LAB 10 – Deploying COBOL/CICS/DB2 application using a GitLab Ci Pipeline (60 minutes)

Created by Mathieu Dalbin , Regi Barosa, Ronnie Geraghty and Wilbert Kho and– July 01-2021,

,

GitLab is an integrated DevOps platform, which fundamentally transforms the way Development, Security, and Ops teams collaborate to build and deploy software. From development to production, GitLab helps teams optimize the development cycle by decreasing time to market and increasing developer productivity through the implementation of DevOps best practice. GitLab enables collaboration by providing features to manage issues and milestones through dashboards. These features allow the team to organize their tasks into groups, projects, epics and issue boards.

This lab will take you through the steps of using GitLab CI along with DBB, ZUnit and UrbanCode Deploy (UCD) on z/OS.

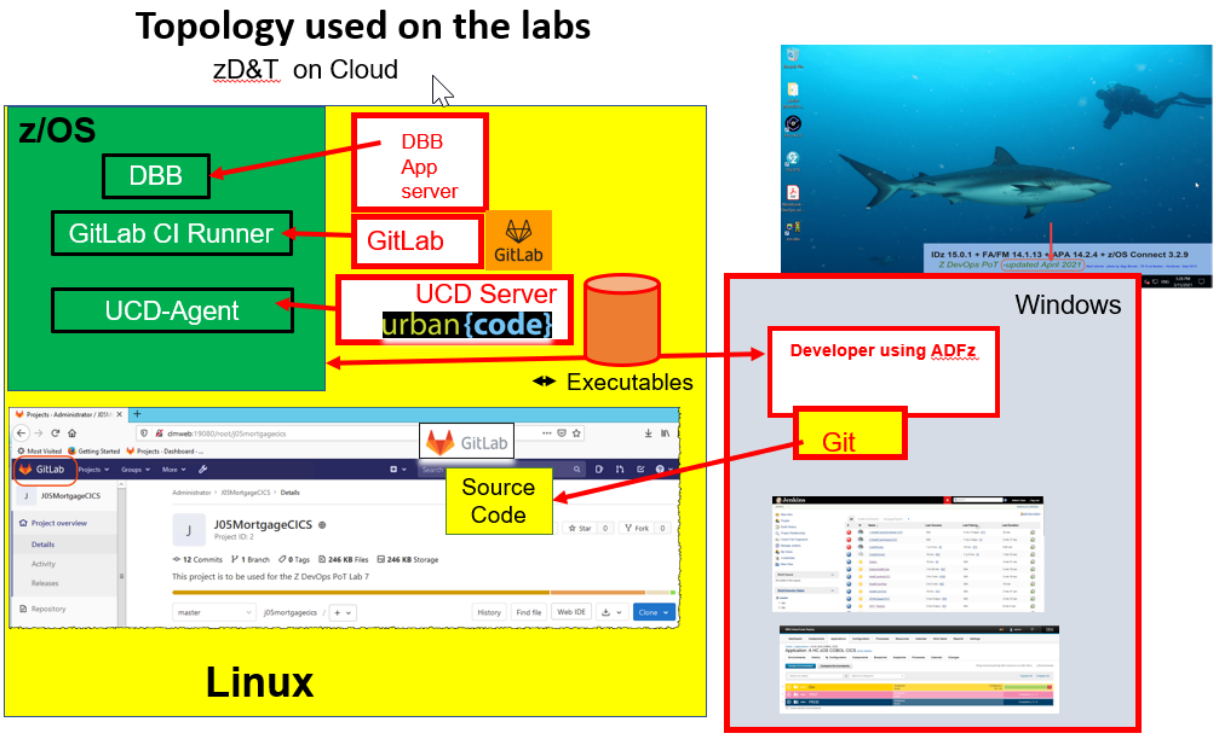
On this lab you will fix a bug of an existing COBOL/CICS application stored in GitLab.
You will use **IDz** to change the code and perform a personal test for later delivery and commit to *GitLab* and then use *GitLab CI* for the final build and continuous delivery.

The updated code will be deployed to CICS using *UrbanCode Deploy* (UCD)

The process would be similar for a PL/I program or IMS instead of CICS.
The environment used on this Lab is pictured below.

Note that we have few “servers” running on Linux like **UCD** server, **Jenkins**, **DBB Application Server**.
Also the **GitLab Repository** is on Linux.

The z/OS has many other running tasks including **CICS, DB2, DBB code** and agents that interact with the Linux servers.



Overview of development tasks

To complete this tutorial, you will perform the following tasks:

1. **Review the GitLab issue and verify the bug using the 3270 terminal**
   → You will start GitLab, verify the issue and using a 3270 emulation execute a transaction named **SSC1** to become familiar with the Application that you intend to modify.
   When using customer number 1 notice that the DOB is 0000-00-00. This is a bug that needs to be fixed
2. **Load the source code from GitLab to the local IDz workspace**
   → using IDz you will clone the GitLab loading all COBOL source code to your windows client.
3. **Use IDz to run the zUnit test case and verify the error using the debug**
   Running zUnit you will verify that the DOB is incorrect,
4. **Modify the COBOL/CICS/DB2 program that has the bug using IDz.**
   → Using *IDz* you will modify the COBOL program LGICDB01 to fix the bug.
5. **Use IDz *DBB User Build* to compile/link and run the zUnit**
   → You will compile and link the modified code using the *DBB User Build* Function.
   When complete you will run zUnit generated test case and verify that the bug is fixed.
6. **Push and Commit the changed code to GitLab .**
   → You will commit the changes to *Git*. This will initiate the GitLab CI pipeline
7. **Verify the GitLab CI Build execution.**
   → You will use Gitlab to build the new changed code, run zUnit and push the executables to be deployed using *UCD*. .
8. **Verify the GitLab CI Package and Deploy to UCD and test the CICS transaction again using 3270**
   → You will verify the results after the final deploy to *CICS* using *UCD*

What is Git and DBB ?

Git is an open Source Code Management tool that is very popular in the distributed world.
In early 2017, Rocket Software ported Git into the mainframe – with the necessary checks to handle EBCDIC to UTF-8 conversions and vice-versa.

In Q3 2017, IBM released an Open Beta of **Dependency Based Build (DBB).** DBB provides a build tool that provides the build framework, dependency understanding, and tracking for builds run on z/OS. This build system is not dependent on any SCM or Continuous integration automation tool. In this lab, we use DBB to build our z/OS COBOL source code which resides in the distributed Git Repositories.
DBB is part of IBM Developer for Z Systems EE (Enterprise Edition) or ADFz and was announced on March 13, 2018.

***GitHub** vs. **Bitbucket** vs. **GitLab** ?*
More at: https://stackshare.io/stackups/bitbucket-vs-github-vs-gitlab

GitHub, ***Bitbucket,*** and ***GitLab*** are code collaboration and version control tools offering repository management. They each have their share of fans, though ***GitHub*** is by far the most used of the three.
Of the three, only ***GitLab*** is open source, though all three support open source projects.
GitHub offers free public repositories; ***Bitbucket*** also offers free private repositories;
GitLab offers a Community Edition which is entirely free

Section 1. Review the GitLab issue and verify the bug using the 3270 terminal

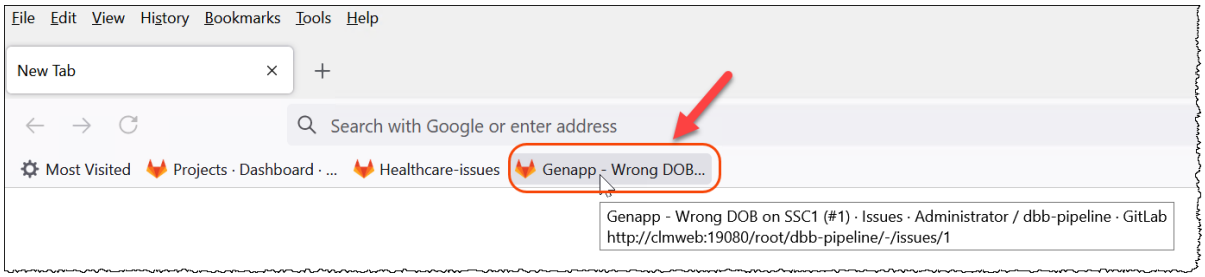
You will access GitLab, verify the issue and using a 3270 emulator execute a transaction named SSC1 to become familiar with the Application that you intend to modify.

When using customer number 1 notice that the DOB is 0000-00-00. This is a bug that needs to be fixed.

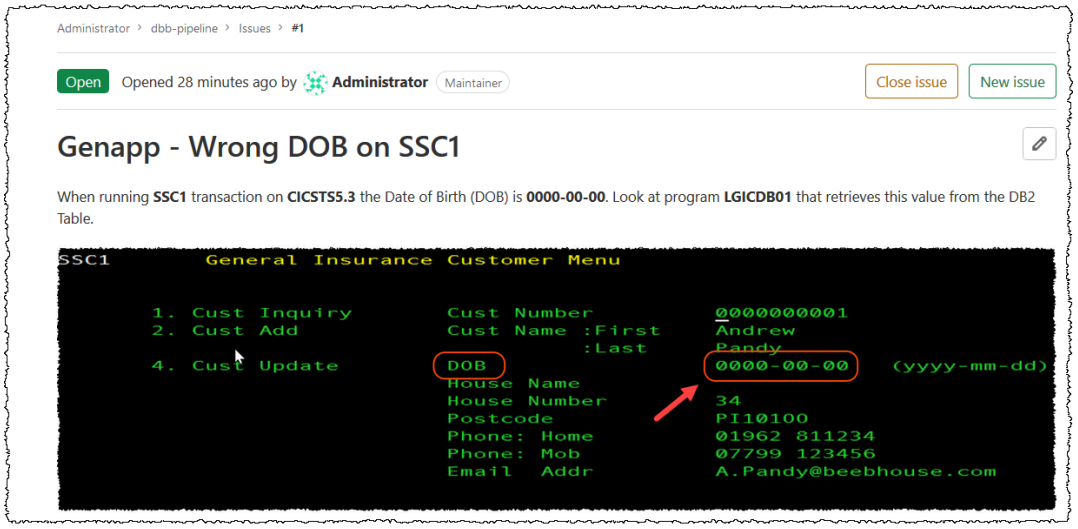
___ 1.0 Access GitLab and verify the issue.

1.0.1 ⏭ Start a browser by clicking the icon in the bottom of your screen

1.0.2 ⏭ Click on the issue **Genapp – Wrong DOB** as below



This explains the current bug on the application



1.0.3 ⏭ Minimize the web browser to go to the Windows desktop

___ 1.1 Connect to z/OS and emulate a CICS 3270 terminal

1.1.1 Start *IBM Developer for z Systems version 15* if it is not already started

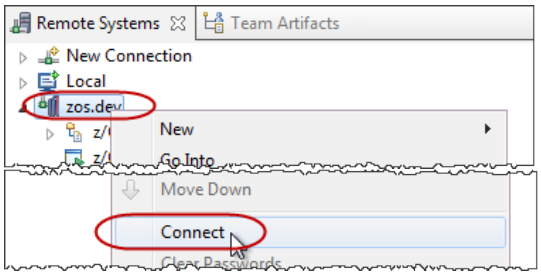
⏭ Using the desktop double click on **IDz V15** icon.

⏭ Verify that the message indicates that it is Version **15.0.1**

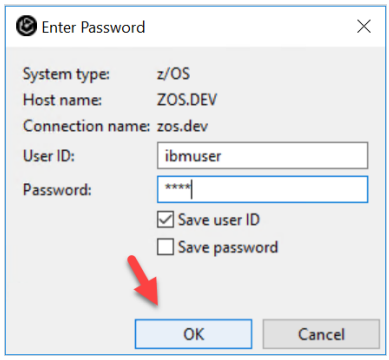
IMPORTANT -> This icon will start an IDz workspace that has already some definitions required for this lab. PLEASE DO NOT start IDz using other way than this icon.

1.1.2 ⏭ Open the **z/OS Projects** perspective by selecting
Window > Perspective > Open Perspective > z/OS Projects

1.1.3 ⏭ Right click on **zos.dev** and select **Connect.**

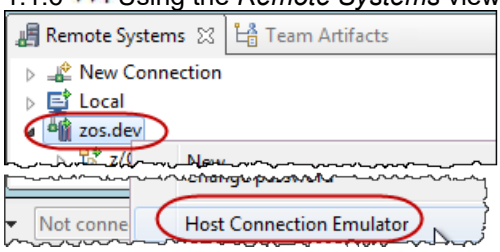


1.1.4 ⏭ Type **ibmuser** as userid and **sys1** as password. Click **OK** to connect to z/OS.

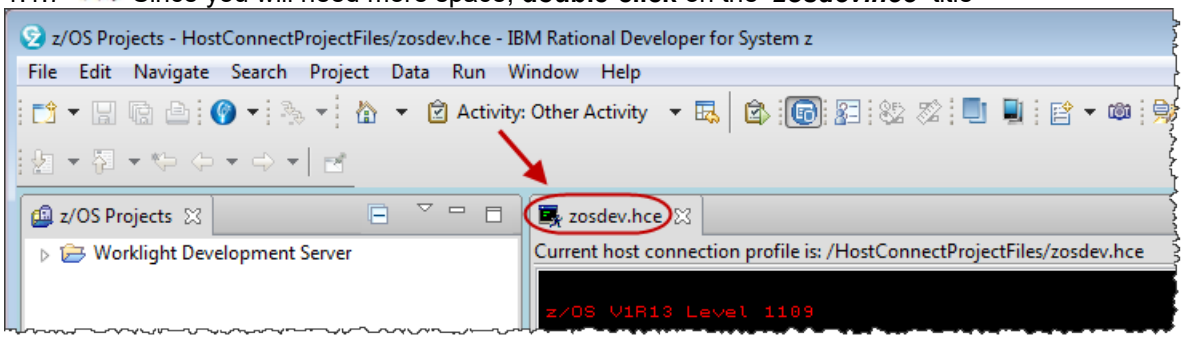


1.1.5 ⏭ Wait until connection is complete (look at the bottom and left until the green bar disappears)

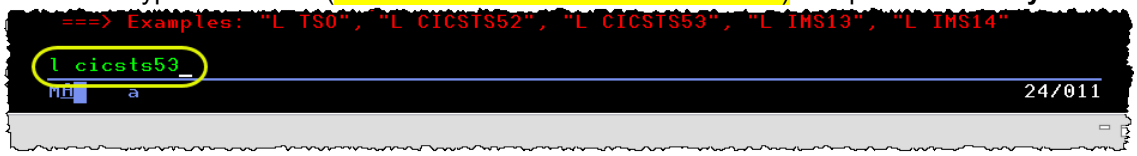
1.1.6 Using the *Remote Systems* view, right click on **zos.dev** and select **Host Connection Emulator.**



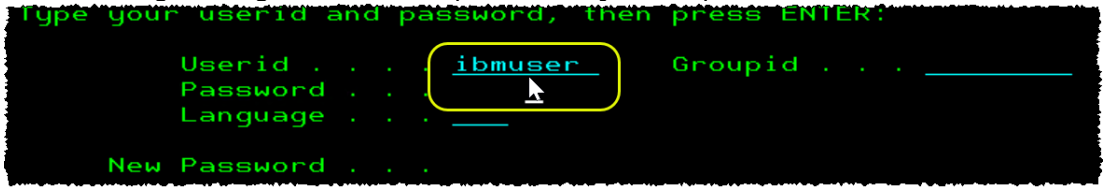
1.1.7 Since you will need more space, **double-click** on the ***zosdev.hce*** title



1.1.8 Type **l cicsts53**. (where "l" is the lower case of letter "L") and press **Enter key**.



1.1.9 Logon using **ibmuser** and password **sys1** and press **Enter**.

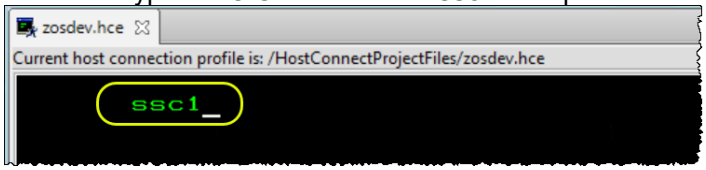


1.1.10 The sign-on message is displayed

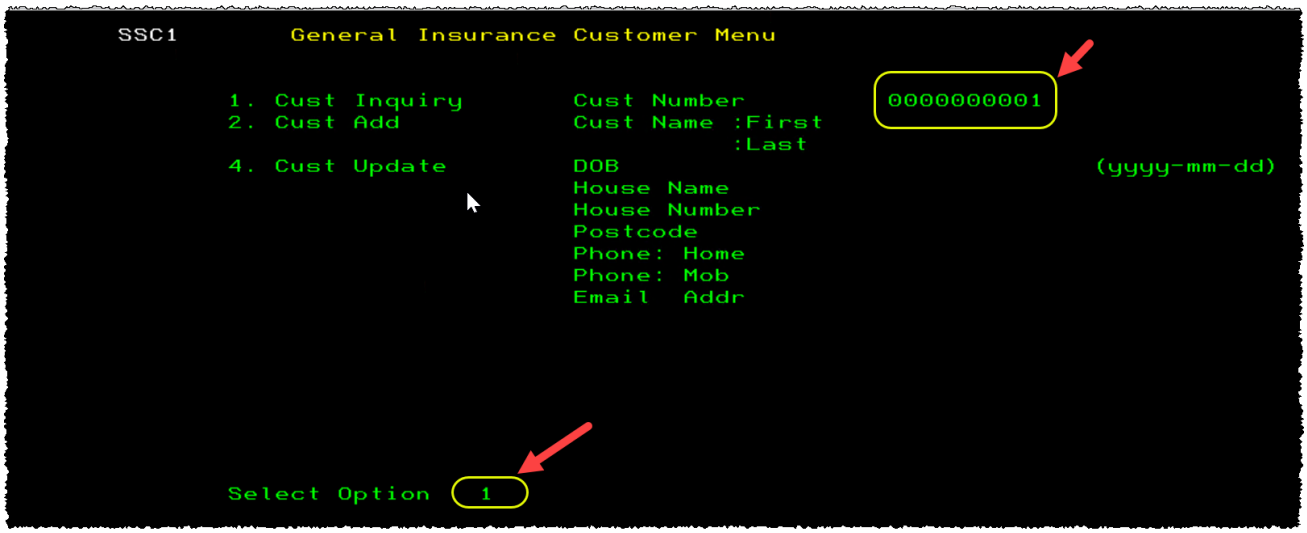
___ 1.2 Run CICS transaction SSC1

You should now be in the z/OS CICS region named *CICSTS53*. This is the CICS instance where you will make the program changes.

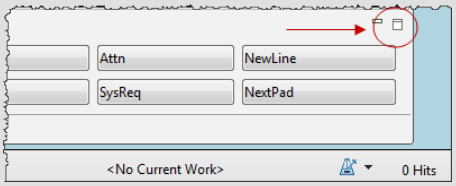
1.2.1 Type the CICS transaction **ssc1** and press the **Enter** key.



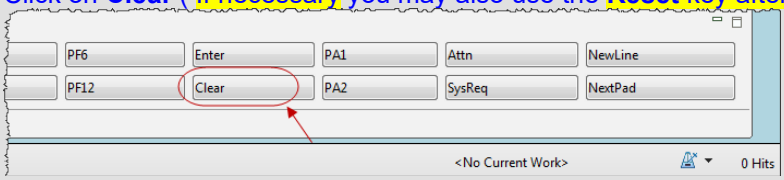
1.2.2 Move the mouse to last 0 and type **1**, use the tab key and type **1** as ***Select option*** and press ***Enter***



You may need to use the **clear** key. If the clear button is not displayed, look in the right lower corner, select this icon . This will display possible keys, including the clear button.

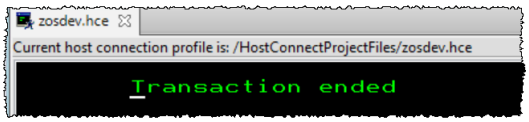


Click on **Clear** (If necessary you may also use the **Reset** key after clicking **NextPad**)



1.2.3 The data below is retrieved from a DB2 table .
Notice that **DOB** (*Date Of Birthday*) is **0000-00-00.** This is a bug that you will need to fix. The correct date should be a valid year, month and day..

1.2.4 ⏭ Press **F3** to end the application.



1.2.5 ⏭ **Close** the terminal emulation clicking on ⊠ → zosdev.hce ⊠.
Or pressing **CTRL + Shift + F4**.

What have you done so far?
You emulated a 3270 terminal using IBM Developer for System z.
You also executed the CICS transaction **SSC1** and verified an existing bug in the Genapp application. The objective here was to show the bug that you will correct.

Section 2 – Load the source code from GitLab to the local IDz workspace

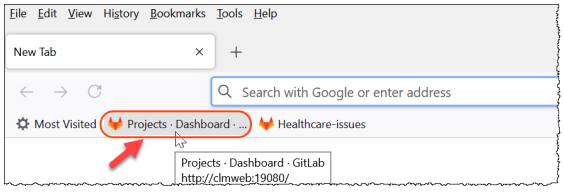
You will load the COBOL code that is stored on GitLab (Linux) to your Windows client to be modified.

___ 2.1 Cloning the Git Repository

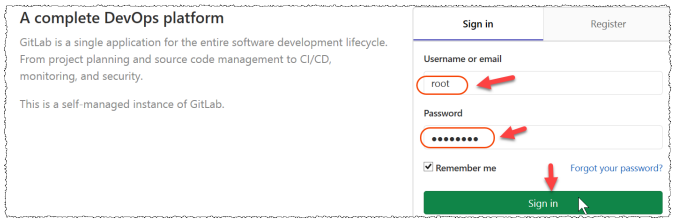
The Genapp Application used in this lab has its source code in the *GitLab Repository* that is on a Linux system. You must connect to the GitLab Repository and clone the Git project into the IDz. This brings the source code into your IDz workspace for edits and build.

