An IBM TechDoc from

IBM Z DevOps Acceleration Program

July, 2020 - Release 2.1

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

Mathieu Dalbin
mathieu.dalbin@fr.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 and configure IBM's Dependency Based Build (DBB) engine and supporting tools to run a basic Continuous Integration pipeline as a Proof of Concept (POC).

1.1 Tools overview

Tools used for the POC include:

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 call 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 Web App – This component is a WebSphere Liberty application installed on a Linux server and is used to store meta data information such as dependencies between source modules, i.e. program to copybook, 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. In order 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 Administrators roles

A z/OS Systems Admin will download DBB & Git onto a USS working folders.

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 working folders will complete the configuration of DBB, Git, Jenkins and any agents needed for the POC like Jenkins.

2 Rocket Git and DBB installations

2.1 Rocket Git installation

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

Rocket's Git provides a standard Git interface on Z with full support of ASCII to EBCDIC conversions. Rocket installs are in https://www.rocketsoftware.com/

First time users must register then navigate to the download page under "zOpenSource" installers. Download the latest versions and change the file names below as needed. You'll need 550 MB of space and a tool to copy the downloads to USS like SCP.

Instructions (change the version number of the tar files as needed):

- From the Rocket site's Download Menu, select z/OpenSource
- Download the latest versions of **Git, BASH, PERL, GZIP**
- Create a USS directory: **mkdir /var/rocket**
- Untar **GZIP**: **tar -C /var/rocket -xovf gzip-1.6-edc_b0005.160229.tar**
- Gzip-unzip the **BASH** tool: **gzip -d bash-4.3_<build>.tar.gz**
- Untar **BASH:** **tar -C /var/rocket -xovf bash-4.3_<build>.tar**
- Untar all the other Rocket tools under the same folder
- Enable the install folder read/write for subsequent configuration by the POC team.

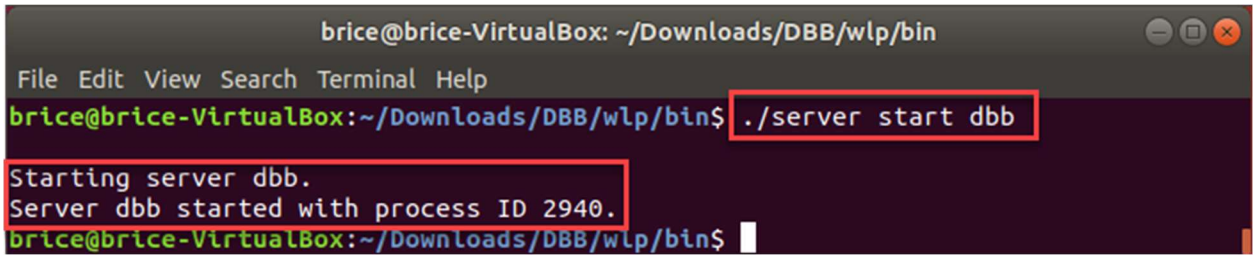
2.2 DBB installation (Linux and z/OS)

DBB has a WebApp that is installed on a Linux Server and a Toolkit that is installed on z/OS.
Download: https://developer.ibm.com/mainframe/products/ibm-dependency-based-build/

2.2.1 DBB WebApp - Linux

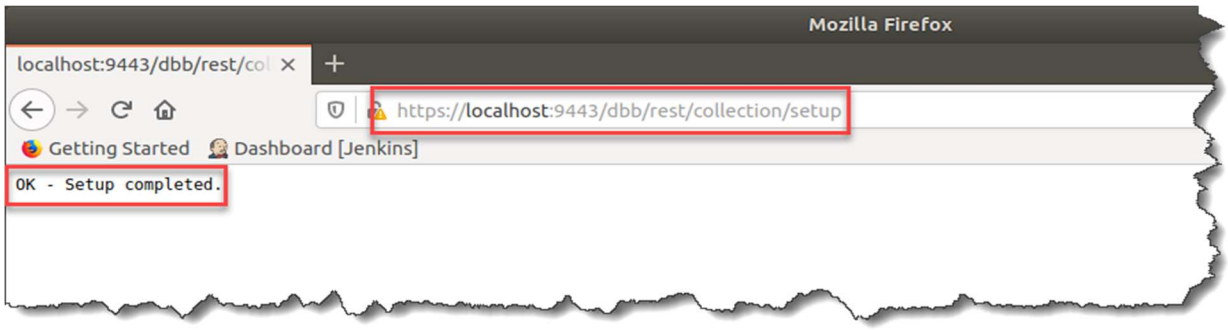
Required skills: Linux Admin (installation of the DBB Web Application)

- The host Linux server will need connectivity to the target z/OS server over port 9443 or 9080.
- Under the Download section follow the link to "Web App" (dbb-server-1.x.x.tar.gz)
- Copy the tar file with SCP or another tool in binary format into **Linux Server's** root folder. You'll need 500mb free space.
- Untar the file: **tar -xzvf dbb-server-x.x.x.tar.gz**
- The DBB Server will be installed under "../wlp"
- Start the DBB Server: **sh ../wlp/bin/server start dbb**



- Verify the installation (add your Linux server's IP):
  - From your browser enter https://***your-srv-ip***:9443/dbb/rest/collection/setup
  - Ignore any "Your connection is not private" messages and continue

- Enter the default DBB user id and password ADMIN/ADMIN
- The message “OK – Setup completed.” should appear indicating success



- Log files are in “../wlp/usr/server/dbb/logs”.
- To stop DBB: **sh ../wlp/bin/server stop dbb**
- Use https://your-srv-ip:9443/dbb to access the results of subsequent DBB Builds

2.2.2 DBB toolkit - USS

Required skills: z/OS Admin (installation of the DBB toolkit)

- Under the Download section follow the link to “as a Standalone” for (dbb-ztoolkit-trial-1.x.x.tar)
- Copy it 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** [1]
  - **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 user progiles (.profile in users’ directories) has “export JAVA_HOME=/usr/lpp/java/J8.0_64 “
- Create or update a z/OS account(s) with an OMVS segment, with at least a 1GB space for Home directory and access to DBB and Git folders. This account will be used by the DevOps(s) Admin to complete the configuration and conduct the POC.

[1] This DBB_HOME should also be the default in the IDz Configuration

3 Environment 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 and a z/OS logon. They will also need privileges to clone an IBM repo from the public Git hub site.

3.1 Update .profile

- Obtain the path of DBB_HOME, Rocket and the URL of DBB WebApp from the Admins
- Edit **.profile** in the home directory as shown (update paths according to your environment):

```
export DBB_HOME=/usr/lpp/IBM/dbb
export DBB_CONF=/usr/lpp/IBM/dbb/conf
export GROOVY_HOME=$DBB_HOME/groovy-2.4.12
export PATH=$JAVA_HOME/bin:$GROOVY_HOME/bin:$DBB_HOME/bin:$PATH
. $DBB_CONF/gitenv.sh
```

3.2 Verifying USS tools installation

Logon to USS and issue these commands:

- **java -version**
  It should display "… JRE 1.8.0 z/OS s390x-64-Bit …"
- **git --version**
- **gzip --version**
- **groovyz -v**
- If any command fails, report it to the z/OS Admin to review the installation steps.

3.3 Install and configure the zAppBuild samples

Note – You may optionally need to create a free ID at github.com for the next steps.
Enter the following commands from USS with the user that performs the installation:

- **cd**
- **git clone git://github.com/IBM/dbb-zappbuild.git**
- **cd dbb-zappbuild**
- **ls -lasT**

Example dbb-zappbuild folder and extended file tags

```
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/dbb-zappbuild #>
```

3.4 Configure zAppBuild framework

You might need help from z/OS Admin to identify system datasets like the cobol compiler.

