORKSPACE} -a poc-app -h $USER -o ${WORKSPACE}/dbb-logs --fullBuild
Properties	java.library.path=$DBB_HOME/lib/ log4j.configurationFile=$DBB_CONF/log4j2.properties

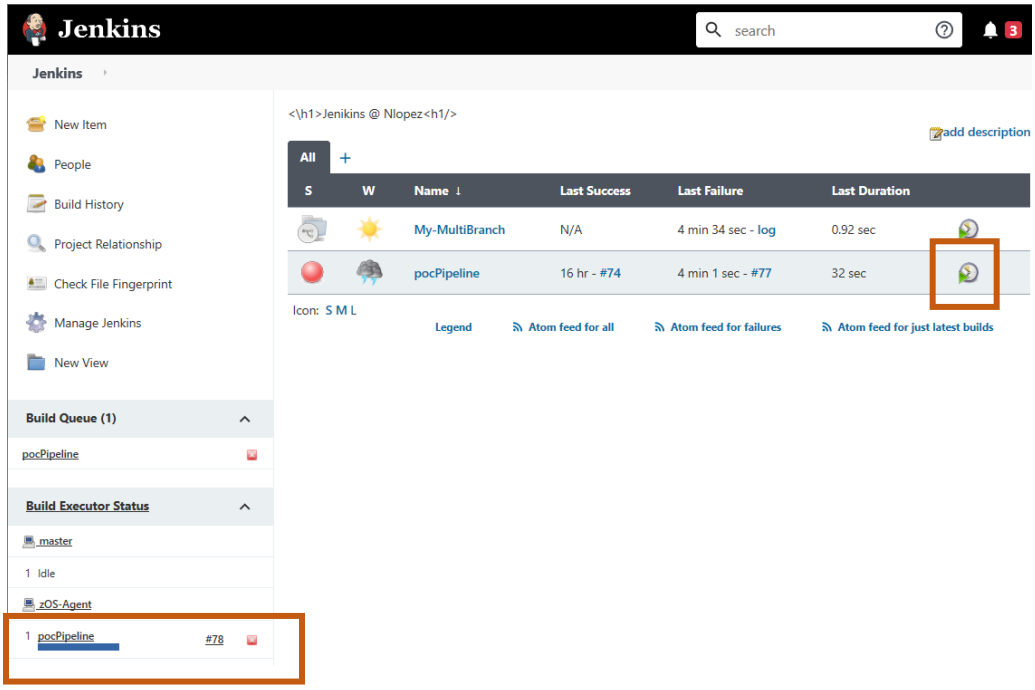


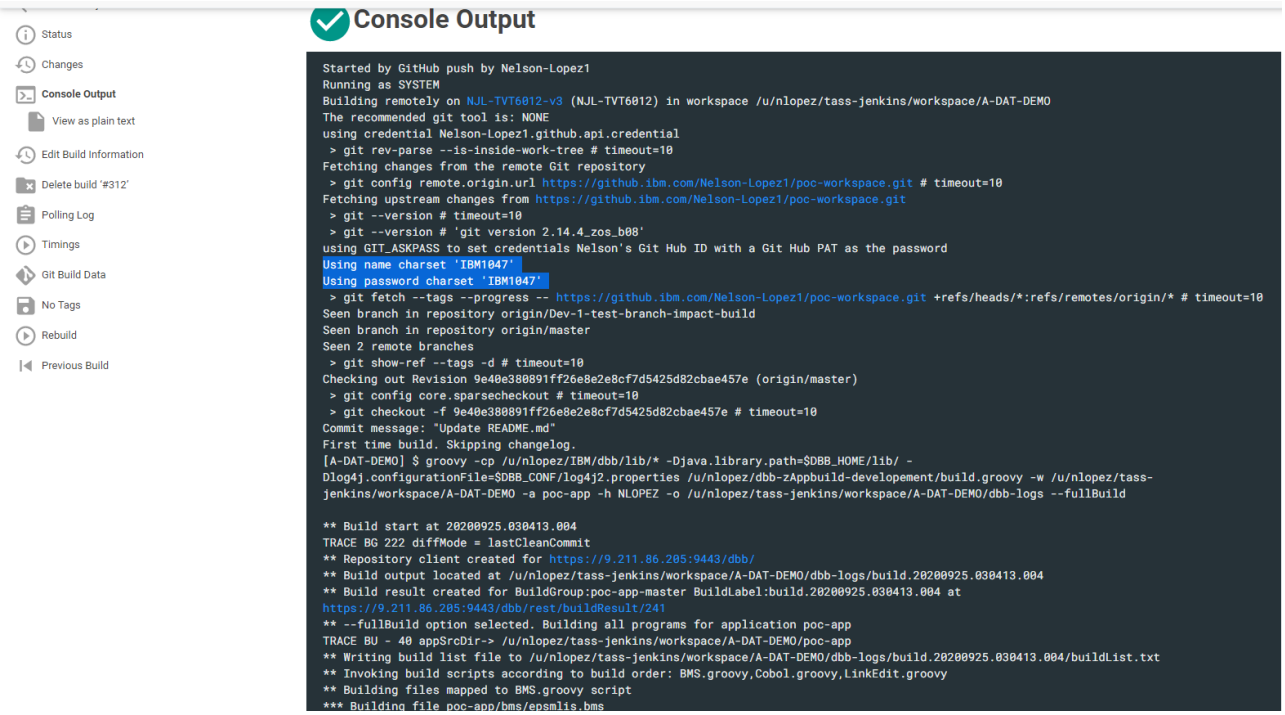
Note: ${WORKSPACE} is a Jenkins system variable that points to the “remote root directory” defined in the agent. The clone, build results and logs are stored there. The above script parms performs a ‘--fullBuild’. Change that to ‘--impactBuild’ to test builds for only changed files.

[12] For details see https://github.com/IBM/dbb-zappbuild/blob/development/BUILD.md

5.6 Jenkins DBB build[13] - IVP

From the Jenkins home page, click on the "Build Now" icon for the new job (far right). In a few seconds, your job will appear on the bottom left. Click on its blue progress bar to jump to the log.





Note: Some values above may not match the recommended values. The log path is provided in "Writing build report to …" towards the bottom. Access the log to view your build output on USS.

[13] Your Jenkins Admin can help set up webhooks to Git that automatically start your job on a Commit or Pull Request.

5.7 View DBB build results in the DBB WebApp

From the job's console log, click on the DBB WebApp URL to view the build result from the DBB WebApp (default userid/password is ADMIN/ADMIN)

```
** Build start at 20200615.092819.028
** Repository client created for https://dbbdev.rtp.raleigh.ibm.com:9443/dbb/
** Build output located at /u/nlopez/dbb-logs/build.20200615.092819.028
** Build result created for BuildGroup:poc-app-master BuildLabel:build.20200615.092819.028 at https://dbbdev.rtp.raleigh.ibm.com:9443/dbb/rest/buildResult/79870
```

Select "View" to see the results.

Main Content

IBM

Collections Build Results

DBB Build Result

Field	Value
id	79870
group	poc-app-master
label	build.20200615.092819.028
Build Report	view download
Build Report Data	download
state	2 (COMPLETE)
status	0 (CLEAN)
owner	
permission	
created	
createdBy	
lastUpdated	
lastUpdatedBy	
attachments	

Build Result Properties (count=3)

Property Name	Property Val
fullBuild	true
filesProcessed	8
:githash:poc-app	563007da829

Main Content

Build Report

Toolkit Version:

