

DevOps on Z

Proof of Technology (PoT)

Exercises Using IBM ZDevops Tools

Updated 2024

OVERVIEW	4
LAB 1 - WORKING WITH MAINFRAME USING COBOL AND DB2	5
SECTION 1. CONNECT TO A z/OS SYSTEM	6
____ 1.1 WORKING WITH THE IDz WORKSPACE	6
____ 1.2 CONNECTING TO THE z/OS REMOTE SYSTEM	9
SECTION 2. EXECUTE THE DB2/COBOL BATCH PROGRAM AND VERIFY THE ABEND	11
____ 2.1 SUBMIT A COBOL/DB2 BATCH TO EXECUTE	11
____ 2.2 ADD z/OS RESOURCES TO THE MVS SUBPROJECT	14
____ 2.3 SUBMIT A JCL TO EXECUTE THE COBOL PROGRAM	16
SECTION 3. USE FAULT ANALYZER TO IDENTIFY THE CAUSE OF THE ABEND	19
____ 3.1 USING FAULT ANALYZER PERSPECTIVE	19
SECTION 4. USING THE IBM z/OS DEBUGGER FOR A TEMPORARY FIX	28
____ 4.0 BE SURE THAT IDz CLIENT IS LISTENING ON PORT 8001.....	28
____ 4.1 SUBMIT THE JCL TO INVOKE THE DEBUG	29
____ 4.2 USING THE VISUAL DEBUGGER FOR STACK PATTERN BREAKPOINTS.....	37
____ 4.3 ACCESSING THE z/OS JES	42
SECTION 5. MODIFY THE COBOL CODE TO FIX THE BUG.....	46
____ 5.1 USING THE EDITOR WITH z/OS COBOL PROGRAMS	46
____ 5.2 FIXING THE PROGRAM REGI0B	54
____ 5.3 COMPILE AND LINK REGI0B.....	59
____ 5.4 EXECUTE THE COBOL/DB2 PROGRAM	65
SECTION 6. USING THE CODE COVERAGE	69
____ 6.1 CREATING A JCL TO RUN THE CODE COVERAGE.....	69
____ 6.2 SUBMIT THE JCL FOR z/OS EXECUTION	72
LAB 2 - CREATE URBancode DEPLOY INFRASTRUCTURE AND DEPLOY TO z/OS.....	84
LAB ARCHITECTURE AND PROPOSED SCENARIO	84
PART 1 – CREATE THE URBancode APPLICATION INFRASTRUCTURE	85
TASK 2 – CREATE A UCD COMPONENT	99
TASK 3 – CREATE A SHIP LIST FILE AND z/OS COMPONENT VERSION	102
TASK 4 - CREATE THE UCD COMPONENT VERSION FROM JCL	111
TASK 5 - CREATE UCD RESOURCES	123
TASK 6 - CREATE AN URBancode APPLICATION	127
TASK 7 - CREATE ENVIRONMENT	129
TASK 8 - CREATE A COMPONENTS PROCESS	133
TASK 9 - CREATE AN APPLICATION PROCESS	160
TASK 10 – ENVIRONMENT PROPERTIES CONFIGURATION.....	165
PART 3 – DEPLOY THE APPLICATION TO z/OS CICS.....	173
TASK 11 – RUN AN APPLICATION PROCESS	173
TASK 12 – VERIFYING THE DEPLOY RESULTS AT z/OS CICS	178
LAB 3 – USING IBM DEPENDENCY BASED BUILD WITH GIT, JENKINS AND UCD ON z/OS	182
SECTION 0. CHECK UCD AGENT IS ONLINE.....	184
SECTION 1. GET FAMILIAR WITH THE APPLICATION USING THE 3270 TERMINAL	193
____ 1.2 RUN CICS TRANSACTION J05P	199
SECTION 2. LOAD THE SOURCE CODE FROM GIT TO THE LOCAL IDz WORKSPACE	201
____ 2.1 CLONING THE GIT REPOSITORY	201
____ 2.2 VERIFY THE CODE CLONED USING z/OS PROJECTS PERSPECTIVE	207
SECTION 3. MODIFY THE COBOL CODE USING IDz	208
____ 3.1 EDIT AND MODIFY THE CODE THAT SEND THE MESSAGE	209
SECTION 4. USE IDz DBB USER BUILD TO COMPILE/BIND AND PERFORM PERSONAL TESTS.	211
____ 4.1 USING DEPENDENCY BASED BUILDING OPTION	211
____ 4.2 VERIFY THE DBB USER BUILD RESULTS	221
____ 4.3 ISSUING A CICS NEW COPY	222
____ 4.4 RUNNING J05P CICS TRANSACTION.....	227
SECTION 5. PUSH AND COMMIT THE CHANGED CODE TO GIT.....	229

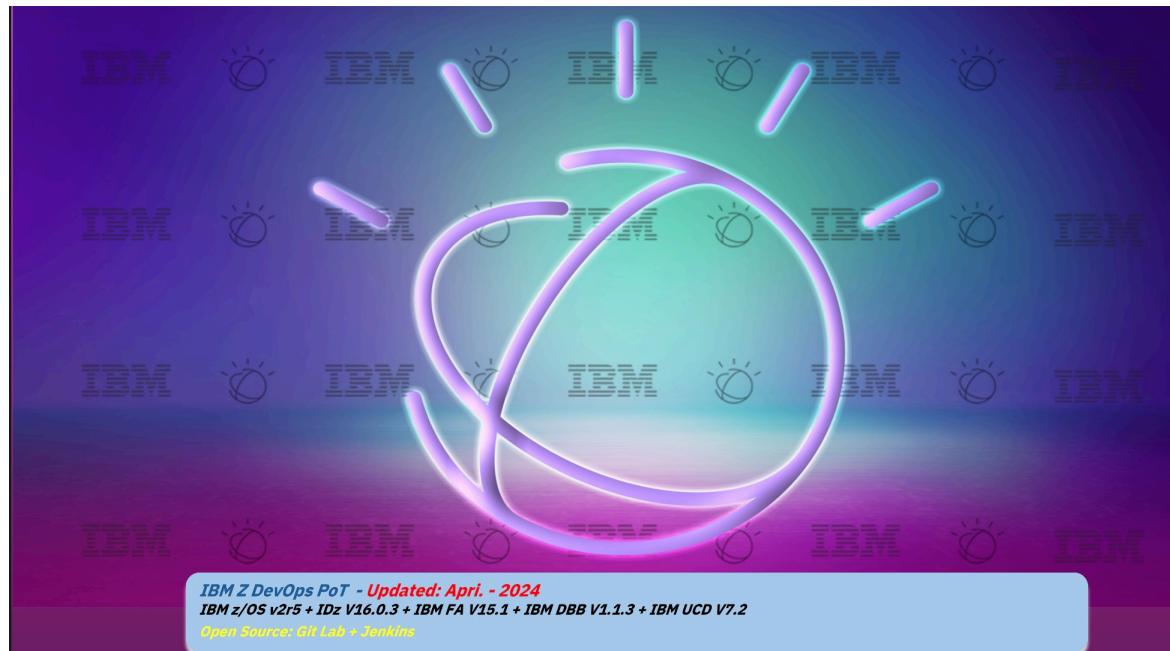
5.1 USING IDz WITH THE GIT PLUGIN TO COMPARE THE CHANGE VERSUS ORIGINAL.....	229
5.2 PUSH AND COMMIT THE CHANGES TO GIT	230
SECTION 6. USE JENKINS WITH GIT PLUGIN TO BUILD ALL THE MODIFIED CODE COMMITTED TO GIT.....	233
6.1 LOGON TO JENKINS USING A WEB BROWSER AND START THE Z/OS AGENT.....	233
6.2 STARTING THE JENKINS PIPELINE.....	235
6.3 CHECKING THE RESULTS	237
6.4 UNDERSTAND THE RESULTS	240
SECTION 7. USE JENKINS AND UCD PLUGIN TO DEPLOY RESULTS AND TEST THE CICS TRANSACTION AGAIN	244
7.1 CHECKING THE DEPLOY RESULTS (SECOND STAGE)	244
7.2 CHECKING URBancode DEPLOY LOGS	247
7.3 USING JENKINS BLUE OCEAN PLUGIN	251
7.4 TESTING THE CICS CODE DEPLOYED TO DEV ENVIRONMENT	254
SECTION 8. UNDERSTANDING DBB Build Reports	259
8.1 LOGIN TO THE DBB SERVER USING THE BROWSER	259
8.2 UNDERSTAND THE DBB COLLECTIONS	260
8.3 UNDERSTAND THE BUILD RESULTS	261

Overview

- Integrated application development and problem analysis ([ADFz](#)) using Fault Analyser and z/OS Debugger
- Managing your source code using modern tools (including [Git](#))
- Building automation ([DBB](#)) for COBOL or PL/1 without a specific source code manager or pipeline automation tool (including [Groovy](#) and [Jenkins](#))
- Using [GitLab CI](#) along with DBB, and UrbanCode Deploy (UCD) on z/OS.
- Application deployments automation to many environments ([UCD](#))

This material was built using a Windows on cloud that was updated on **April 2024**

Here the desktop background of this image:



The following symbols appear in this document at places where additional guidance is available.

Icon	Purpose	Explanation
	Important!	This symbol calls attention to a particular step or command. For example, it might alert you to type a command carefully because it is case sensitive.
	Information	This symbol indicates information that might not be necessary to complete a step, but is helpful or good to know.
	Trouble-shooting	This symbol indicates that you can fix a specific problem by completing the associated troubleshooting information.

LAB 1 - Working with mainframe using COBOL and DB2

Updated June,2024 (revised by Carlos Hirata/Wilbert Kho, Reviewed by Joe Huang and Frank Hernandez)

This lab will take you through the steps of using the [Application Delivery Foundation for z Systems](#) (ADFz) to work with a z/OS system. It will familiarize you with some of the capabilities of this product using a DB2 COBOL batch program that is ABENDING.

On this lab you will connect to a remote z/OS system, submit and execute a program (which ABENDs), identify the program abend, set up a MVS project, edit, compile, and debug a COBOL application. The process would be similar for a PL/I program.

Overview of development tasks

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

- 1. Connect to a z/OS System.**
→ You will connect to the z/OS system using a provided z/OS logon userid.
- 2. Execute the DB2/COBOL batch program and verify the ABEND.**
→ You will submit a COBOL/DB2 batch to execute and verify the bug.
- 3. Use Fault Analyzer to identify the cause of the ABEND**
→ You will use Fault Analyzer to identify what is causing the ABEND.
- 4. Use the IBM Debug for a temporary fix**
→ You will modify the field content to bypass the bug
- 5. Modify the COBOL code to fix the bug.**
→ You will be able to fix the bug changing the COBOL code.
- 6. Use Code Coverage**
→ You can now verify program areas covered by the test case with Code Coverage

Section 1. Connect to a z/OS System

You will connect to the z/OS. This section is like LOGON to a TSO. You will use **empot01** as userid and password.

1.1 Working with the IDz workspace

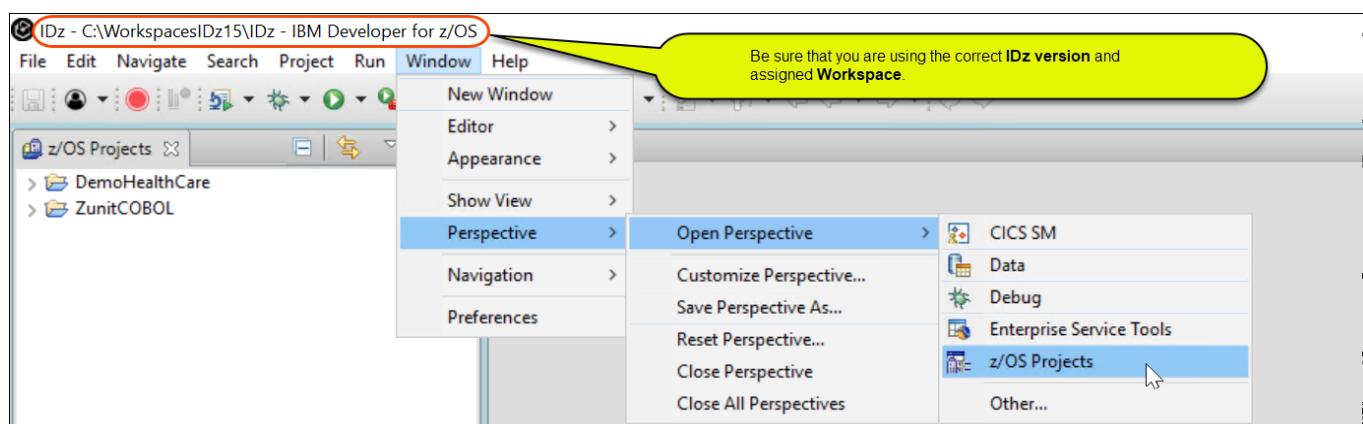
1.1.1 Start IBM Developer for z Systems version 16

- ▶▶ Using the windows desktop double click on **IDz V16** icon.
- ▶▶ Verify that the message indicates that it is **Version 16**

IMPORTANT -> This icon will start an eclipse workspace that has already some definitions required for this lab. PLEASE DO NOT start IDz using other way than this icon. The starting process might take some time .. Please be Patient.



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



z/OS Projects perspective

Use the z/OS Projects perspective to define the connection, connect to, and work with remote systems, and to create, edit, and build projects, subprojects, and files on remote UNIX, Linux for z Systems, and z/OS systems.

The z/OS Projects perspective contains the many views like Remote Systems view, z/OS Projects view, Properties view, Outline view, Remote Error List view, z/OS File System Mapping view, Remote System Details view, Team view, Property Group Manager view and Snippets view



You can close and open views to customize the perspective.

To open one of the views that were closed:

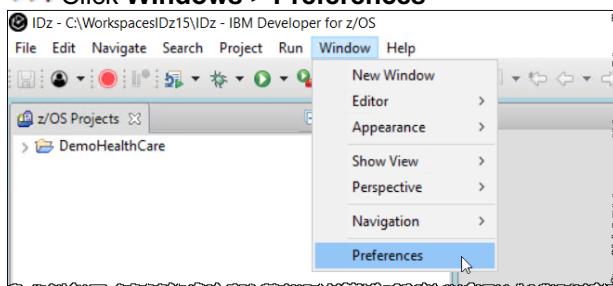
1. In the workbench, select **Window > Show View**.

A menu that lists the views associated with the z/OS Projects perspective is displayed.

2. Click the name of the view you want to open.

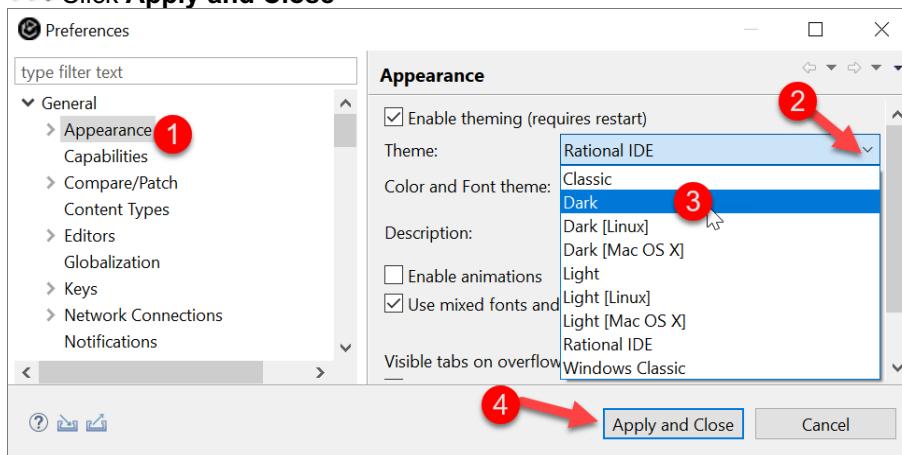
1.1.3 Starting on version 616 the developer can change the workspace appearance. If you want to try that

►| Click **Windows > Preferences**



1.1.4 ►| Under **General** select **Appearance** and on **Theme** select **Dark**.

►| Click **Apply and Close**

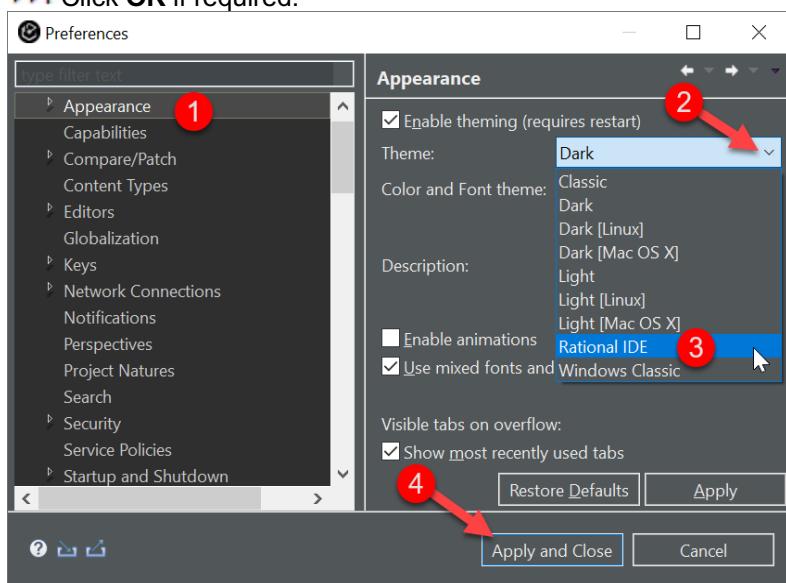


1.1.5 Since all picture here were done using “Rational IDE” theme we recommend to return to the default.

►| Again, click **Windows > Preferences**

►| Under **General** select **Appearance** and on **Theme** select **Rational IDE** and click **Apply and Close**

▶ Click OK if required.