Use IDz or ISPF to access the folder where zAppBuild is installed and navigate to the build-conf sobfolder (**dbb-zappbuild/build-conf)**. Edit the **build.properties** file and update the DBB Web Application URL as provided by the Linux Admin.

```
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=
```

Edit **datasets.properties** and update the various system libraries. The z/OS Admin may be able to provide the DSNs. Line 20 can be ignored if the Cobol compiler V6 is used.

```
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=
```

3.4.1 Test of a DBB build

Logon to USS and issue these commands:

- **cd**
- **mkdir dbb-logs**

- ***Note** - make sure the next command is entered as one line. You can cut and right click on the putty term to paste.
- **groovyz dbb-zappbuild/build.groovy -w dbb-zappbuild/samples -a MortgageApplication -h $USER -o dbb-logs --fullBuild**

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

View the build Logs:

- **cd dbb-logs**
- **ls** list the build folders
- **cd build.xxx** replace xxx with the latest build folder name
- **cat EPSMLIST.cobol.log** show the sysprint output of a sample build

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.
```

4 Jenkins installation

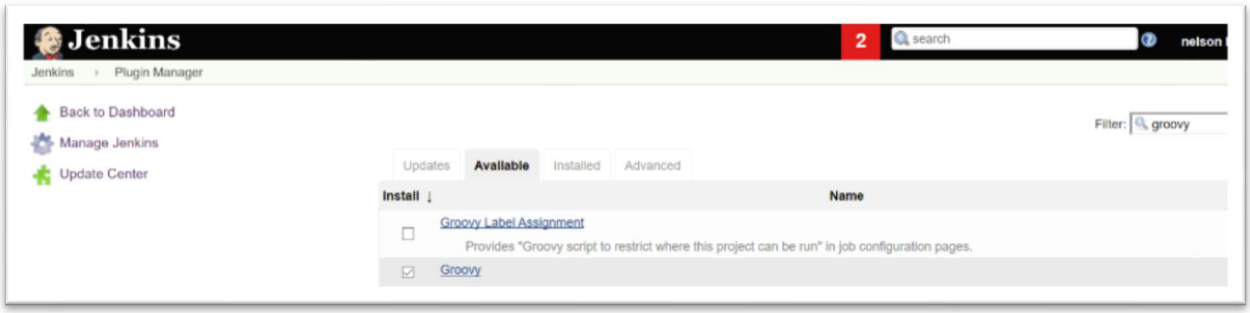
Required skills: DevOps Admin & Jenkins Admin

The installation of the Jenkins server (also known as Jenkins Master) is out of the scope of this document. If needed, the Jenkins web site has details on how to download and install Jenkins. The download can be found here. Installation instructions can be found here. For a new Jenkins installation, you'll need to add and configure the Git and Groovy plugins. Depending on company policies, this step can be performed by the DevOps Admin or a Jenkins Admin.

4.1 Jenkins install/update plugins

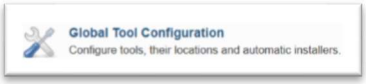
Follow these steps to configure plugins:

- From the home page, navigate to "Manage Jenkins/Manage Plugins"
- In the "Available" tab, scroll and check off Groovy and Git then press Install.
  *Note: If these plugins are not "available" then they have been pre-installed.

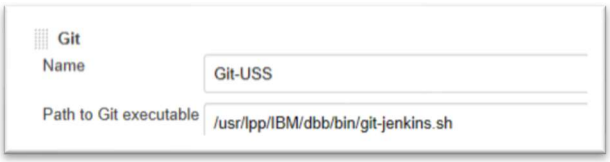


4.1.1 Jenkins Global Plugin - Git

- From the home page, navigate to "Manage Jenkins/Global Tools Configuration".



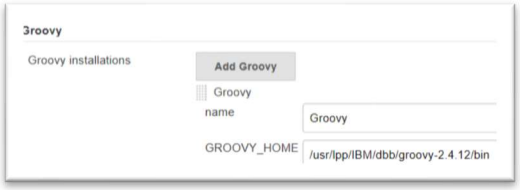
- Scroll to the Git section and ADD a name and path **/usr/lpp/IBM/dbb/bin/git-jenkins.sh**



4.1.2 Jenkins Global Plugin - Groovy

Find Groovy in the Global Tools Page:

- Press ADD and disable install automatically
- enter a name and path **/usr/lpp/IBM/dbb/groovy-2.4.12/bin**



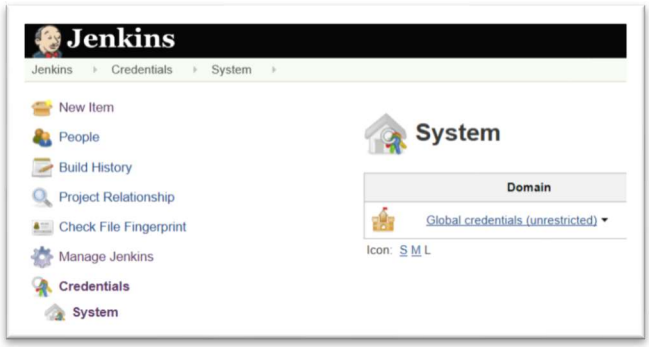
*Note: These plugin names are used when configuring the agent