2.1.1 Before cloning the Git repository, you should visualize the code stored at GitLab.
⏭ Go back to the web browser clicking in the icon in the bottom of your screen

2.1.2 ⏭ Click on this bookmark below to access ***GitLab***. The URL used is http://clmweb:19080/

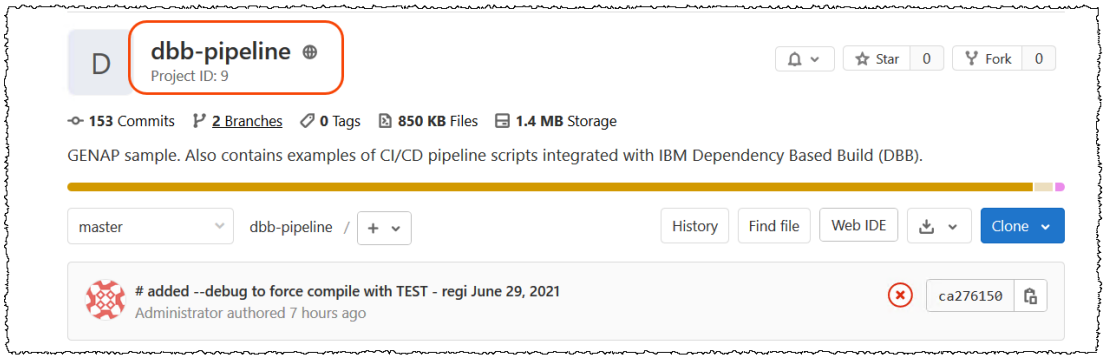


⏭ If asked for credentials use **root** and password: **zdtlinux**



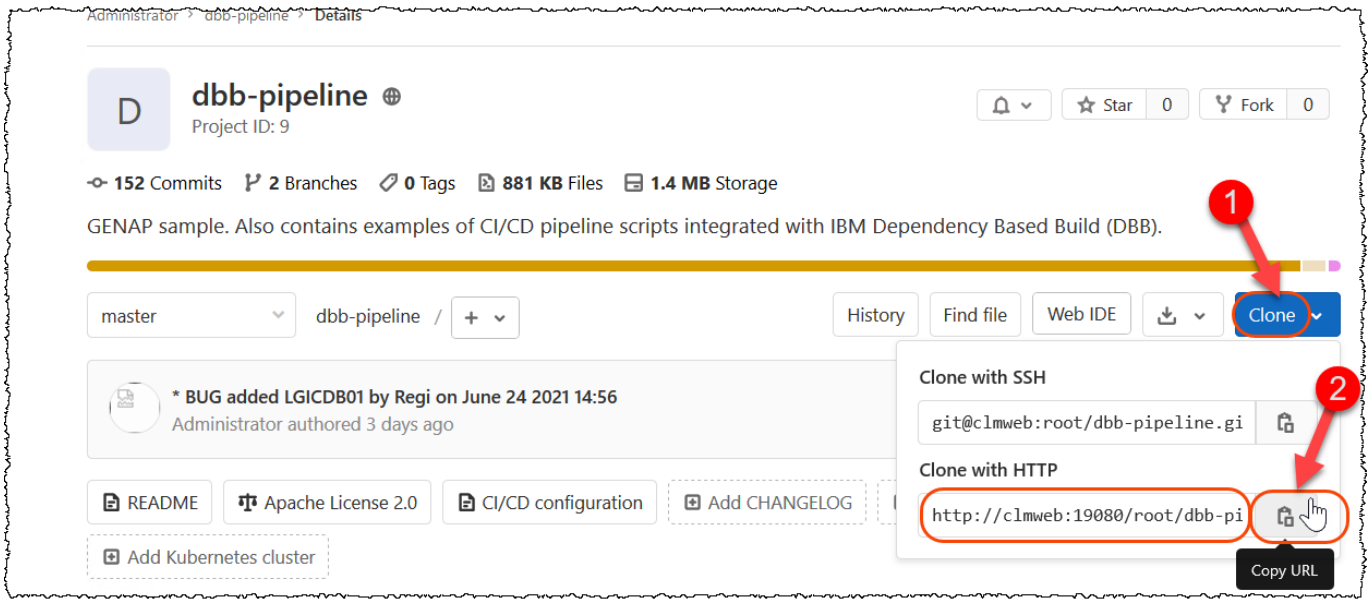
2.1.3 **Click** on the hyperlink below to see the **dbb-pipeline** repository.

2.1.4 You can see the **dbb-pipeline** repository.
You may browse the content if you want. The source code to be used on this lab is there



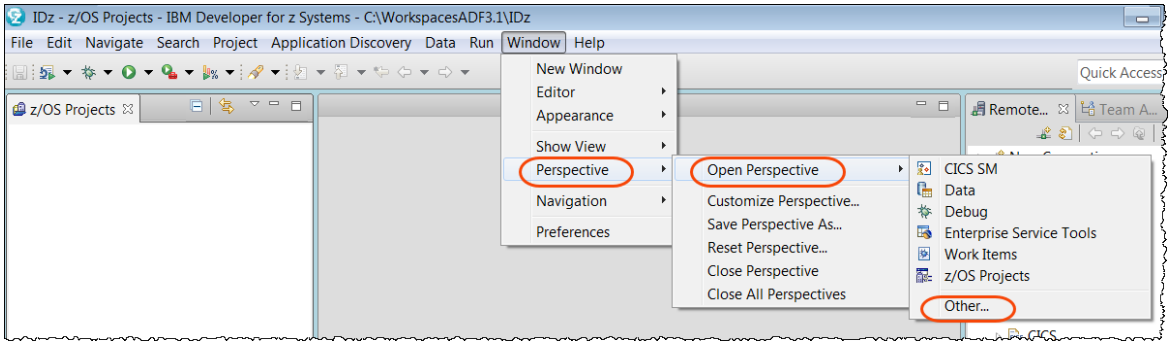
2.1.5 In order to clone it at your window desktop you could use *HTTP* or *SSH.*
Since *SSH* is blocked in our environment we need to use ***HTTP.***

1 Click on **Clone** (the blue button) and 2 in the icon to **copy the URL**
This value will be kept in the windows clipboard.

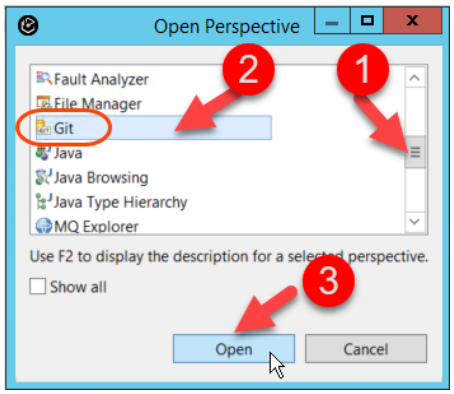


2.1.6 **Go back to IDz** using the icon that is on the base of your screen:

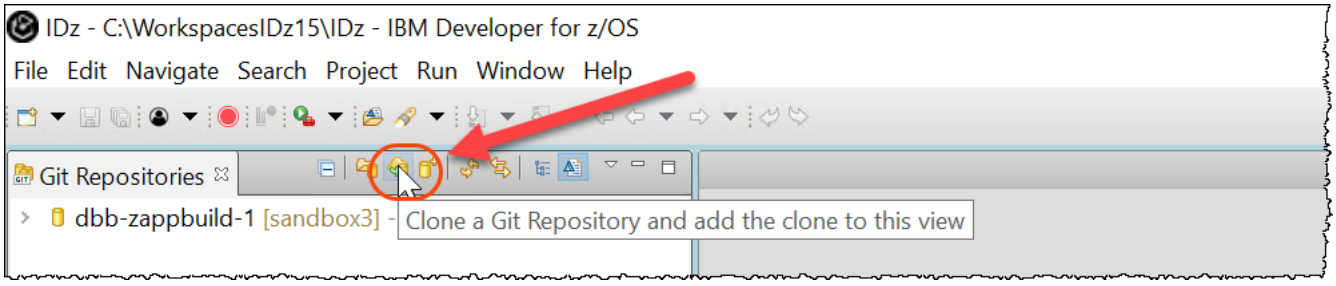
2.1.7 Open the **Git** perspective by selecting
Window > Perspective > Open Perspective > Other…



2.1.8 Select **Git** and click **Open**

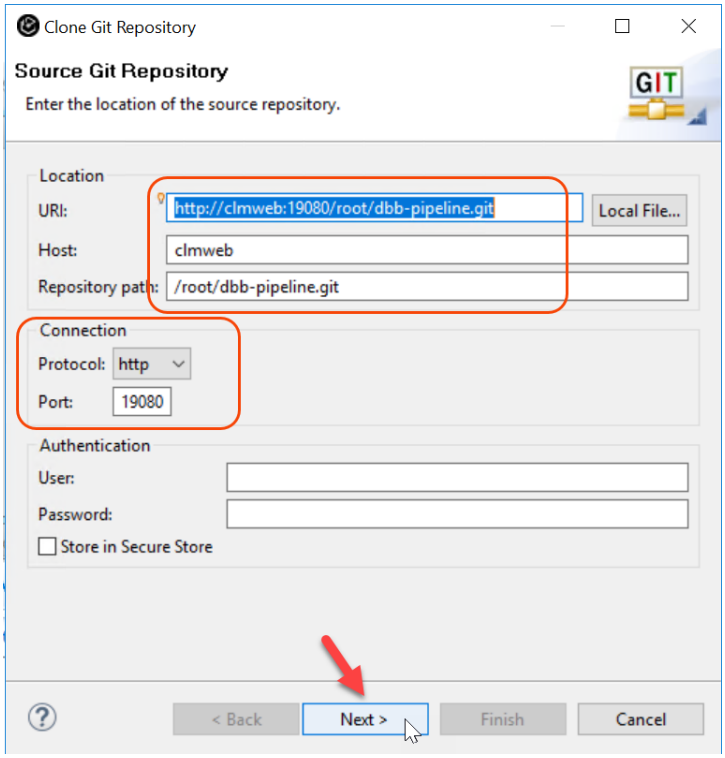


2.1.9 In the *Git Repositories* tab, click on the icon to '**Clone a Git repository**'.



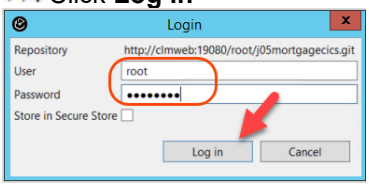
2.1.9 ⏭ The values copied from the web page (copy URL) will be shown in the Clone Git Repository.
Tip: In case you don't see it, got back to the page and copy it again (steps 2.1.1-2.1.5 above).

⏭ Click **Next**

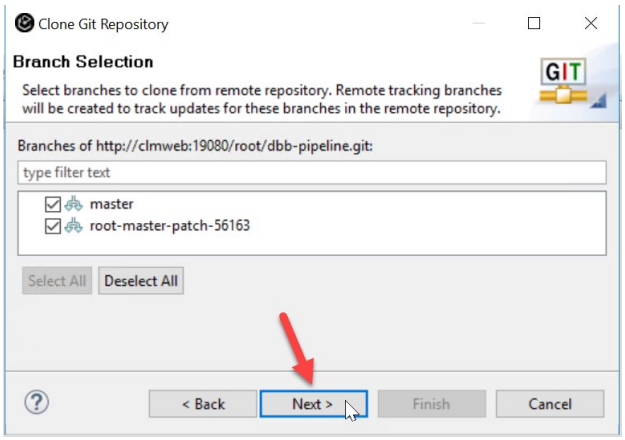


2.1.10 If a dialog asks for login the credentials are a use user: **root** and password **zdtlinux**

⏭ Click **Log in**



2.1.11 We will clone all branches. ⏭ Click **Next**

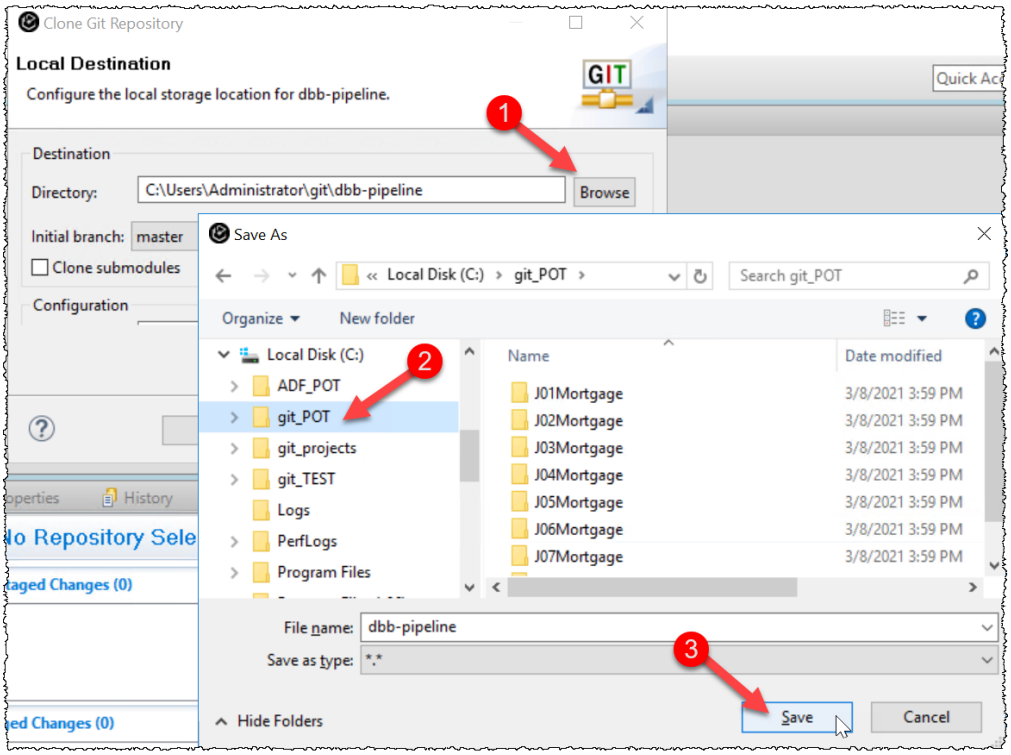


2.1.12 We will clone the repository on your local windows and import to be shown on IDz

1 ⏭ Use **Browse** button to select the directory `C:\git_POT\` **(on left)**.

2 ⏭ Click folder **git_POT** **on left**

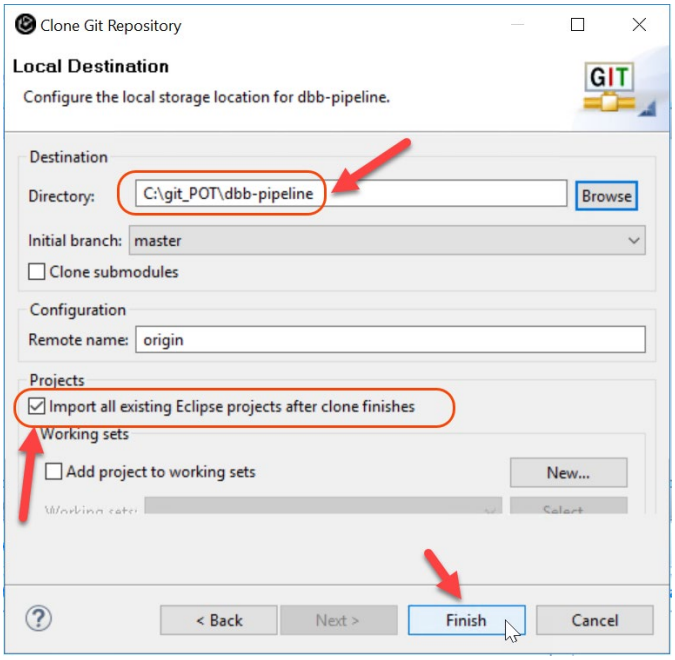
3 ⏭ Click **Save**



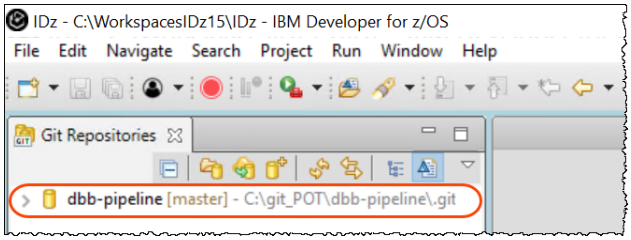
2.1.13 The *Directory* now should point to `C:\git_POT\dbb-pipeline`

⏭ Be sure that you selected **Import all existing Eclipse projects after clone finishes**

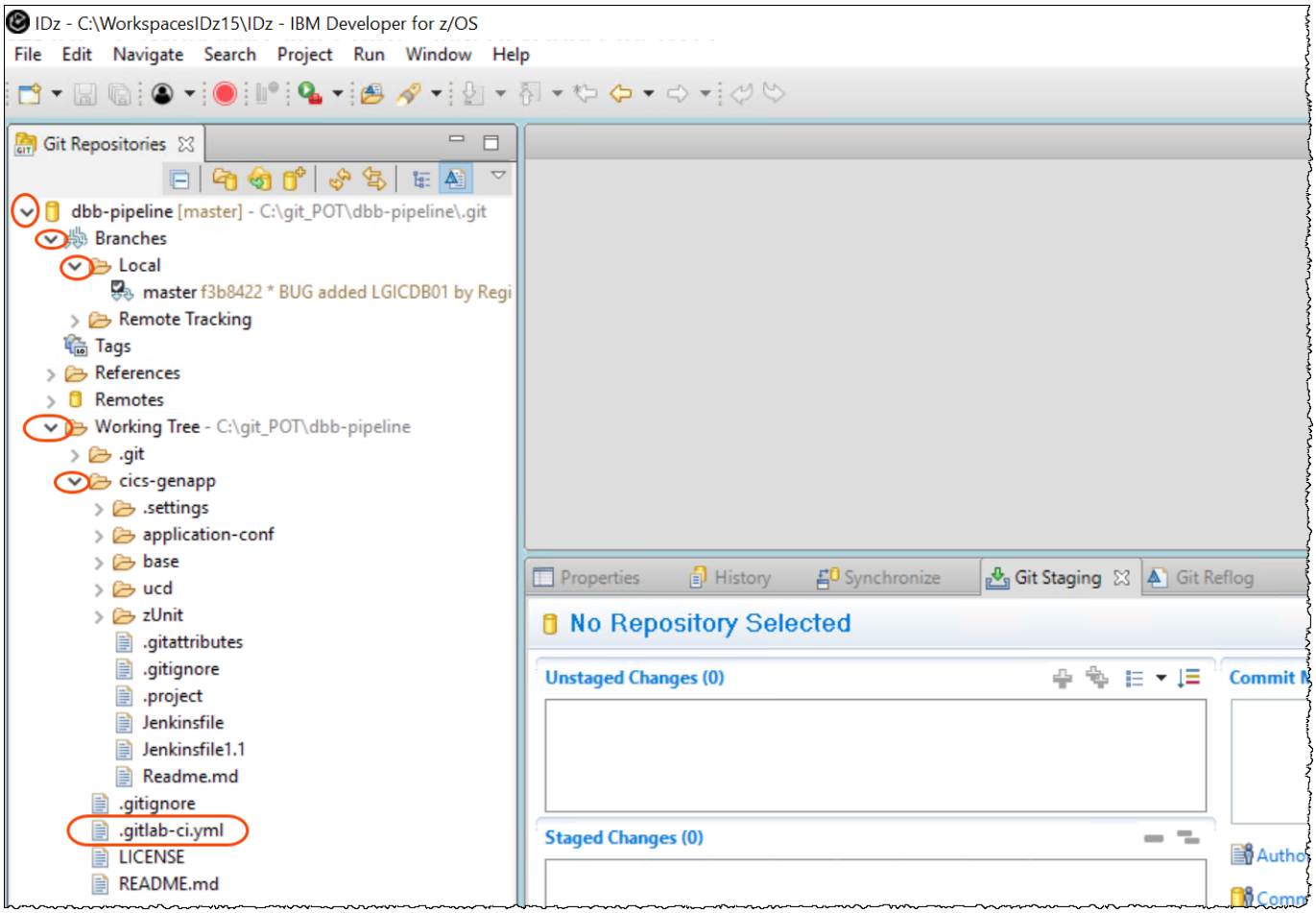
⏭ Click **Finish**



2.1.14 The repository that has the Genapp Application will be cloned from the Remote master repository and will appear in the *Git Repositories* view.



2.1.15 **Expand the nodes** by left clicking on the icon as shown below:
Notice the **`.gitlab-cy.yml`** that will drive the deploy later

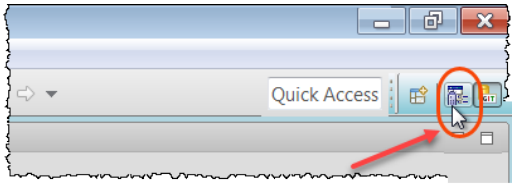


___ 2.2 Verify the code cloned using z/OS projects perspective

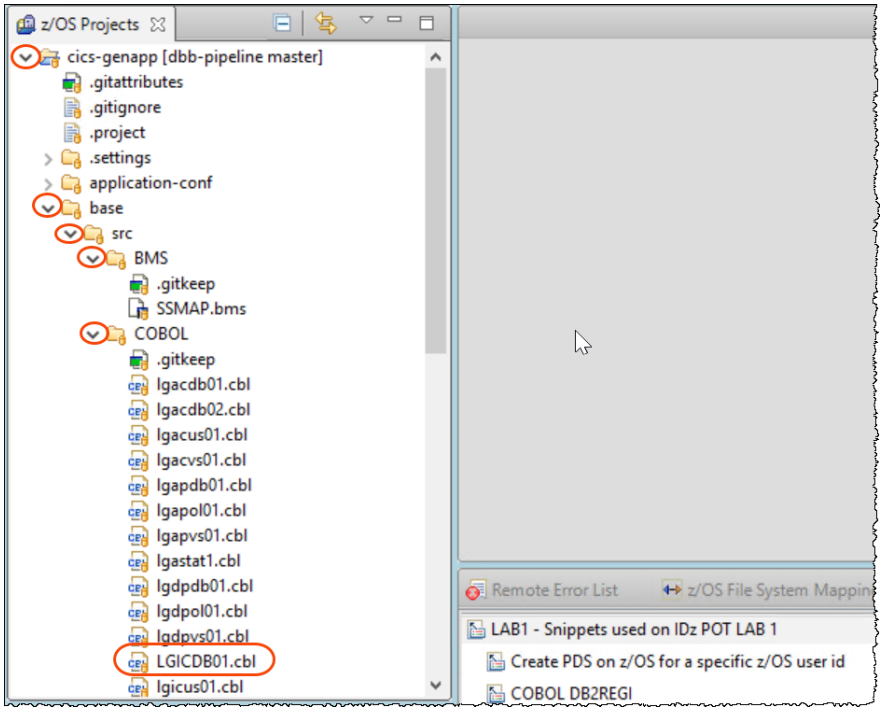
The z/OS projects perspective is the IDz perspective that developers use to work with the source code.

Notice that you have cloned the project at your local workstation but later you will need to connect to z/OS to be able to compile and build your programs.

2.2.1 Switch to the **z/OS Projects** perspective clicking on icon on the top right corner



2.2.2 Expand the project **cics-genapp** clicking on icon and see the source code loaded
Notice that IDz knows that this code is under a repository and the yellow decorator on the icon indicates that.
You will see later that the program that is causing the bug is **LGICDB01.cbl**



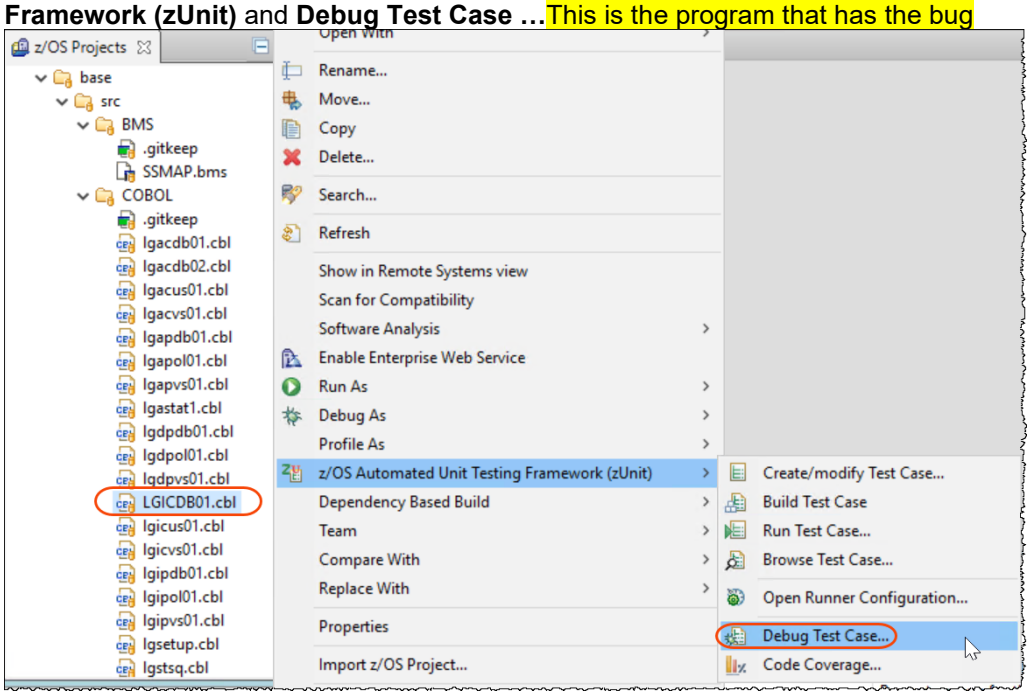
Section 3 – Use IDz to run the zUnit test case and verify the error using the debug.