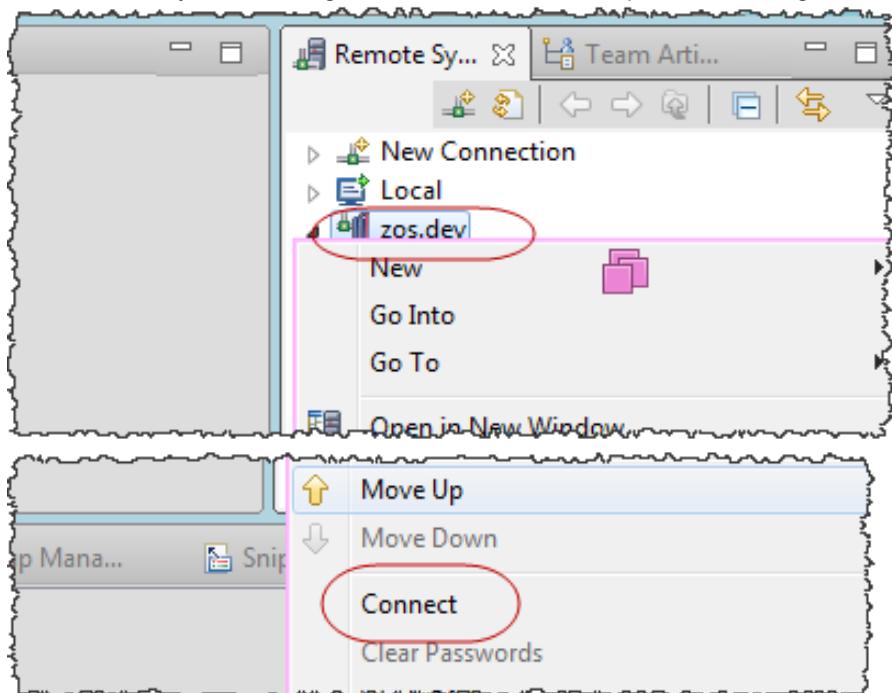
1.2 Connecting to the z/OS Remote system

1.2.1 To Connect to the z/OS system:

► Using the *Remote Systems* view on the top and right side of the screen:

Right-click on **zos.dev** and select **Connect**

Notice: Since you are using a cloud instance the response to the right click may be delayed first time.

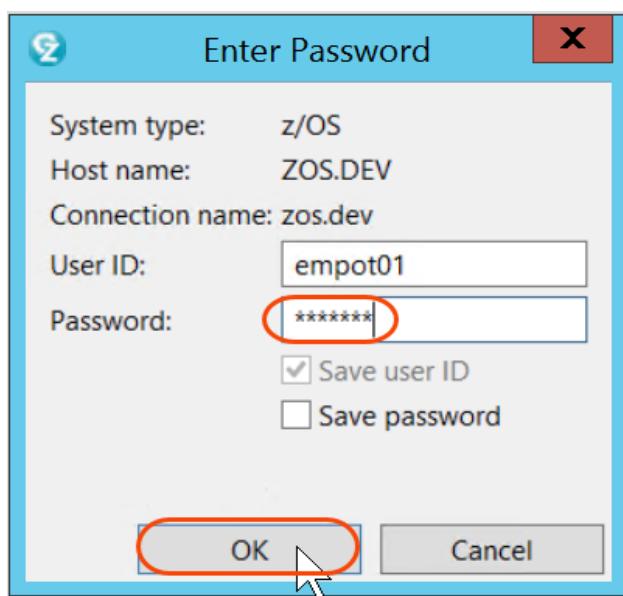


1.2.2 You will be prompted for your z/OS userid and password.

► Type **empot01** as userid (might be already there) and **empot01** as password.

The userid and password **can be any case; don't worry about having it in UPPER case.**

1.2.3 ► Click **OK** to connect to z/OS.



Be Patient! The connection could take a while depending on the network condition.

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

Notice that some folder icons changed after connected. A small **green arrow**  is added.



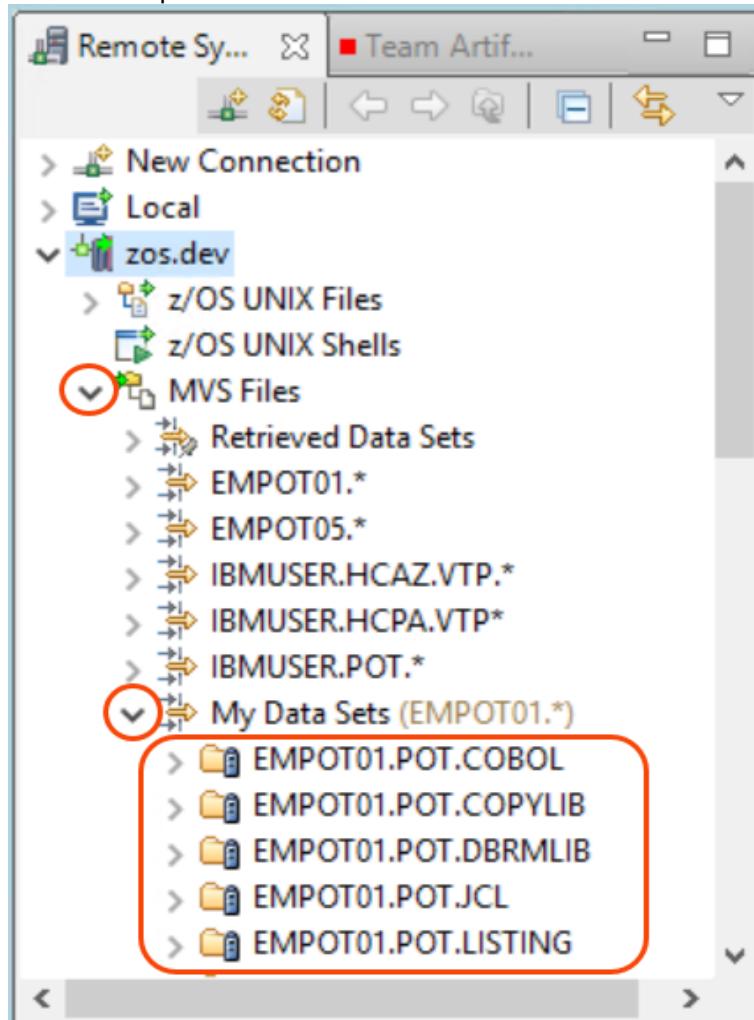
In case of connection errors...

If you have errors during the connection it is because the z/OS system is not available.
You also may have network issues.



An easy way to find if z/OS is up is opening a 3270 emulation clicking on the icon  that is on the base of your windows client. (zos.dev is not "pingable").
If you do not receive a reply, Contact the instructor. Your connection is broken.

1.2.5 ► Expand  MVS Files and  My Data Sets  to see MVS data sets that starts with **EMPOT01.***



Remote Systems view

The Remote Systems view shows all existing connections to remote systems. Connections are persisted, containing the information needed to access a remote host. The view contains a prompt to create new connections, and pop-up menu actions to rename, copy, delete, and reorder existing connections. Connections contain attributes, or data, that are saved between sessions of the workbench. These attributes are the connection name, the remote system's host name and system type, an optional description, and a user ID that is used by default by each subordinate subsystem, at connection time.



Section 2. Execute the DB2/COBOL batch program and verify the ABEND.

To make your job easier, you will group together all the assets that you will work with. This is a new development concept for TSO/ISPF users, since TSO/ISPF does not provide such capability. To accomplish this task, you will create a **z/OS project** and select which assets will be part of this project.

What is a z/OS project?

After you define a z/OS-connection and connect to that system, you can define a z/OS project under that system. You can define the z/OS project, however, only when you are connected to the system. Application Delivery Foundation for z Systems assigns a set of default properties from the set of system properties. However, changes that you make to system properties do not affect the definition of an existing project. If you change your system properties to reference a new compiler release, for example, the reference affects only those projects that are defined subsequent to the change. This isolation of system data from existing projects is beneficial because it lets you develop your code without disruption. You can introduce changes to the project definition at a time of your choosing.

Creating a new MVS subproject

MVS subprojects contain the development resources that reside on an MVS system. You can create multiple subprojects in a z/OS project.

Before you create an MVS subproject, you need to have completed the following tasks:

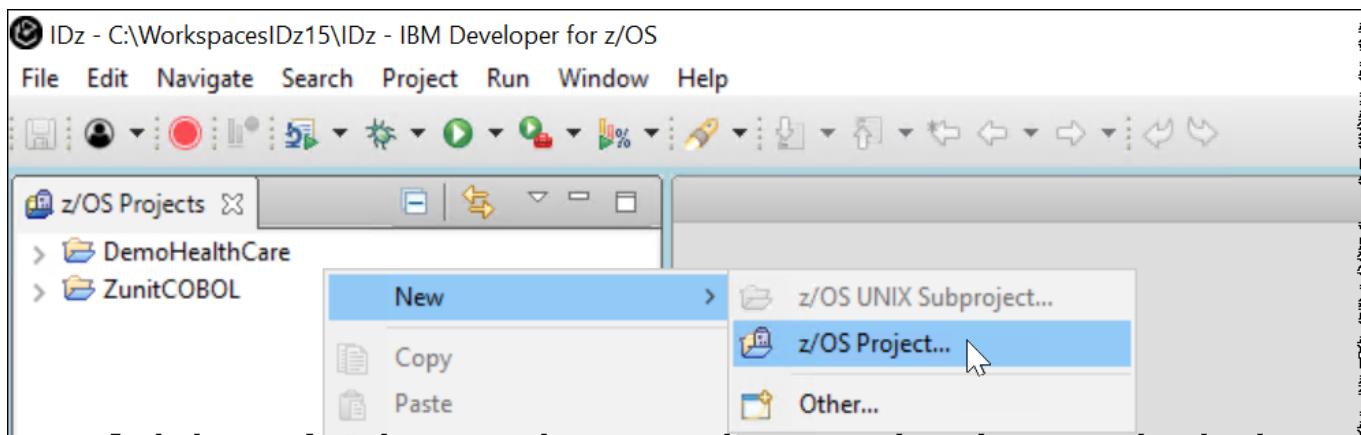
- Connecting to a remote system
- Creating a z/OS project



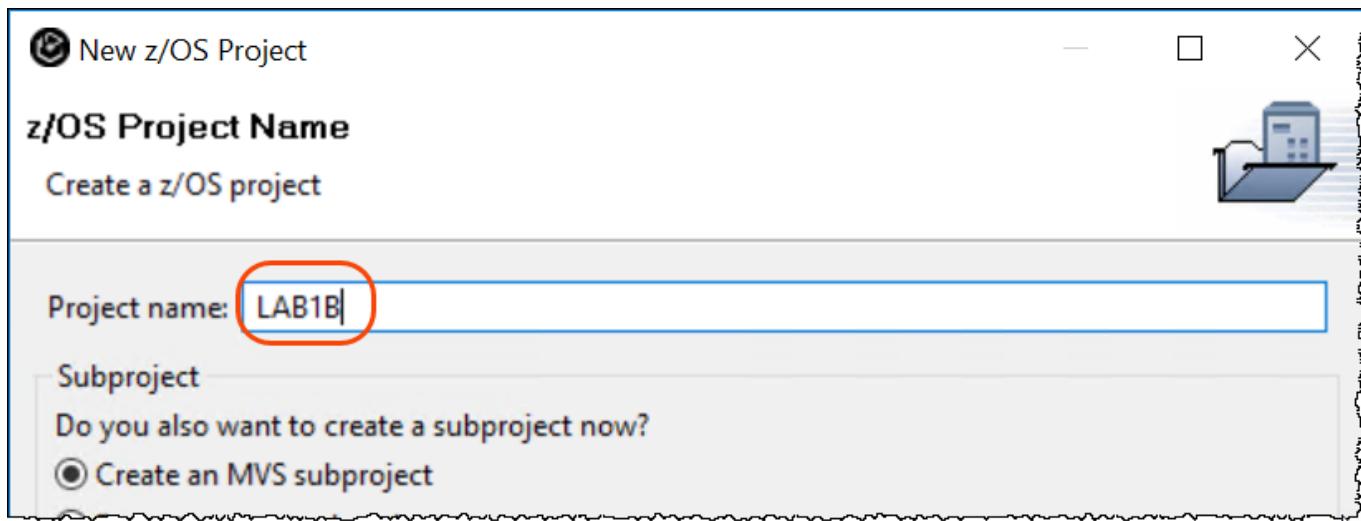
2.1 Submit a COBOL/DB2 batch to execute

The advantage of creating a z/OS Project is that we just focus on those datasets and members that are being constructed or updated, instead of having all the mainframe datasets or members. At any time if you need to see a dataset not added to the project, just go to the z/OS Systems view and add it. In addition, at any time, you can remove from your project the datasets that you don't need anymore.

2.1.1 ►| Using the **z/OS Projects** view (on the top and left), right click and select **New > z/OS Project...**

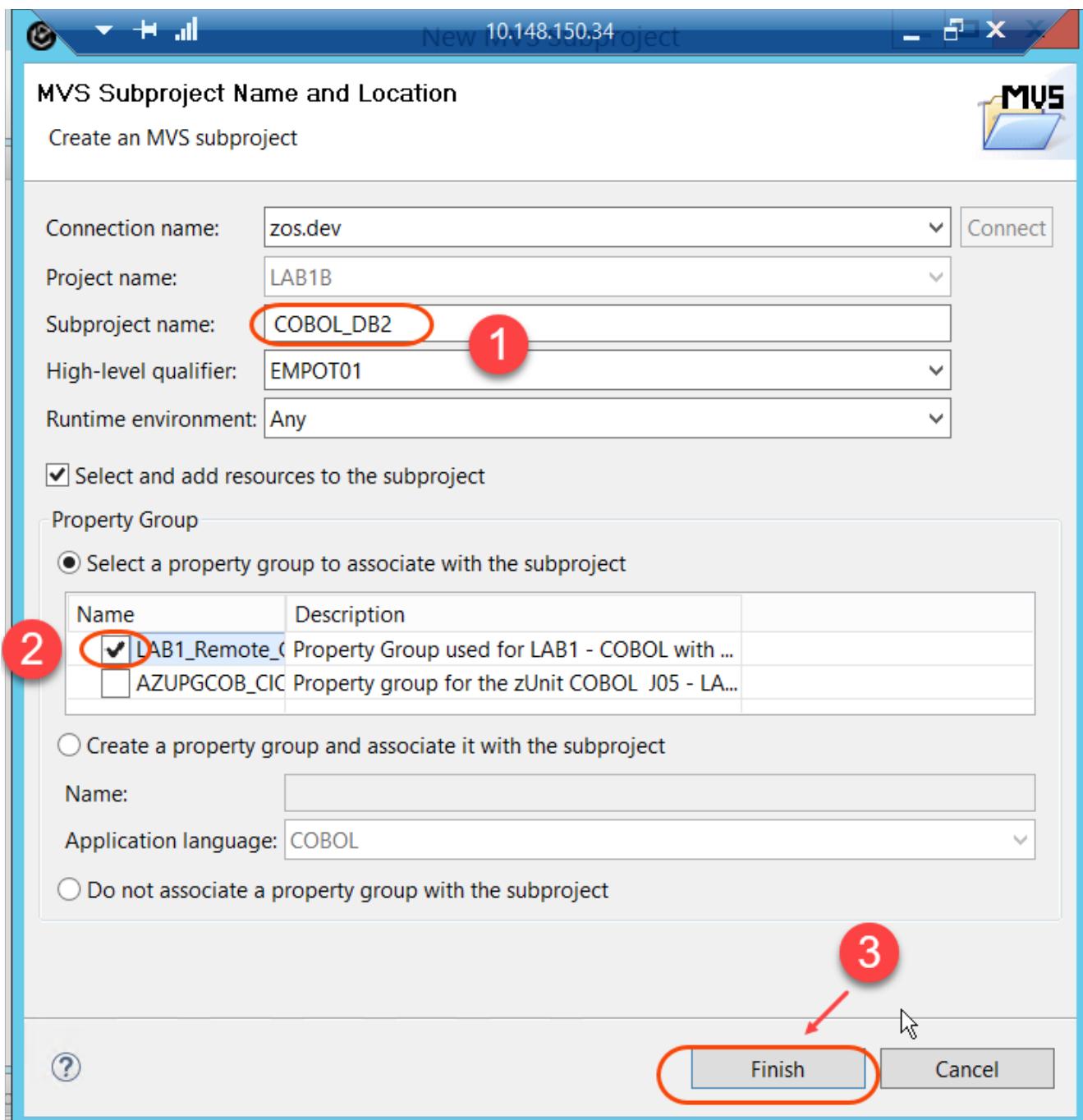


2.1.2 ►| Type **LAB1B** as the *Project name*, select **Create an MVS subproject** and click **Finish**.



2.1.3 ►| Type **COBOL_DB2** as *Subproject Name*, select **LAB1_Remote_Cobol..** as the property group and click **Finish**.

TIP: If using *Spanish or Brazilian* keyboard you must activate the correct language in the bottom, right corner, otherwise you will not be able to generate the “_”.



This dialog will be displayed:

Select the data sets or members you want to add to the subproject

- >  **EMPOT01.***
- >  **EMPOT05.***
- >  **IBMUSER.HCAZ.VTP.***
- >  **IBMUSER.HCPA.VTP***
- >  **IBMUSER.POT.***
- >  **Retrieved Data Sets**
- >  **My Data Sets**

Can't you find the property group above?

 It is because you are using a wrong workspace. Close IDz, go back to the step 1.1.1 and be sure you start IDz from the icon that is on the windows desktop.