Version: 1.0.8

Build: 16

Date: 03-Mar-2020 21:30:54

Build Summary

Number of files being built: 8

	File	Commands	RC	Data Sets	Outputs	Deploy Type	Logs
1	poc-app/bms/epsmlis.bms	ASMA90	0	NLOPEZ.BMS(EPSMLIS)	NLOPEZ.BMS.COPY(EPSMLIS)		EPSMLIS.bms.log
		ASMA90	0				
		IEWBLINK	0		NLOPEZ.LOAD(EPSMLIS)	MAPLOAD	
2	poc-app/bms/epsmort.bms	ASMA90	0	NLOPEZ.BMS(EPSMORT)	NLOPEZ.BMS.COPY(EPSMORT)		EPSMORT.bms.log
		ASMA90	0				
		IEWBLINK	0		NLOPEZ.LOAD(EPSMORT)	MAPLOAD	
3	poc-app/cobol/epsnbrvl.cbl Show Dependencies	IGYCRCTL	0	NLOPEZ.COBOL(EPSNBRVL)	NLOPEZ.OBJ(EPSNBRVL)		EPSNBRVL.cobol.log
4	poc-app/cobol/epsmpmt.cbl Show Dependencies	IGYCRCTL	4	NLOPEZ.COBOL(EPSMPMT)			EPSMPMT.cobol.log
		IEWBLINK	0		NLOPEZ.LOAD(EPSMPMT)	LOAD	
5	poc-app/cobol/epsmlist.cbl Show Dependencies	IGYCRCTL	0	NLOPEZ.COBOL(EPSMLIST)	NLOPEZ.OBJ(EPSMLIST)		EPSMLIST.cobol.log
6	poc-app/cobol/epscsmrd.cbl Show Dependencies	IGYCRCTL	0	NLOPEZ.COBOL(EPSCSMRD)			EPSCSMRD.cobol.log
		IEWBLINK	0		NLOPEZ.LOAD(EPSCSMRD)	LOAD	
7	poc-app/cobol/epscmort.cbl Show Dependencies	IGYCRCTL	4	NLOPEZ.COBOL(EPSCMORT)	NLOPEZ.DBRM(EPSCMORT)	DBRM	EPSCMORT.cobol.log
		IEWBLINK	0		NLOPEZ.LOAD(EPSCMORT)	LOAD	
8	poc-app/link/epsmlist.lnk Show Dependencies	IEWBLINK	0	NLOPEZ.LINK(EPSMLIST)	NLOPEZ.LOAD(EPSMLIST)	LOAD	EPSMLIST.linkedit.log

6 DBB User Build in IDz

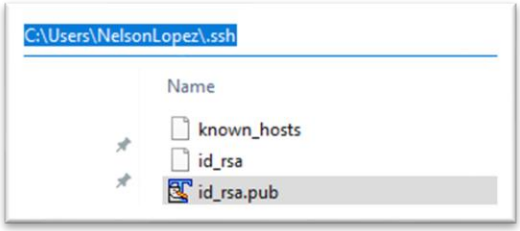
Required skills: DevOps Admin

As a developer, setup IDz[14] to perform a DBB user build of the poc-workspace repo.

6.1 Add your Windows' SSH key to Git

Perform these steps to use SSH to connect from IDz to your Git server (GitHub for instance). From a windows terminal:

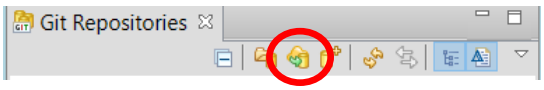
- **ssh-keygen -t rsa -b 4096 -C "your GitHub acct"**[15]
- Navigate to your windows "user_directory/.ssh" folder and open "id_rsa.pub" file.
- Cut/paste the key into Git. See "Add your SSH Key to Git" step 1 in the "Jenkins and Git Credentials" section above