Running zUnit you will verify that the DOB is incorrect . A zUnit test case was created by other developer when the dialog was correct and the DOB date was displayed correct, For details how to create zUnit test cases you can see the zUnit labs provided.

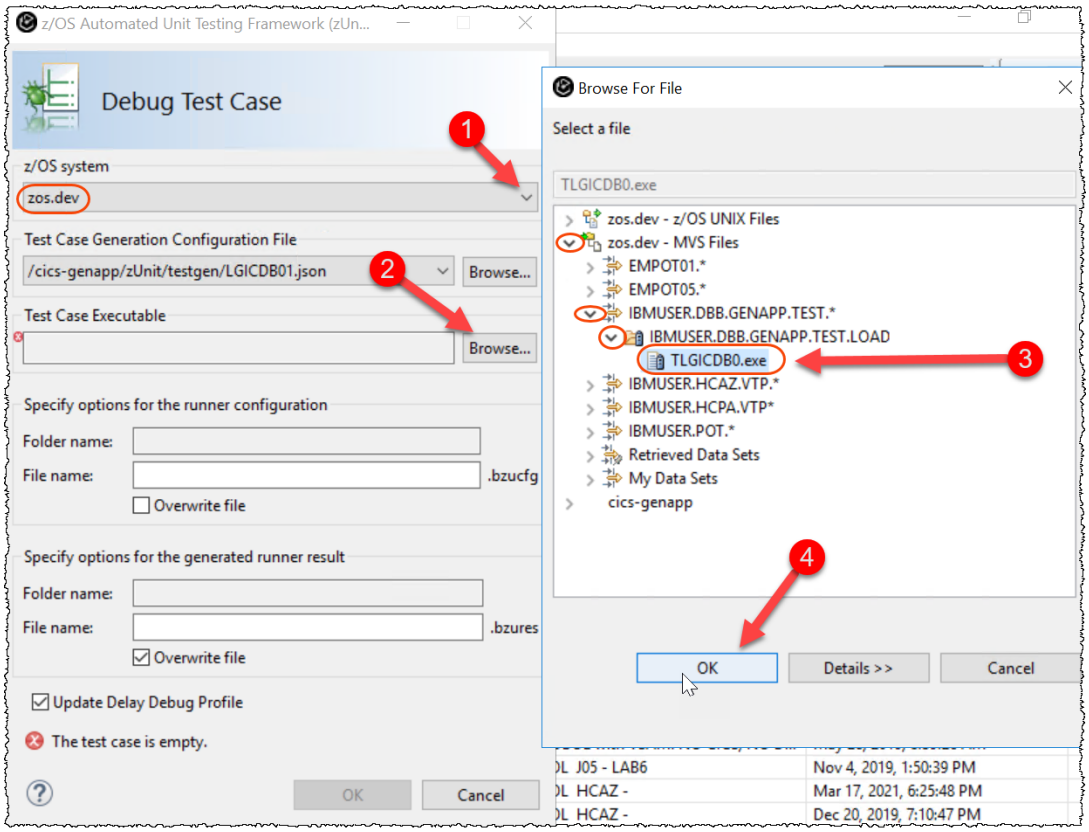
___ 3.1 Running the zUnit in debug mode

A zUnit test case is available, and the developer will run it using the debug option. This way he will be able to see the bug..

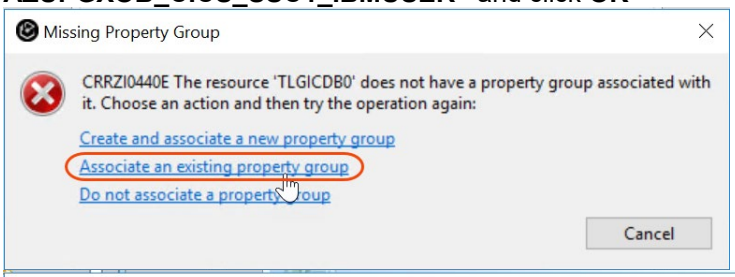
3.1.1 ⏭ Using *z/OS Projects* view right click **LGICDB01.cbl** and select **z/OS Automated Unit Testing Framework (zUnit)** and **Debug Test Case …**This is the program that has the bug

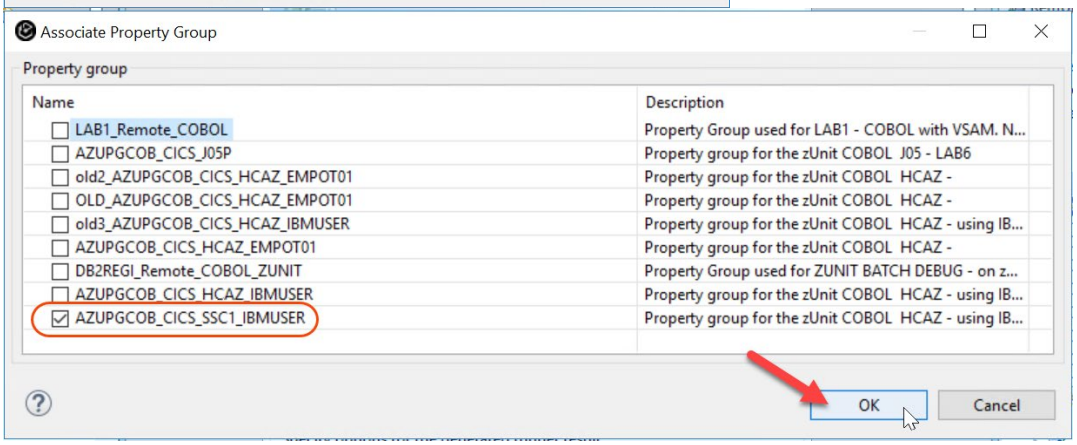


3.1.2 ⏭ Select **zos.dev** for *z/OS system*, for *Test Case Executable* use the **Browse…** button and choose **`IBMUSER.DBB.GENAPP.TEST.LOAD(TLGICDB0.exe)`** and Click **OK**.

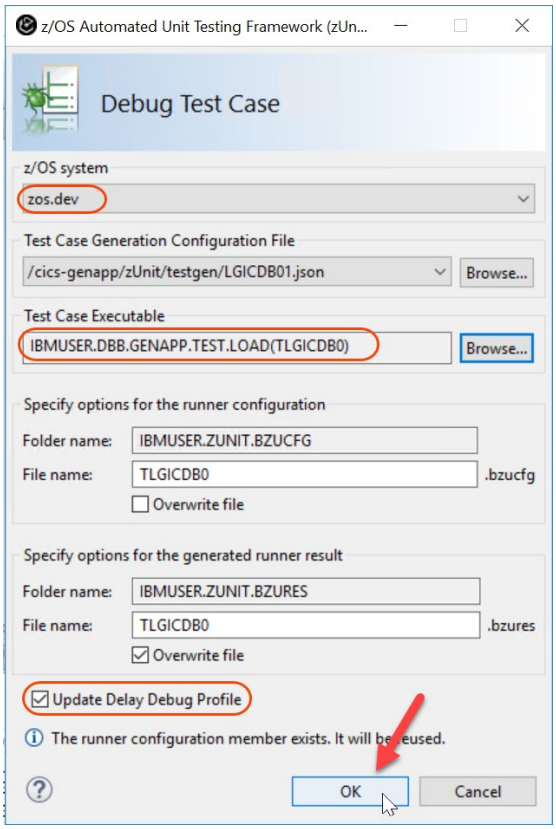


3.1.3 ⏭ If the dialog below appears select **Associate an existing property group** point to **AZUPGXOB_CICS_SSC1_IBMUSER** and click **OK**



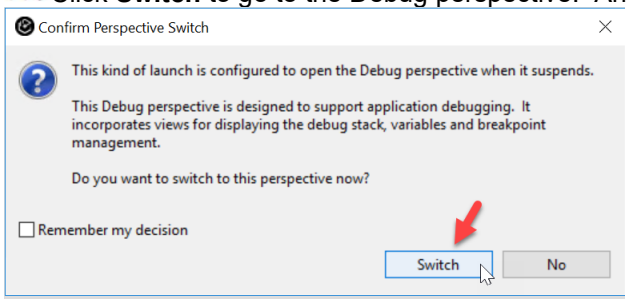


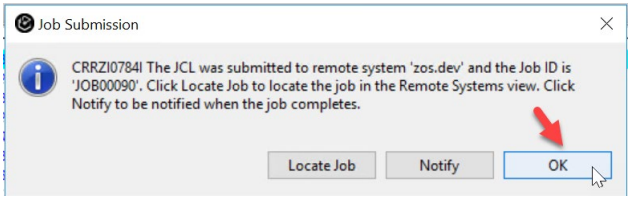
3.1.3 ⏭ Be sure that your dialog values below match the screen below and click **OK.**
Notice that the "*Update Delay Debug Profile"* will cause the debug to start at the program that we are interested in debug (LGICBDB01) , not the other modules.



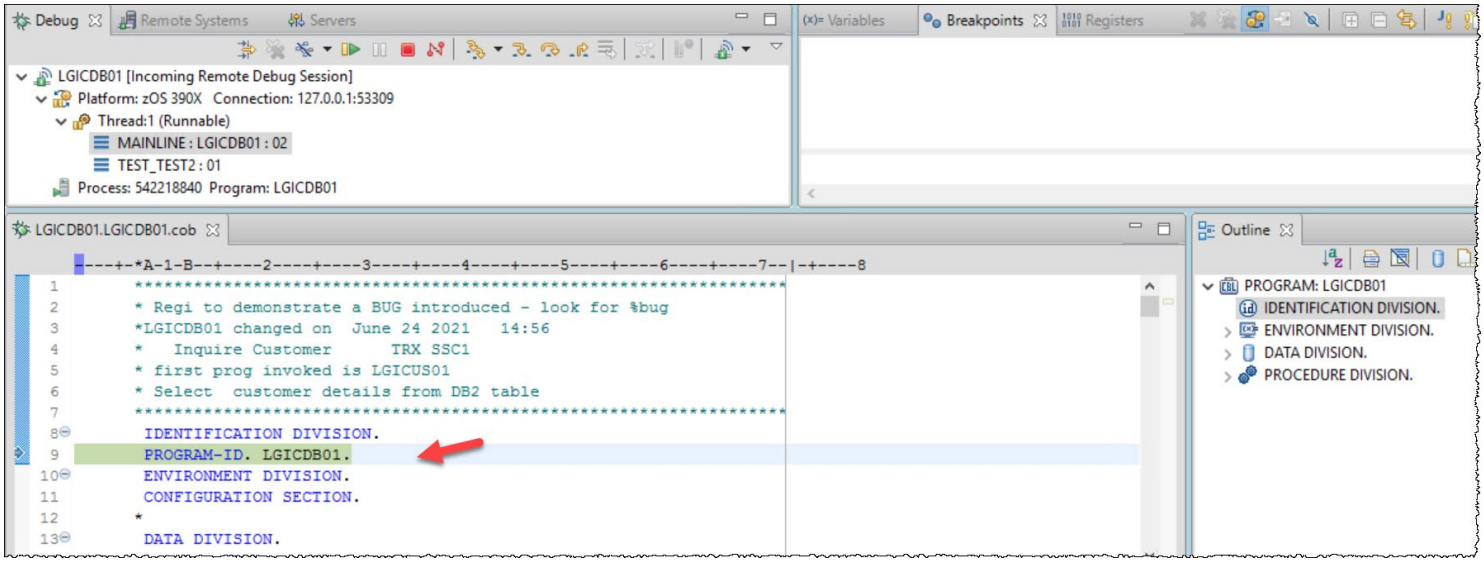
3.1.4 This dialog will send a batch jon and the debug will start.

⏭ Click **Switch** to go to the Debug perspective. And **OK** for the Job Submission dialog



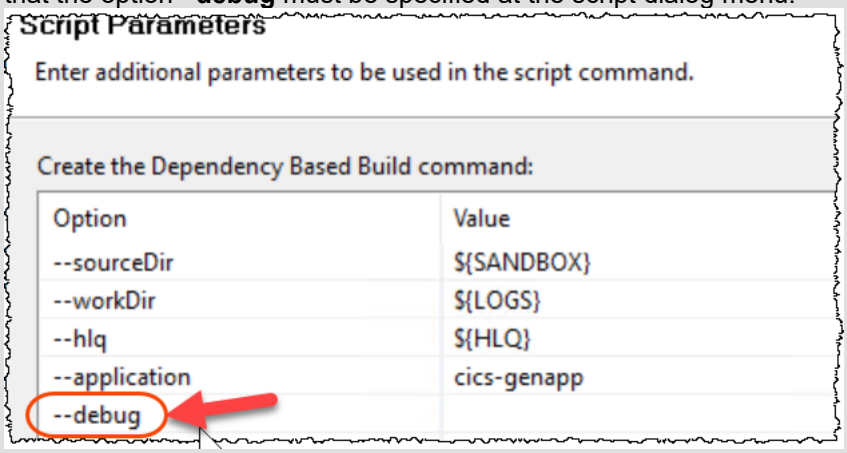


3.1.5 The Debug will start at the program **LGIDDB01**



The Debug shows assembly code?

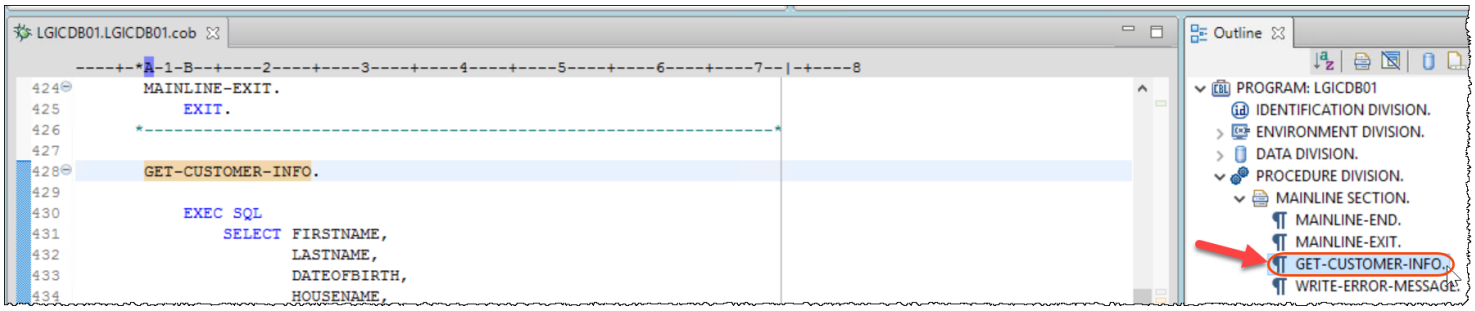
This would happen if the program **LGICDB01** was not compiled with the **TEST** option. To fix that. Use DBB User build as explained on the step 5.1 to rebuild the program. Notice that the option **–debug** must be specified at the script dialog menu.



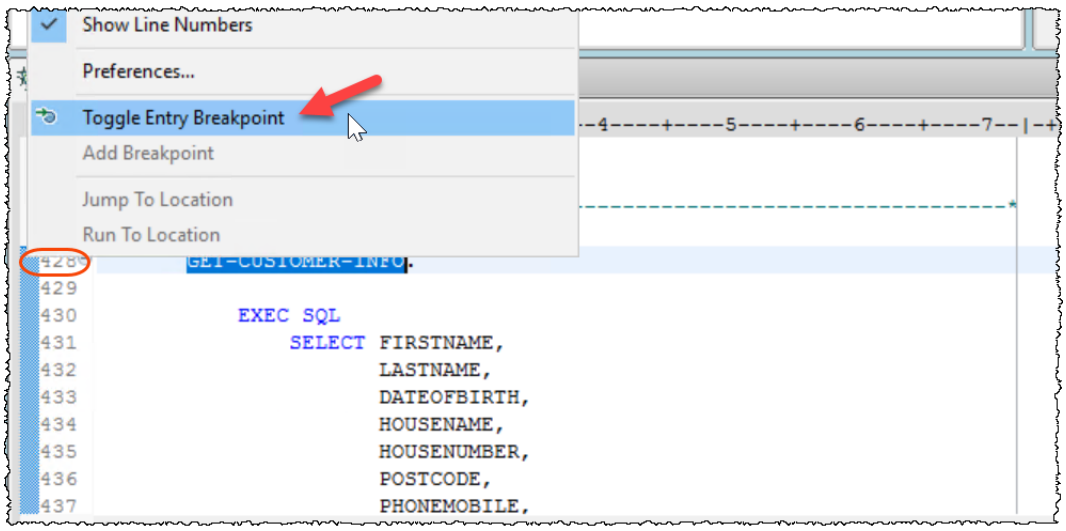
___ 3.2 Adding a Breakpoint and verify the bug

Since the DOB is retrieved from a DB2 table, one idea would be making a breakpoint at the statement that reads the DB2 table value.

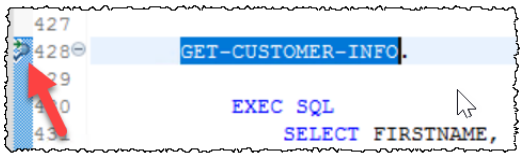
3.2.1 ⏭ Using the Debug *Outline* view, expand the **PROCEDURE DIVISION** and click on **GET-CUSTOMER-INFO.** This where the DB2 table data is selected.



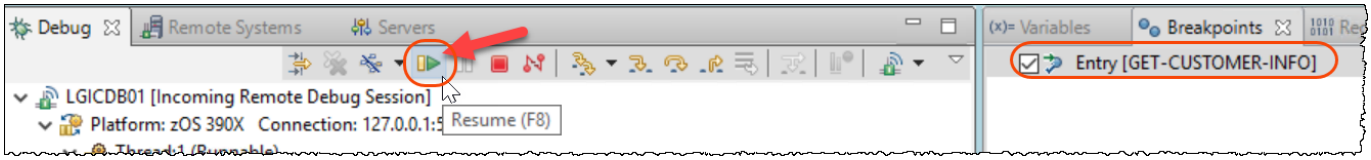
3.2.2 ⏭ Using the *LGICDB91.LGICDB01.cbl* view right click on line **428** and select **Toggle Entry Breakpoint**



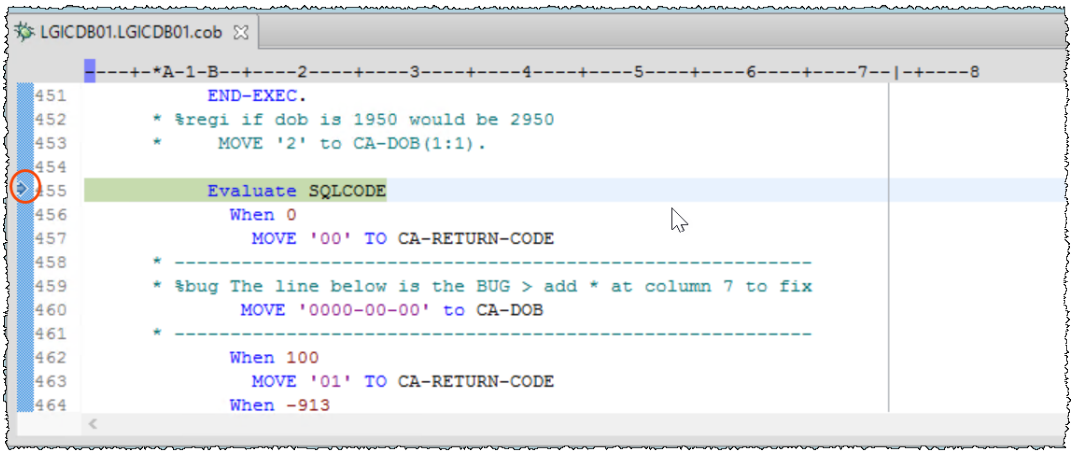
3.2.3 You will notice a small mark that identify the breakpoint



3.2.4 ⏭ Click ▶ or press **F8** to resume the execution. The execution will stop at the breakpoint.

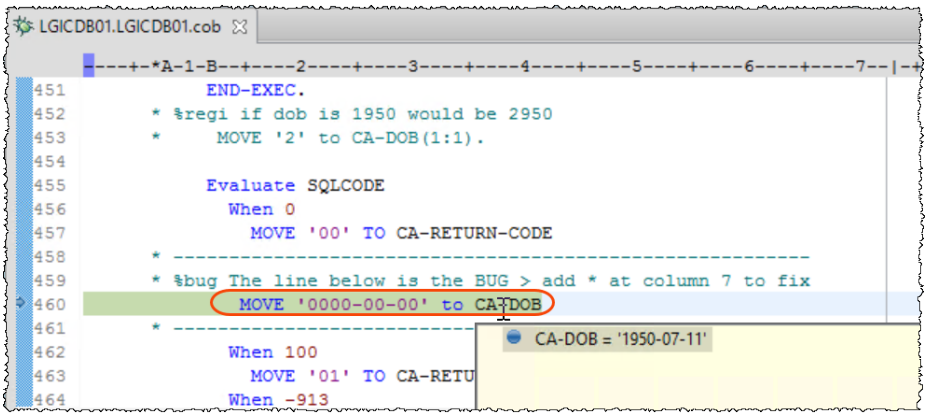


3.2.5 Keep clicking or pressing **F5** (step Into) until you see the statement below.

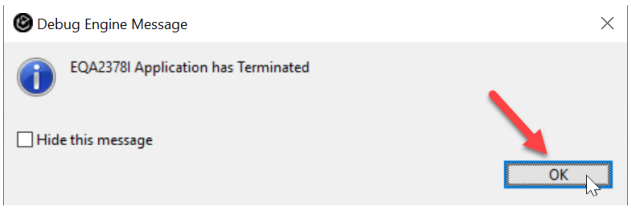


B

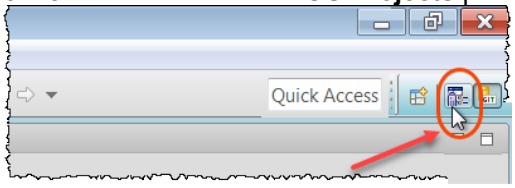
3.2.6 As you saw the data is not coming from DB2 tables but from playback files recorded during the test cases.
Keep clicking or pressing **F5** until you see the statement **460** below.
This is the bug. So the fix would be commenting this line and we will do on next section
This statement moves 0000-00-00 to a field that has a valid date (1950-07-11)



3.2.7 Click or press **F8** few times to terminate the execution and **OK** to terminate the dialog



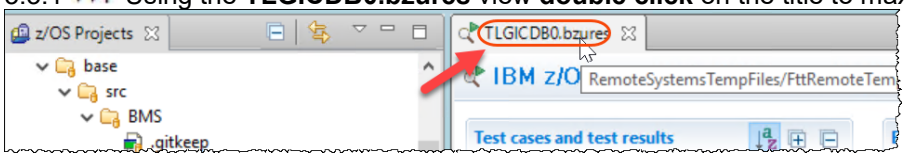
3.2.8 Switch to the **z/OS Projects** perspective clicking on icon on the top right corner



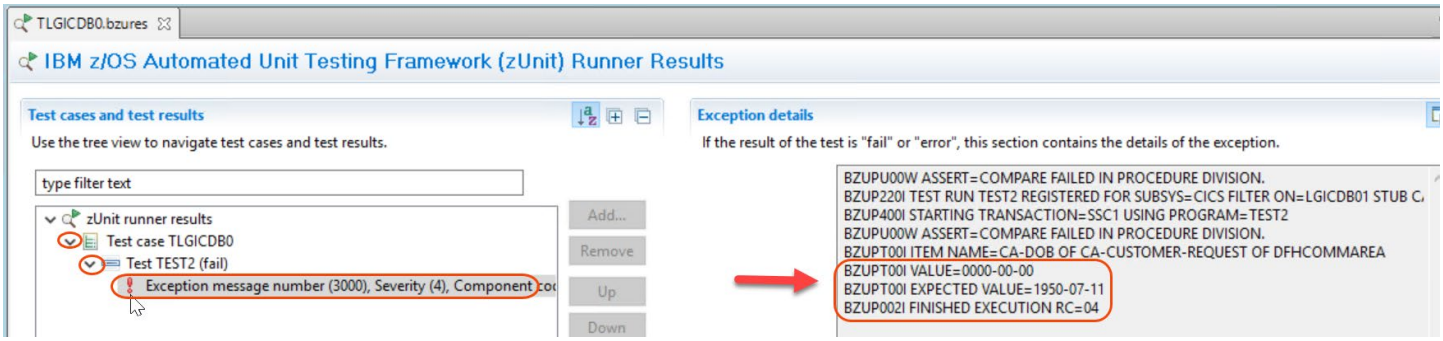
___ 3.3 Verifying the zUnit Results

This Batch job can be seeing at the JES but also a dialog is displayed with the results.