- PROPERTY GROUP

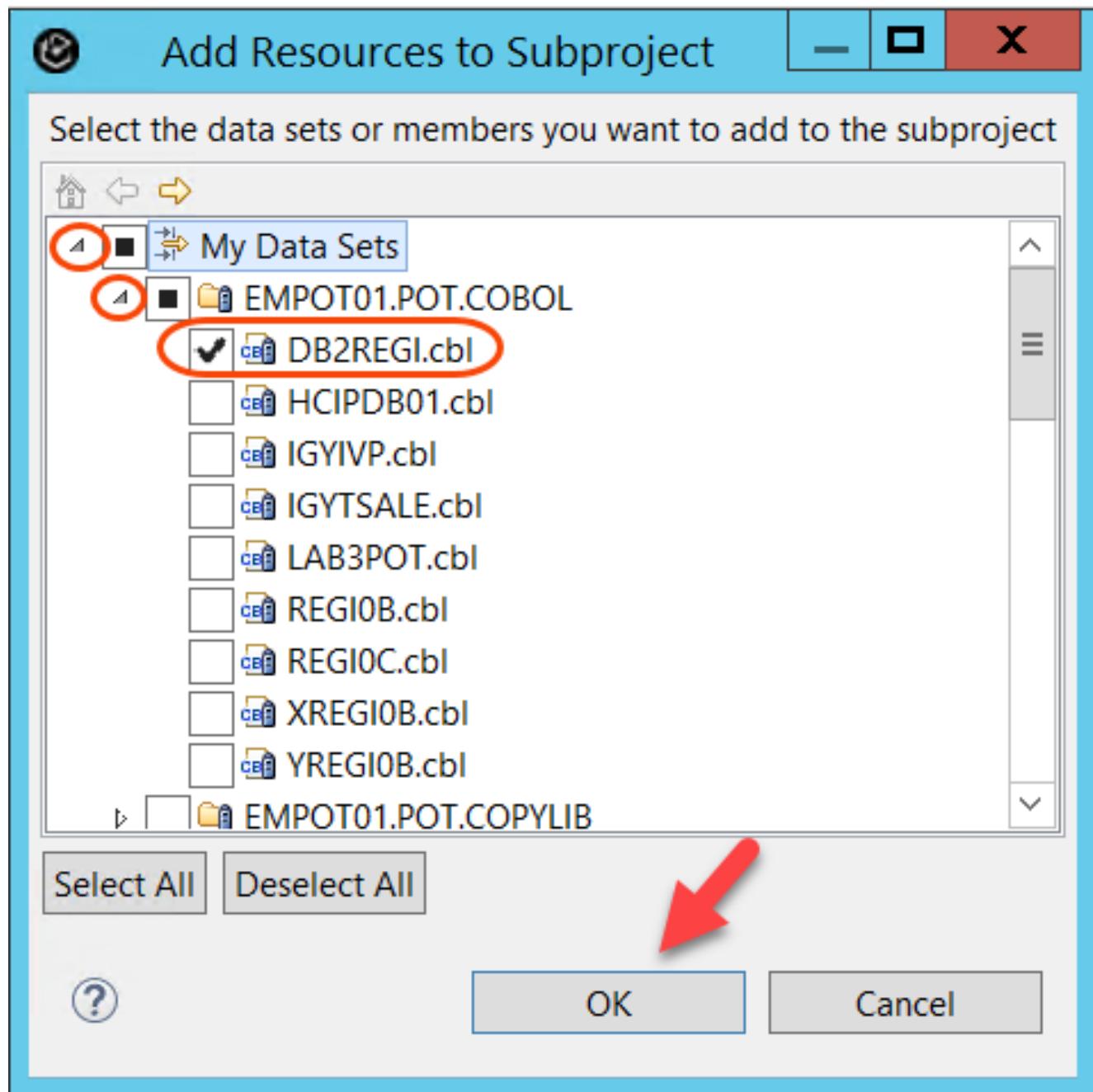
 *Property groups provide a way to manage resource properties, share them easily across systems, projects, resources, and users, and maintain consistency in your development and build environment.*

You can, for example, define a property group that points to copybooks. This is something like //STEP LIB; otherwise, some of the variables used in the program are not resolved.

2.2 Add z/OS resources to the MVS subproject

To make the data sets available to your remote project named *LAB1B*, you will need to add them. For this lab we just want to add one member..

2.2.1 ► On the dialog below, expand **My Data Sets**, **EMPOT01.POT.COBOL** , select **DB2REGL.cbl**, and click **OK**

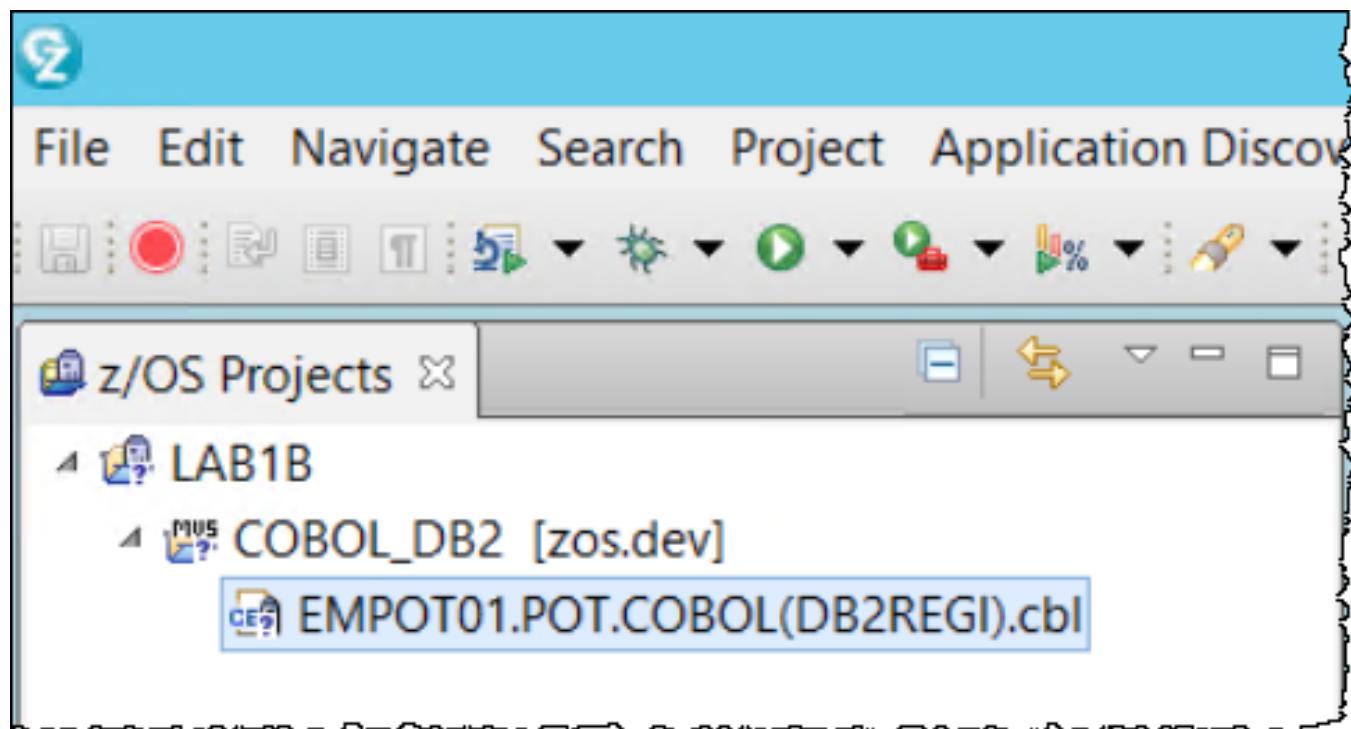


2.2.2 You should see a z/OS Project named **LAB1B** in your *z/OS Projects view*.

► Click **z/OS Projects view** (on left) and under **COBOL_DB2**, you will now see **DB2REGI** member added to your project.

Notice that creating a Remote project is a good practice to isolate the code you are working on, but you could work on your code without creating this project.

Our lab involves a small update so you could work directly on the *Remote System* perspective without having to create a Remote project.



2.3 Submit a JCL to execute the COBOL program

The JCL to execute the program is available on the local windows folder: "C :\ADF_POT\empot01". You will use this JCL to execute a batch COBOL/DB2 on z/OS.

2.3.1 ► Using *Remote System View* (on top and right), **scroll up** to locate the file **pot01run.jcl** under **Local/Local Files/ADF_POT/empot01** and **double click** to edit.

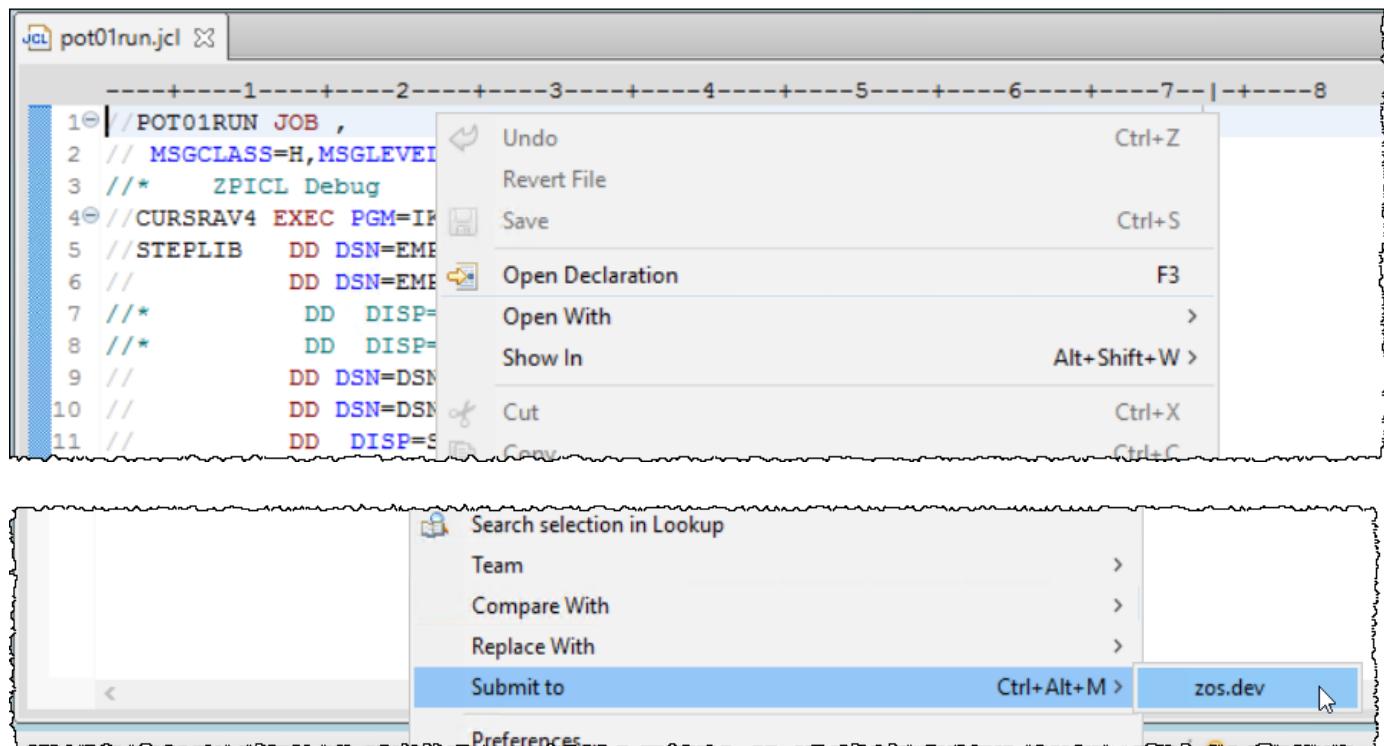
The screenshot shows a development environment with two main windows. On the left is a code editor window titled 'DB2REGI.cbl' containing a COBOL program. The program includes various JCL statements like 'POT01RUN', 'STEPLIB', and 'LIB'. A yellow callout bubble points to the text '(NOT pot01db.jcl)' in the code. On the right is a 'Remote Systems X' browser window showing a file tree under 'Local'. The 'pot01run.jcl' file is highlighted with a red border.

```

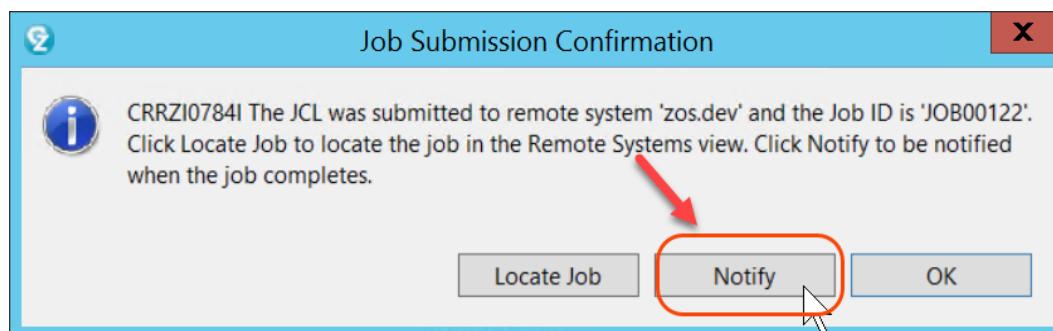
1 //> P01RUN JOB ,
2 // MSGCLASS=H,MSGLEVEL=(1,1),TIME=1440,REGION=70M,COND=(16,LT)
3 /* ZPICL Debug
4 // CURSAV4 EXEC PGM=IKJEFT01,DYNAMNBR=20
5 // STEPLIB DD DSN=EMPOT01.P01.LOAD,DISP=SHR
6 //          DD DSN=EMPOT.ZPICL.LOAD,DISP=SHR
7 /*          DD DISP=SHR,DSN=IBMUSER.VER1.SEQAMOD
8 /*          DD DISP=SHR,DSN=IBMUSER.VER1.SEQAUTH
9 //          DD DSN=DSNC10.SDSNLOAD,DISP=SHR
10 //         DD DSN=DSNC10.DBCG.RUNLIB.LOAD,DISP=SHR
11 //         DD DISP=SHR,DSN=ID2EE.V16R0.IEXP.SFEKAUTH
12 //SYSPRINT DD SYSOUT=*
13 //SYSOUT  DD SYSOUT=*
14 //CEE0OPTS DD *
15 TER(UADUMP)
16 /*
17 //SYSTSPRT DD SYSOUT=*
18 //SYSTSIN DD *
19 TSOLIB ACTIVATE DA('DSNC10.SDSNLOAD')
20 DSN SYSTEM(DBCG)
21 RUN PROGRAM(DB2REGI) PLAN(DB2REGI) -
22 LIB('EMPOT.ZPICL.LOAD')
23 END
24 /*
25 //

```

2.3.2 ►| Right click on the JCL being edited and select **Submit to → zos.dev**

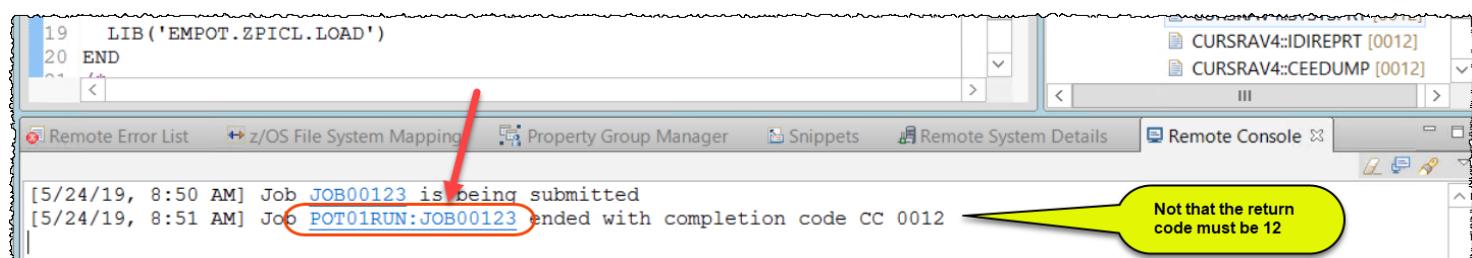


2.3.3 ►| When the dialog pop up appears, click **Notify**. This allows you to be notified when the execution is ended.



2.3.4 Under *Remote Console*, you will be notified when execution is completed.

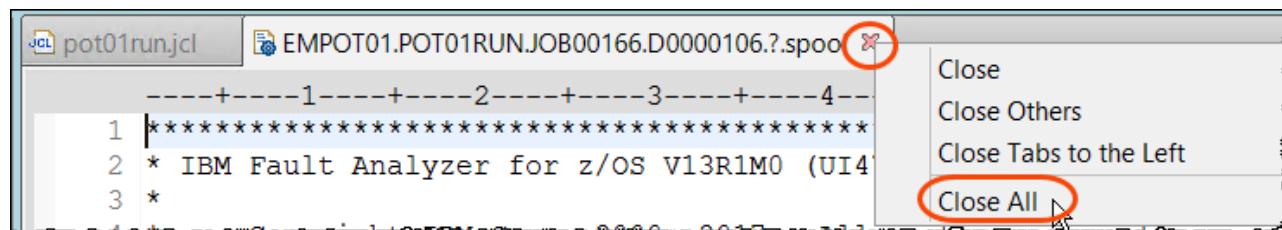
►| Once the execution ends, **click on the link POT01RUN:JOB00xxx** (where 00xxx is a number) to point to the Job output. This number vary depending on z/OS.



2.3.5 ► Under **Remote Systems** view **scroll down**, expand **JES > Retrieved Jobs > POT01RUN:JOB00xxx** and **double click CURSRAV4::IDIREPRT** step and you will see the *Fault Analyzer* report showing an **OCB ABEND**. Your mission is to fix that bug.

Tip: If there is no jobs under “Retrieved Jobs”, is because you did not click on link as stated at 2.3.4. You can also see this output once you right click on “My Jobs” under **JES** and select **Refresh**.

362.3.6 ► Close all the opened editors using **Ctrl + Shift + F4**,
Or right click on the and choose **Close all**.