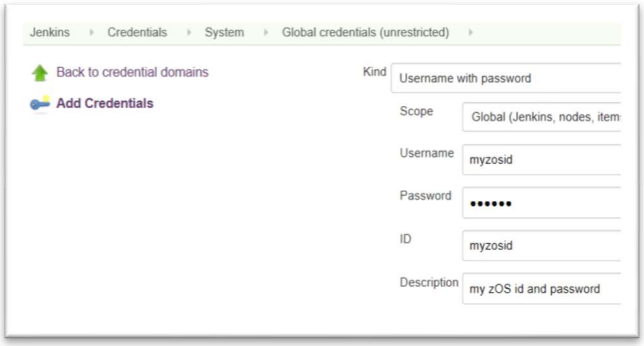
4.2 Jenkins and Git credentials

The DevOps Admin will use their z/OS ID to run an agent and their Git ID to clone a repo.

4.2.1 Add your z/OS ID to Jenkins

- From the home page, navigate to “Manage Jenkins/Credentials/System”.
- Select “Global Credentials”, “Add Credentials”
- “Kind’ will be **Username with password** and enter your z/OS account.





4.2.2 Add your SSH Key to Git

These steps may vary due to installation security policies. For example, Git access may be controlled by LDAP. If that is the case work with the Git Admin to define how to access Git from a remote system. These notes explain adding SSH keys for remote access.

Also see https://git-scm.com/book/en/v2/Git-on-the-Server-Generating-Your-SSH-Public-Key

1 – From USS:

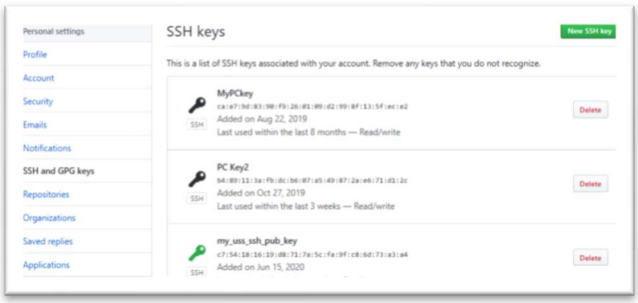
- **cd**
- **ssh-keygen -t rsa -b 4096 -C “user@myOrganization.com”** - where user@... is your Git acct
- press enter and accept all defaults
- **cat .ssh/id_rsa.pub**

- copy the output as shown

USS – generate SSH Keys

2 – From your "GitHub/Account/settings", SSH option, paste your **Public** key

Git Hub - add a new SSH Key

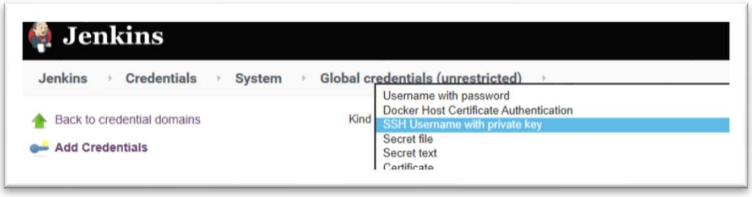


3 – Cut/Paste your **Private** key from USS:

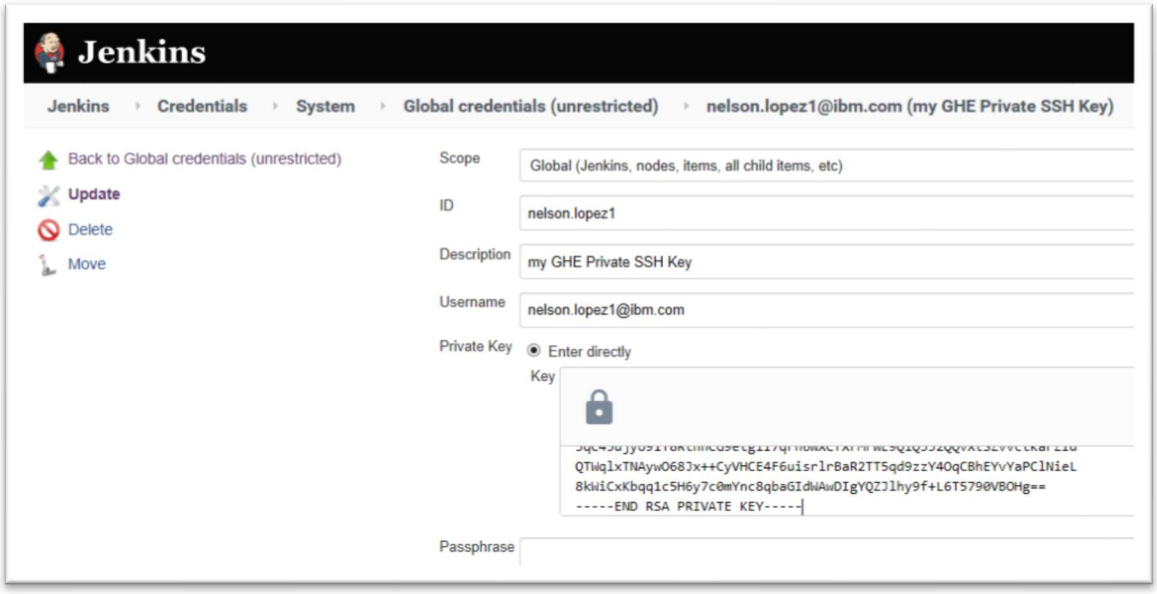
- **cd**
- **cat .ssh/id_rsa**
- cut the new cat output. Careful highlight the text as shown starting from "-----BEGIN …" to "…KEY-----" with no trailing blanks.

- In Jenkins, navigate to "Manage Jenkins/Credentials/System".

- Select “Global Credentials”, “Add Credentials”
- “Kind’ will be “SSH …”

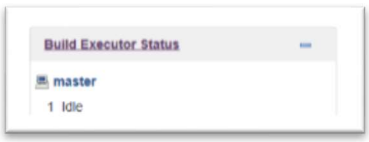


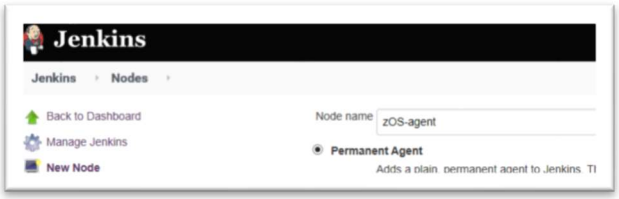
- Enter an ID, Description, Username “Git Hub Accout”
- Enable Private Key “Enter directly” and paste your key



4.3 Configure a Jenkins Agent on USS

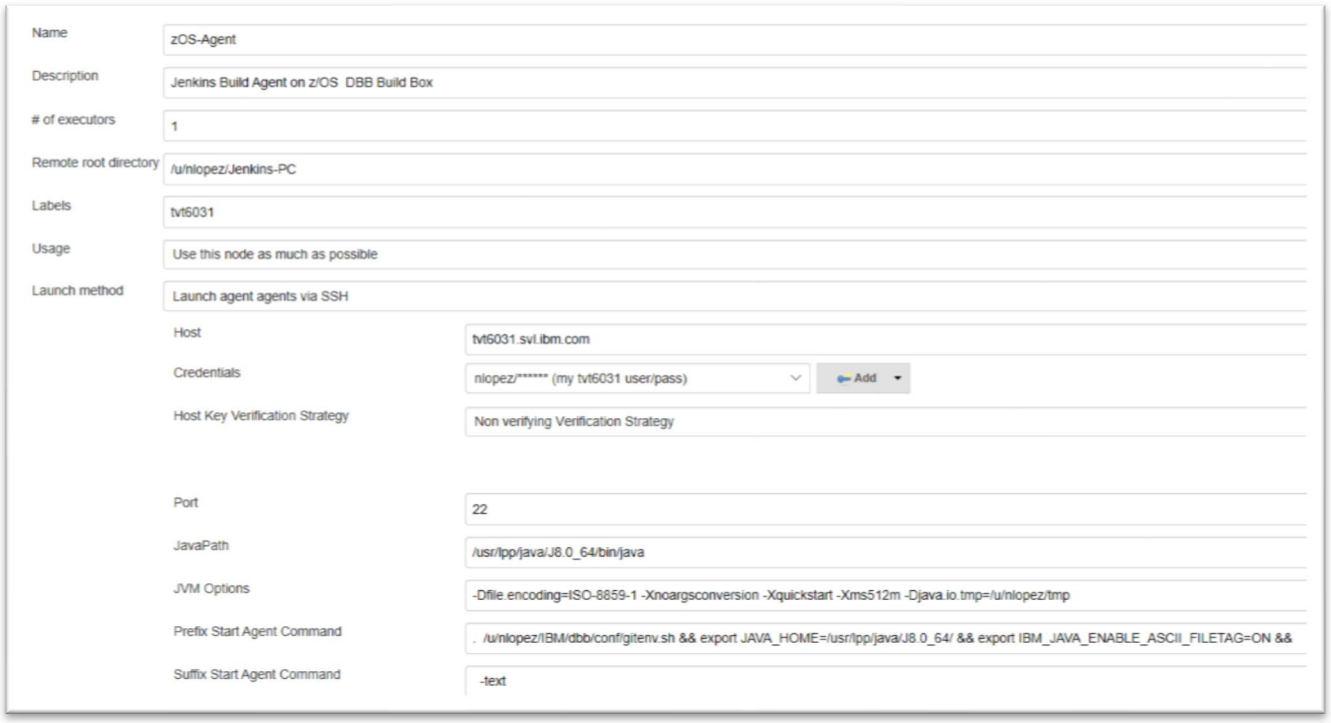
A Jenkins agent is a Java application that is deployed from a Jenkins Server to a remote z/OS Host. From "Build Executor Status” on the home page, click "New Node" and provide a name for your new agent. Select "Permanent Agent" and "OK". Click on **“Advanced…”** and fill in the required fields for your installation.





Fields	Values	Note
# of executors	1	
Remote root directory	/?/jenkins-remote	Where “?” is an absolute path to the z/OS User’s home directory and a folder that is automatically created.
Labels	?	A name for the agent to be used in pipeline jobs
Usage	Use this node as much as possible	
Launch Method	Launch agents via SSH	
Host	?	The URL or IP of the of the host z/OS

Credentials	?	From the dropdown select the previously defined z/OS credential (user id/passwd)
Host Key Verification…	Non verifying …	
Port	22	Ensure firewall rules allow access between the Jenkins Server and USS on the port.