3.3.1 ⏭ Using the **TLGICDB0.bzures** view **double click** on the title to maximize it



3.3.2. ⏭ **Expand the test case** and verify that the expected value is **1950-07-11** but the current value is **0000-00-00.**



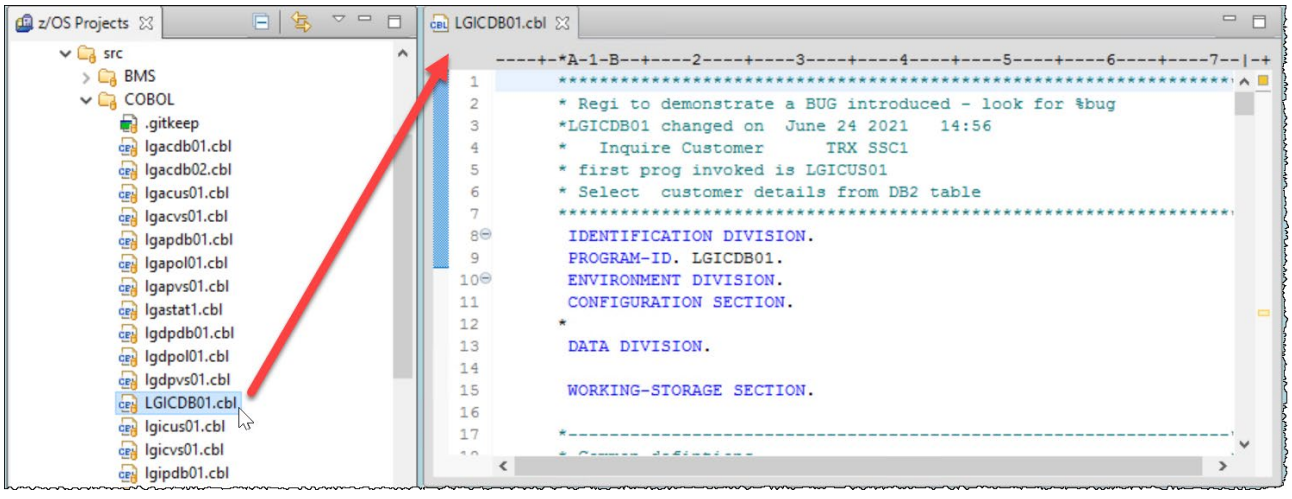
3.3.3. ⏭ Use **Ctrl + Shift + F4** to close all opened editors.

Section 4.Modify the COBOL/CICS/DB2 program that has the bug using IDz..

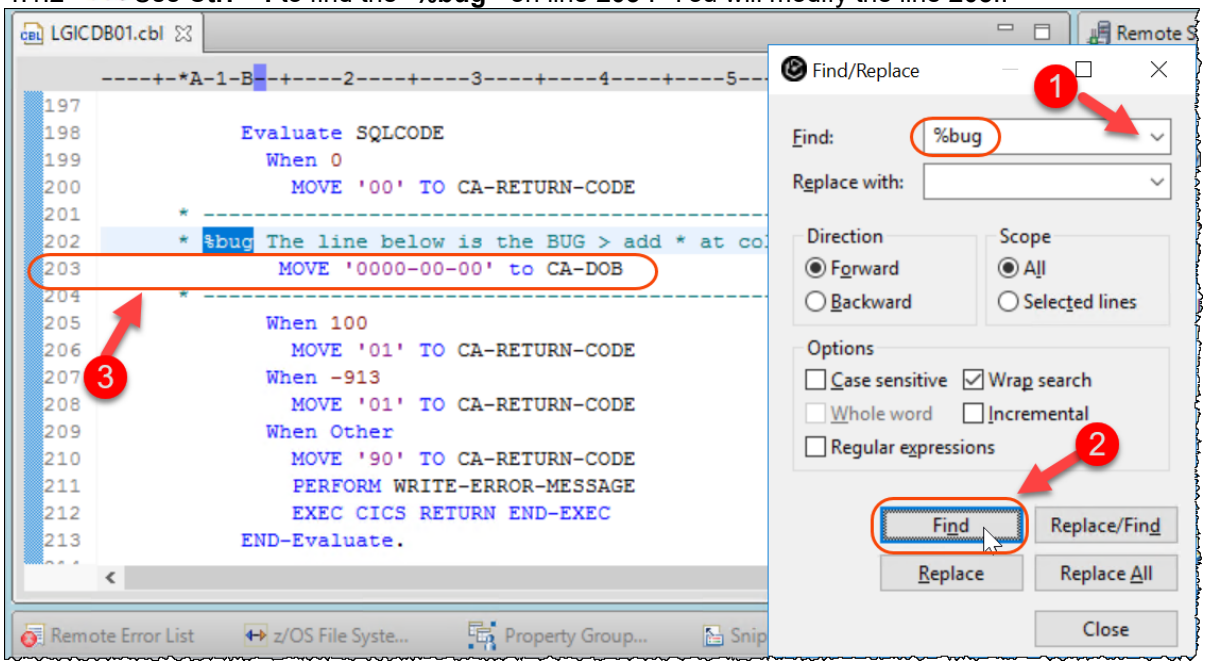
Using IDz you will fix the bug modifying the COBOL program that has the defect..

___ 4.1 Edit and modify the code that send the message

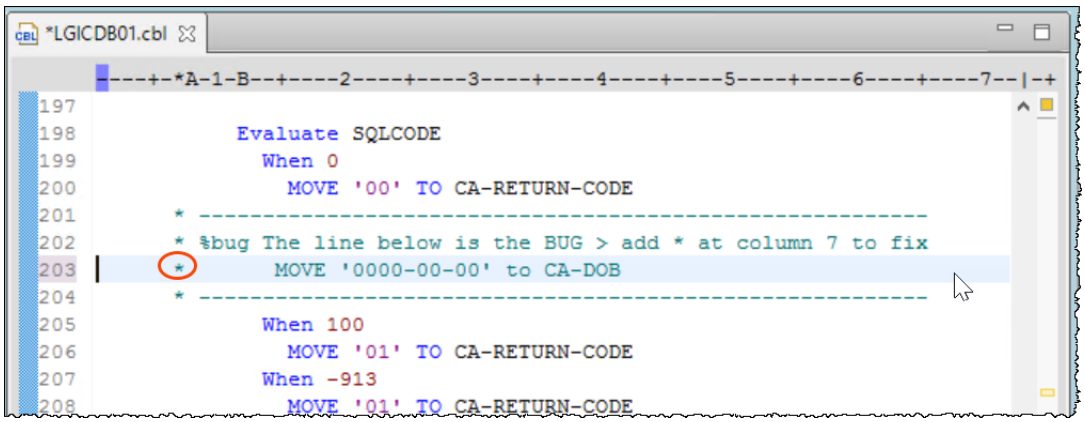
4.1.1 ⏭ Using *z/OS Projects* view double click **LGICDB01.cbl** under *COBOL* folder. This is the program that you will update. To make easier to see we made it UPPER case



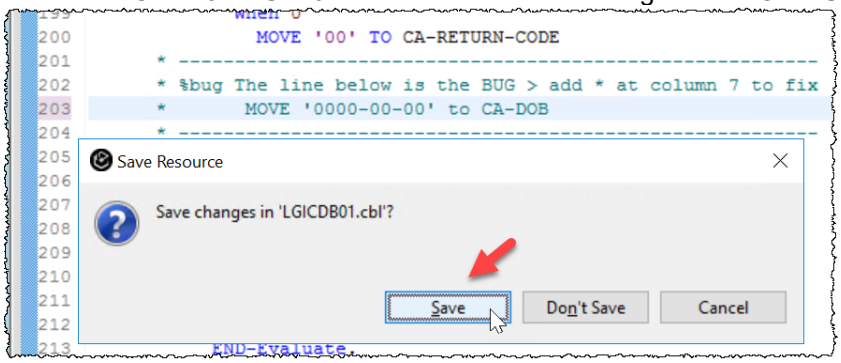
4.1.2 ⏭ Use **Ctrl + f** to find the "**%bug**" on line **203** . You will modify the line **203**..

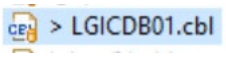


3.1.3 ⏭ **Close** the find dialog and modify the line **203** adding an * on **column 7**
Tip → Could right click on this line and select **Source > Toggle Comment**



4.1.4 ⏭ Use **Ctrl + Shift + F4** to close all code being edited. Click **Save** to save the modified code.



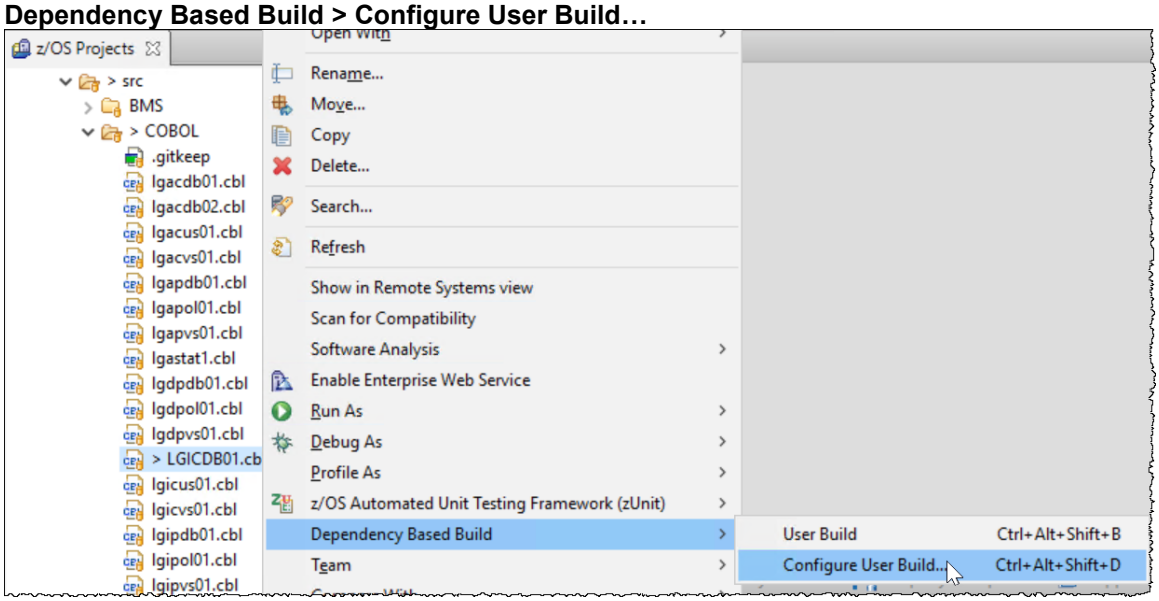
Notice that since Git is managing this code a mark on the program that indicates a change 

Section 5. Use IDz DBB User Build to compile/link and run the zUnit.

You will compile and link the modified code using the DBB User Build Function. When complete you will run zUnit generated test case and verify that the bug is fixed.

___ 5.1 Using Dependency Based Building option

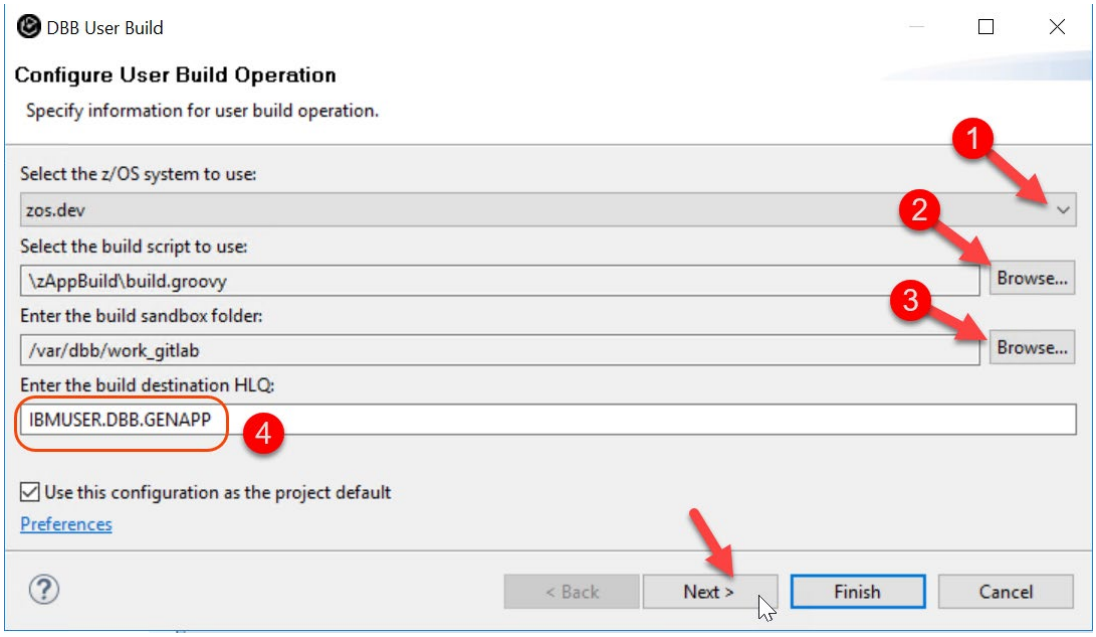
5.1.1 ⏭ Under *z/OS Projects* view right click **LGICDB01.cbl** and select **Dependency Based Build > Configure User Build…**



5.1.2 ⏭ Be sure that those values are assigned for the build.

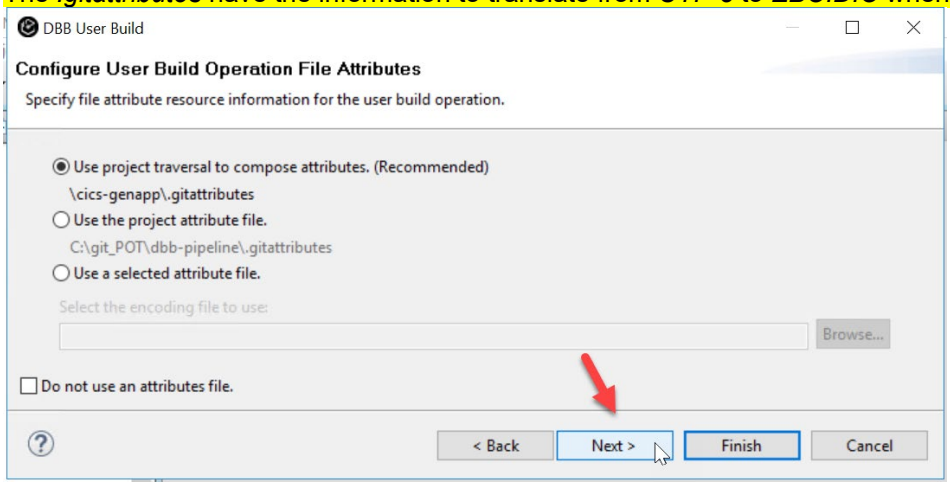
4 On *build destination HLQ*: be sure that you are using **`IBMUSER.DBB.GENAPP`** (must type)

⏭ Click **Next**



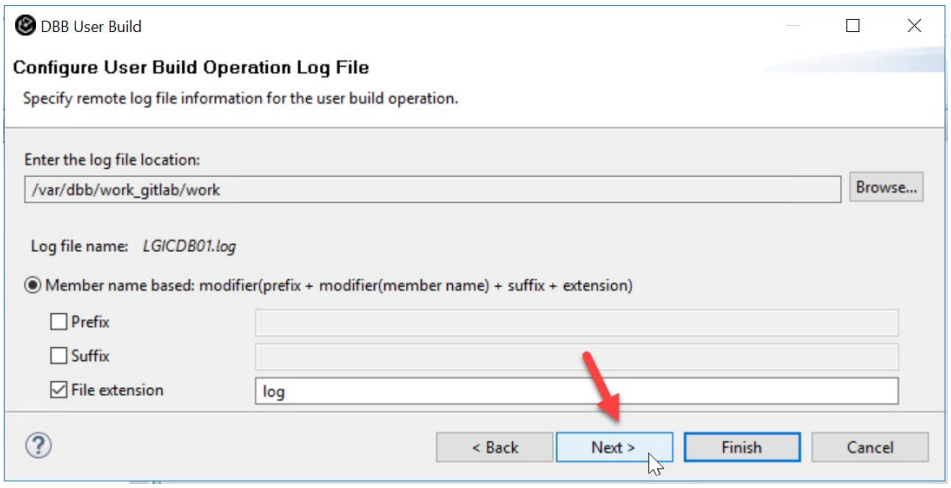
5.1.3 ⏭ On *Configure User Build Operation File Attributes* click **Next**

The *.gitattributes* have the information to translate from *UTF-8* to *EBCIDIC* when moving the code to z/OS.



5.1.4 ⏭ On *Configure User Build Operation Log File* click **Next**

You will use the default values.

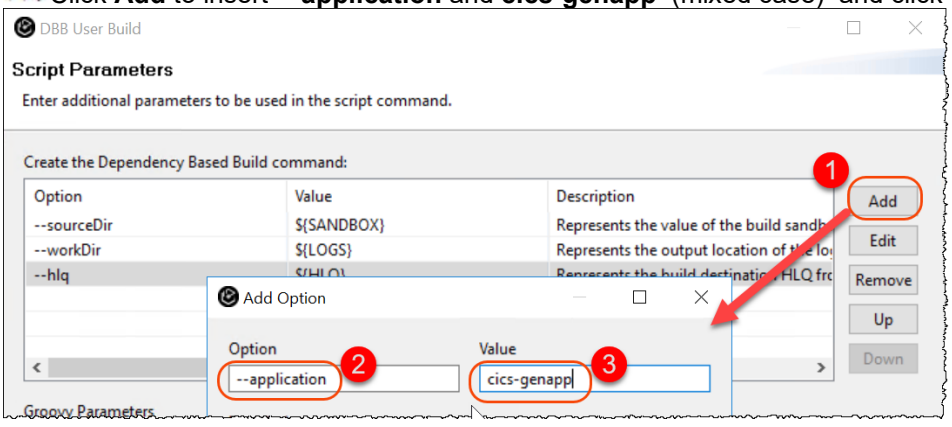


5.1.5 ⏭ On dialog *Script Parameters*, if the "**—application**" and "**--debug**" are not there you must add

⏭ Otherwise skip to 5.1.7

In case you need to add **--application**.

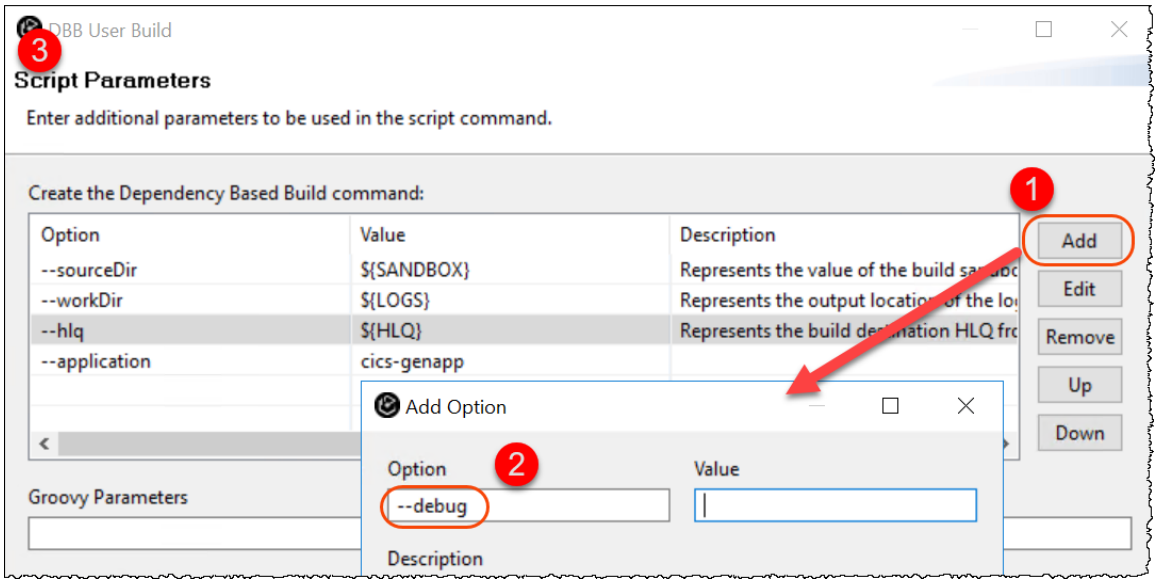
⏭ Click **Add** to insert **--application** and **cics-genapp** (mixed case) and click **OK**



5.1.6 In case you need to add **--debug**.

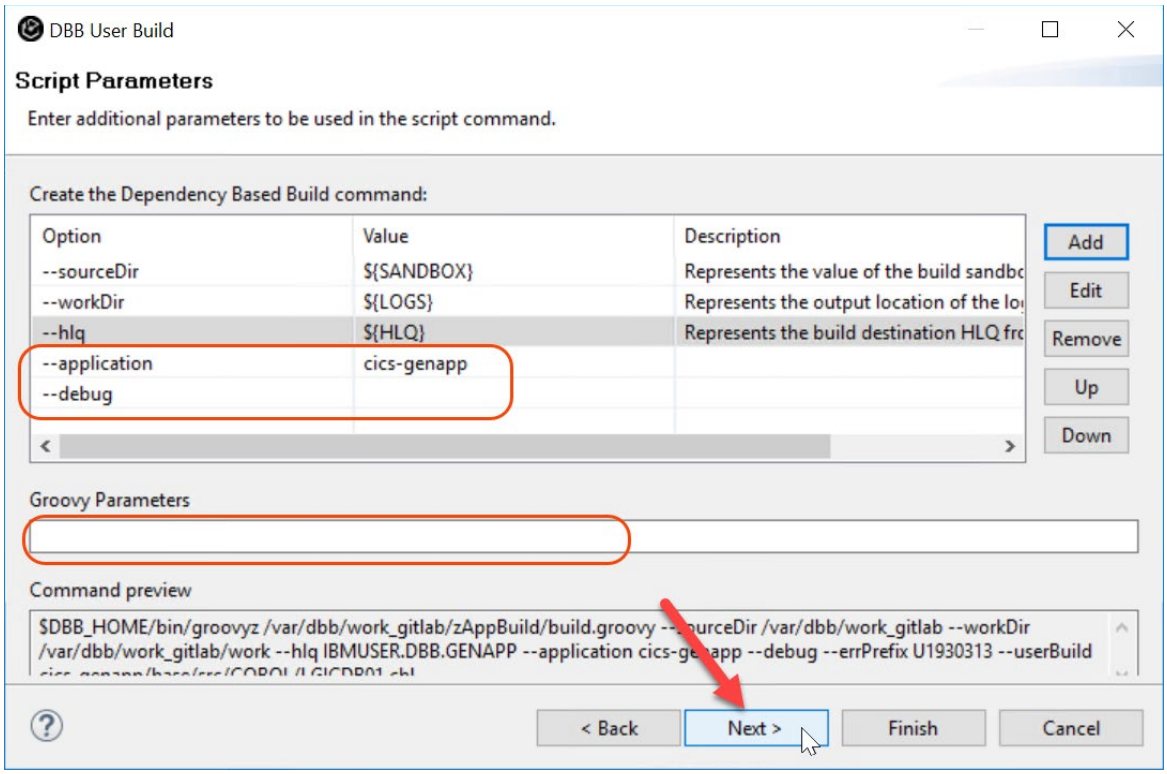
⏭ Click **Add** to insert **--debug** and click **OK**

Verify that the options below were inserted. Notice that there is an "**--**" before each option.