Section 3. Use Fault Analyzer to identify the cause of the ABEND

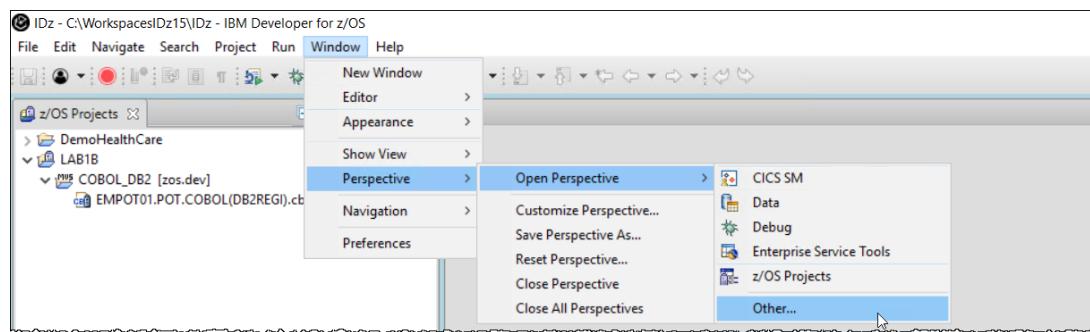
You will now take advantage of **IBM Fault Analyzer** (that is part of Application Delivery Foundation- ADF) to verify the cause of the ABEND.

What is IBM Fault Analyzer for z/OS?

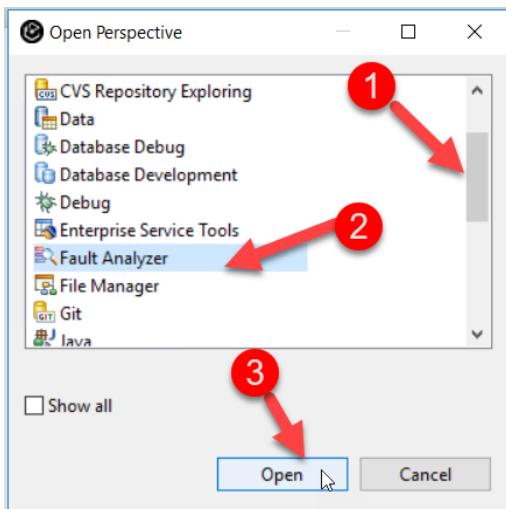
 *IBM Fault Analyzer for z/OS* helps developers analyze and fix system and application failures for CICS, WebSphere MQ, IMS and DB2 environments. When an application ends abnormally, Fault Analyzer is automatically engaged, gathering real-time information about the event and its environment at the time of failure. This helps the development team to identify the cause, analyze what went wrong and resolve the problem in a more timely and efficient manner to avoid costly interruptions that could jeopardize application schedules and outcomes.

3.1 Using Fault Analyzer perspective

3.1.1 ► Open the *Fault Analyzer* perspective using **Window > Perspective > Open Perspective > Other...**

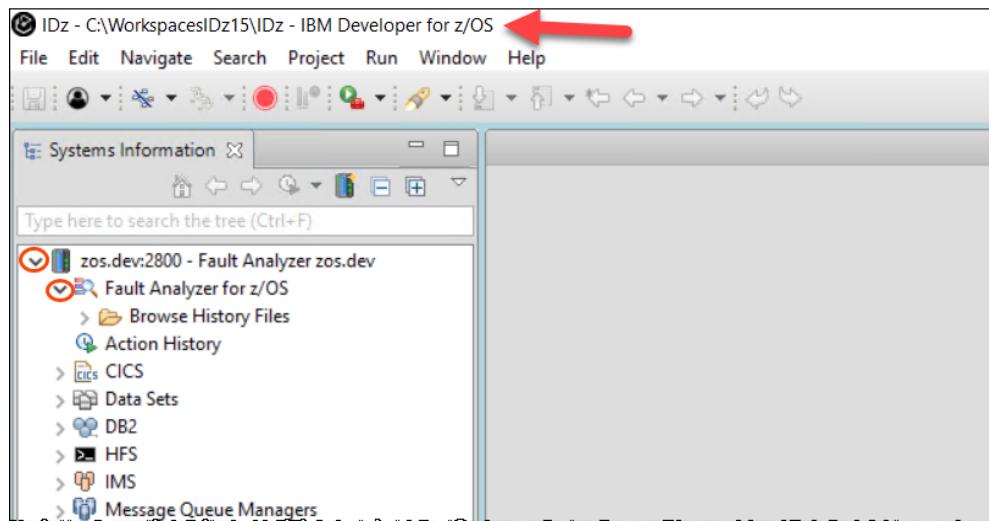


3.1.2 ► Scroll down, select **Fault Analyzer** and click **Open**

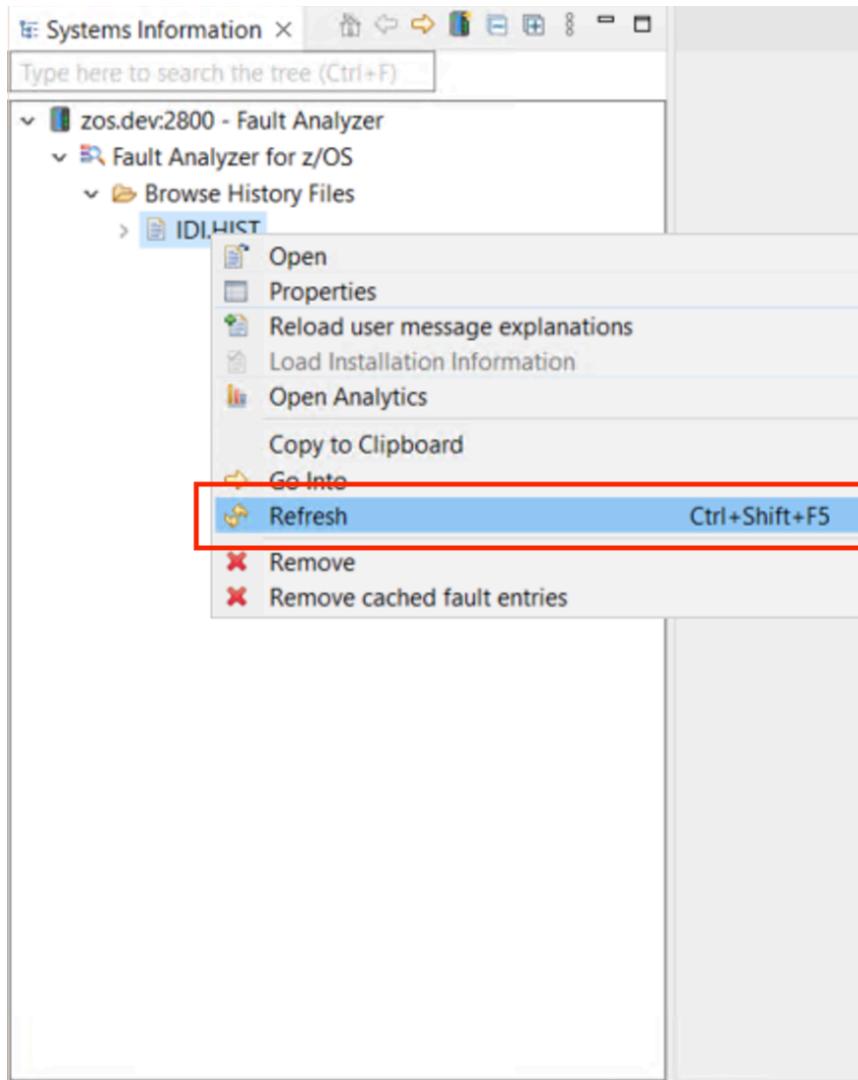


3.1.3 ► Under *Systems Information* view (on left) be sure that **Fault Analyzer for z/OS** is expanded

TIP: If you do not have this entry, it is because you are using a wrong IDz workspace. Please contact the instructor.



- 3.1.4 ►| Right click **Fault Analyzer for z/OS** and select **Refresh**
►| Expand **Browse History File**. You will see the folder **IDI.HIST**
►| Right click **IDI.HIST** and select **Refresh** (or **Ctrl + Shift + F5**)



- 3.1.5 ►| If you get a **Sign on** dialog for *Fault Analyzer* use **empot01** credentials and click **OK**
You see a list of *FAULT_IDs* on the z/OS system that you are connected to...

Systems Information X

Type here to search the tree (Ctrl+F) ...

- zos.dev2800 - Fault Analyzer
 - Fault Analyzer for z/OS
 - Browse History Files
 - IDI.HIST
 - F00131

Outline X ZOS.DEV : 2800/IDI.HIST X Lookup Markers

There is no active editor that provides an outline.

Filter Criteria: None

FAULT_ID	JOB/TRAN	USER_ID	SYS/IOB	ABEND	I_ABEND	JOB_ID	JOBNAME	USERNAME	DATE
> F00131	POT01RUN	EMPOT01	S0W1	S0CB	U4039	JOB00964	POT01RUN		2024/05/20
F00130	POT01RUN	EMPOT01	S0W1	S0CB	U4039	JOB00939	POT01RUN		2024/05/14
F00125	DB2CLLAB	IBMUUSER	S0W1	S0C1		JOB00877	DB2CLLAB		2024/05/13
F00124	RSED4	STCRSE	S0W1	SEC6	SEC6	STC00069	RSED4		2024/05/09
F00123	RSED5	STCRSE	S0W1	SEC6	SEC6	STC00069	RSED5		2024/05/09
F00122	IBMUUSER1	IBMUUSER	S0W1	S0B6	S0B6	JOB00780	IBMUUSER1		2024/05/09
F00121	IBMUUSER1	IBMUUSER	S0W1	S0B6	S0B6	JOB00774	IBMUUSER1		2024/05/09
F00119	RSED4	STCRSE	S0W1	SEC6	SEC6	STC00688	RSED4		2024/05/06
F00118	RSED8	STCRSE	S0W1	SEC6	SEC6	STC00693	RSED8		2024/05/06
F00117	RSED5	STCRSE	S0W1	SEC6	SEC6	STC00689	RSED5		2024/05/06
F00116	RSED5	STCRSE	S0W1	SEC6	SEC6	STC00691	RSED5		2024/05/06
F00115	BATCHCL	IBMUUSER	S0W1	S013	S013	JOB00597	BATCHCL		2024/05/03
F00114	RSED3	STCRSE	S0W1	SEC6	SEC6	STC00574	RSED3		2024/05/03
F00113	RSED7	STCRSE	S0W1	SEC6	SEC6	STC00579	RSED7		2024/05/03

3.1.6 ► On the **ZOS.DEV:2800/IDI.HIST** view locate the **latest** job name **POT01RUN** and **double click** on it. You will see the report being downloaded from the z/OS to your Windows client. The report below will be displayed

```

1  ****
2 * IBM Fault Analyzer for z/OS V15R1M02 (UI92187 2023/06/12)
3 *
4 *
5 * Copyright IBM Corp. 2000, 2017. All rights reserved.
6 * Copyright HCL Technologies Ltd 2017, 2023. All rights reserved.
7 ****
8
9 JOBNAME: POT01RUN SYSTEM ABEND: OCB S0W1 2024/05/20 15:16:21
10
11
12 NOTE: This report was saved after reanalysis of the current fault entry--it was
13 not generated during real-time processing.
14
15
16
17 Module DB2REGI, program DB2REGI, source line # 364: Abend S0CB (Decimal-Divide Exception)
18 SYNOPSIS
19
20
21 A system abend OCB occurred in module DB2REGI program DB2REGI at offset X'BCE'.
22
23 A program-interruption code 000B (Decimal-Divide Exception) is associated with
24 this abend and indicates that:
25
26 The divisor was zero in a signed decimal division.
27
28 The cause of the failure was program DB2REGI in module DB2REGI. The COBOL
29 source code that immediately preceded the failure was:
30

```

Main Report Event Details Abend Information System-Wide Information Miscellaneous									
ZOS.DEV: 2800/IDI.HIST Lookup Markers									
Filter Criteria: None									
Fault_ID	JOB/TRAN	USER_ID	SYS/JOB	ABEND	L_ABEND	JOB_ID	JOBNAME	USERNAME	DATE
> F00131	POT01RUN	EMPOT01	S0W1	S0CB	U4039	JOB00984	POT01RUN		2024/05/20
F00130	POT01RUN	EMPOT01	S0W1	S0CB	U4039	JOB00939	POT01RUN		2024/05/14
F00125	DB2CLLAB	IBMUSER	S0W1	S0C1	S0C1	JOB00877	DB2CLLAB		2024/05/13
F00124	RSED4	STCRSE	S0W1	SEC6	SEC6	STC0069	RSED4		2024/05/09
F00123	RSED5	STCRSE	S0W1	SEC6	SEC6	STC0069	RSED5		2024/05/09
F00122	IBMUSER1	IBMUSER	S0W1	S0B6	S0B6	JOB00780	IBMUSER1		2024/05/09
F00121	IBMUSER1	IBMUSER	S0W1	S0B6	S0B6	JOB00774	IBMUSER1		2024/05/09
F00119	RSED4	STCRSE	S0W1	SEC6	SEC6	STC00688	RSED4		2024/05/06
F00118	RSED8	STCRSE	S0W1	SEC6	SEC6	STC00693	RSED8		2024/05/06
F00117	RSED5	STCRSE	S0W1	SEC6	SEC6	STC00689	RSED5		2024/05/06
F00116	RSED5	STCRSE	S0W1	SEC6	SEC6	STC00691	RSED5		2024/05/06
F00115	BATCHCL	IBMUSER	S0W1	S013	S013	JOB00597	BATCHCL		2024/05/03
F00114	RSED3	STCRSE	S0W1	SEC6	SEC6	STC00574	RSED3		2024/05/03
F00113	RSED7	STCRSE	S0W1	SEC6	SEC6	STC00579	RSED7		2024/05/03

3.1.7 ► Scroll the report down and you will see that the field **RECEIVED-FROM-CALLED** used on the division has "0". So the abend is because of a divide by zero.

► Also notice that there are other tabs on this panel (like Event Details, Abend Information, etc.). You may try those tabs to get more information about the abend.

Systems Information X Type here to search the tree (Ctrl+F)

ZOS.DEV:2800/IDI.HIST(F00131)-Report X

```

22
23 A program-interruption code 000B (Decimal-Divide Exception) is associated with
24 this abend and indicates that:
25
26 The divisor was zero in a signed decimal division.
27
28 The cause of the failure was program DB2REGI in module DB2REGI. The COBOL
29 source code that immediately preceded the failure was:
30
31 Source
32 Line #
33 -----
34 000364 DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
35
36 The COBOL source code for data fields involved in the failure:
37
38 Source
39 Line #
40 -----
41 000199 03 RECEIVED-FROM-CALLED PIC 99.
42 000200 03 VALUE1 PIC 99.
43 000205 03 RESULT PIC 99.
44
45 Data field values at time of abend:
46
47 RECEIVED-FROM-CALLED = 0
48 RESULT = X'0000'
49 VALUE1 = 66
50
51

```

Main Report Event Details Abend Information System-Wide Information Miscellaneous

3.1.8 ► Using the *Main Report* view Click on the blue link
This will show the COBOL statement that caused the *OCB abend*

Systems Information X Type here to search the tree (Ctrl+F)

ZOS.DEV:2800/IDI.HIST(F00131)-Report X

```

21 A system abend OCB occurred in module DB2REGI program DB2REGI at offset X'BCE'.
22
23 A program-interruption code 000B (Decimal-Divide Exception) is associated with
24 this abend and indicates that:
25
26 The divisor was zero in a signed decimal division.
27
28 The cause of the failure was program DB2REGI in module DB2REGI. The COBOL
29 source code that immediately preceded the failure was:
30
31 Source
32 Line #
33 -----
34 000364 DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
35
36 The COBOL source code for data fields involved in the failure:
37
38 Source
39 Line #
40 -----
41 000199 03 RECEIVED-FROM-CALLED PIC 99.
42 000200 03 VALUE1 PIC 99.
43 000205 03 RESULT PIC 99.
44
45 Data field values at time of abend:
46
47 RECEIVED-FROM-CALLED = 0
48 RESULT = X'0000'
49 VALUE1 = 66
50

```

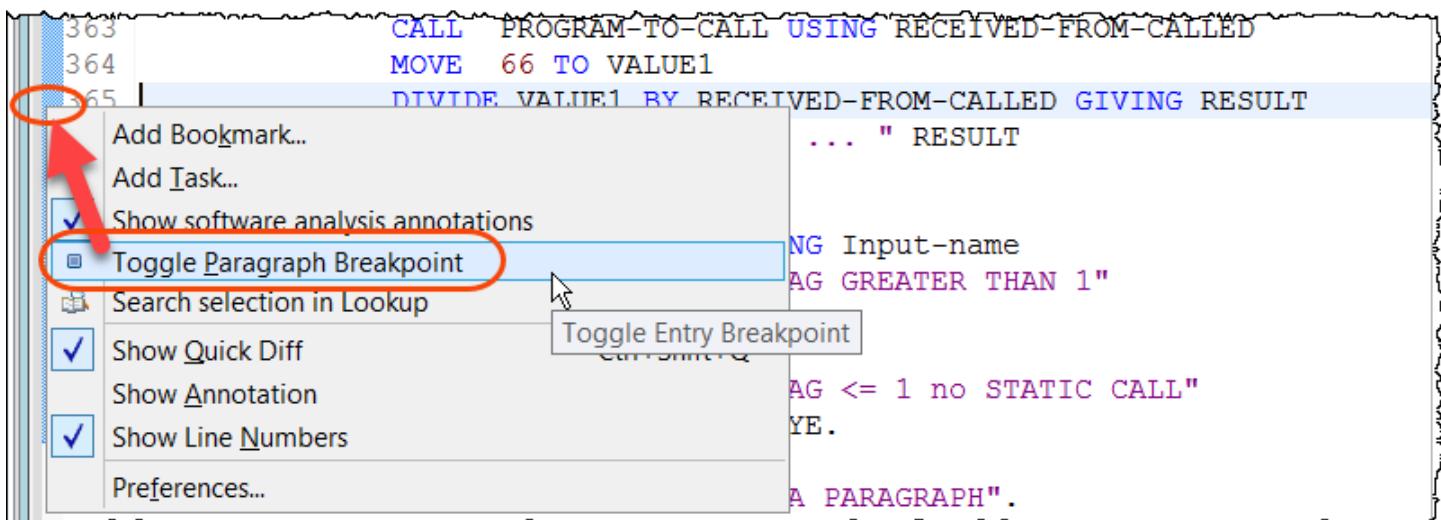
3.1.9 The program editor shows the line with the statement that is causing the abend.
Notice that this is NOT the COBOL source code. This what is on the "minidump" downloaded by Fault Analyzer.
There is no sense to make changes here. But you will see what caused the abend.