Java path	/usr/lpp/java/J8.0_64/bin/java	
JVM Options	-Dfile.encoding=ISO-8859-1 -Xnoargsconversion -Xquickstart -Xms512m -Djava.io.tmp=/u/tmp	All one line. Confirm /u/tmp directory name with at least 1gb of free space.
Prefix Agent Start Command	. /usr/lpp/IBM/dbb/conf/gitenv.sh && export JAVA_HOME=/usr/lpp/java/J8.0_64/ && export IBM_JAVA_ENABLE_ASCII_FILETAG=ON && env &&	All one line. "gitenv.sh" was customized during the install
Suffix Start Agent Command	-text	**Ensure there is a leading space**. Accept the defaults for all other fields in this section.

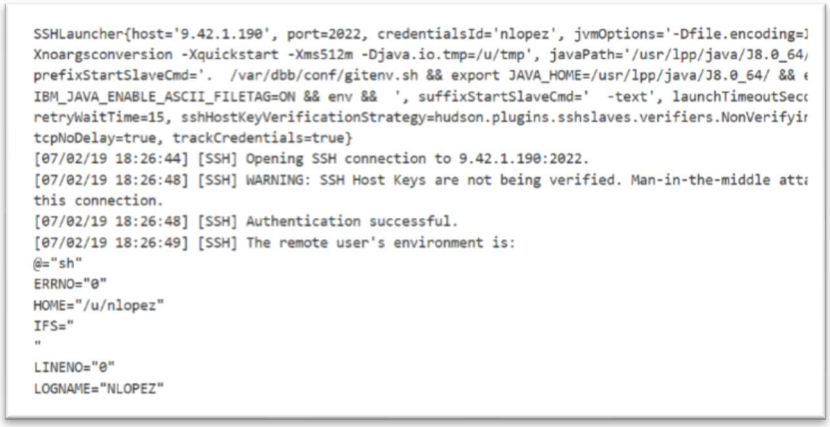


Scroll to the “Node Properties”, check off “Environment Variables” and “Tools Locations” add these names and values (*Note: the screen may not reflect your installation values):

Variable	Values	Note
JAVA_HOME	usr/lpp/java/J8.0_64	
PATH	/usr/lpp/java/J8.0_64/bin:/usr/lpp/java/J8.0_64/bin/classic: /usr/lpp/IBM/dbb/groovy-2.4.12/bin: /usr/lpp/IBM/dbb/bin:/var/rocket/bin:/bin	All one line
Git-USS	/usr/lpp/IBM/dbb/bin/git-jenkins.sh	The variable is the Git plugin name defined in the Global Plugin page
Groovy	/usr/lpp/IBM/dbb/groovy-2.4.12/bin	The variable is the Groovy plugin name defined in the Global Plugin page

Press “Save” and then “Launch the Agent”. A successful connection message is displayed at the bottom of the log.

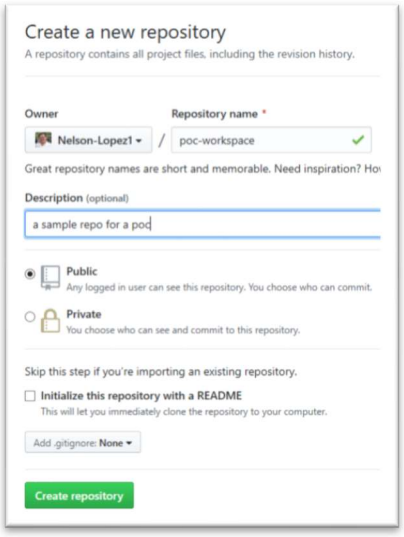
If the agent does not start, carefully review all configuration values. Use “Disconnect” and then relaunch to apply changes.



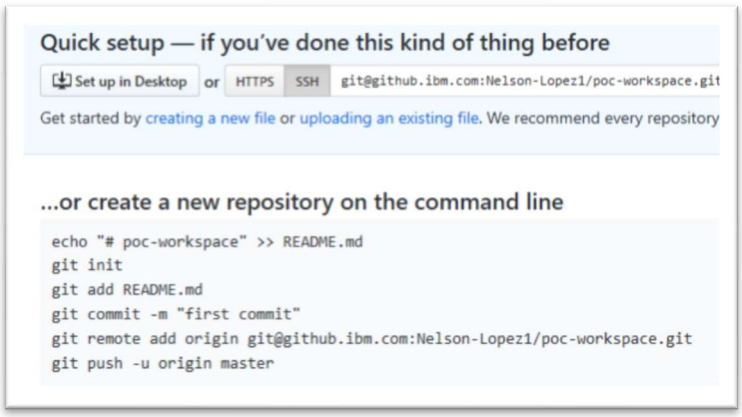
4.4 Create a GitHub repository

A DevOps Admin will follow these steps to initialize a new Git Repo as part of the POC. Other non-GitHub environments have similar steps.

1 - In Git Hub, create a repo called “poc-workspace”. After pressing “Create repository”, a screen with instructions will appear. Do not close it.



Instructions after "create repository" is pressed

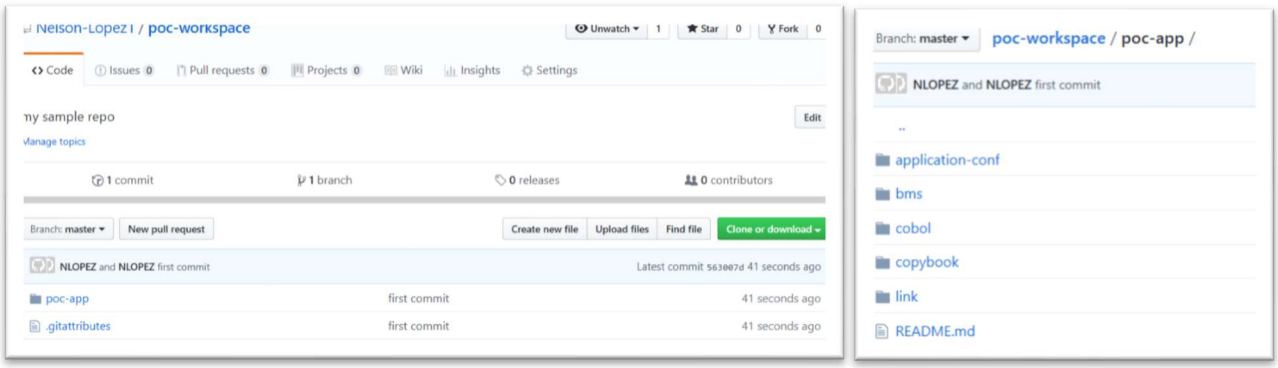


2 - In USS:

- **cd**
- **mkdir -p poc-workspace/poc-app**
- **cp $HOME/dbb-zappbuild/.gitattributes poc-workspace**
- **cp -r $HOME/dbb-zappbuild/samples/MortgageApplication/** poc-workspace/poc-app**
- **cd poc-workspace**
- **git init**
- **git add .**
- **git commit -m "first commit"**
- copy/paste the **"git remote ..."** and **"git push ..."** lines from the git instructions page into USS command line

3 - In Git Hub, refresh your repo page to view the results

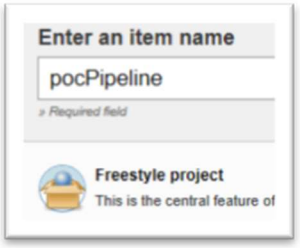
New Sample Repo



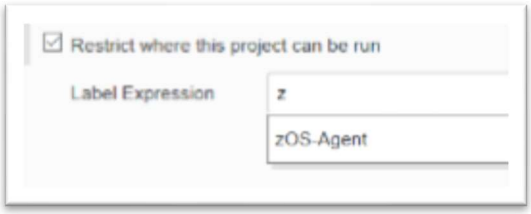
4.5 Create a Jenkins Pipeline Job

In Jenkins, select "New Item" on the home and:

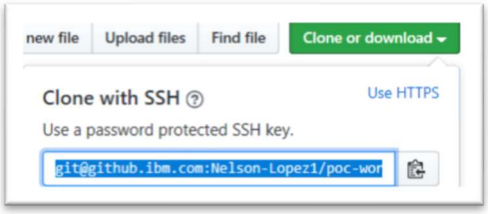
- Enter a name like “pocPipeline”, choose "Freestyle" and "OK"