5.1.7 The field **Groovy Parameters** must be empty

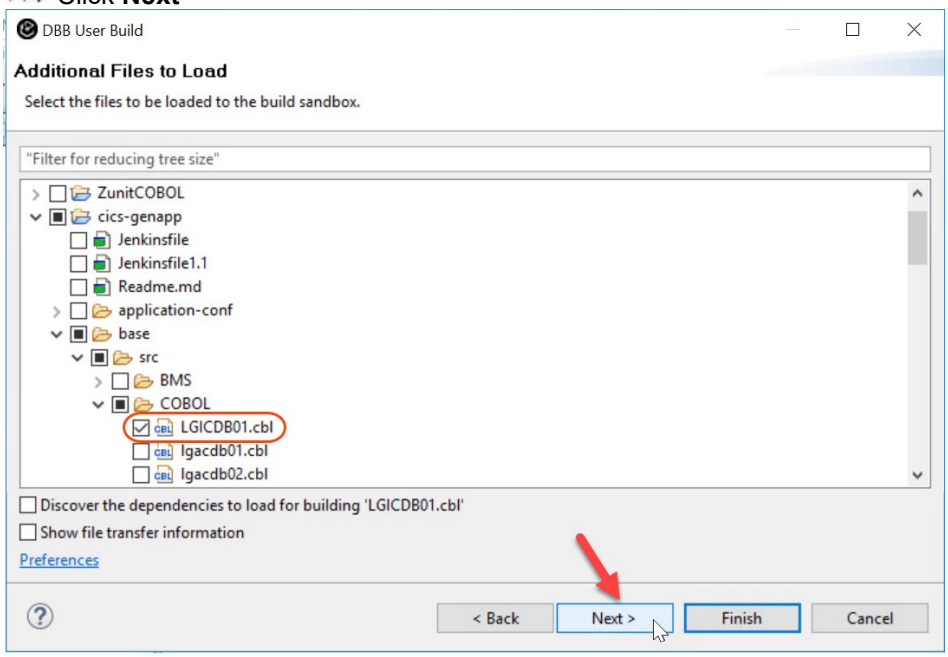
⏭ Click **Next**



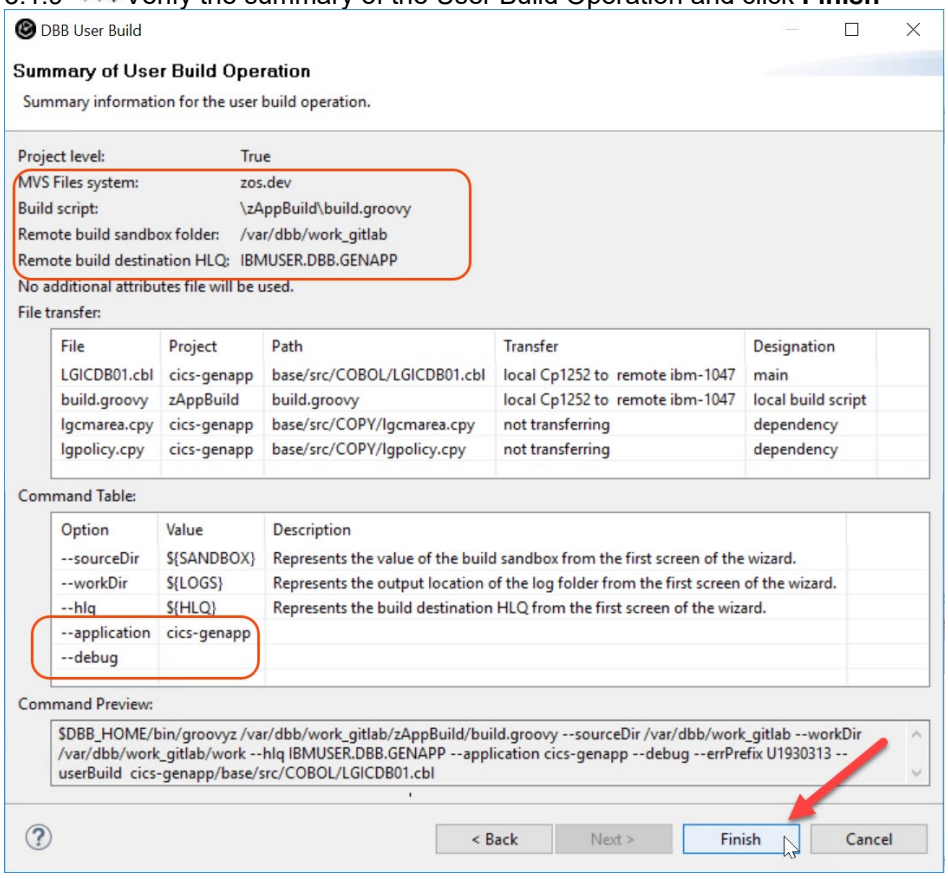
5.1.8 The selected files will be moved from your local workspace to a z/OS USS directory and the execution of the groovy scripts will interact with the DBB framework that will invoke the compiler, linkage editor, etc..

⏭ Expand **cics-genapp** and ***base/src/COBOL*** to verify that the **LGICDB01.cbl** is selected

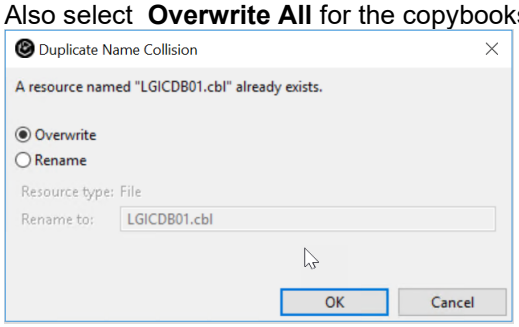
⏭ Click **Next**

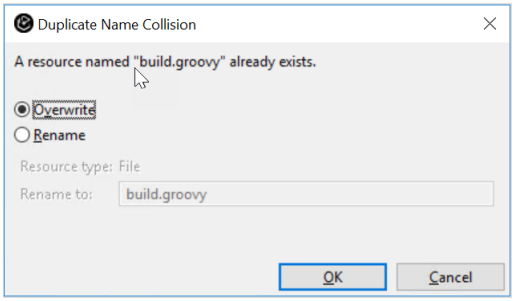


5.1.9 ⏭ Verify the summary of the User Build Operation and click **Finish**



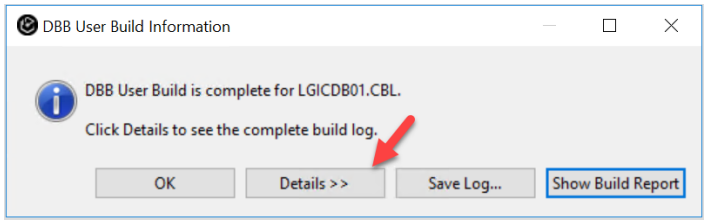
5.1.10 ⏭ If the dialogs below pops up select **Overwrite** and click **OK**
Also select **Overwrite All** for the copybooks





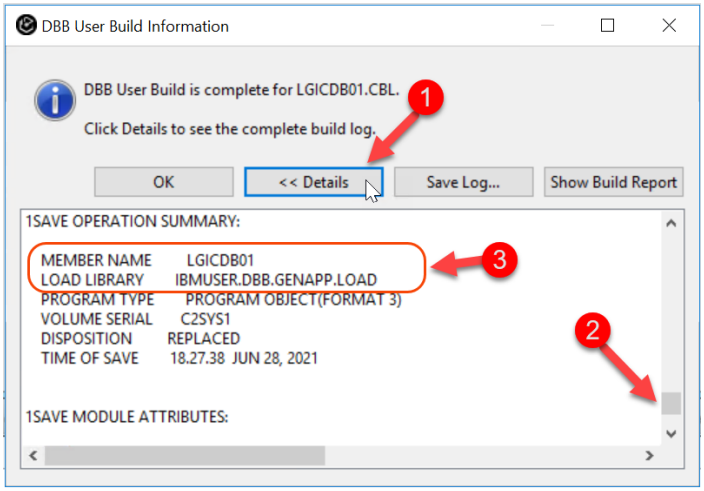
The DBB User build will be running and the messages will be shown at the Console view..
⏭ Click on **Console** tab on lower right corner The execution will start, and this will take a while (2 Minutes) .

5.1.11 When complete you will have the message below:
⏭ Click **Details**

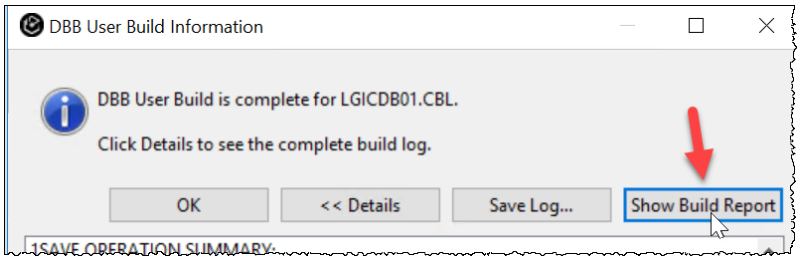


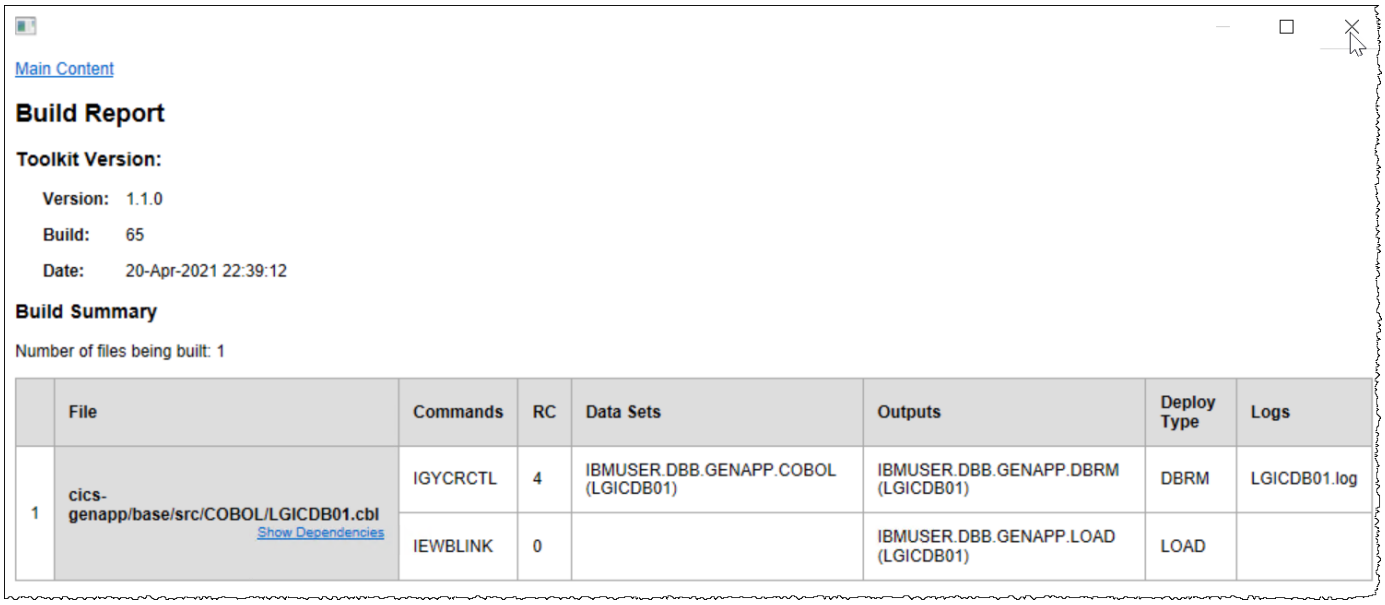
___ 5.2 Verify the DBB User Build results

5.2.1 ⏭ After clicking **Details >>** scroll down and verify that a new load module named **LGICDB01** was replaced on the dataset ***IBMUSER.DBB.GENAPP.LOAD***
⏭ Click **OK** to close this dialog.



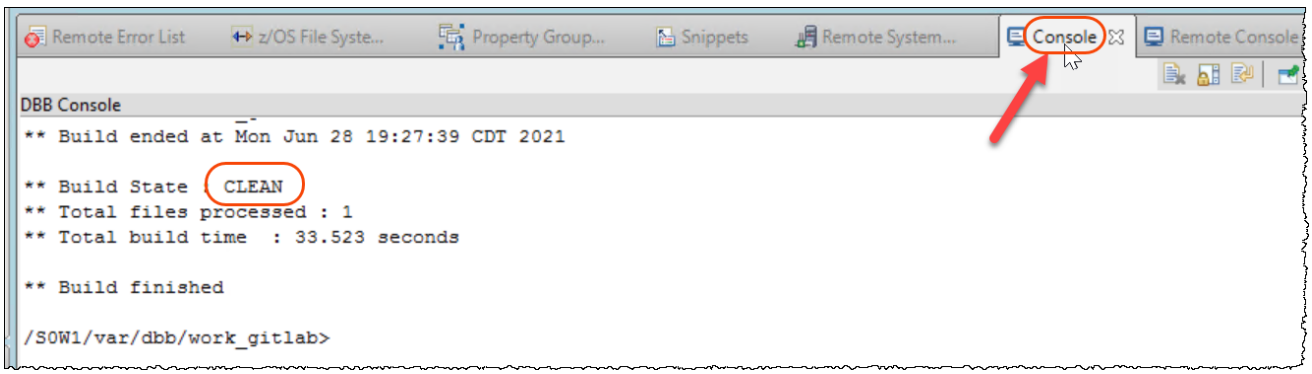
5.2.2 Click **Show Build Report** to see the report created:





This report will also be stored at the DBB Application Server

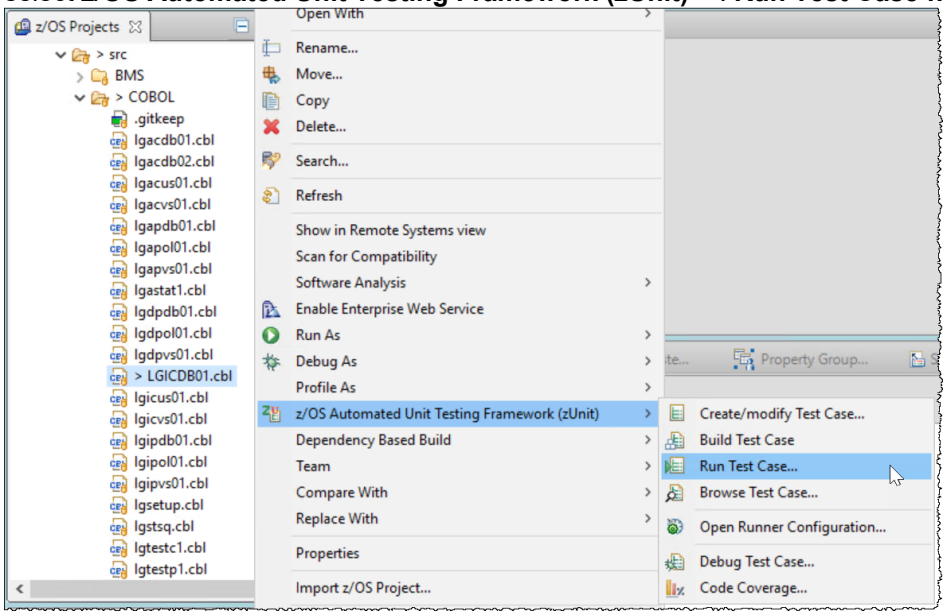
5.2.3 Close the dialogs and Verify at the Console view that the Build State should show a *CLEAN* build.



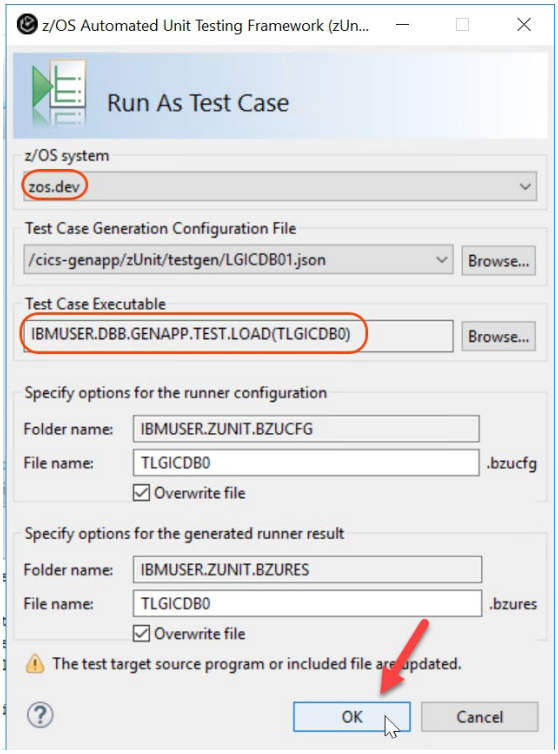
___ 5.3 Running the zUnit again

You should now run the zUnit test case again to verify that the bug has been fixed.

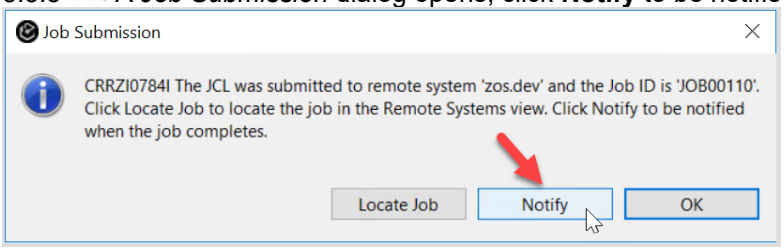
5.3.1 ⏭ Right click on **LGICDB01.cbl** and
select **z/OS Automated Unit Testing Framework (zUnit)** > . **Run Test Case ….**



5.3.2 ⏭ Be sure that the values below are in effect and click **OK**

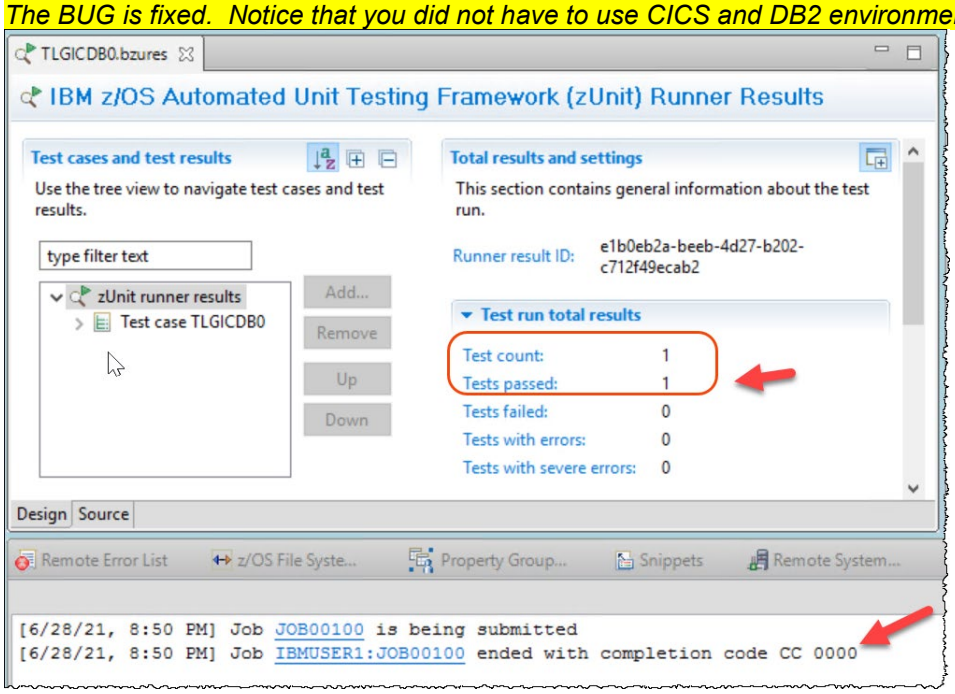


5.3.3 A *Job Submission* dialog opens, click **Notify** to be notified of job completion.



5.3.4 Once the unit test run has completed, the results screen is displayed showing test cases ran and passed.

The BUG is fixed. Notice that you did not have to use CICS and DB2 environment to correct the bug.



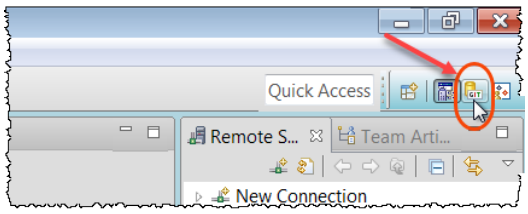
5.3.5 Use **Ctrl + Shift + F4** to close all opened editors.

Section 6. Push and Commit the changed code to GitLab.