ZOS.DEV:2800/IDID10.HIST(F00327)-Report DB2REGI.COB

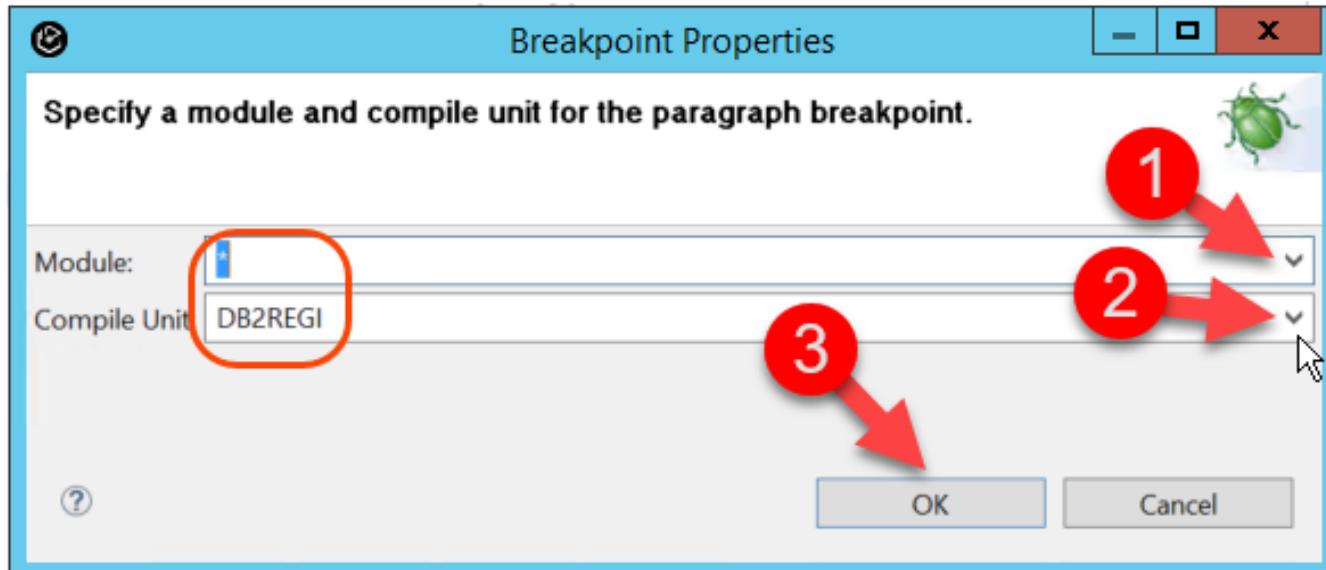
```
-----*A-1-B---+---2---+---3---+---4---+---5---+---6---+---7
      MOVE "LAB2" to WHICH-LAB.
359  520-LOGIC.
360      IF WHICH-LAB = 'LAB2'
361      * If is LAB2 lets do a dynamic CALL.. and force a divide by ZERO
362          MOVE "REG10B" TO PROGRAM-TO-CALL
363          CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
364          MOVE 66 TO VALUE1
365          DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
366          DISPLAY "The result is ... " RESULT
367      END-IF
368      IF BRANCHFLAG > 1
369          CALL 'REG10C' USING Input-name
370          DISPLAY "BRANCHFLAG GREATER THAN 1"
371          PERFORM 530-SEEEYA
372      ELSE
373          DISPLAY "BRANCHFLAG <= 1 no STATIC CALL"
374          PERFORM 540-GOODBYE.
375  530-SEEEYA.
376          DISPLAY "EXECUTED SEEYA PARAGRAPH".
```

Tip: If the font is not clear on the editor, it is because we increased the font to 150%. To make this smaller start the Control Panel > Display > set a custom scaling level > reduce to 125% (we did set to 150%).

3.1.10 ► On line “ DIVIDE VALUE1 BY RECEIVED-FROM-CALLED ... ” , right-click in the ruler area (the blue line on left) and select **Toggle Paragraph Breakpoint**. (also known as Toggle Entry Breakpoint)



- 3.1.11 ► On the Breakpoint Properties dialog use drop down to select *, and select or type **DB2REGI** as Compile Unit and click **OK**



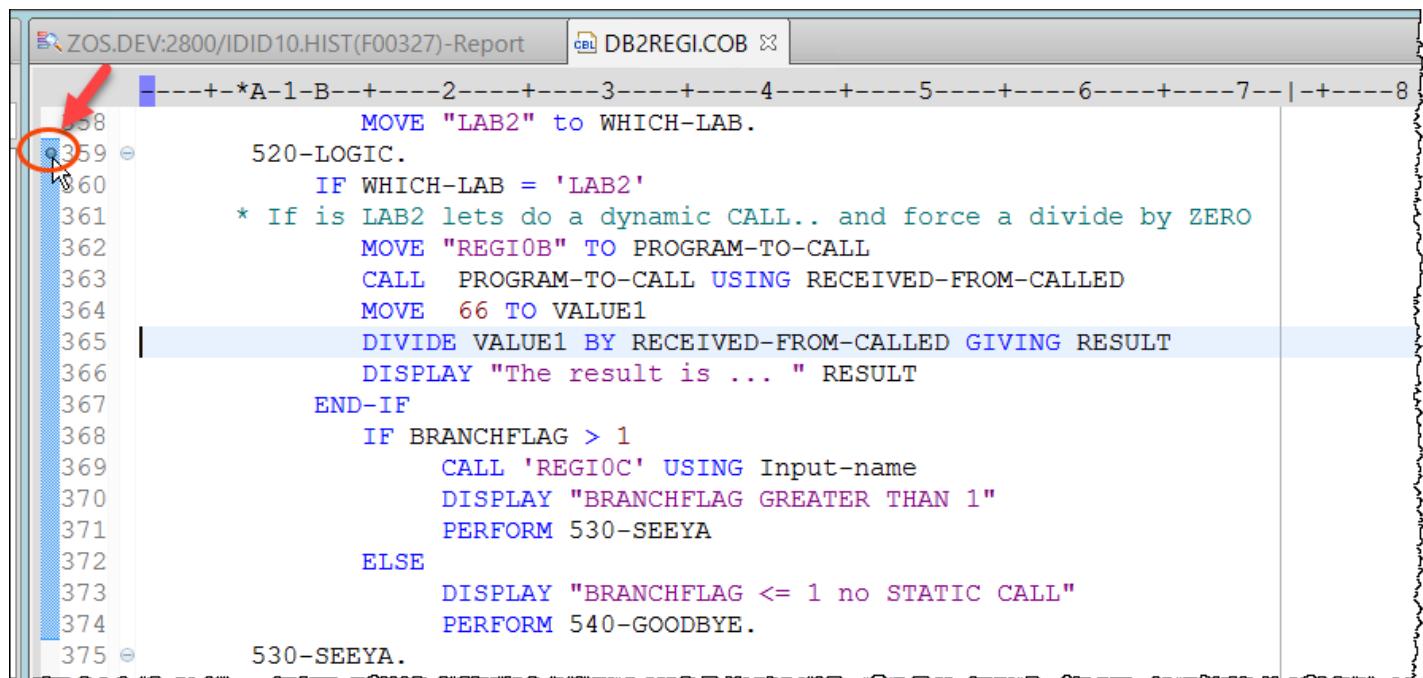
What is *Toggle Paragraph Breakpoint*?

 Toggle Paragraph Breakpoint provides the ability to set paragraph breakpoints prior to starting a debug session. Because these breakpoints are set using the original source files, they persist between debug sessions.

On previous versions, breakpoints could only be set at the level of the generated program listing files.

This allows a more natural edit, compile and debug workflow on the original source files.

- 3.1.12 ► Scroll up a bit. The *Toggle Paragraph Breakpoint* is set at the **520-LOGIC** paragraph. This break point may optionally be used when debugging the COBOL code.



```
--+--*A-1-B---+---2---+---3---+---4---+---5---+---6---+---7---+---8
      MOVE "LAB2" to WHICH-LAB.
  520-LOGIC.
      IF WHICH-LAB = 'LAB2'
      * If is LAB2 lets do a dynamic CALL.. and force a divide by ZERO
          MOVE "REGI0B" TO PROGRAM-TO-CALL
          CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
          MOVE 66 TO VALUE1
          DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
          DISPLAY "The result is ... " RESULT
      END-IF
      IF BRANCHFLAG > 1
          CALL 'REGI0C' USING Input-name
          DISPLAY "BRANCHFLAG GREATER THAN 1"
          PERFORM 530-SEEEYA
      ELSE
          DISPLAY "BRANCHFLAG <= 1 no STATIC CALL"
          PERFORM 540-GOODBYE.
  530-SEEEYA.
```

3.1.13 ► Close all opened editors using **CTRL+ Shift + F4**.

Or just click on the  of each opened editor.

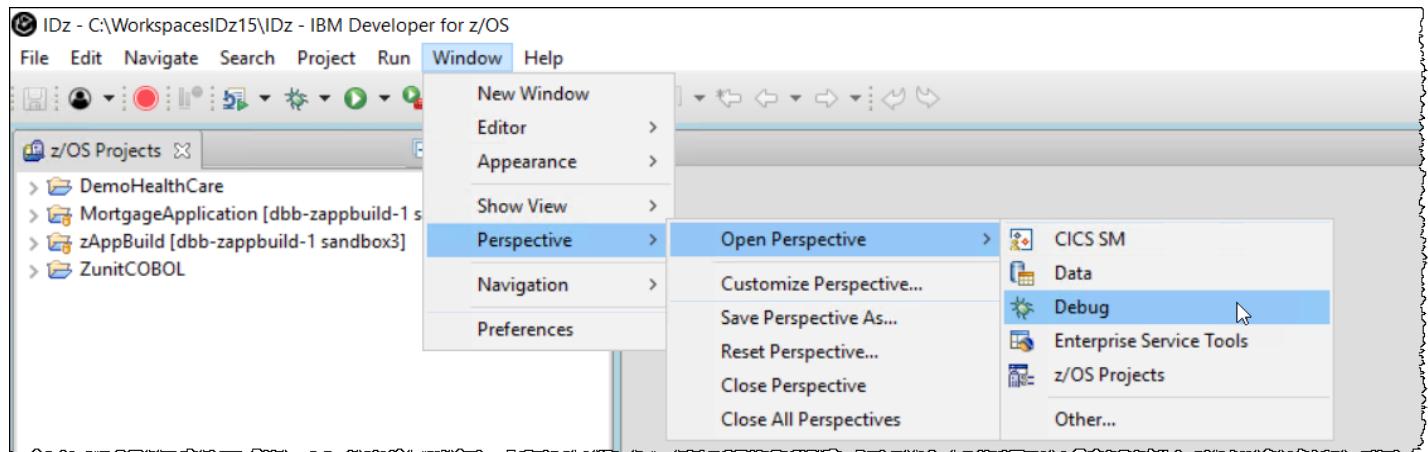
Section 4. Using the IBM z/OS Debugger for a temporary fix

You will use the Debug to verify the ABEND and make a runtime fix.

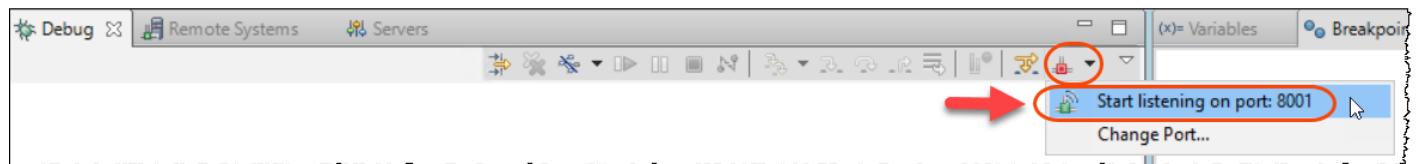
4.0 Be sure that IDz client is listening on port 8001

Usually this step is NOT required, but in our cloud environment we have timeouts that force ports to be closed.

4.0.1 ► Go to the *Debug Perspective* by selecting **Windows > Perspective > Open perspective > Debug**



4.0.2 ► If the icon is red, click and select **Start listening on port 8001**



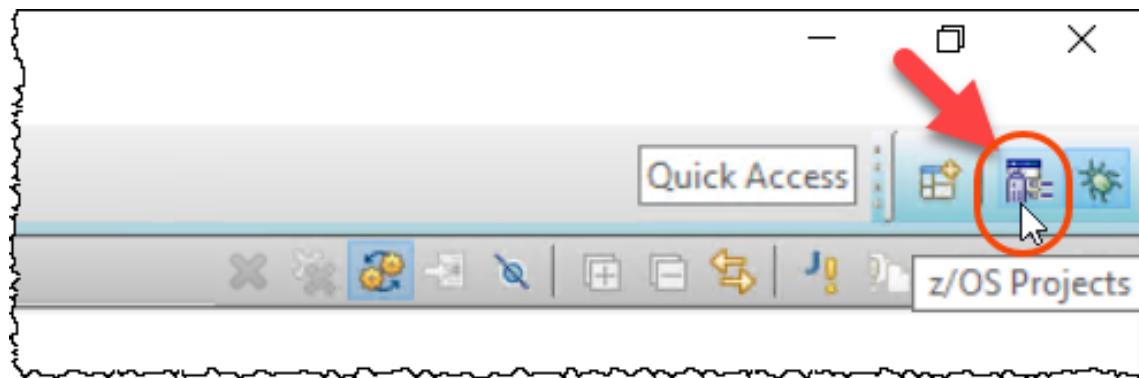
4.0.3 The listening icon will turn green and the IDz client is listening on port 8001.



4.1 Submit the JCL to invoke the Debug

You can now submit the JCL to execute again with the Debug option.

- 4.1.1 ► Using the top right corner Go to the **z/OS Projects** Perspective by clicking on the icon below.



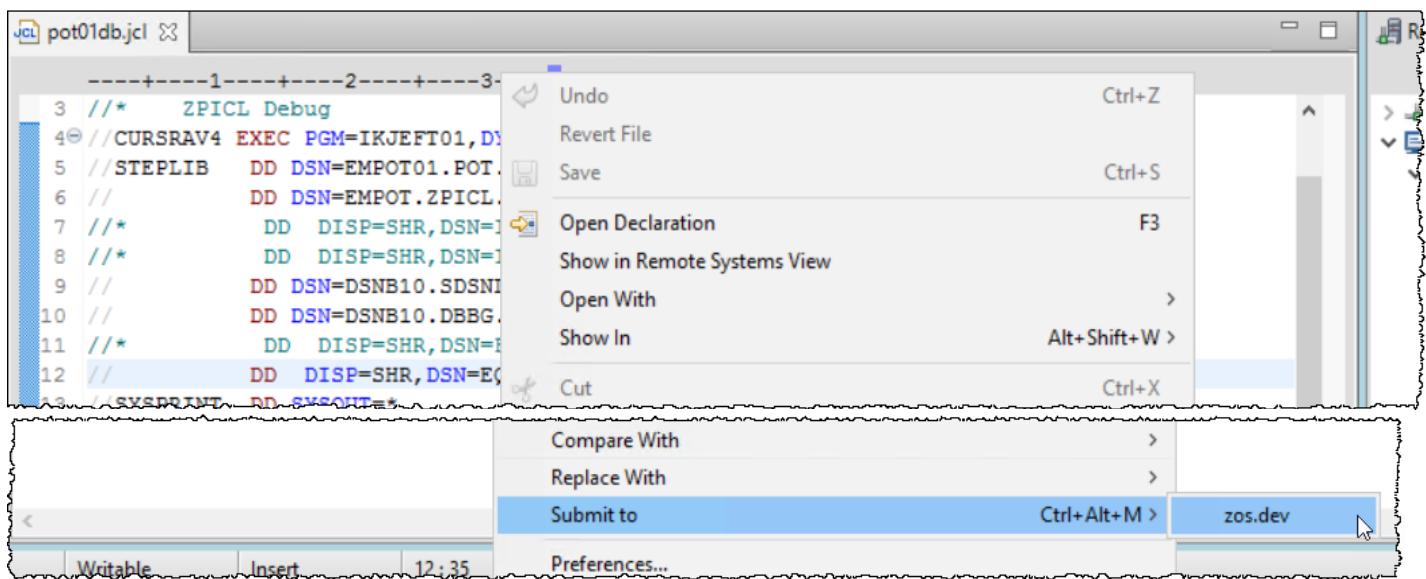
- 4.1.2 ► Using **Remote System View**, scroll up to locate the file **pot01db.jcl** under **Local/Local Files/ADF_POT/empot01** and double click to edit..

```

pot01db.jcl x
-----+-----+-----+-----+-----+-----+-----+-----+
1 //POT0101 JOB , 00000100
2 // MSGCLASS=H,MSGLEVEL=(1,1),TIME=1440,REGION=70M,COND=(16,LT) 00000200
3 /* ZPICL Debug 00000300
4 //CURSRAV4 EXEC PGM=IKJEFT01,DYNAMNBR=20 00000400
5 //STEPLIB DD DSN=EMPOT01.POT.LOAD,DISP=SHR 00000500
6 // DD DSN=EMPOT.ZPICL.LOAD,DISP=SHR 00000600
7 /* DD DISP=SHR,DSN=IMBUSER.VER1.SEQAMOD 00000700
8 /* DD DISP=SHR,DSN=IMBUSER.VER1.SEQAUTH 00000800
9 // DD DSN=DSNC10.SDSNLOAD,DISP=SHR 00000900
10// DD DSN=DSNC10.DBCG.RUNLIB.LOAD,DISP=SHR 00001000
11// DD DISP=SHR,DSN=IDZEE.V1GRO.IEXP.SFEKAUTH 00001100
12// DD DISP=SHR,DSN=ISM330.SEQAMOD 00001200
13// DD DISP=SHR,DSN=IDZEE.V1GRO.IDBG.SEQAMOD 00001300
14//SYSPRINT DD SYSOUT=*
15//SYSOUT DD SYSOUT=*
16//SYSPRINT DD SYSOUT=*
17//SYSTIN DD *
18 TSOLIB ACTIVATE DA('DSNC10.SDSNLOAD')
19 DSN SYSTEM(DBCG)
20 RUN PROGRAM(DB2REGI) PLAN(DB2REGI) -
21 LIB('EMPOT.ZPICL.LOAD')
22 END
23/*
24 //***** Launch Debug Tool
25 //CEEOPTS DD *
26 TEST(,,,DEMGT%EMPOT01:*)
27 /*
28 <