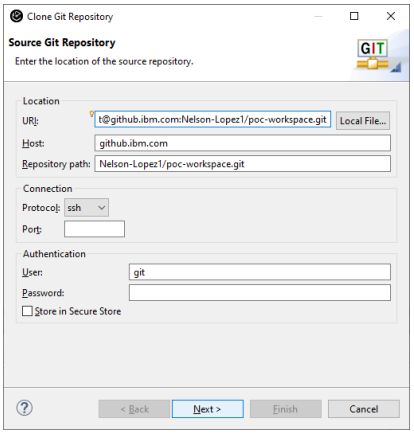
6.2 Clone with eGit plugin

Ensure the eGit plugin is installed (or pre-installed) using the "eclipse marketplace" from the IDz's help menu. From the IDz Git perspective:

- Select the clone icon:



- Paste your new repo's URL and follow the prompts:

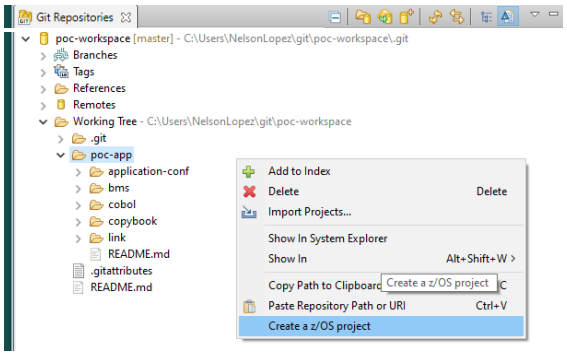


[14] This assumes IDzEE has been previously installed and your using SSH access to your Git server. These screen shots are based on ver 14.2.3.

[15] "-C" is optional. Check with your Git for standards at your site.

6.3 Create an IDz z/OS .project file

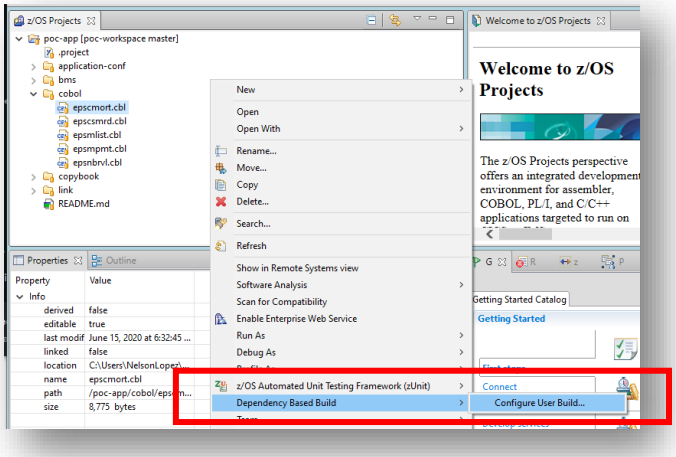
In the Git perspective, open the poc-application folder and right click to “Create a z/OS project”