Using IDz you will commit the changes you made to GitLab This will initiate the GitLab CI pipeline This will initiate the GitLab CI pipeline

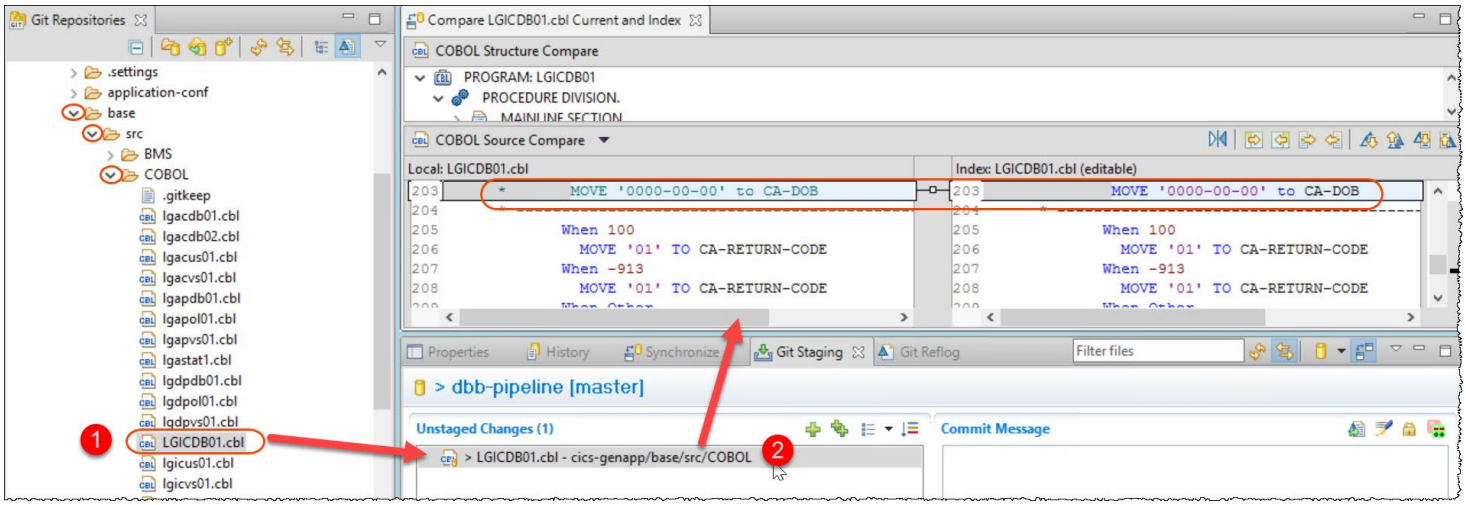
___ 6.1 Using IDz and Git perspective to compare the change versus original

6.1.1 Switch to the **Git Perspective** selecting the icon on top and right



6.1.2 ⏭ 1 Expand the **base/src/COBOL** folders and click on the program **LGICDB01.cbl** that you have modified.
Notice that the changed program is listed in the *Git Staging* view..
⏭ 2 Double click on the **LGICDB01.cbl** that is listed under ***Unstaged Changes*** and noticed the comparison among the program that you updated versus the one that is in the Git repository.

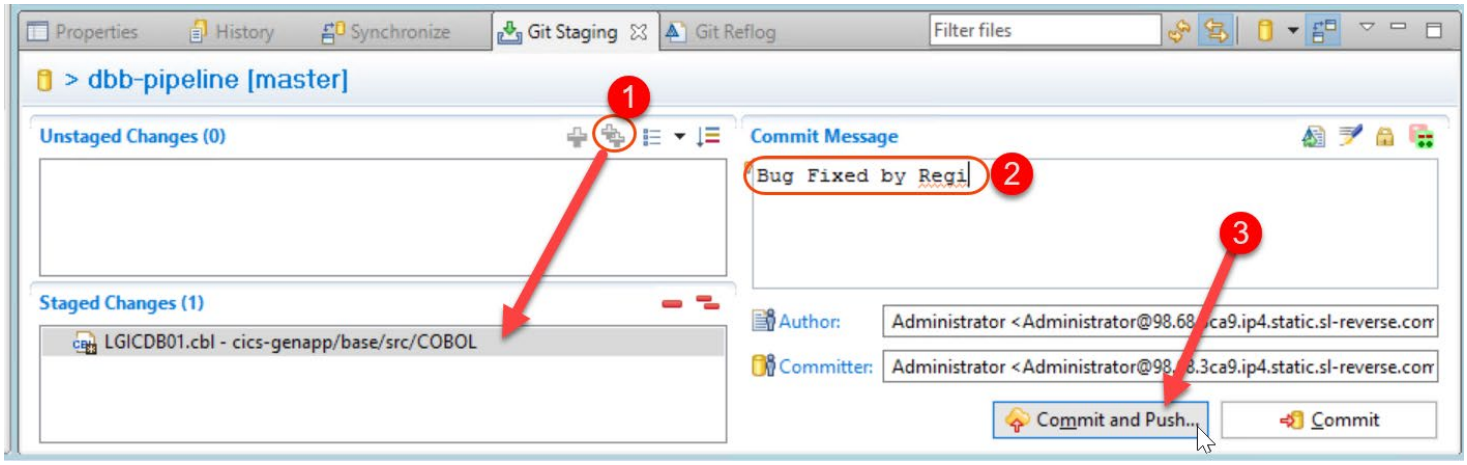
You will need to commit the changes to the Git repository to make a final build later.



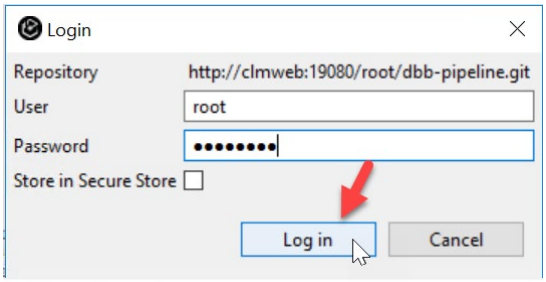
6.1.3 ⏭ Use **Ctrl + Shift + F4** to close the editors.

___ 6.2 Push and Commit the changes to GitLab

.
6.2.1 ⏭ Using **Git Staging** view, ,
1 Click on the icon ,
2 Write a commit message like: **Bug Fixed by *your name***
3 Click **Commit and Push …**

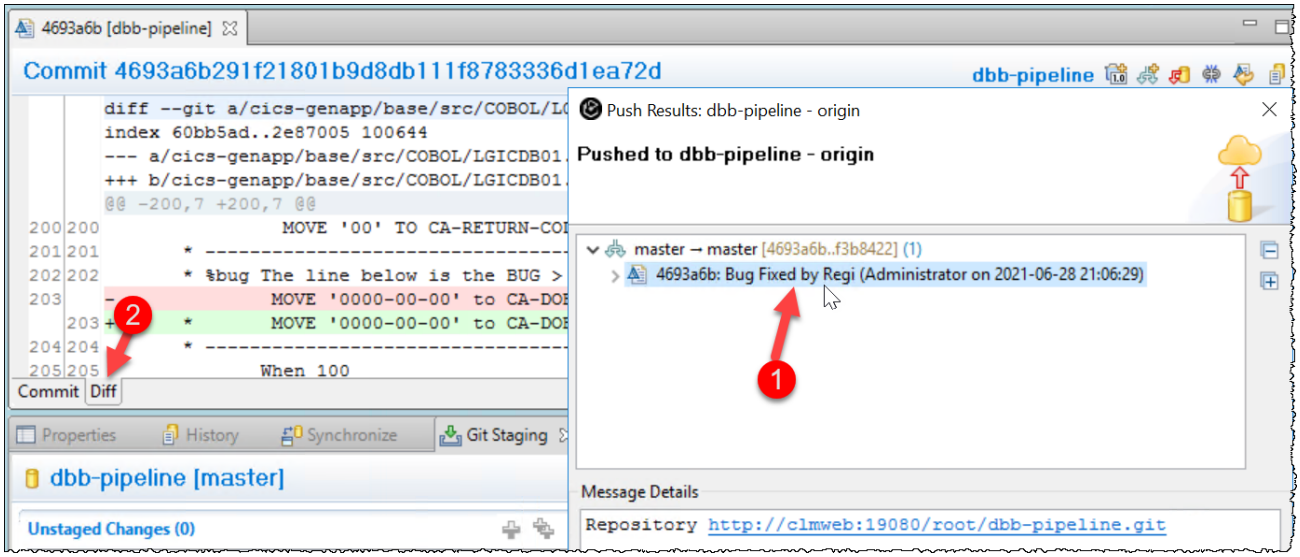


6.2.2 ⏭ For login credentials use **root** and password **zdtlinux** and click **Log in**



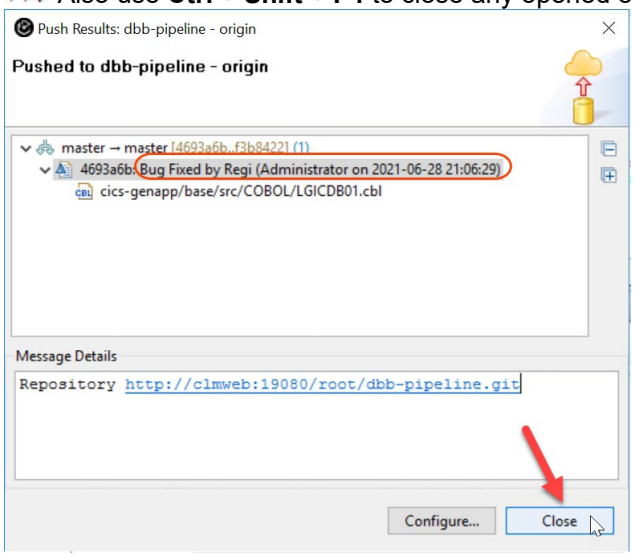
6.2.3 To see the changes you committed to Git:

⏭ 1 Double click on **Git hash string** and 2 select **Diff** tab

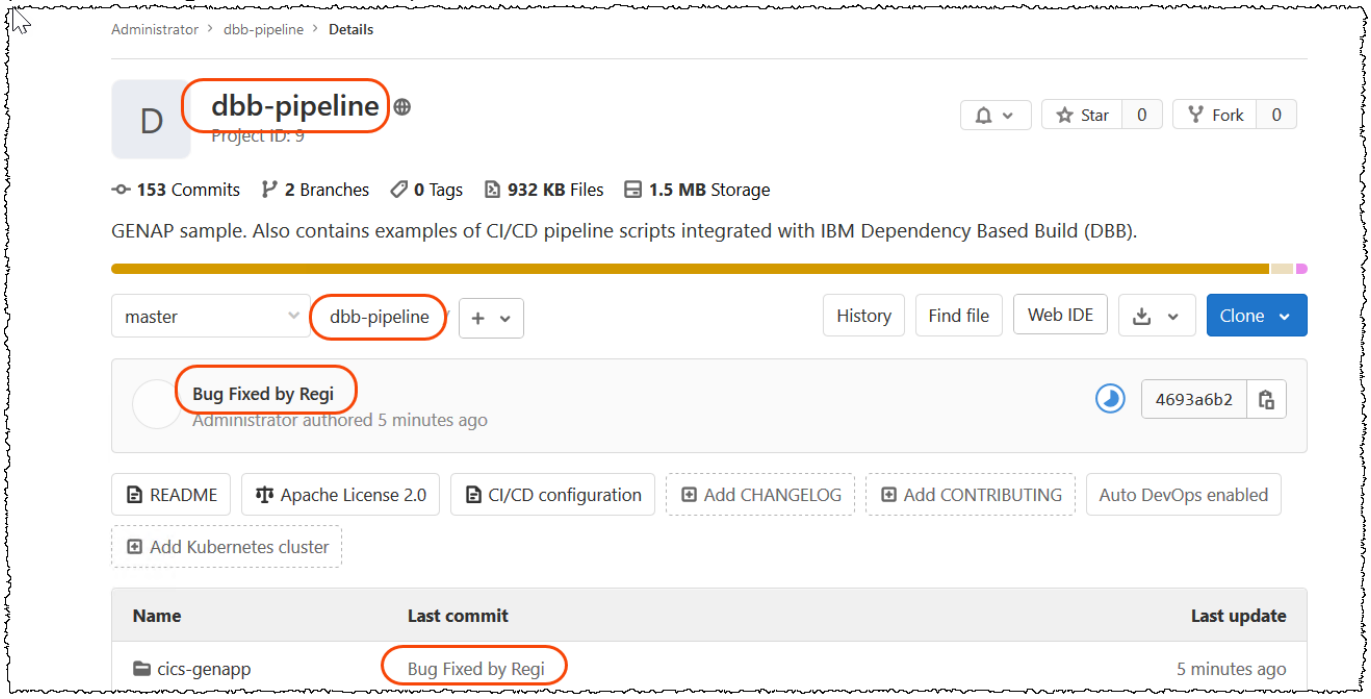


6.2.4 ⏭ Verify that the commit was successful and click **Close**

⏭ Also use **Ctrl + Shift + F4** to close any opened editor



6.2.5 To see this changes in *GitLab* use the **Firefox browser** and do a **Refresh** (**F5**) (details how to get the link at 2.1.1).

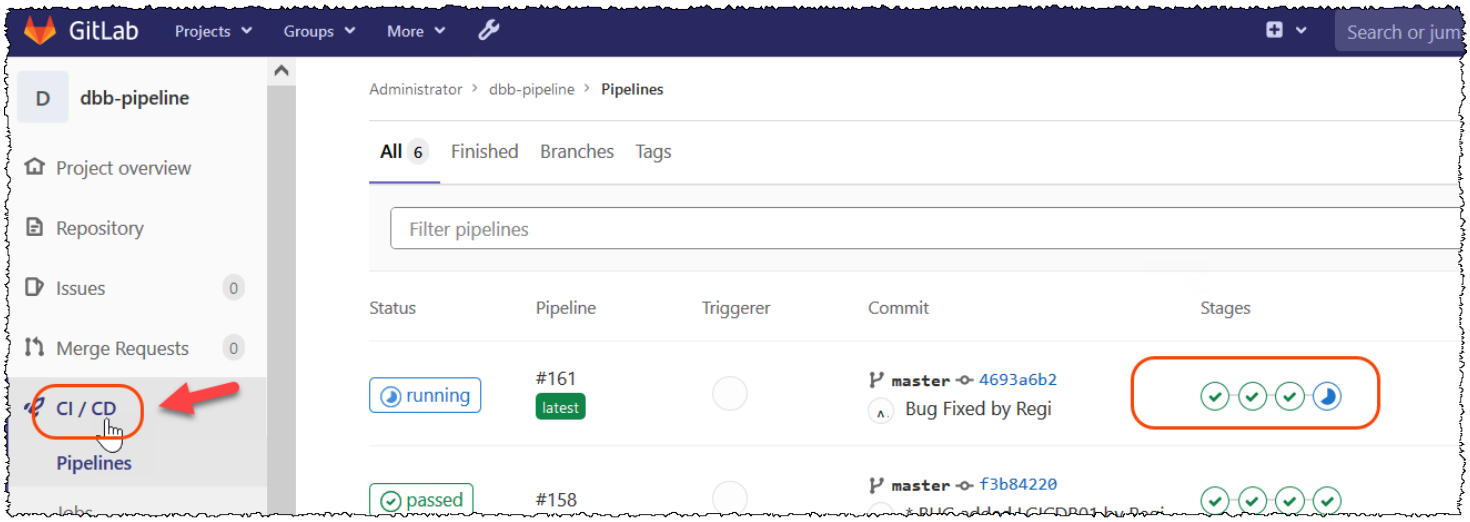


Section 7. Verify the GitLab CI Build execution..

At this point the changed code is delivered into the GitLab repository that is on Linux. You will use GitLab CI pipeline in action

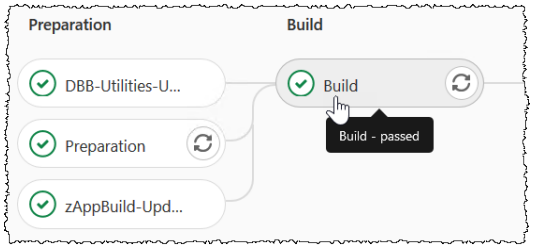
___ 7.1 Verifying the Build

7.1.1 Click on **CI / CD** on left to verify the Pipeline in action



7.1.2 ⏭ Click on **Running** to see the results

7.1.3 ⏭ Click on **Build**

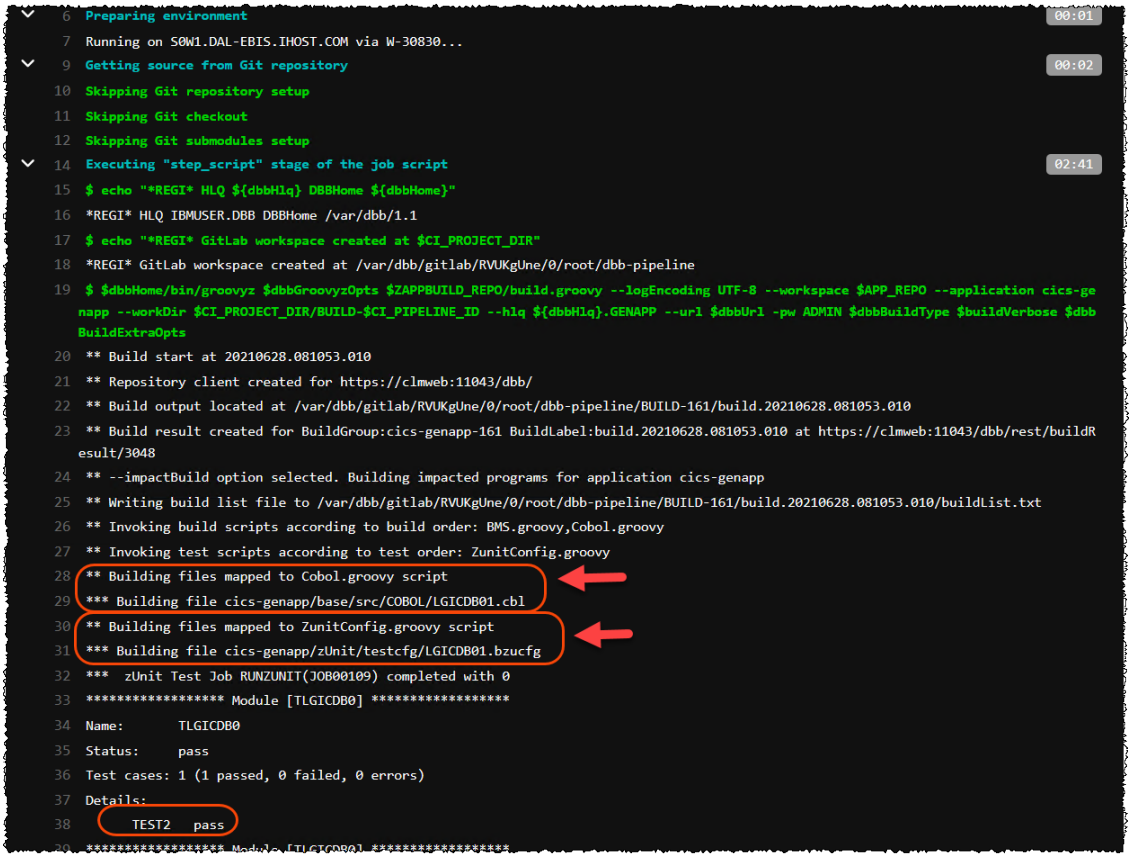


The connection to z/OS fails with the message below?
There has been a runner system failure,

```
Running with gitlab-runner 13.12.0 (7a6612da)
  on zos.dev runner RVUKgUne
Preparing the "ssh" executor
Using SSH executor...
ERROR: Preparation failed: ssh command Connect() error: ssh Dial() error: dial tcp 10.1.1.2:1022: connect: connection refused
```

Ask the instructor .. He needs to go to SDSF log and issue /**S SSHD**

7.1.4 Verify that the program **LGICDB01.cbl** is built and also a zUnit is executed since a zUnit test case for **LGICDB01** does exist



```
Preparing environment                                                                                                    00:01
Running on S0W1.DAL-EBIS.IHOST.COM via W-30830...
Getting source from Git repository                                                                                       00:02
Skipping Git repository setup
Skipping Git checkout
Skipping Git submodules setup
Executing "step_script" stage of the job script                                                                          02:41
$ echo "*REGI* HLQ ${dbbHlq} DBBHome ${dbbHome}"
*REGI* HLQ IBMUSER.DBB DBBHome /var/dbb/1.1
$ echo "*REGI* GitLab workspace created at $CI_PROJECT_DIR"
*REGI* GitLab workspace created at /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline
$ $dbbHome/bin/groovyz $dbbGroovyzOpts $ZAPPBUILD_REPO/build.groovy --logEncoding UTF-8 --workspace $APP_REPO --application cics-genapp --workDir $CI_PROJECT_DIR/BUILD-$CI_PIPELINE_ID --hlq ${dbbHlq}.GENAPP --url $dbbUrl -pw ADMIN $dbbBuildType $buildVerbose $dbbBuildExtraOpts
** Build start at 20210628.081053.010
** Repository client created for https://clmweb:11043/dbb/
** Build output located at /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010
** Build result created for BuildGroup:cics-genapp-161 BuildLabel:build.20210628.081053.010 at https://clmweb:11043/dbb/rest/buildResult/3048
** --impactBuild option selected. Building impacted programs for application cics-genapp
** Writing build list file to /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010/buildList.txt
** Invoking build scripts according to build order: BMS.groovy,Cobol.groovy
** Invoking test scripts according to test order: ZunitConfig.groovy
** Building files mapped to Cobol.groovy script
*** Building file cics-genapp/base/src/COBOL/LGICDB01.cbl
** Building files mapped to ZunitConfig.groovy script
*** Building file cics-genapp/zUnit/testcfg/LGICDB01.bzucfg
***  zUnit Test Job RUNZUNIT(JOB00109) completed with 0
****************** Module [TLGICDB0] ******************
Name:       TLGICDB0
Status:     pass
Test cases: 1 (1 passed, 0 failed, 0 errors)
Details:
    TEST2    pass
****************** Module [TLGICDB0] ******************
```

7.1.5 Verify that the build was successful

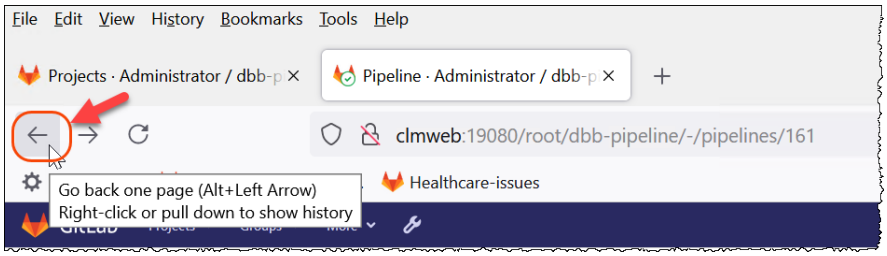
```
****************** Module [TLGICDB0] ******************
** Writing build report data to /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010/BuildReport.json
** Writing build report to /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010/BuildReport.html
** Updating build result BuildGroup:cics-genapp-161 BuildLabel:build.20210628.081053.010 at https://clmweb:11043/dbb/rest/buildResult/3048
** Build ended at Mon Jun 28 20:12:28 CDT 2021
** Build State : CLEAN
** Total files processed : 2
** Total build time   : 1 minutes, 34.599 seconds
** Build finished
$ cd $CI_PROJECT_DIR/BUILD-$CI_PIPELINE_ID/build*
$ for f in `find . -name "*.zunit.report.log"`; do echo "Process zUnit $f"; Xalan -o $f.xml $f /tmp/AZUZ2J30.xsl; done;
Process zUnit ./LGICDB01.zunit.report.log
Job succeeded
```

Section 8. Verify the GitLab CI Package and Deploy to UCD and test the CICS transaction again using 3270

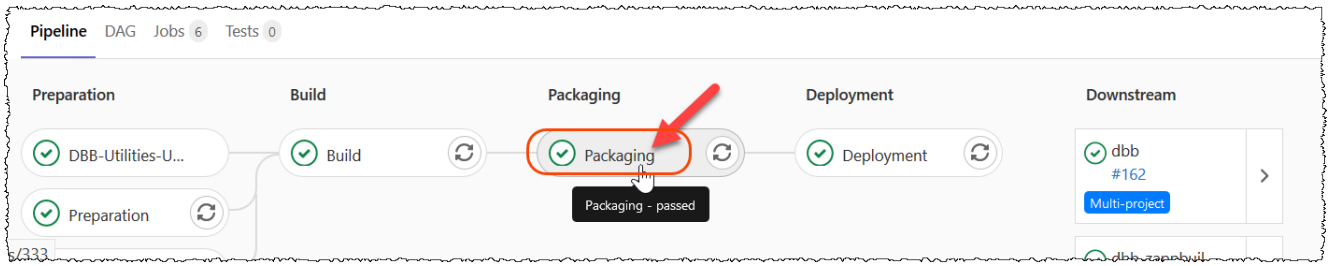
You will verify the results of the second stage (deploy using UCD) and also test the deployed application using CICS.