```

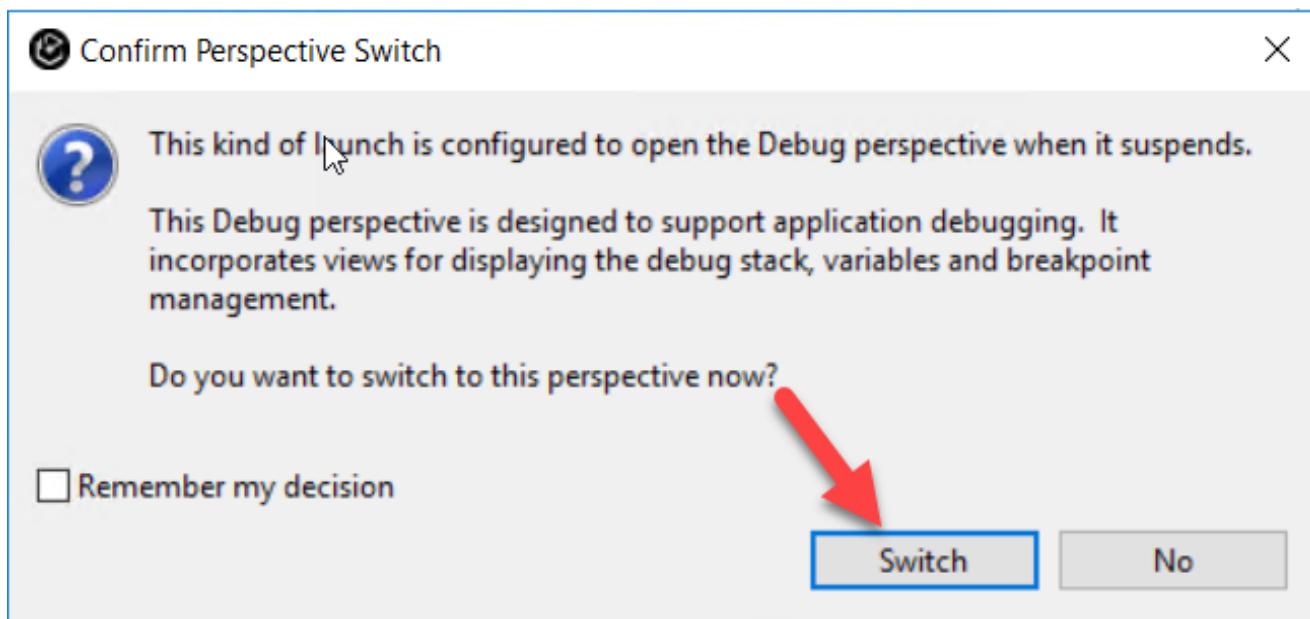
4.1.3 ►| Right click on the JCL being edited and select **Submit to > zos.dev**



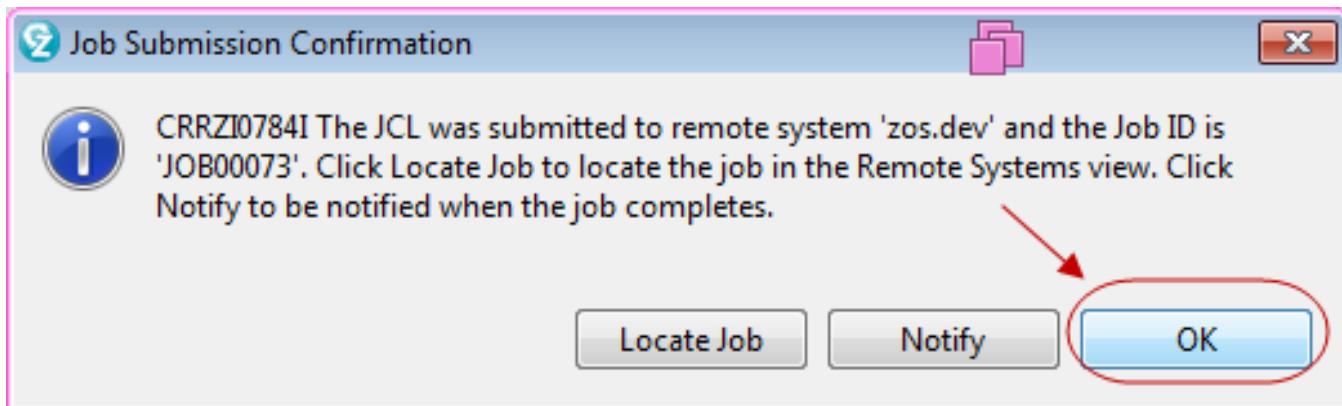
4.1.4 Once the job starts the z/OS debug will “talk” with you. Notice that the communication will be via the IDz connection (RSE), you don’t need to specify IP addresses. This may work even when firewalls are in place.

►| Click **Switch** to open the *Debug Perspective*

Notice: Depending how fast are you the dialogs order here could be different.

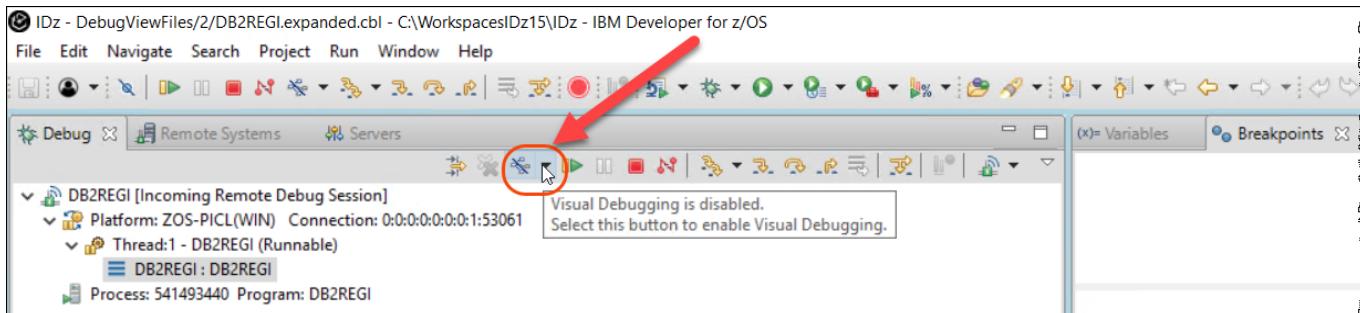


4.1.5 ► Also click **OK** for this dialog below



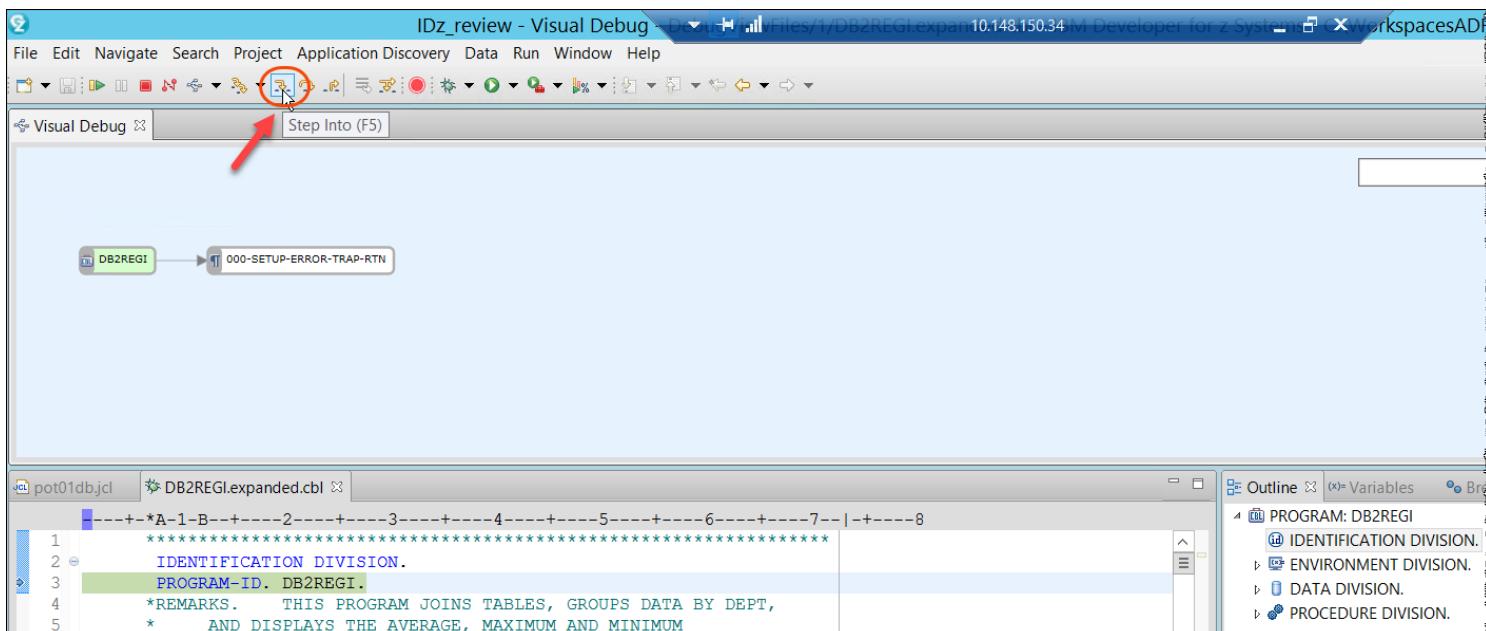
4.1.6 Using the *Debug* perspective:

► Click the icon to enable Visual Debugging which shows the **Visual Debug** view. If a dialog pops up click **YES**.



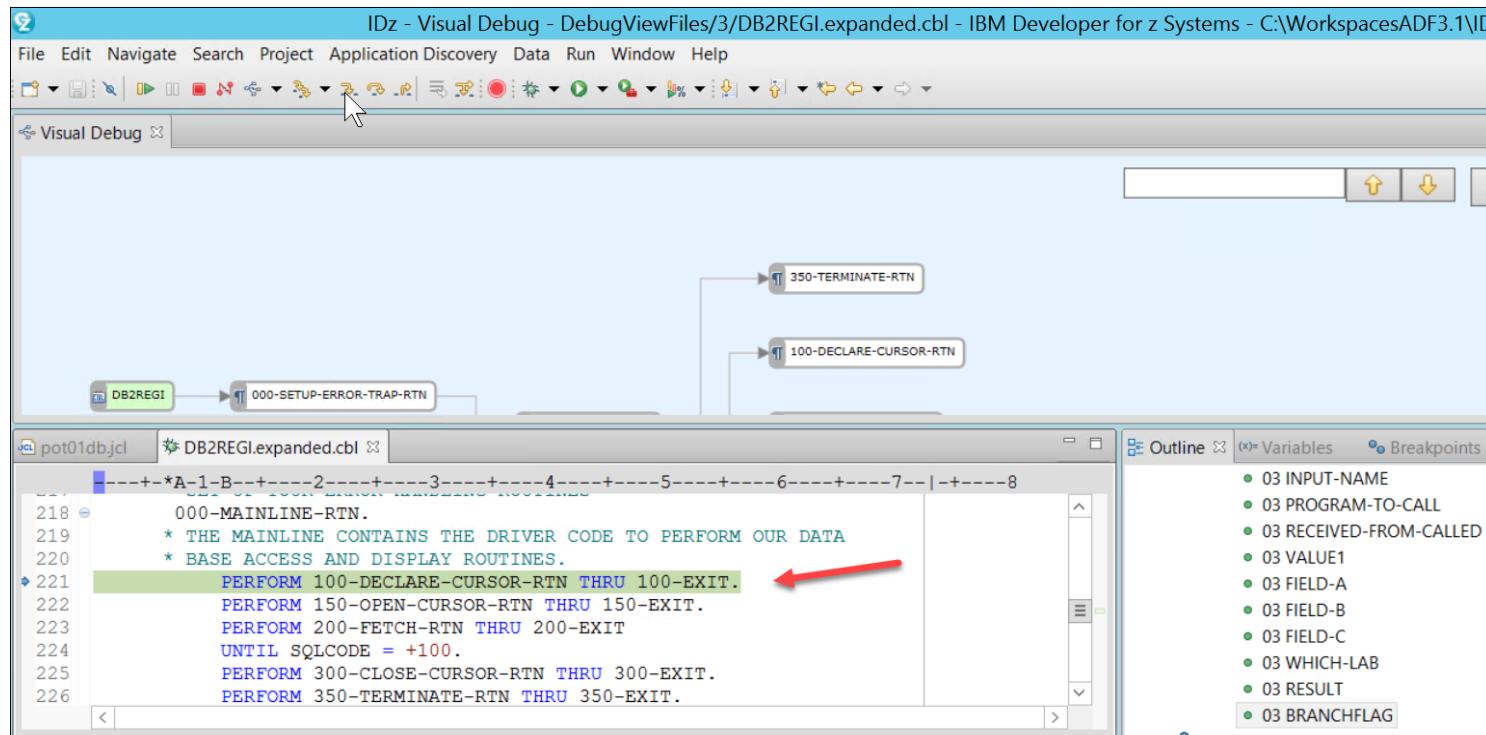
4.1.7 The *Visual Debug* view show the paragraphs being executed (in the top)

► Click the icon or **F5** (*Step into*) to execute first line of code.



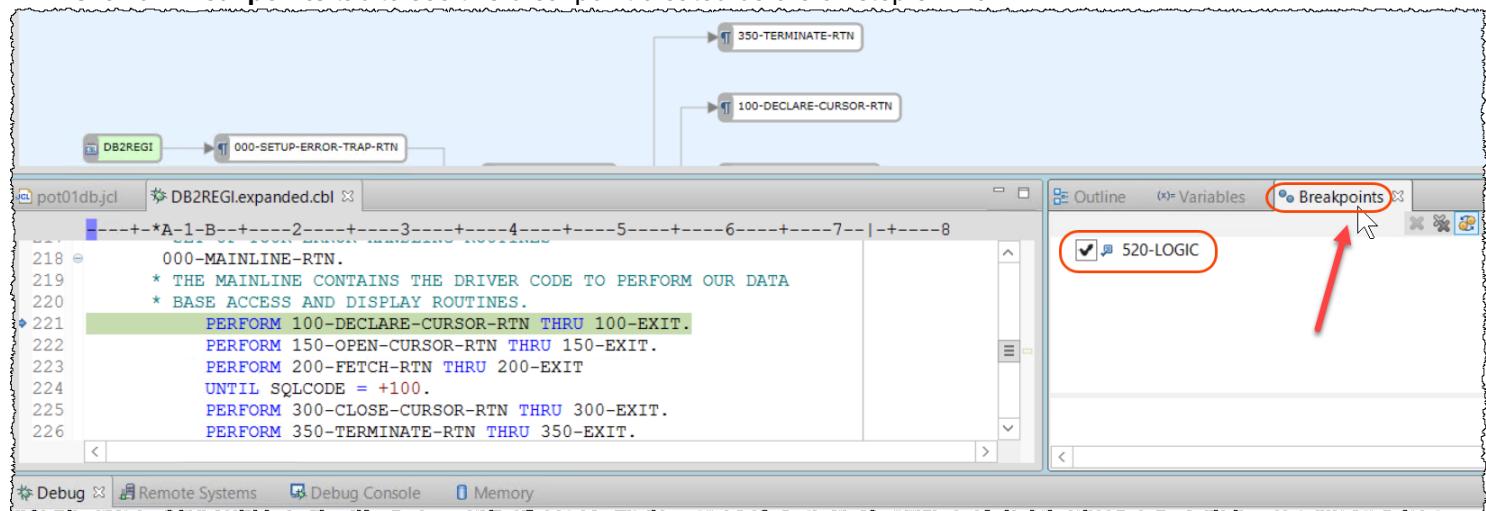
4.1.8 Press F5 or until this the statement

```
PERFORM 100-DECLARE-CURSOR-RTN THRU 100-EXIT
```



4.1.9 On the step 3.1.10 when using the *Fault Analyzer*, you created a *paragraph breakpoint* even without having the execution started. This is handy since it will show the area that needs to be debugged..

▶ Click on **Breakpoints** tab to see this breakpoint created before on step 3.1.10.



What is the Visual Debugging?