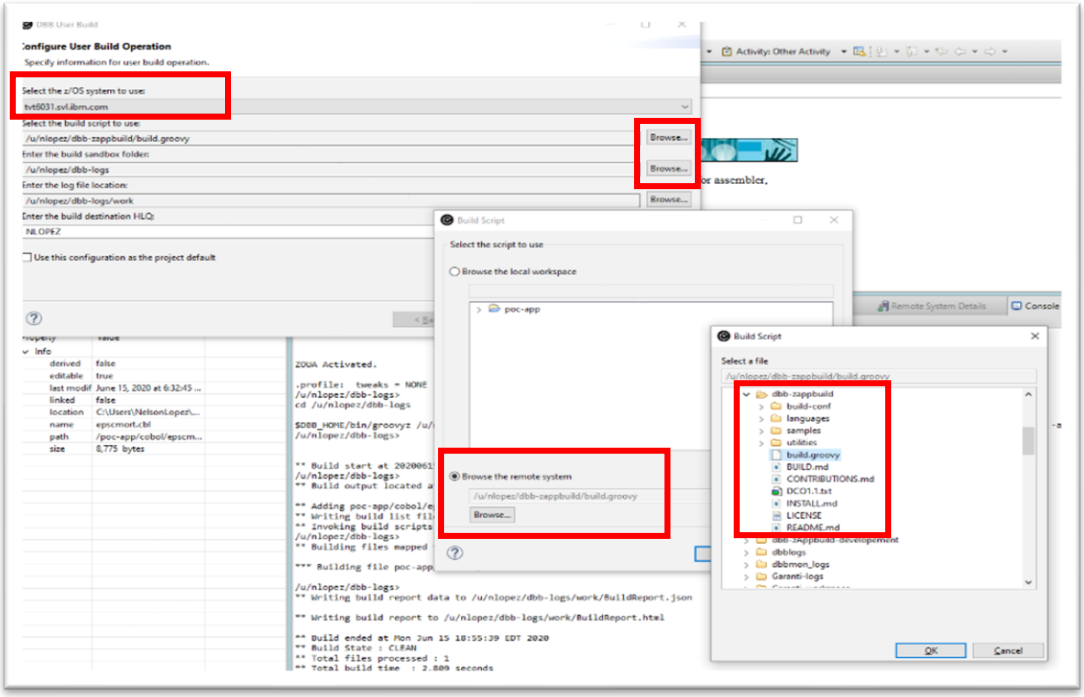


6.4 Configure and run a DBB User Build

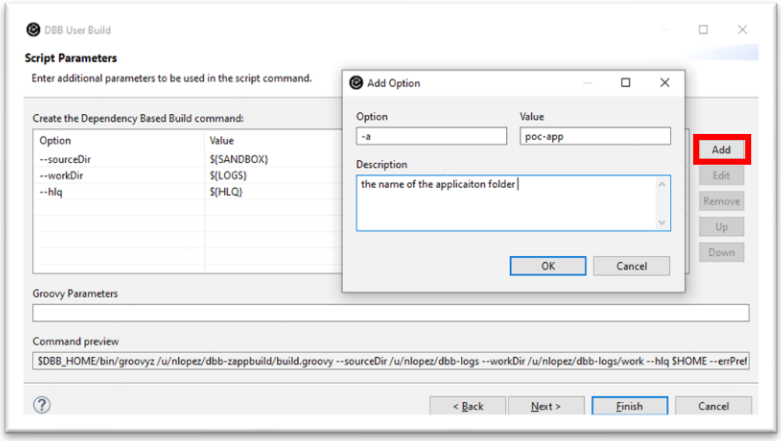
From the z/OS Projects perspective:

- Right click on any cobol program
- Select “Dependency Based Build/Configure User Build”.
- On the next screen, under the field “Select the z/OS …”, enter your z/OS host DNS or IP
- For “Select the build script …” and “…sandbox…”, press the “Browse” on the right of each field to navigate the remote USS filesystem and select the path for “?/dbb-zappbuild/build.groovy” and the previously created “dbb-logs” sandbox folder.
- The value for “Enter the log file …” is auto filled and needs no change.
- “HLQ” - enter your z/OS ID.

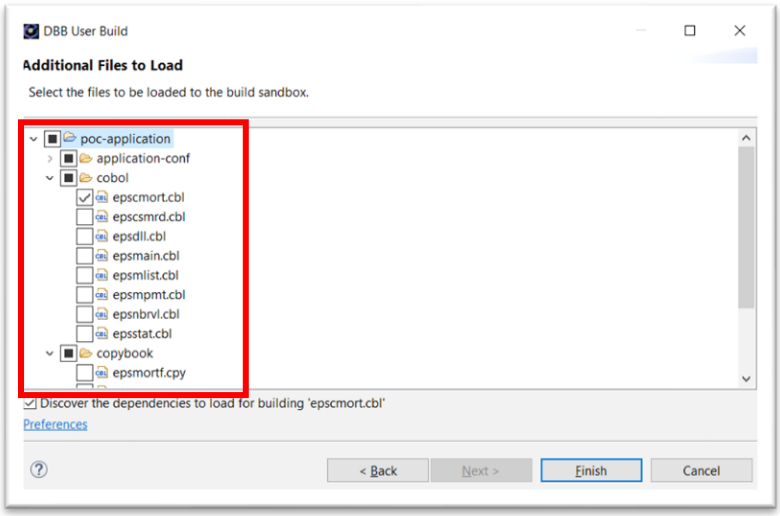




- Press “Next” and then “ADD” to enter the Option **“-a”** with a value of “poc-app” and then “OK”.