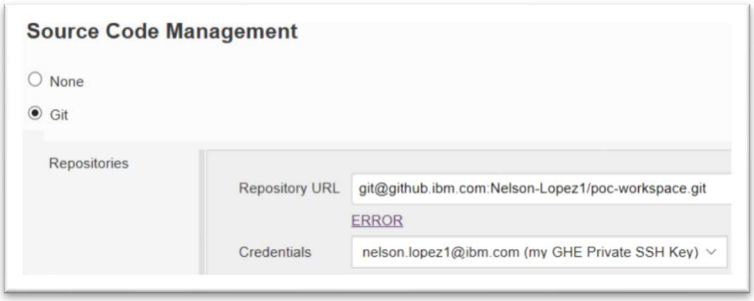
- Enable "Restrict where …” and enter the name of Jenkins Agent previously created.



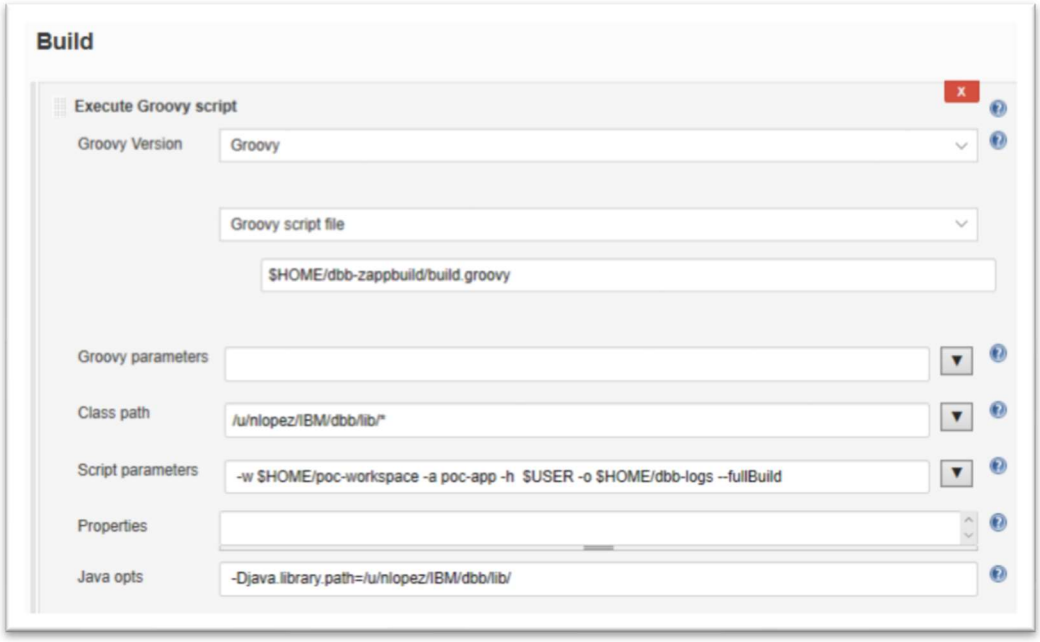
- “Repository URL” - enable “Git” and in Git Hub, press the green “Clone…” button for the new repo URL and cut/paste it.



- “Credentials” – select the private SSH key previously created.



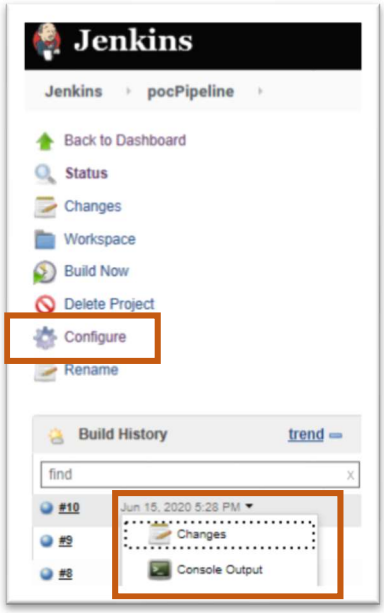
- Scroll to the “Build” section, press “Add Build Task” and select "Execute Groovy Script"
- Click "Advanced" and enter the values below:



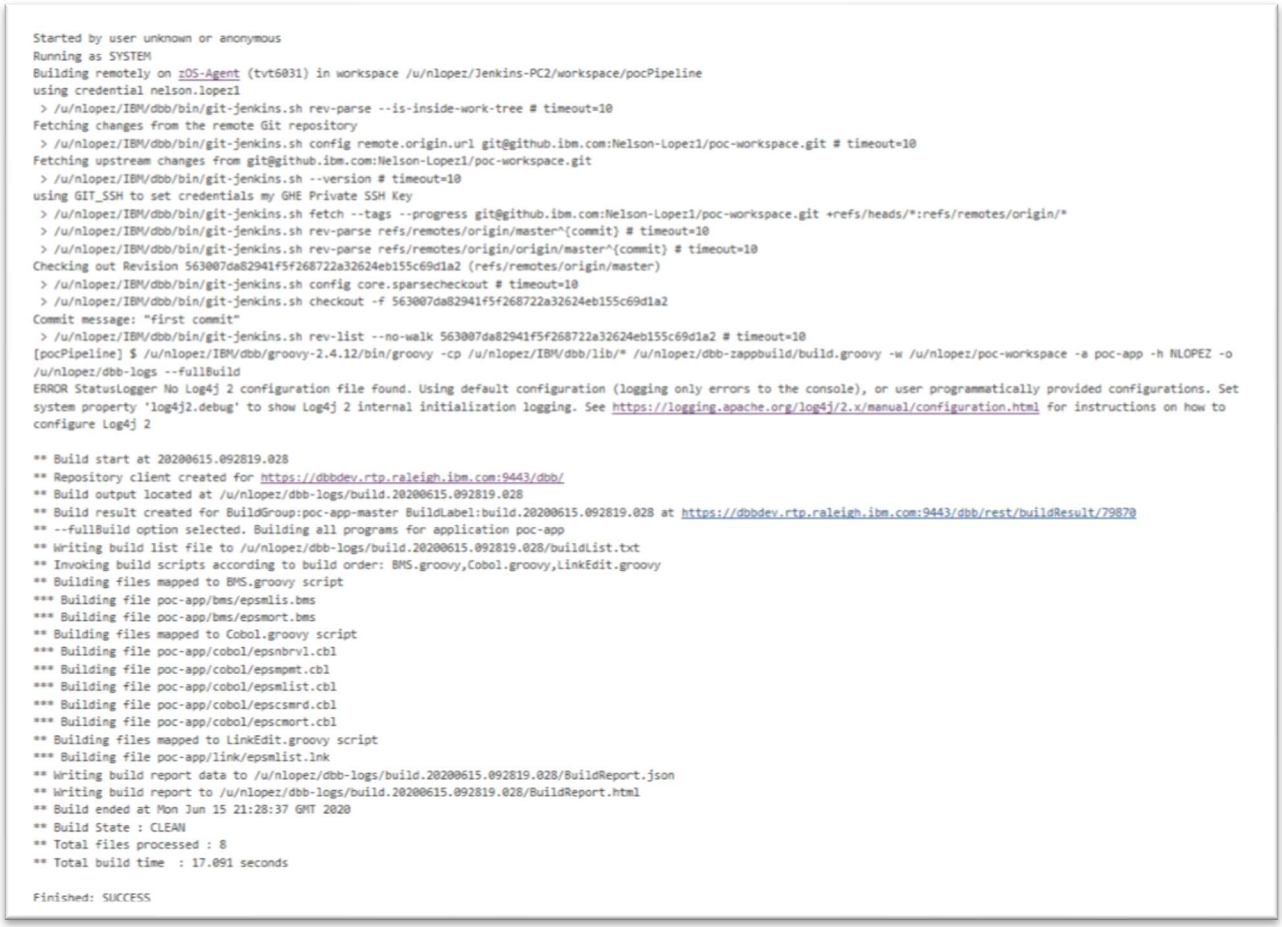
4.6 Run a Jenkins DBB build – Jenkins Admin

- From the Jenkins home page, click on the new job name.

- Select “Build Now” to run the job



- Press the dropdown by the new build timestamp and select “Console Output”



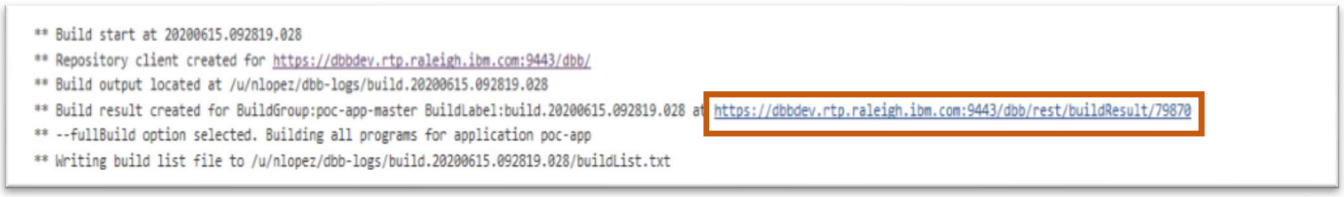
```
Started by user unknown or anonymous
Running as SYSTEM
Building remotely on zOS-Agent (tvt6031) in workspace /u/nlopez/Jenkins-PC2/workspace/pocPipeline
using credential nelson.lopez1
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh config remote.origin.url git@github.ibm.com:Nelson-Lopez1/poc-workspace.git # timeout=10
Fetching upstream changes from git@github.ibm.com:Nelson-Lopez1/poc-workspace.git
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh --version # timeout=10
using GIT_SSH to set credentials my GHE Private SSH Key
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh fetch --tags --progress git@github.ibm.com:Nelson-Lopez1/poc-workspace.git +refs/heads/*:refs/remotes/origin/*
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh rev-parse refs/remotes/origin/master^{commit} # timeout=10
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 563007da82941f5f268722a32624eb155c69d1a2 (refs/remotes/origin/master)
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh config core.sparsecheckout # timeout=10
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh checkout -f 563007da82941f5f268722a32624eb155c69d1a2
Commit message: "first commit"
 > /u/nlopez/IBM/dbb/bin/git-jenkins.sh rev-list --no-walk 563007da82941f5f268722a32624eb155c69d1a2 # timeout=10
[pocPipeline] $ /u/nlopez/IBM/dbb/groovy-2.4.12/bin/groovy -cp /u/nlopez/IBM/dbb/lib/* /u/nlopez/dbb-zappbuild/build.groovy -w /u/nlopez/poc-workspace -a poc-app -h NLOPEZ -o
/u/nlopez/dbb-logs --fullBuild
ERROR StatusLogger No Log4j 2 configuration file found. Using default configuration (logging only errors to the console), or user programmatically provided configurations. Set
system property 'log4j2.debug' to show Log4j 2 internal initialization logging. See https://logging.apache.org/log4j/2.x/manual/configuration.html for instructions on how to
configure Log4j 2