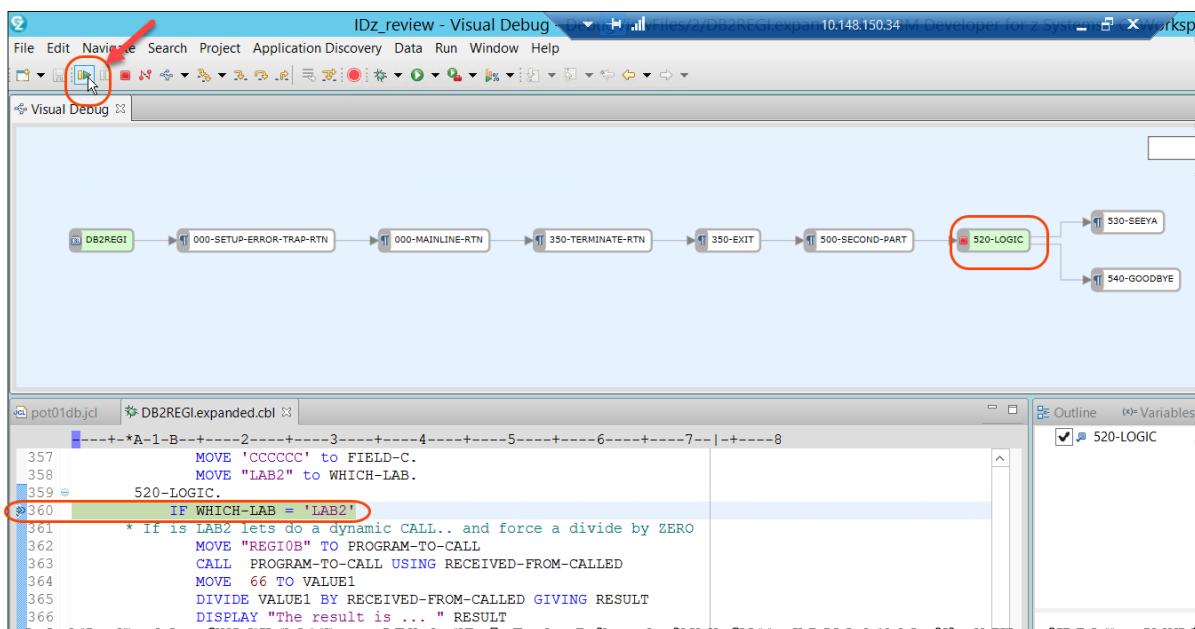


Visual debugging allows you to interact with your COBOL or PL/I debug session using the program control flow diagram.

With this diagram, you can visualize the stack trace, set breakpoints, and run to a selected call path. In COBOL, the stack trace represents the paragraph call chain.

In PL/I, the stack trace represents the procedure call chain.

- 4.1.10 ► Click on or press **F8** and notice that the execution will stop on the line since a *Paragraph Entry Breakpoint* was created before. This line is about to be executed.



- 4.1.11. As you verified using Fault Analyzer, the abend occurred on line “ **DIVIDE VALUE1 BY RECEIVED-FROM-CALLED ..** ” where you had a divide by zero

(see step 3.1.10). Now you will add a breakpoint on this line and change the values to avoid the abend.

- On the COBOL editor move the mouse to the **blue column** on left and **double click** on the line **DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT** to create a breakpoint.

```

358      MOVE "LAB2" to WHICH-LAB.
359  520-LOGIC.
360  IF WHICH-LAB = 'LAB2'
361    * If is LAB2 lets do a dynamic CALL.. and force a divide by ZERO
362    MOVE "REG10B" TO PROGRAM-TO-CALL
363    CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
364    MOVE 66 TO VALUE1
365    DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
366    DISPLAY "The result is ... " RESULT

```

4.1.12 Notice that a small circle is shown on the left of line “ DIVIDE VALUE1 BY RECEIVED-FROM-CALLED .. ”.

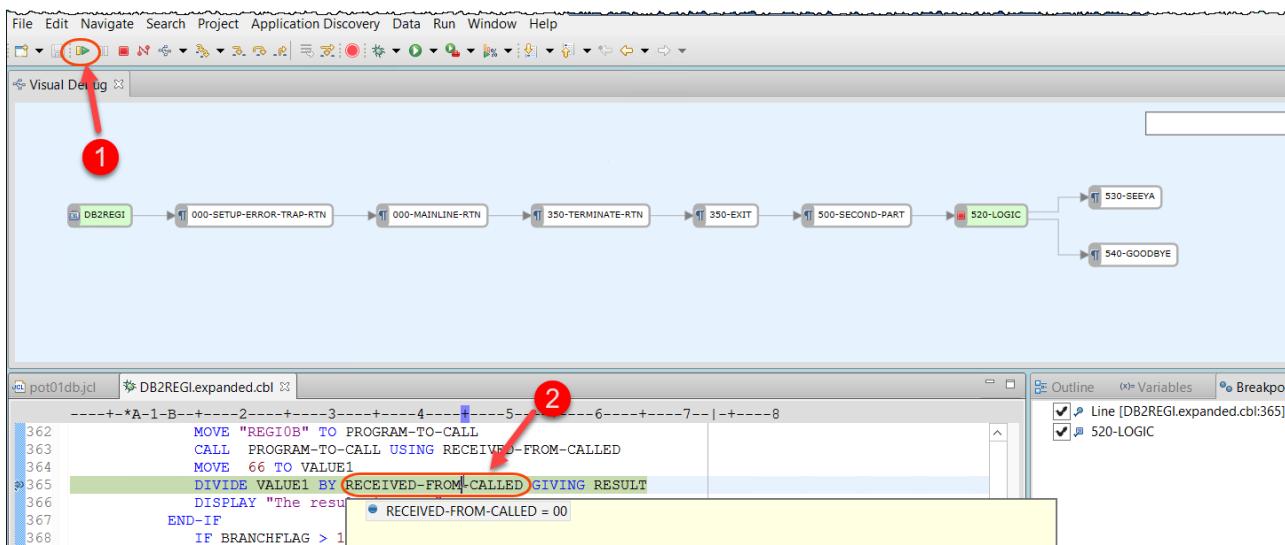
4.1.13 Continue by clicking on or using F5 until you will see that a program named REG10B is called and this program is returning a value of 0.

4.1.14 Click on or press F8. The execution will stop at the breakpoint that you created (line DIVIDE VALUE1 BY RECEIVED-FROM-CALLED).

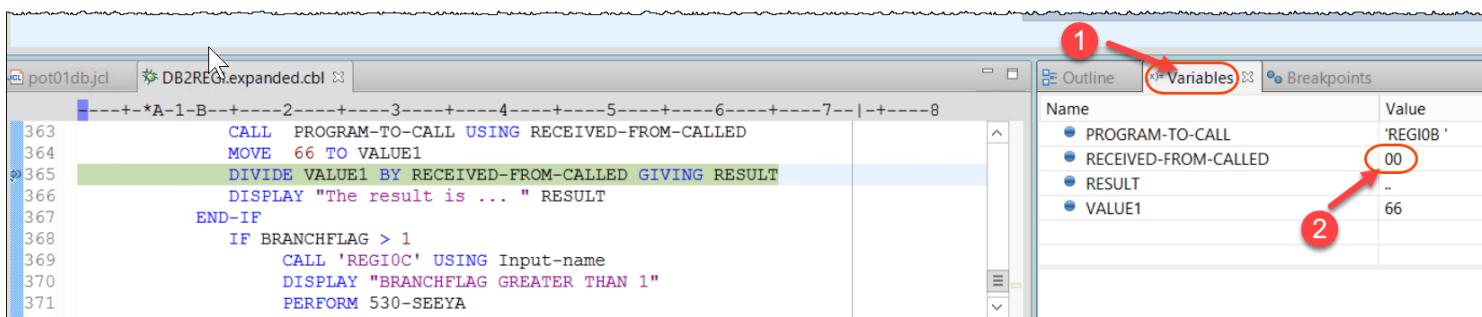
....).

► ② Move the cursor to **RECEIVED-FROM-CALLED** variable and click to see the **00** value. .

(Tip: under the cloud instance the behavior may act different and you may need to click on the editor area before moving the mouse to that field).

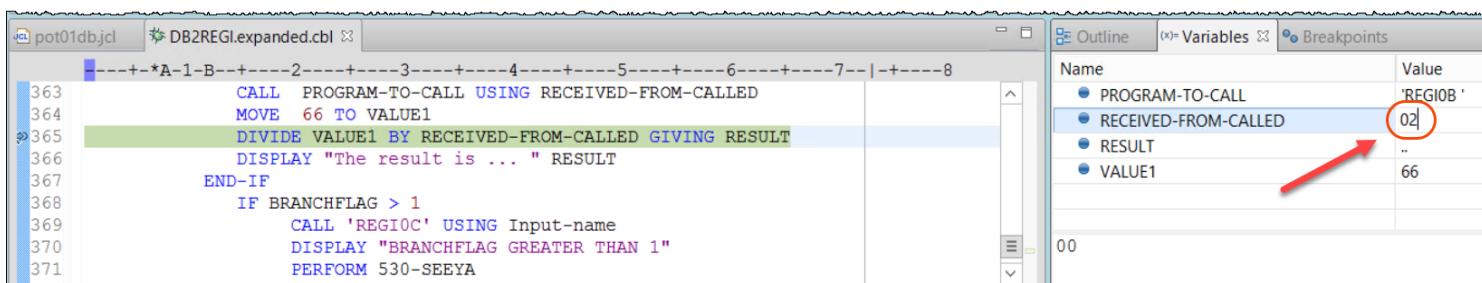


4.1.15 ► ① Click the **Variables** tab (right) and ② verify that **RECEIVED-FROM-CALLED** is **00**.



4.1.16 ► This is the abend cause. You must change the value to something different than 0.

► Click on **RECEIVED-FROM-CALLED** value of **00** and modify to **2**, just overtyping and press **enter**.



4.1.17 ► Again, move the cursor to **RECEIVED-FROM-CALLED** variable and now see the value **02**.

(Tip: under the cloud instance the behavior may act different and you may need to click on the editor area before moving the mouse to that field).

DB2REGI.expanded.cbl

```

363      CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
364      MOVE 66 TO VALUE1
365      DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
366      DISPLAY "The result is ... " RESULT
367      END-IF
368      IF BRANCHFLAG > 1

```

Outline Variables Breakpoints

Name	Value
PROGRAM-TO-CALL	REGI00B
RECEIVED-FROM-CALLED	02
RESULT	..
	66

4.1.18 ➡ Click the icon (Step into) or press F5 to see the next line being executed
This time the divide by 2 give you a result of 33.

DB2REGI.expanded.cbl

```

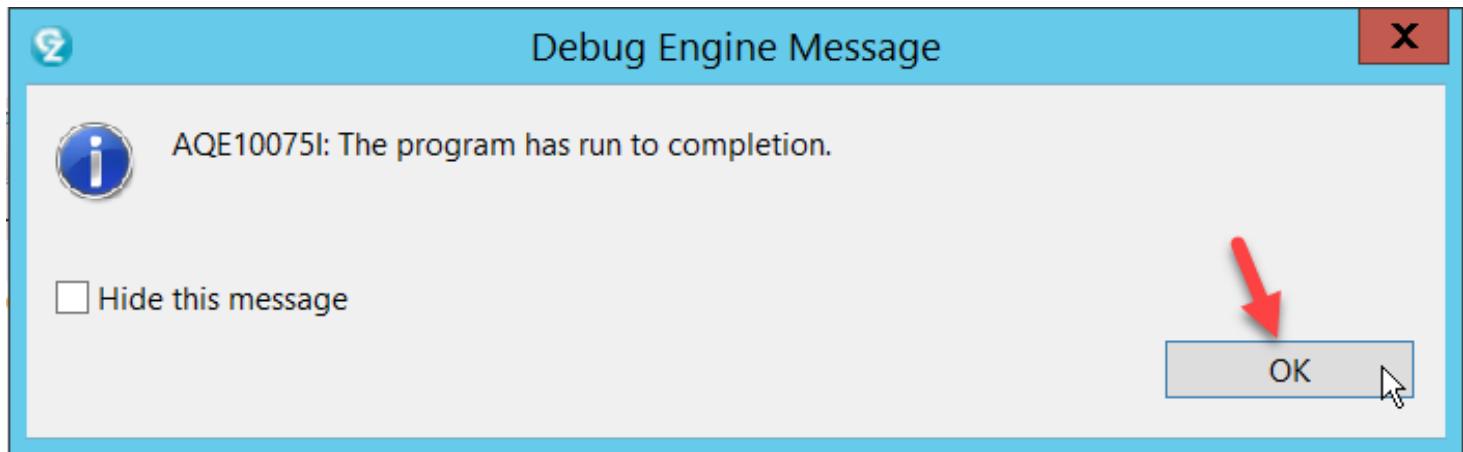
363      CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
364      MOVE 66 TO VALUE1
365      DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
366      DISPLAY "The result is ... " RESULT
367      END-IF
368      IF BRANCHFLAG > 1
369          CALL 'REGI0C' USING Input-name
370          DISPLAY "BRANCHFLAG GREATER THAN 1"
371          PERFORM 530-SEEEYA

```

Outline Variables Breakpoints

Name	Value
BRANCHFLAG	02
RECEIVED-FROM-CALLED	02
RESULT	33
VALUE1	66

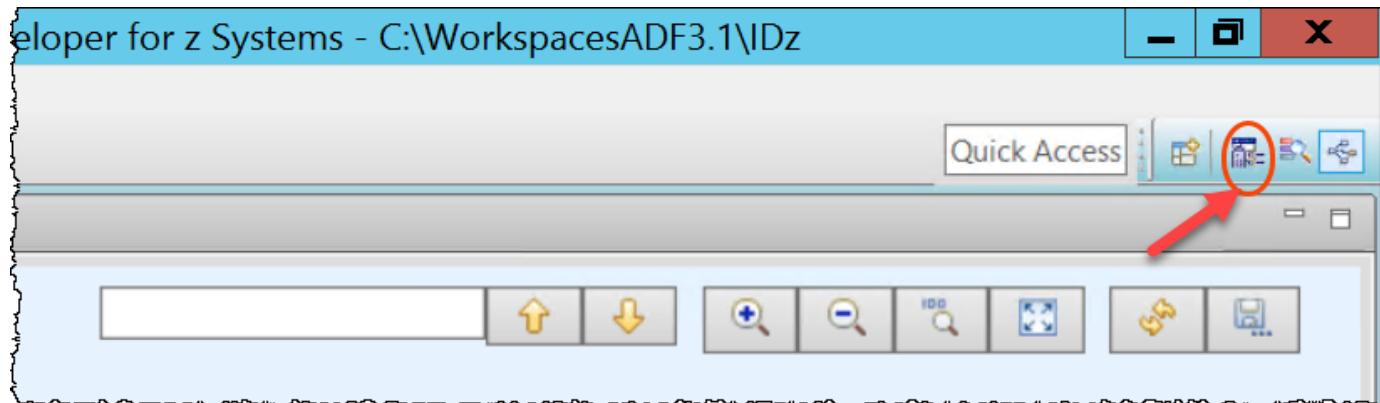
4.1.19 ➡ Click on (or F5) few times
➡ and Resume (F8) when you are satisfied so the program will execute until the end.



4.1.20 ➡ Click OK to close the dialog above.

To fix this bug definitely you will modify the program **REGI00B** that is returning 0 to be used in a division.
You will do that later. For now, optionally you can play again with the Debugger and use the Visual Debugger.
Or continue after the optional steps.

4.1.21 ► Go back to the **z/OS Projects Perspective** by clicking on the icon  in the top right corner.



4.2 Using the Visual Debugger for Stack pattern breakpoints

If you are not running late and interested on this subject, execute the steps below, otherwise jump to 4.3

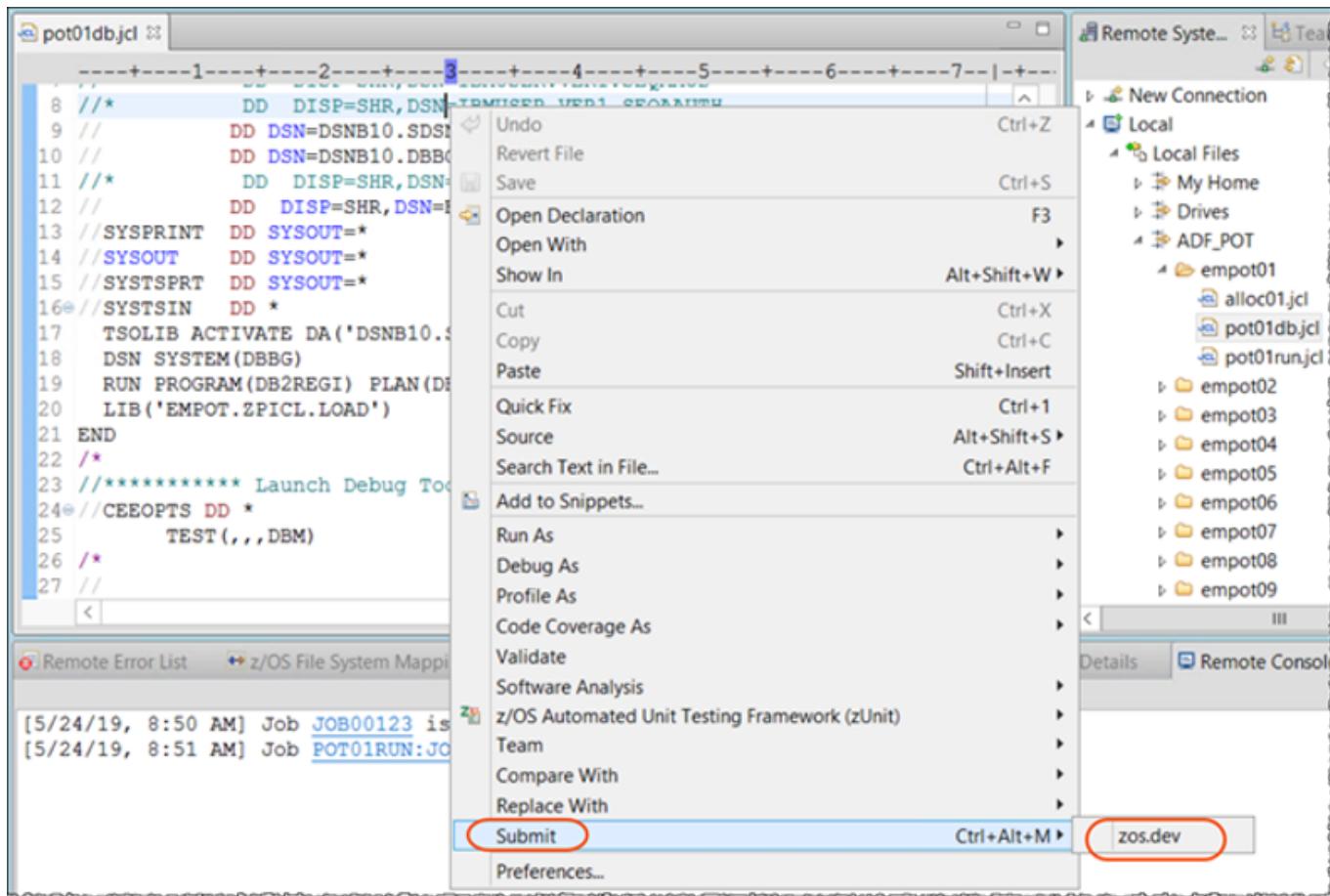
Stack pattern breakpoint



A stack pattern breakpoint is a special type of conditional breakpoint. With this feature, you can specify that you only want to stop at a location when the current stack trace contains a predefined stack pattern.