___ 8.1 Checking the Packaging results (second stage)

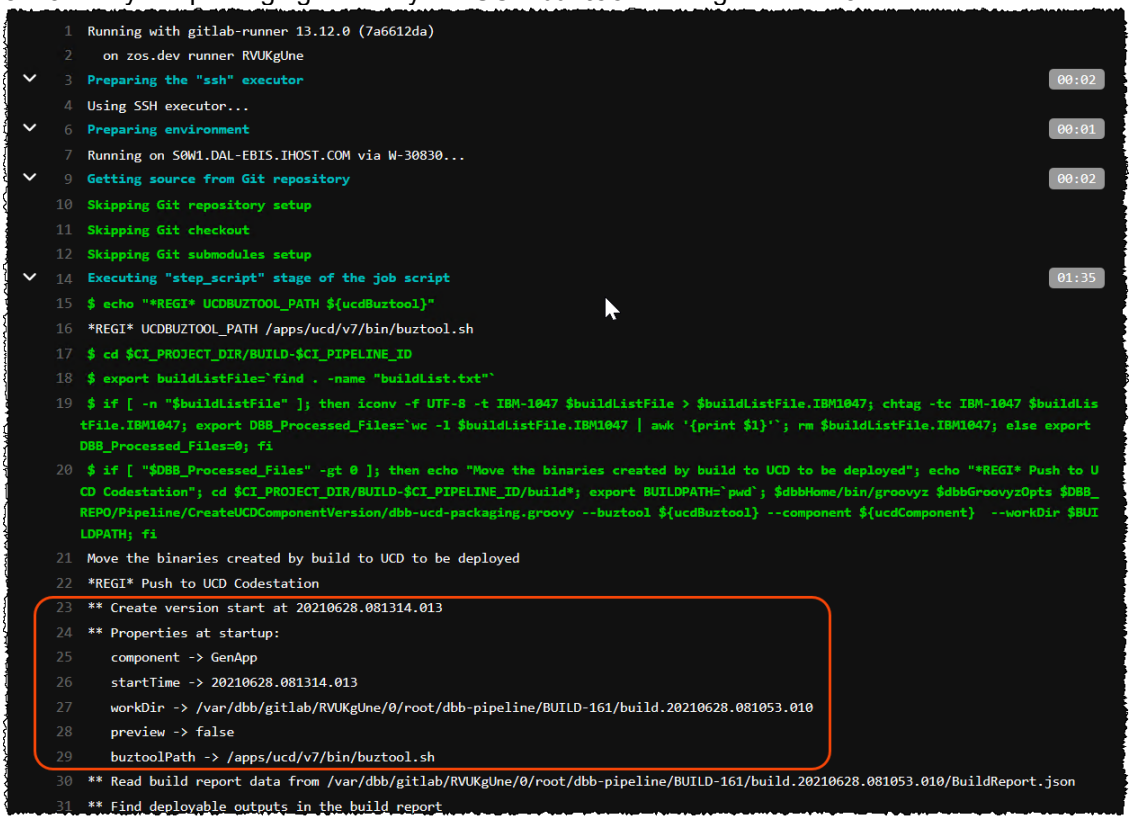
8.1.1 ⏭ Click on the **Go back** in the top left of the browser:



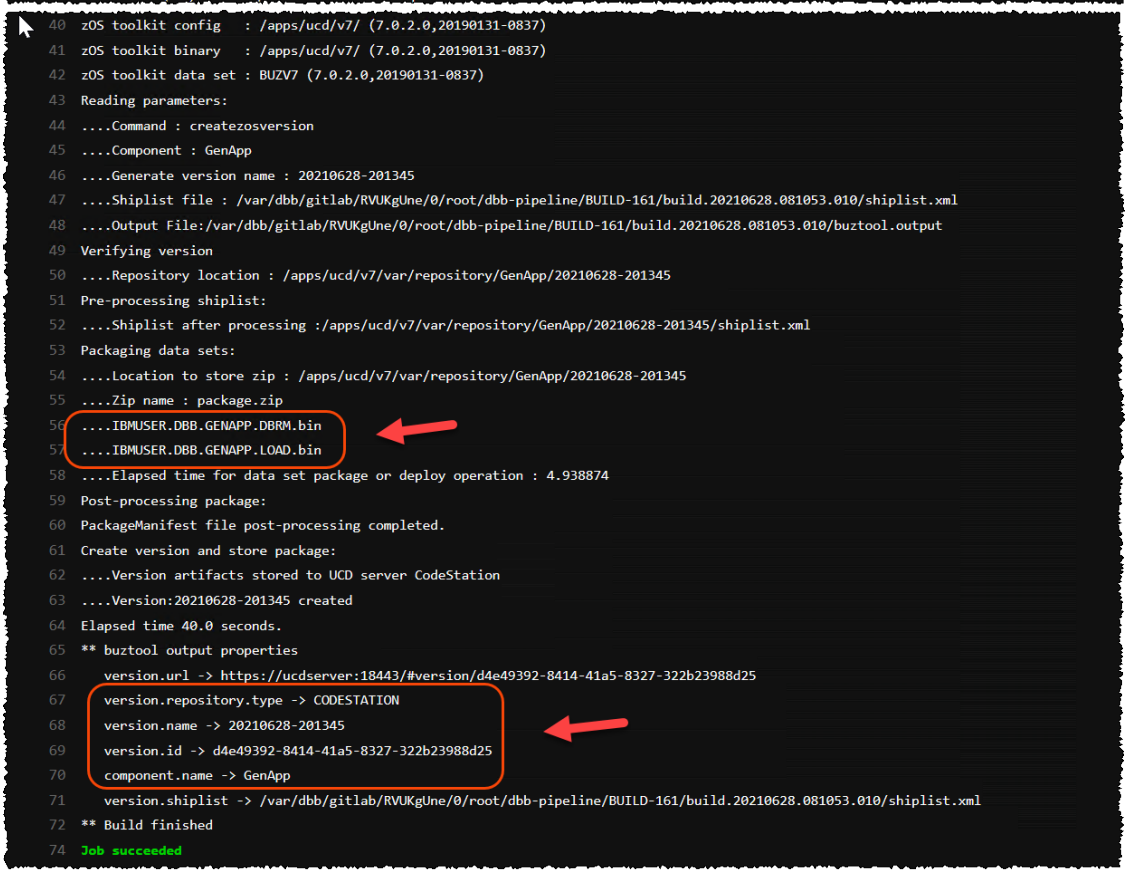
8.1.2 ⏭ Click on **Packaging**



8.1.3 Verify the packaging done by the UCD **buztool** starting on line **23**



```
Running with gitlab-runner 13.12.0 (7a6612da)
  on zos.dev runner RVUKgUne
Preparing the "ssh" executor                                                    00:02
Using SSH executor...
Preparing environment                                                           00:01
Running on S0W1.DAL-EBIS.IHOST.COM via W-30830...
Getting source from Git repository                                              00:02
Skipping Git repository setup
Skipping Git checkout
Skipping Git submodules setup
Executing "step_script" stage of the job script                                 01:35
$ echo "*REGI* UCDBUZTOOL_PATH ${ucdBuztool}"
*REGI* UCDBUZTOOL_PATH /apps/ucd/v7/bin/buztool.sh
$ cd $CI_PROJECT_DIR/BUILD-$CI_PIPELINE_ID
$ export buildListFile=`find . -name "buildList.txt"`
$ if [ -n "$buildListFile" ]; then iconv -f UTF-8 -t IBM-1047 $buildListFile > $buildListFile.IBM1047; chtag -tc IBM-1047 $buildLis
tFile.IBM1047; export DBB_Processed_Files=`wc -l $buildListFile.IBM1047 | awk '{print $1}'`; rm $buildListFile.IBM1047; else export
DBB_Processed_Files=0; fi
$ if [ "$DBB_Processed_Files" -gt 0 ]; then echo "Move the binaries created by build to UCD to be deployed"; echo "*REGI* Push to U
CD Codestation"; cd $CI_PROJECT_DIR/BUILD-$CI_PIPELINE_ID/build*; export BUILDPATH=`pwd`; $dbbHome/bin/groovyz $dbbGroovyzOpts $DBB_
REPO/Pipeline/CreateUCDComponentVersion/dbb-ucd-packaging.groovy --buztool ${ucdBuztool} --component ${ucdComponent}  --workDir $BUI
LDPATH; fi
Move the binaries created by build to UCD to be deployed
*REGI* Push to UCD Codestation
** Create version start at 20210628.081314.013
** Properties at startup:
   component -> GenApp
   startTime -> 20210628.081314.013
   workDir -> /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010
   preview -> false
   buztoolPath -> /apps/ucd/v7/bin/buztool.sh
** Read build report data from /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010/BuildReport.json
** Find deployable outputs in the build report
```

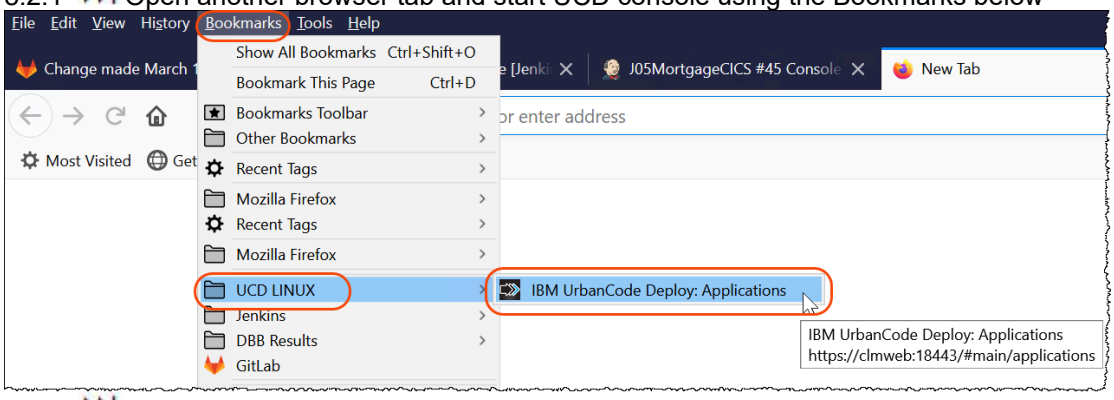


```
zOS toolkit config   : /apps/ucd/v7/ (7.0.2.0,20190131-0837)
zOS toolkit binary   : /apps/ucd/v7/ (7.0.2.0,20190131-0837)
zOS toolkit data set : BUZV7 (7.0.2.0,20190131-0837)
Reading parameters:
....Command : createzosversion
....Component : GenApp
....Generate version name : 20210628-201345
....Shiplist file : /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010/shiplist.xml
....Output File:/var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010/buztool.output
Verifying version
....Repository location : /apps/ucd/v7/var/repository/GenApp/20210628-201345
Pre-processing shiplist:
....Shiplist after processing :/apps/ucd/v7/var/repository/GenApp/20210628-201345/shiplist.xml
Packaging data sets:
....Location to store zip : /apps/ucd/v7/var/repository/GenApp/20210628-201345
....Zip name : package.zip
....IBMUSER.DBB.GENAPP.DBRM.bin
....IBMUSER.DBB.GENAPP.LOAD.bin
....Elapsed time for data set package or deploy operation : 4.938874
Post-processing package:
PackageManifest file post-processing completed.
Create version and store package:
....Version artifacts stored to UCD server CodeStation
....Version:20210628-201345 created
Elapsed time 40.0 seconds.
** buztool output properties
   version.url -> https://ucdserver:18443/#version/d4e49392-8414-41a5-8327-322b23988d25
   version.repository.type -> CODESTATION
   version.name -> 20210628-201345
   version.id -> d4e49392-8414-41a5-8327-322b23988d25
   component.name -> GenApp
   version.shiplist -> /var/dbb/gitlab/RVUKgUne/0/root/dbb-pipeline/BUILD-161/build.20210628.081053.010/shiplist.xml
** Build finished
Job succeeded
```

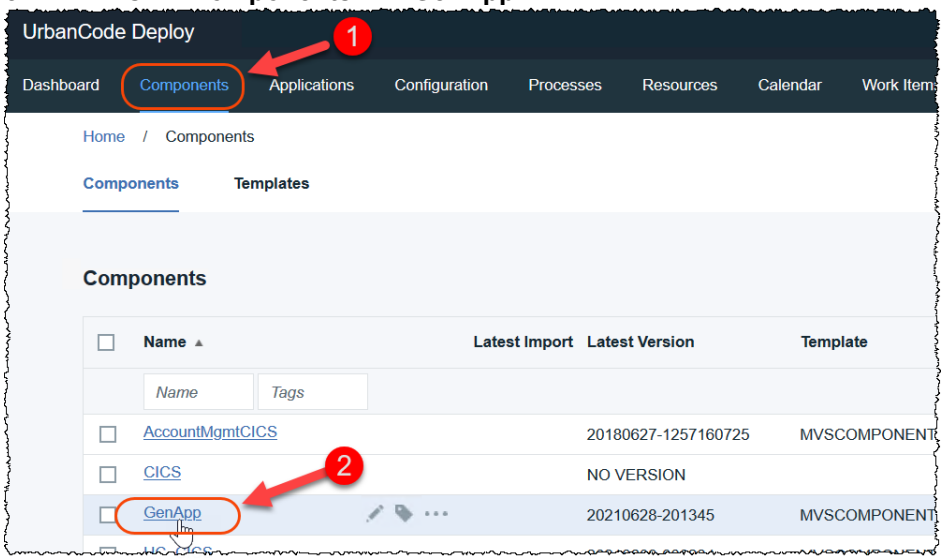
___ 8.2 Checking UrbanCode Created version

You can logon to UCD and verify the version created there

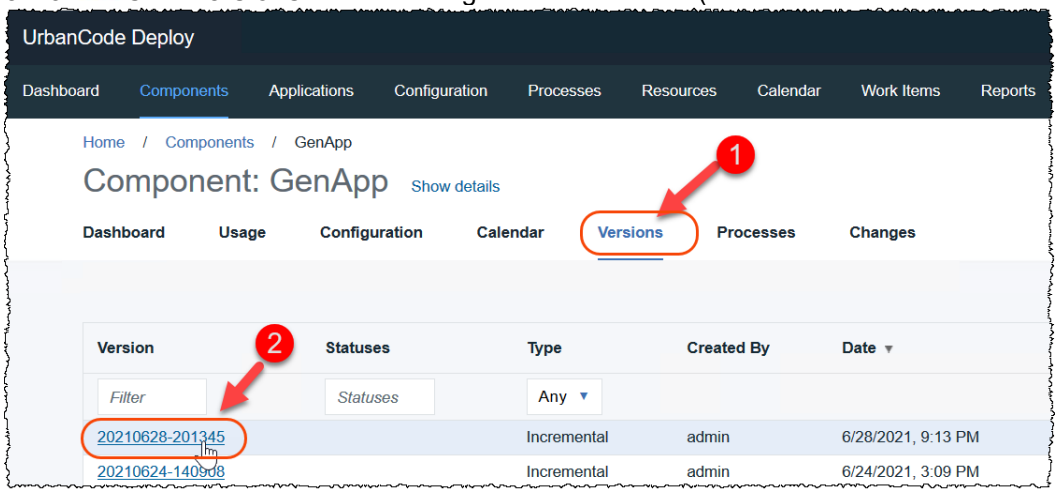
8.2.1 ⏭ Open another browser tab and start UCD console using the Bookmarks below



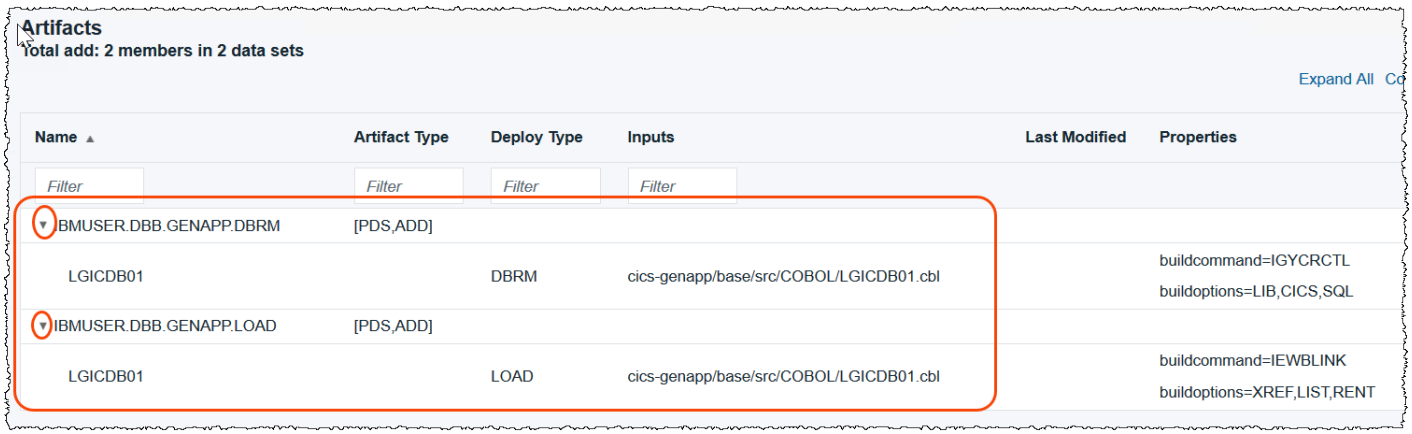
8.2.2 ⏭ Click **Components** and **GenApp**



8.2.3 ⏭ Click **Versions** and the last generated version (this name is also shown on the GitLab log)

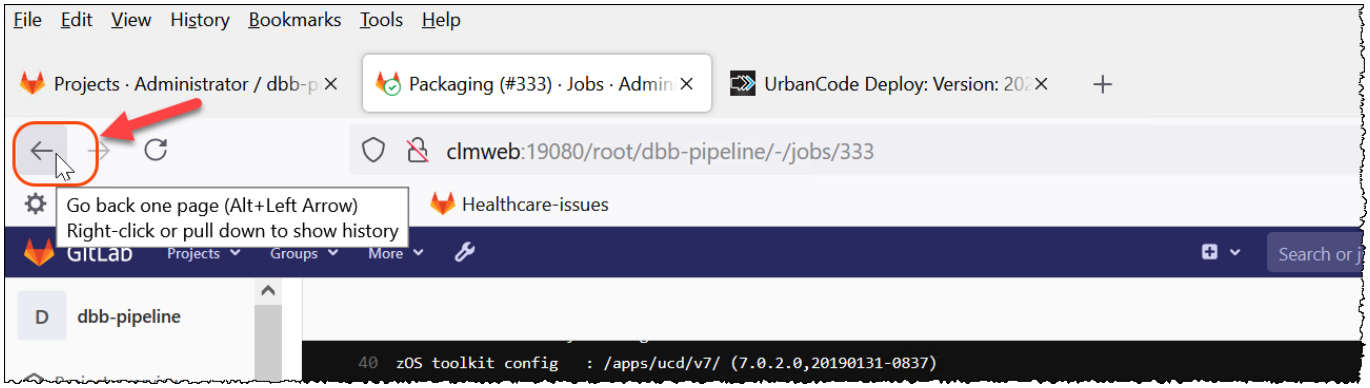


8.2.4 Expand the artifacts and you will see the binaries created at the UCD Code station

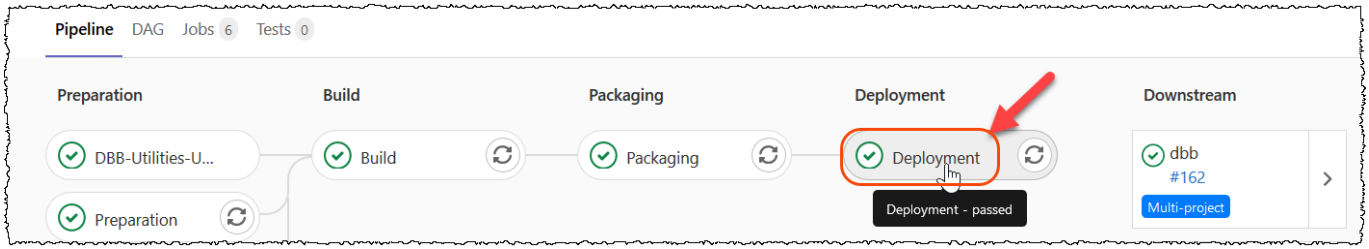


___ 8.3 Checking the Deployment results (third stage)

8.3.1 Open the GitLab Packaging that an click on the **Go back** in the top left of the browser:



8.3.2 Click on **Packaging**



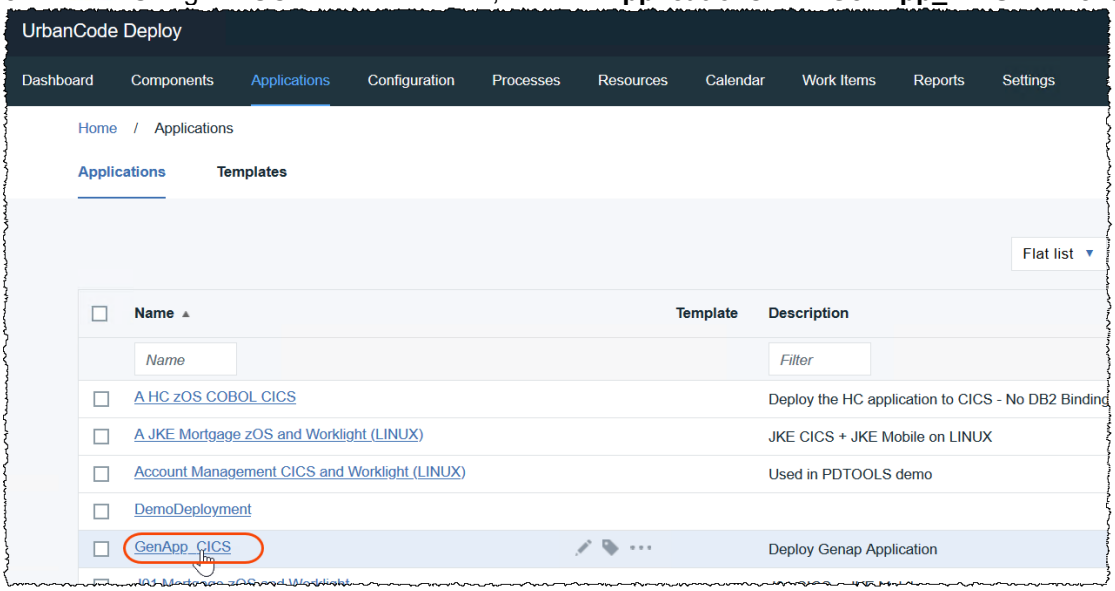
8.3.4 The Deploy was successful as seem on lines 17-26

```
1  Running with gitlab-runner 13.12.0 (7a6612da)
2    on zos.dev runner RVUKgUne
3  Preparing the "ssh" executor                                                    00:02
4  Using SSH executor...
6  Preparing environment                                                           00:02
7  Running on S0W1.DAL-EBIS.IHOST.COM via W-30830...
9  Getting source from Git repository                                              00:02
10 Skipping Git repository setup
11 Skipping Git checkout
12 Skipping Git submodules setup
14 Executing "step_script" stage of the job script                                 05:08
15 $ cd $CI_PROJECT_DIR/BUILD-$CI_PIPELINE_ID
16 $ export buildListFile=`find . -name "buildList.txt"`
17 $ if [ -n "$buildListFile" ]; then iconv -f UTF-8 -t IBM-1047 $buildListFile > $buildListFile.IBM1047; chtag -tc IBM-1047 $buildLis
   tFile.IBM1047; export DBB_Processed_Files=`wc -l $buildListFile.IBM1047 | awk '{print $1}'`; rm $buildListFile.IBM1047; else export
   DBB_Processed_Files=0; fi
18 $ if [ "$DBB_Processed_Files" -gt 0 ]; then echo "Invoke UCD and wait for the deployment to be completed"; echo "*REGI* Deploy Appl
   ${ucdApplication} Process  ${ucdProcess}  to Environment :${ucdEnv} on CICSTS53"; cd $CI_PROJECT_DIR/BUILD-$CI_PIPELINE_ID/build*; e
   xport BUILDPATH=`pwd`; $DBB_HOME/bin/groovyz $DBB_REPO/Pipeline/DeployUCDComponentVersion/ucd-deploy.groovy  -a "$ucdApplication" -e
   "$ucdEnv" -U $ucdUser -P $ucdPassword -u $ucdSite -d "$ucdComponent:latest" -p "$ucdProcess" -k -t 3000000; fi
19 Invoke UCD and wait for the deployment to be completed
20 *REGI* Deploy Appl  GenApp_CICS Process  Deploy_to_CICS  to Environment :DEV on CICSTS53
21 **  Deploying component versions: GenApp:latest
22 *** Starting deployment process 'Deploy_to_CICS' of application 'GenApp_CICS' in environment 'DEV'
23 *** Follow Process Request: https://clmweb:18443//#applicationProcessRequest/17a55554-6e1b-43b7-ecbd-385707b4df5f
   .................................................
24 *** The deployment result is SUCCEEDED. See the UrbanCode Deploy deployment logs for details.
25 ** Build finished
27 Job succeeded
```