** 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
** --fullBuild option selected. Building all programs for application poc-app
** Writing build list file to /u/nlopez/dbb-logs/build.20200615.092819.028/buildList.txt
** Invoking build scripts according to build order: BMS.groovy,Cobol.groovy,LinkEdit.groovy
** Building files mapped to BMS.groovy script
*** Building file poc-app/bms/epsmlis.bms
*** Building file poc-app/bms/epsmort.bms
** Building files mapped to Cobol.groovy script
*** Building file poc-app/cobol/epsnbrvl.cbl
*** Building file poc-app/cobol/epsmpmt.cbl
*** Building file poc-app/cobol/epsmlist.cbl
*** Building file poc-app/cobol/epscsmrd.cbl
*** Building file poc-app/cobol/epscmort.cbl
** Building files mapped to LinkEdit.groovy script
*** Building file poc-app/link/epsmlist.lnk
** Writing build report data to /u/nlopez/dbb-logs/build.20200615.092819.028/BuildReport.json
** Writing build report to /u/nlopez/dbb-logs/build.20200615.092819.028/BuildReport.html
** Build ended at Mon Jun 15 21:28:37 GMT 2020
** Build State : CLEAN
** Total files processed : 8
** Total build time  : 17.091 seconds

Finished: SUCCESS
```

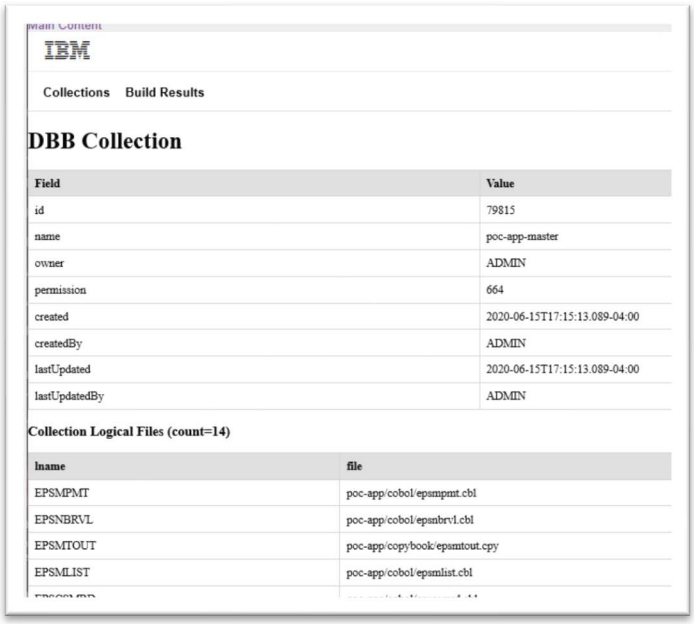
4.7 View DBB build results in the DBB WebApp

From the console log, click on the DBB Web App URL to view the build results (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
** --fullBuild option selected. Building all programs for application poc-app
** Writing build list file to /u/nlopez/dbb-logs/build.20200615.092819.028/buildList.txt
```