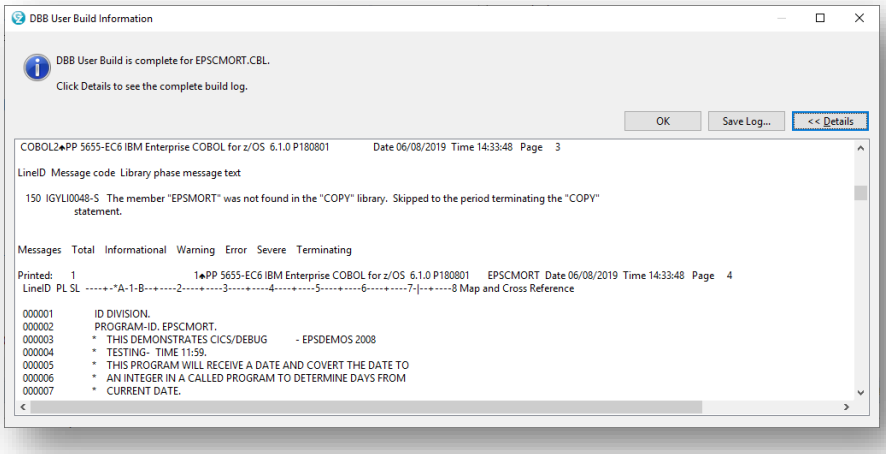


- Press “Next” and on the next page select the “application-conf” folder (you only need to do this the first time or any time you change something in this folder). All other folder(s) are pre-selected.



- Press Finish to start a DBB user build.

Within a few seconds, IDz will display the results. Press details to view the compiler/link output. If a system error is returned, double check your settings and MVS privileges and try again.



7 Migrating a Sample Application from PDS(s) to Git

A DevOps Admin can follow these steps to migrate a sample application from PDS(s) to git. This example reuses the sample repo created during the IVP as a template. This same technique can be used to migrate any application to git using a more descriptive repo name and the standard DBB folder layout.

7.1 Cleanup the POC sample folder in USS

- **cd**
- **cd poc-workspace/poc-app**
- **rm -r cobol**
- **rm -r bms**
- **rm -r copybook**
- **rm -r link**

7.2 Migrate source code

The migrate tool resides in **$DBB_HOME/migration/bin/migrate.sh** and requires these arguments:[16]

- **-r** a local **r**epo path. In the example below, "**$HOME/poc-workspace/poc-app/**" is a local repo with 2 parts:
  - "poc-workspace" is the local repo's root folder
  - "poc-app" is the application folder of all source files
- **-m** **m**apping rule:
  - **hlq:** the PDS minus it's LLQ. For example, 'TST.ACCTS.COBOL' would be "hlq:TST.ACCTS"
  - **targetDir**[17]**:** a directory created by the tool under the application folder ("poc-app") in this example.
  - **extension:** added to each member. Standard defaults are **cbl**=cobol, **bms**=cics-maps, **pli**, **mac,** and **cpy**=copybooks
  - **pdsMapping:** and **toLower:** enter as shown
- **LLQ** is the last argument and the LLQ of the PDS. It can include a member or a pattern. If member is not provided all are copied.