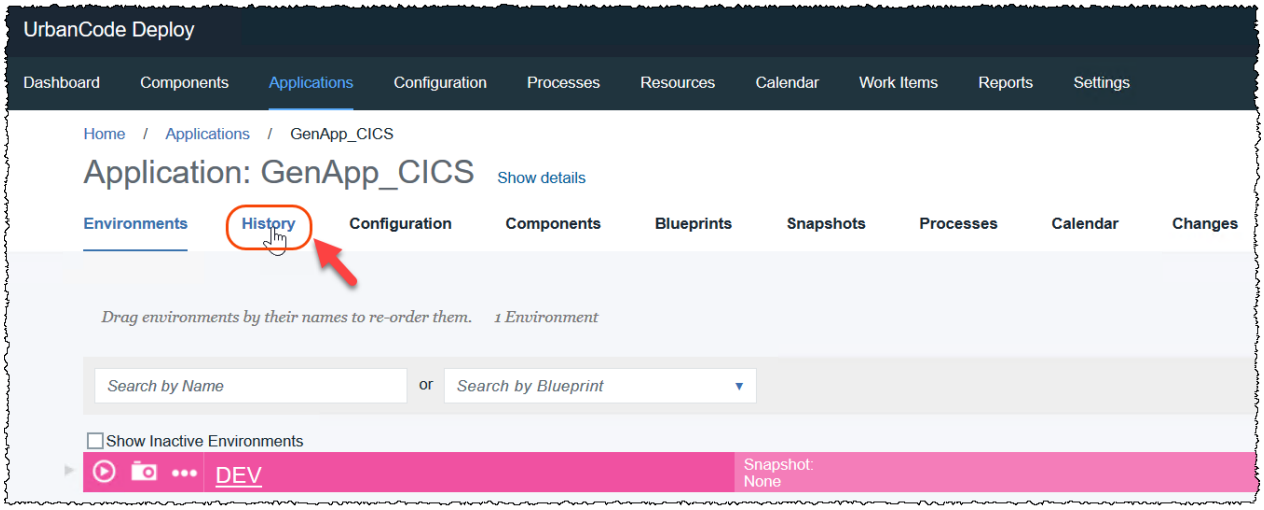
___ 8.4 Checking UrbanCode Deploy results

You can logon to UCD and verify the deploy results created there

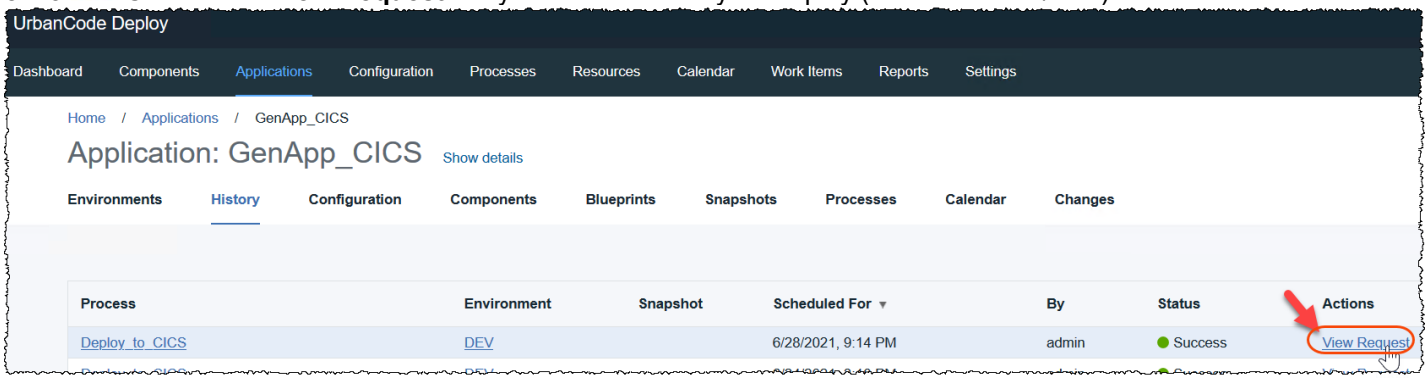
8.4.1 Using the UCD tab on browser, click on **Applications** find **GenApp_CICS** and **click** on it



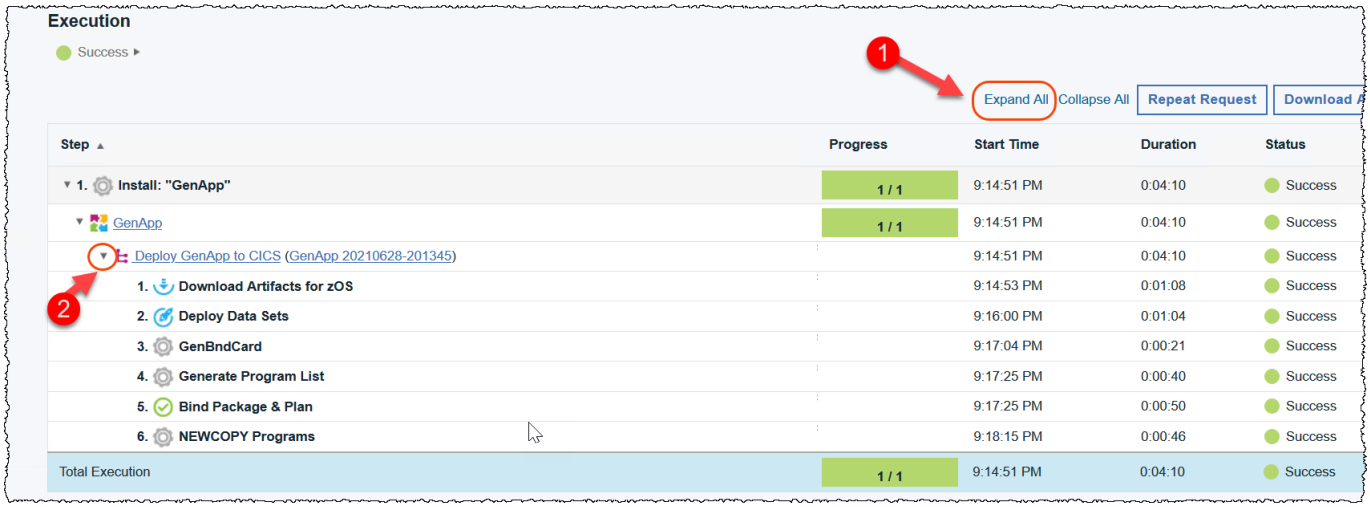
8.4.2 Click on the **History** tab



8.4.3 Click on the **View Request** entry that is related to your deploy (check the date/time)

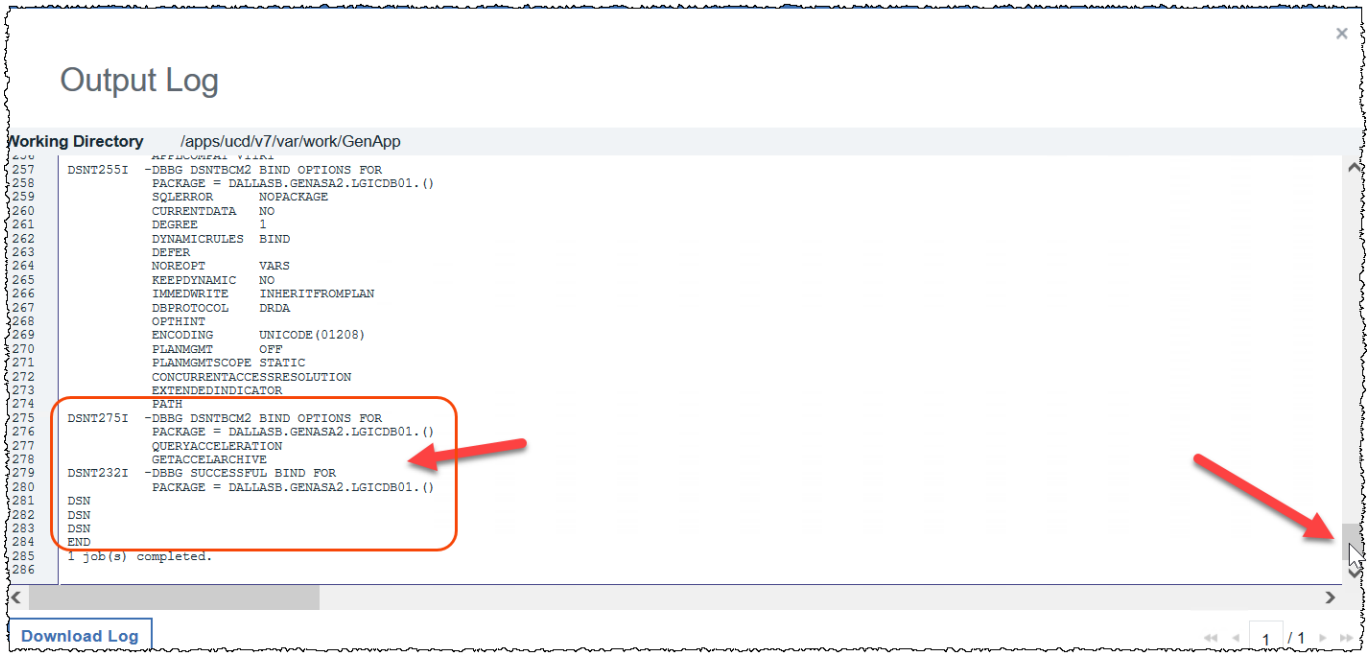


8.4.4 Click **Expand All** and **expand the node** as show in the #2 below
Once the step is green means that it is completed.

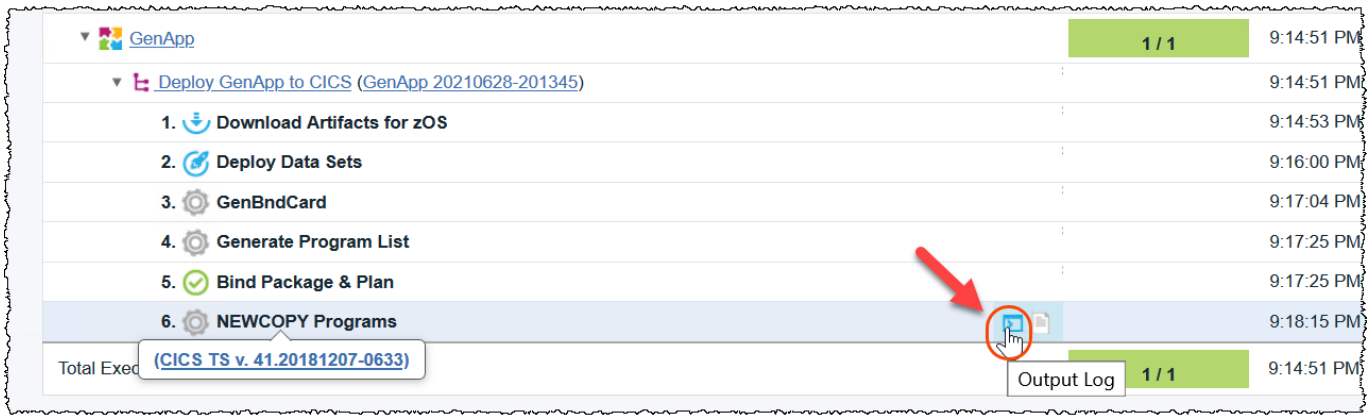


8.4.5 Once the **Bind Package & Plan** is green, move the mouse as below and click **output log**

8.4.6 Notice that The **bind** was successful.



8.4.7 Once the **NEWCOPY Programs** is green, move the mouse as below and click **output log**

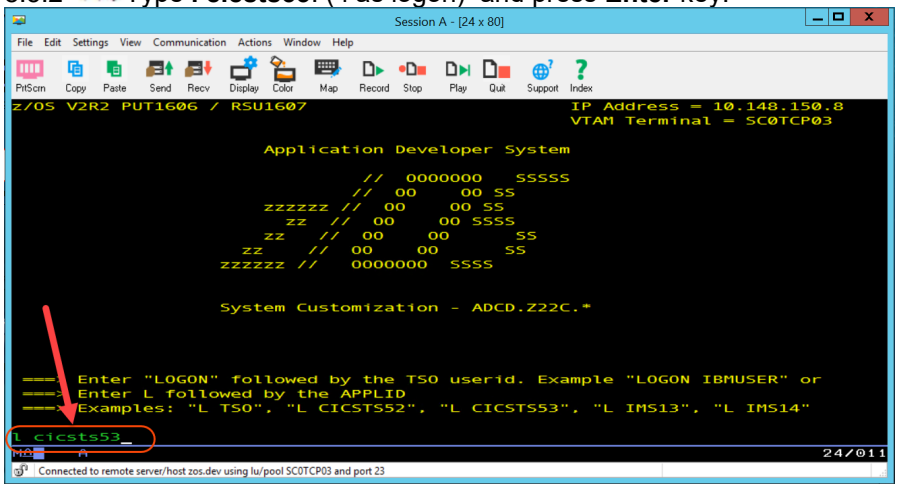


8.4.8 Notice that **LGICDB01** program was deployed to CICS Dev environment. and a CICS NEWCOPY was succeeded.

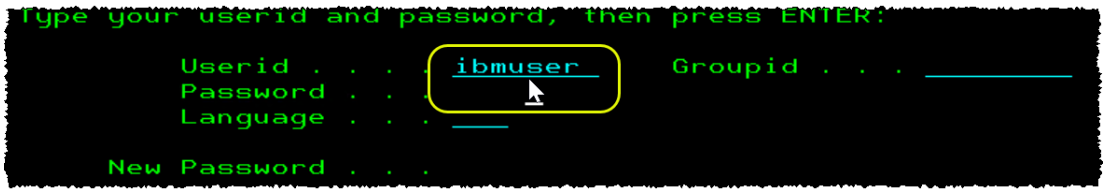
___ 8.5 Testing the CICS code deployed to Dev environment

8.5.1 Double click on the **3270 terminal emulator** that is on the Windows desktop

8.5.2 Type **l cicsts53**. (l as logon) and press **Enter** key.

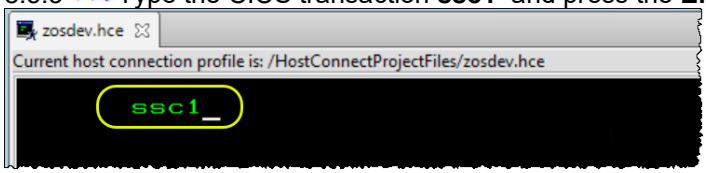


8.5.3 Logon using your z/OS user id and password (**ibmuser/sys1**) and press **Enter**

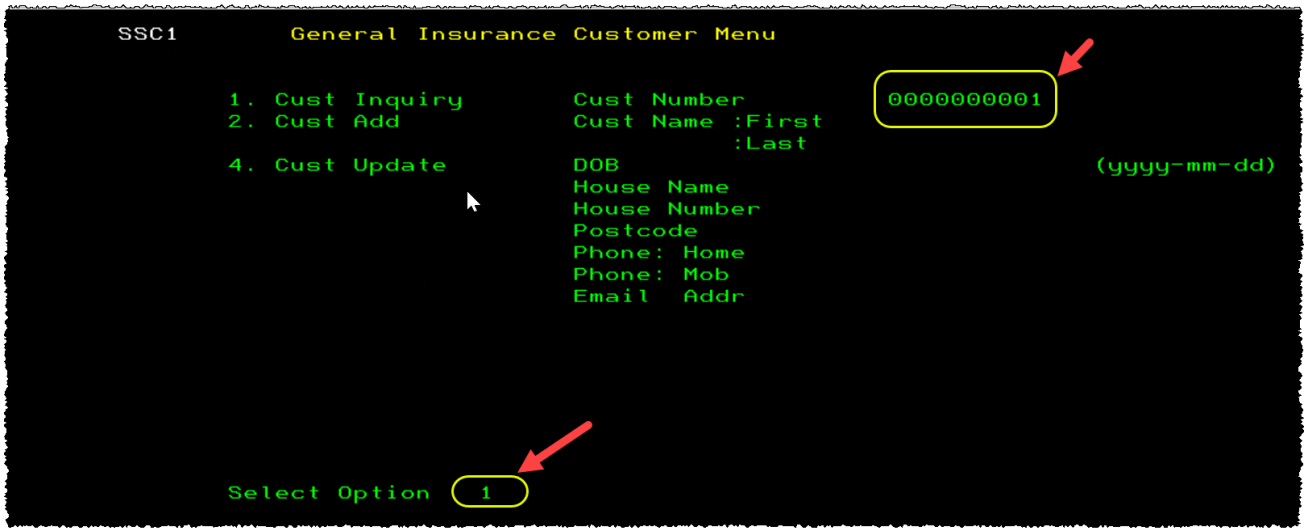


8.5.4 The sign-on message is displayed

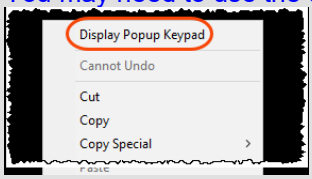
8.5.5 Type the CICS transaction **ssc1** and press the **Enter** key.

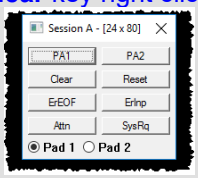


8.5.6 Move the mouse to last **0** and type **1**, use the tab key and type **1** as ***Select option*** and press ***Enter***

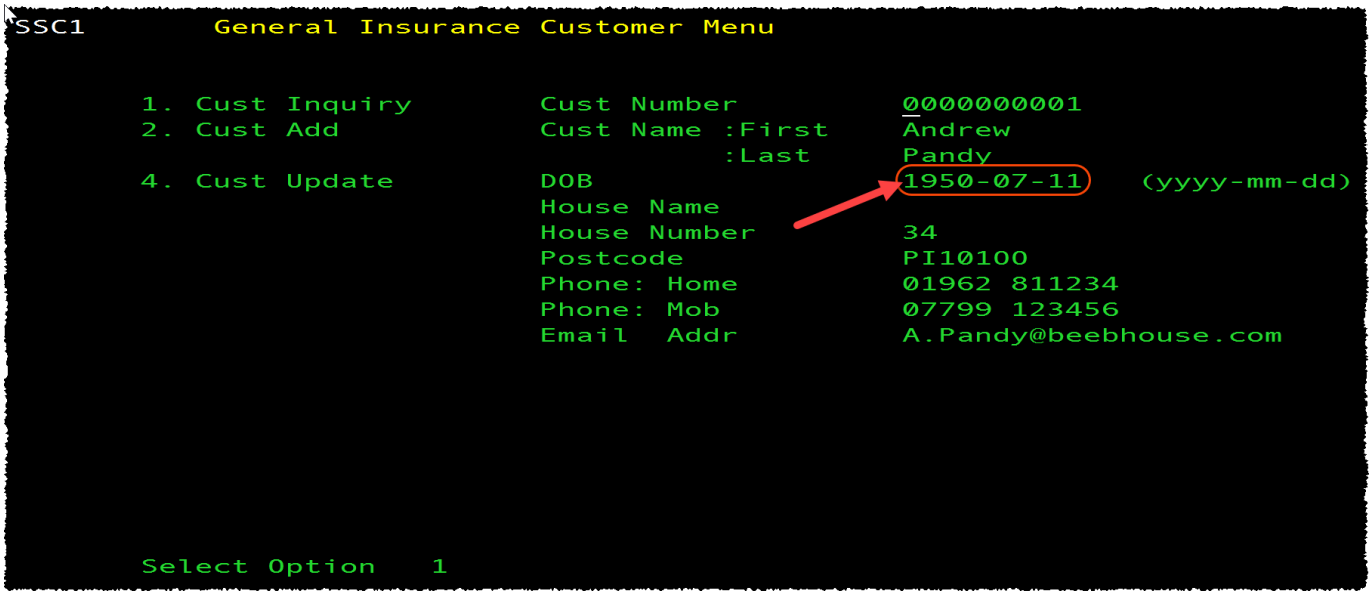


You may need to use the **clear** key right click and select display Popup Keypad

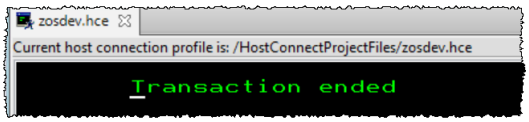




8.5.7 ⏭ The data below is retrieved from a DB2 table .
Notice that **DOB** (*Date Of Birthday*) is now correct. The bug is fixed.



8.5.8 ⏭ Press **F3** to end the application.



Congratulations! You have completed the Lab 10. .