View the application's DBB Collection named "poc-app-master" in the WebApp.



Main Content

IBM

Collections Build Results

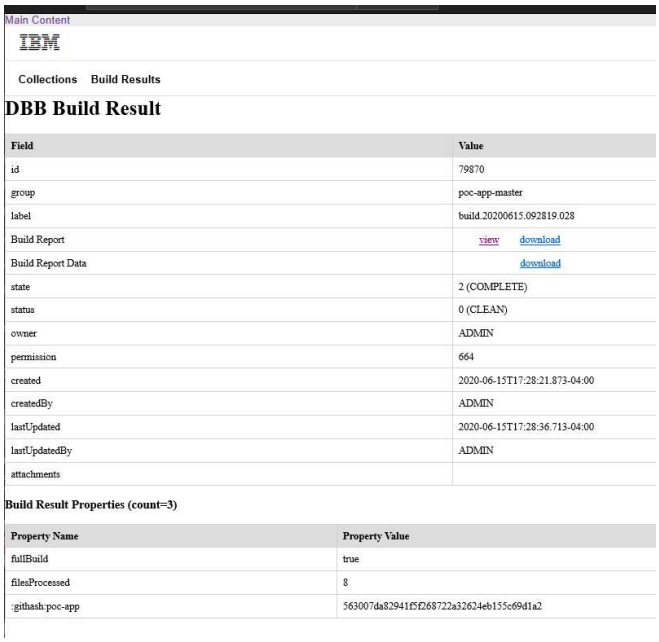
DBB Collection

Field	Value
id	79815
name	poc-app-master
owner	ADMIN
permission	664
created	2020-06-15T17:15:13.089-04:00
createdBy	ADMIN
lastUpdated	2020-06-15T17:15:13.089-04:00
lastUpdatedBy	ADMIN

Collection Logical Files (count=14)

lname	file
EPSMPMT	poc-app/cobol/epsmpmt.cbl
EPSNBRVL	poc-app/cobol/epsnbrvl.cbl
EPSMTOUT	poc-app/copybook/epsmtout.cpy
EPSMLIST	poc-app/cobol/epsmlist.cbl

Check the build report related to this collection.



Main Content

IBM

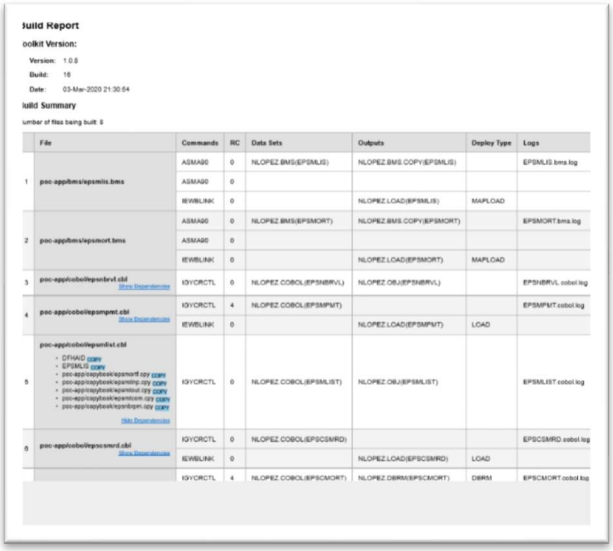
Collections Build Results

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	ADMIN
permission	664
created	2020-06-15T17:28:21.873-04:00
createdBy	ADMIN
lastUpdated	2020-06-15T17:28:36.713-04:00
lastUpdatedBy	ADMIN
attachments	

Build Result Properties (count=3)

Property Name	Property Value
fullBuild	true
filesProcessed	8
:githash:poc-app	563007da82941f5f268722a32624eb155c69d1a2



Build Report

oolkit Version:

Version: 1.0.8
Build: 16
Date: 03-Mar-2020 21:30:54

uild Summary

umber 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 • DFHAID copy • EPSMLIS copy • poc-app/copybook/epsmort.cpy copy • poc-app/copybook/epsmtrp.cpy copy • poc-app/copybook/epsmtrxd.cpy copy • poc-app/copybook/epsmtcom.cpy copy • poc-app/copybook/epsmtrpm.cpy copy Hide 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	
		IGYCRCTL	4	NLOPEZ.COBOL(EPSCMORT)	NLOPEZ.DBRM(EPSCMORT)	DBRM	EPSCMORT.cobol.log

5 IDz configuration for user build

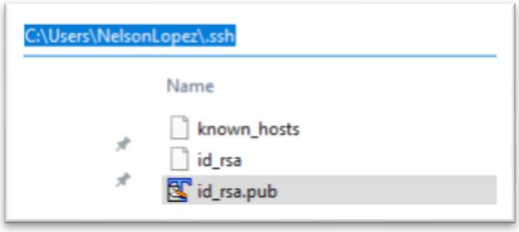
Required skills: DevOps Admin

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

5.1 Add your Windows' SSH key to Git

Optionally, perform these steps if you plan to use SSH to connect from IDz to your Git server (GitHub for instance). From a windows terminal:

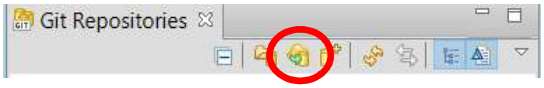
- **ssh-keygen -t rsa -b 4096 -C "your Git Hub acct "**
- Navigate to your windows "user_directory/.ssh" folder and open "id_rsa.pub" file.
- Cut/paste the key into Git. See "Add your SSH Key to Git" step 1 in the "Jenkins and Git Credentials" section above



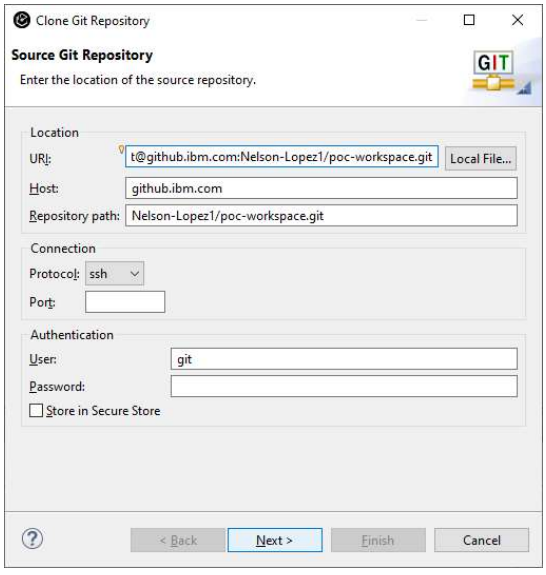
5.2 Clone with eGit plugin

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

- Select the clone icon:

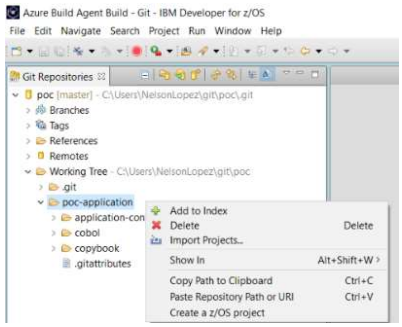


- Paste your new repo's URL and follow the prompts:



5.3 Create an IDz z/OS .project file

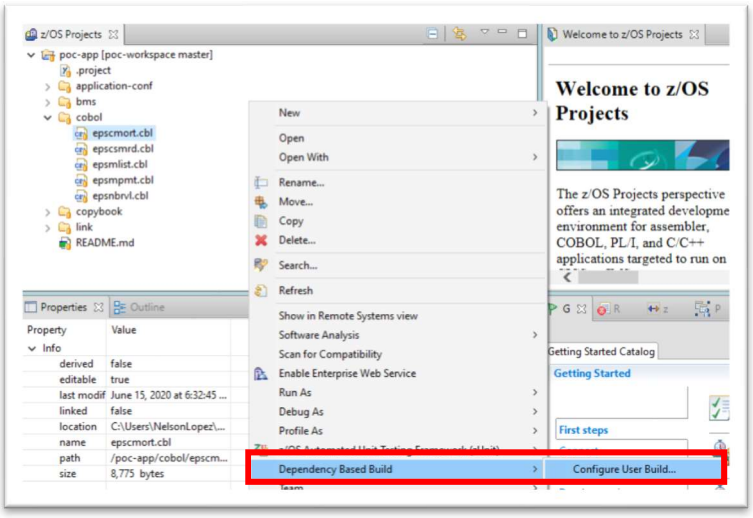
- Open the poc-application folder
- Right click and select the “Create a z/OS project”