Example 1- migrate Cobol members 'ABC*' from PDS 'TST.ACCTS.COBOL'. Using the command below (all one line) from your home directory, each member is copied to the local **r**epo path (**-r**) into the **targetDir** 'cobol' with an extension of **'cbl'**. Bolded text indicates values to be review or changed.

```
$DBB_HOME/migration/bin/migrate.sh  -r $HOME/poc-workspace/poc-app/
-m MappingRule[hlq:TST.ACCTS,pdsMapping:false,toLower:true,targetDir:cobol,extension:cbl]
"COBOL(ABC*)"
```

Example 2- migrating copybook members 'AT*' in 'TST.COMMON.COPY' to the targetDir "copybook' with an extension of '.cpy'

```
$DBB_HOME/migration/bin/migrate.sh  -r $HOME/poc-workspace/poc-app/
-m MappingRule[hlq:TST.COMMON,pdsMapping:false,toLower:true,targetDir:copybook,extension:cpy]
"COPY(AT*)"
```

7.3 Push your source to Git

- **cd**
- **cd poc-workspace**
- **git add .**
- **git commit -m 'my sample app is now in git'**
- **git push**

You can now clone your app in IDz for a User Build or from a Jenkins job.

[16] For a complete reference see https://www.ibm.com/support/knowledgecenter/SS6T76_1.0.9/migration.html

[17] For SYSLIB folders like copybook, plinc and maclib refer to zAppBuild's "application-conf/application.properties" file for default naming rules like "$copybookRule"

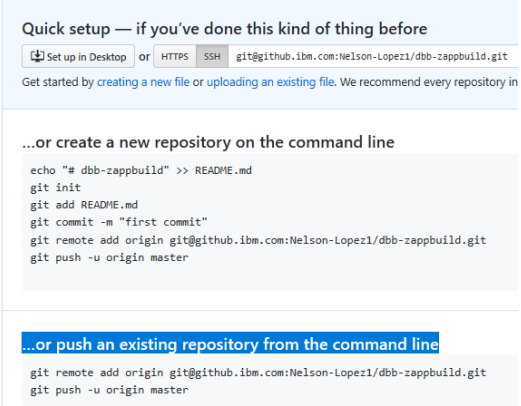
8 Appendix – Cloning dbb-zappbuild

If you don't have access to clone "https://github.com/IBM/dbb-zappbuild" from zOS you can either clone these samples to a PC or install them from a ZIP file provided by your IBM representative to a PC. Once on a PC you can push the samples to an internal git server to then clone it to a zOS/USS file system folder.

These steps show how to copy the public IBM DBB sample to a new repo on your internal git server using GitHub. Feel free to reach out to your local Git Admin for help with the server at your site.

Assumptions:

- You have installed git on your PC (see https://git-scm.com/downloads)
- You have access to an internal Git Server to create repo(s). This example uses GitHub but any git compliant server will work.
- Rocket Git is installed on Unix System Services and you have SSH access to clone from your internal git server.



1. Create a new public repo on your internal git server and call it dbb-zappbuild.
2. After creating the repo, a "Quick setup" page explains how to initialize it from your PC.
3. If you are working with a ZIP file:
   a. Expand the ZIP file on your PC and make sure the main folder is called dbb-zappbuild
   b. From a windows terminal, navigate to the new folder (cd ?/dbb-zappbuild) and issue the following git commands:
      - git init
      - git add .
      - git commit -m "initialize dbb-zappbuild"
      - git remote add origin ? *(where ? is your URL from the "Quick setup")*
      - git push -u origin master
4. If you can clone from the public IBM GitHub site:
   a. From a windows terminal, navigate to any working folder and issue:
      - git clone https://github.com/IBM/dbb-zappbuild
      - cd dbb-zappbuild
      - git remote add **dbb** ? (where ? is your URL from the "Quick setup")
      - git push -u **dbb** master

   Note you can use the SSH version of the above clone command based on your public GitHub account setup.
5. Review that the contents of the remote repo match your local copy.
6. Logon to USS and clone your remote dbb-zappbuild repo from your home folder.
7. The local PC repo can be deleted.