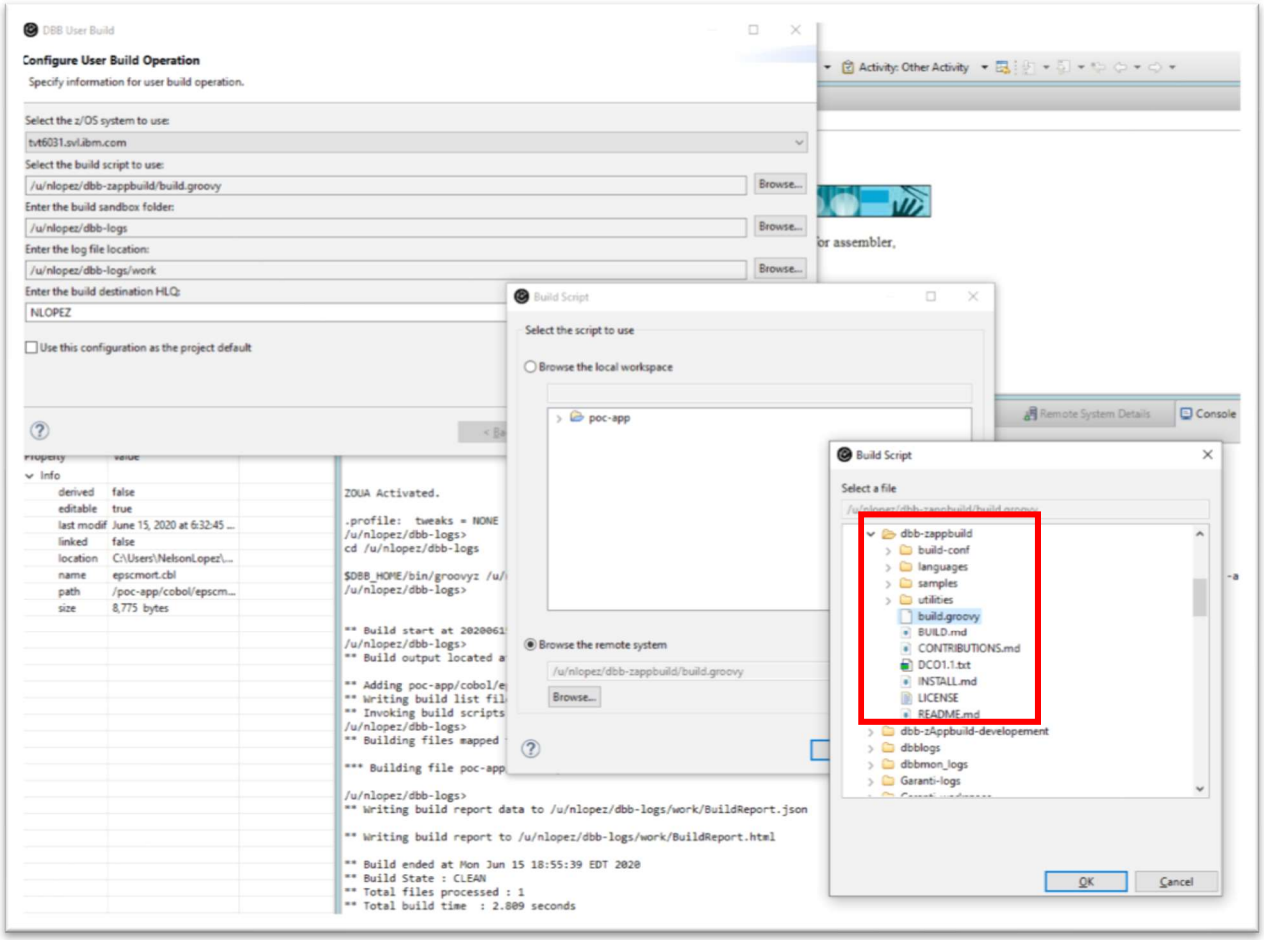
5.4 Configure and run a DBB user build

From the z/OS perspective:

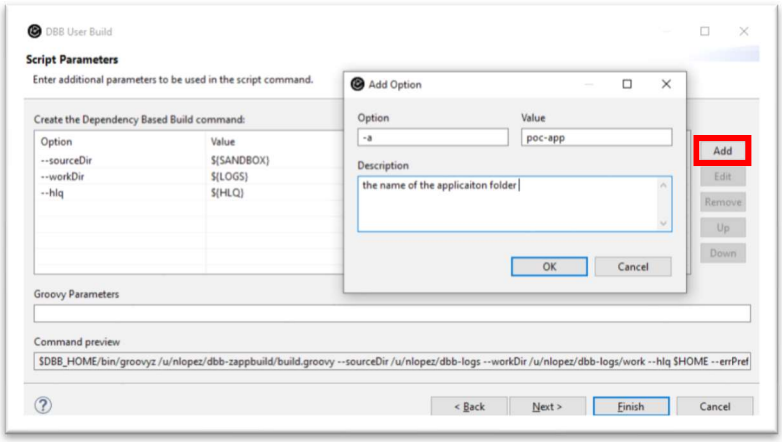
- Right click on a cobol program
- Select “Dependency Based Build/Configure User Build”.



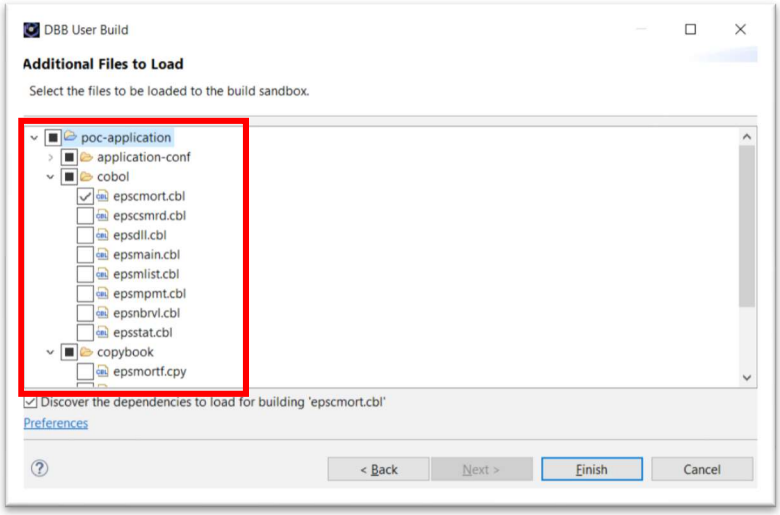
- On the next screen (see below), “Select the z/OS …” from the dropdown to point to zAppBuild.
- For “Select the build script …” and “…sandbox…”, press “Browse”, and browse on the next page.
- Navigate to your USS home folder and select “dbb-zappbuild/build.groovy” and then “dbb-logs” as the sandbox folder.



- "Enter the log file …" is auto-filled
- "HLQ" - enter your z/OS ID.
  Press "Next"and then "ADD" the **"-a"** option as shown with a value of "poc-app".



- Press "Next" and select the "application-conf" folder (you only need to do this the first time and each time you change something in this folder).



- 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.