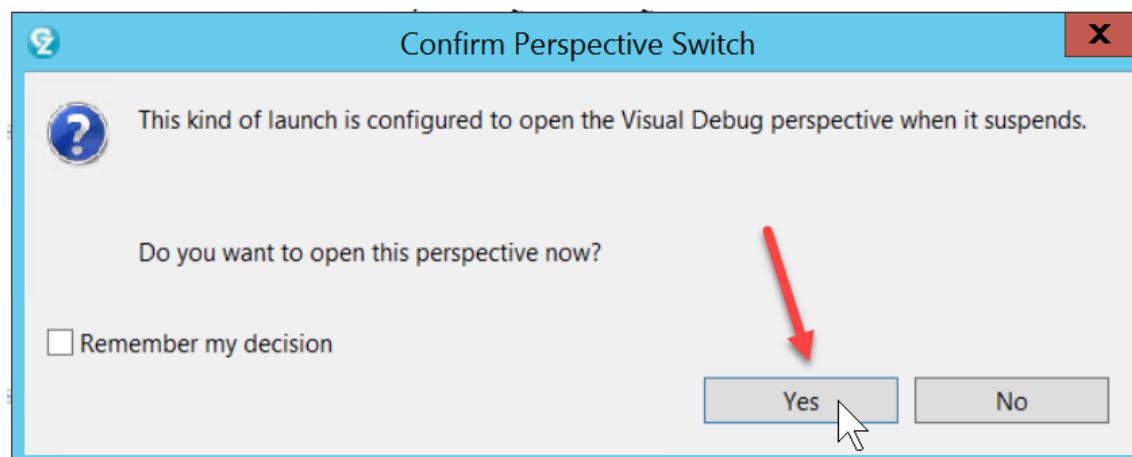
Visual debugging supports a stack pattern integration feature, which allows you to select a connected path from the program control flow diagram and set a stack pattern breakpoint using the selected path as a stack pattern.

4.2.1 ► Right click on the JCL being edited and select **Submit > zos.dev**

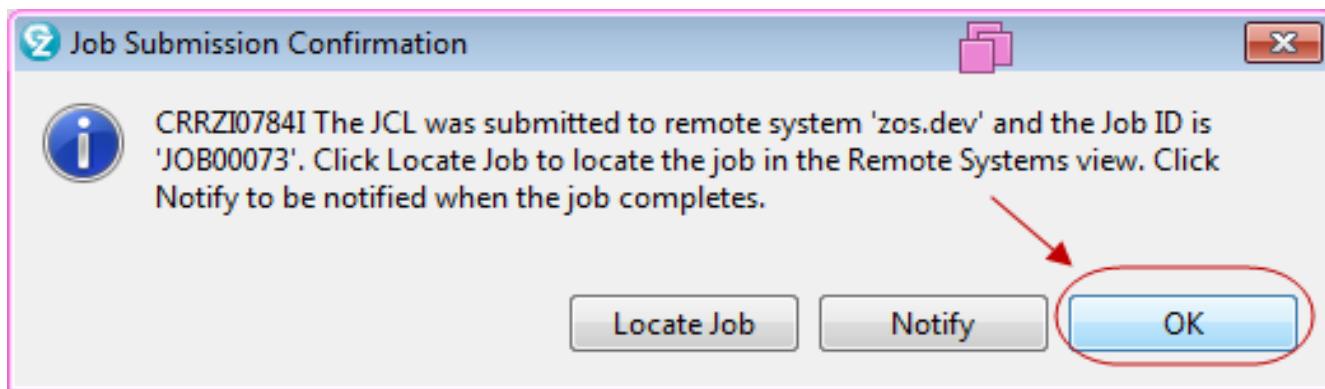


4.2.2 Once the job starts the z/OS debug will “talk” with you.

► Click **Yes** to open the *Visual Debug* Perspective
(depending on the z/OS the order here could be the next step first).

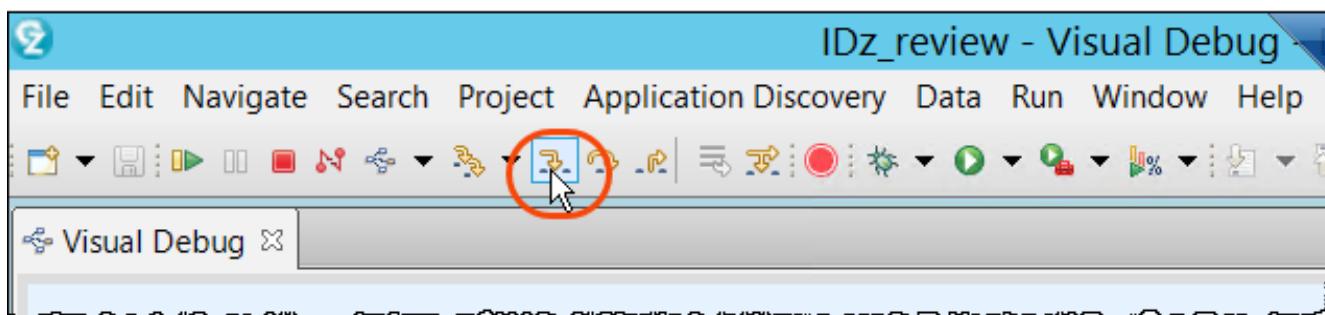


4.2.3 ► Also click **OK** for this dialog below

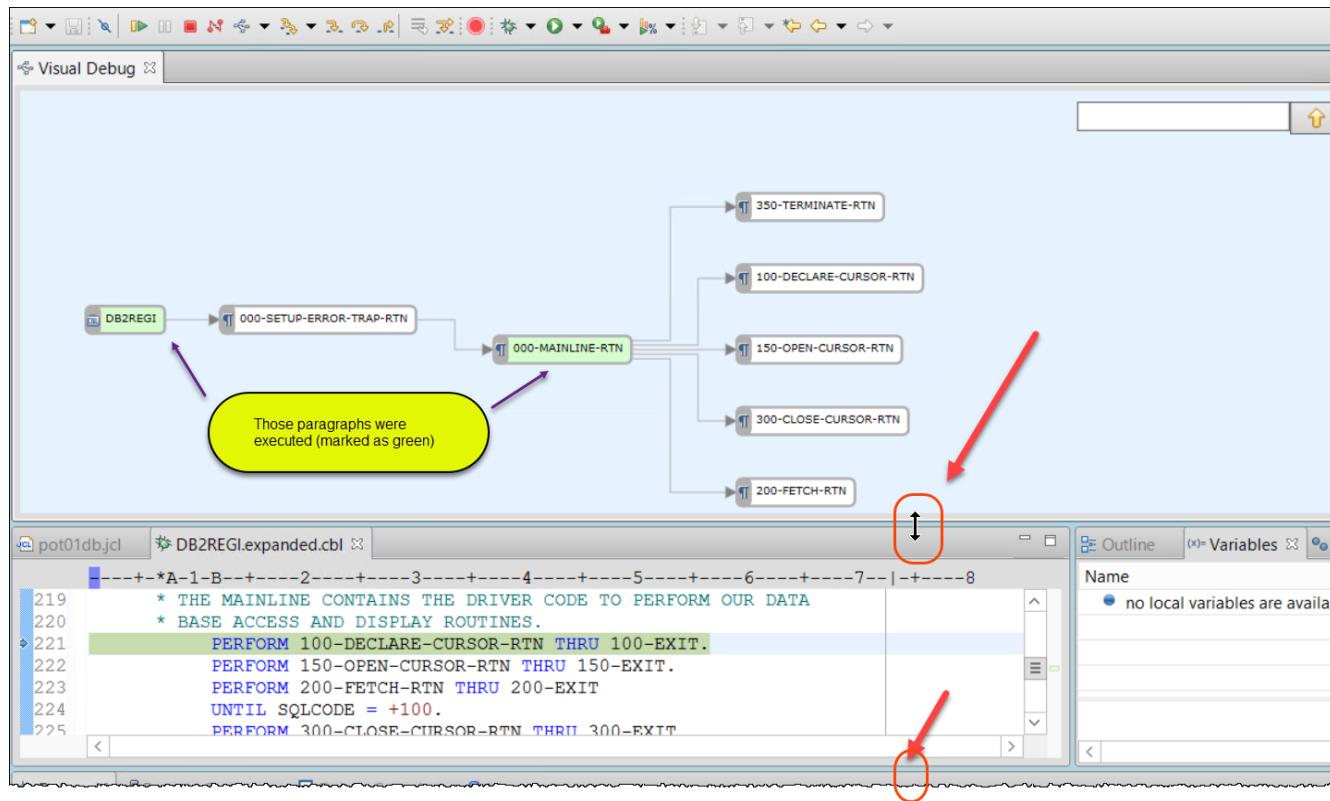


4.2.4 Using the *Visual Debug* perspective:

► Click on icon (or Press F5)



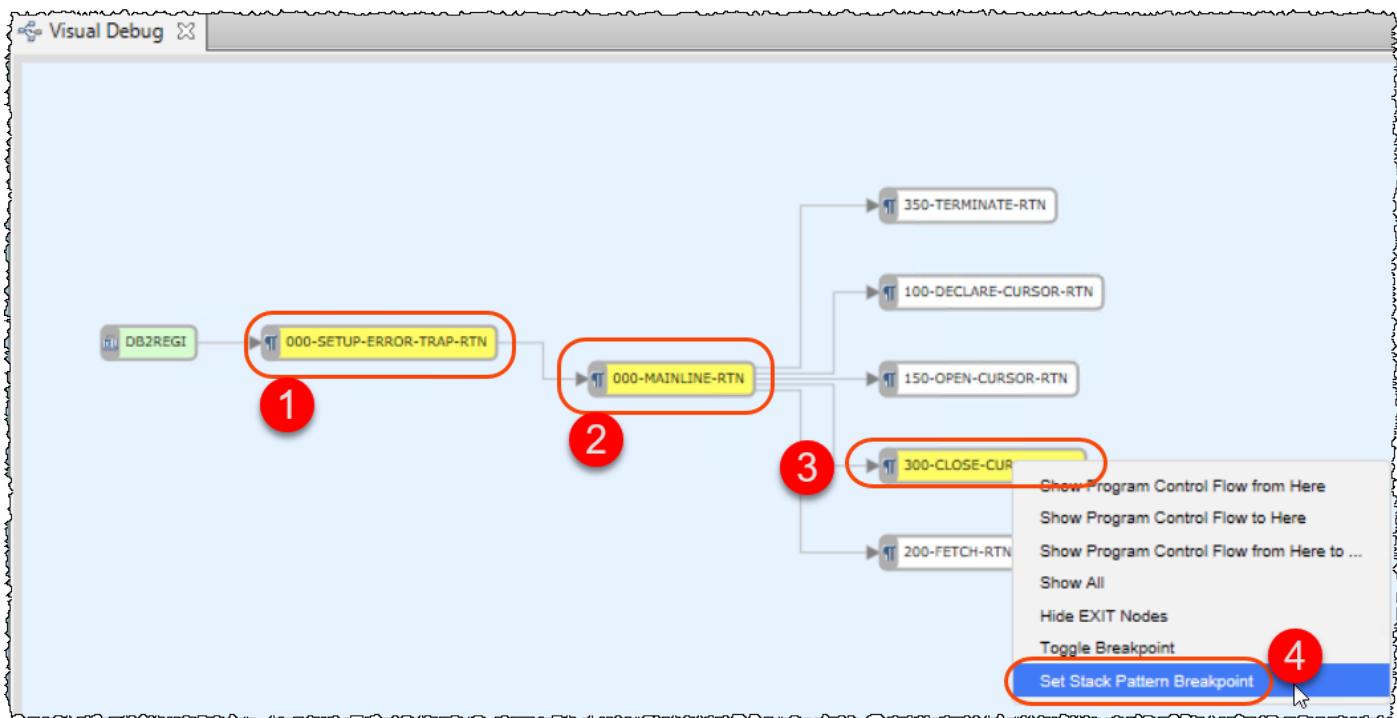
4.2.5 ►| Resize the **Visual Debug** view and you will notice that the paragraphs executed are marked as green



4.2.6 ►| Using the **Visual Debug** view, scroll down

►| Press **CTRL** Key and select the 3 paragraphs as shown below, right click and choose **Set Stack Pattern Breakpoint**

This breakpoint that will happen when the path selected happens.

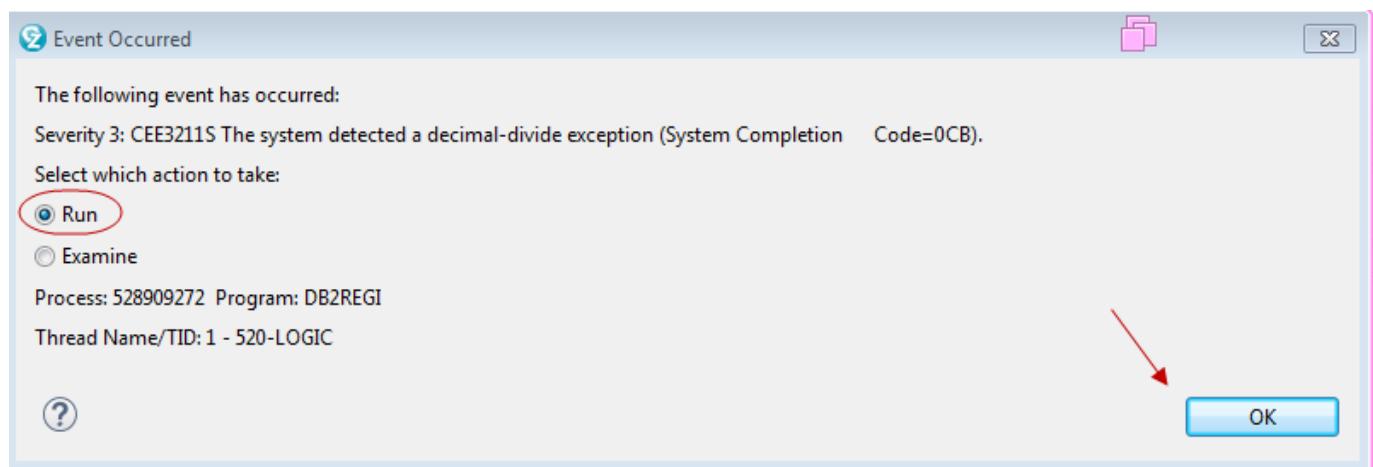


4.2.7 ► Click on **Breakpoints** view and you will see this breakpoint created.

The screenshot shows the COBOL editor with the Breakpoints view open. Three breakpoints are listed: Entry [300-CLOSE-CUR-RTN] [Stack Pattern], Line [DB2REGI.expanded.cbl:365], and 520-LOGIC. The 'Entry [300-CLOSE-CUR-RTN] [Stack Pattern]' entry is circled with a red box and has a red arrow pointing to it from the previous diagram.

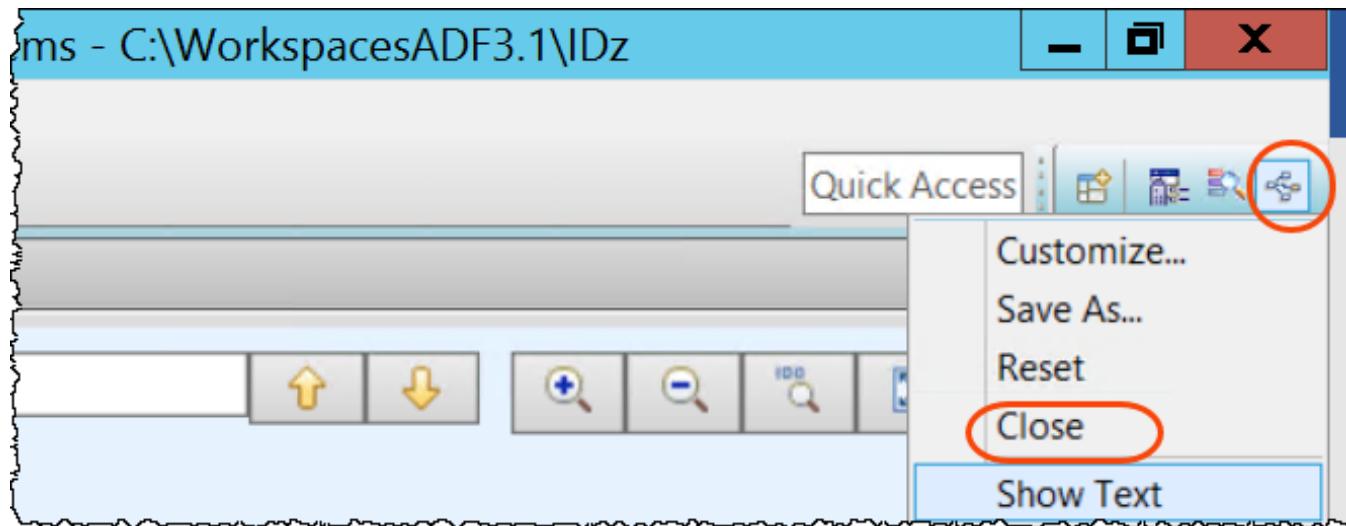
4.2.8 ► Click on (or press F8) few times to resume the execution

4.2.9 ► Select **Run** and click **OK** to continue

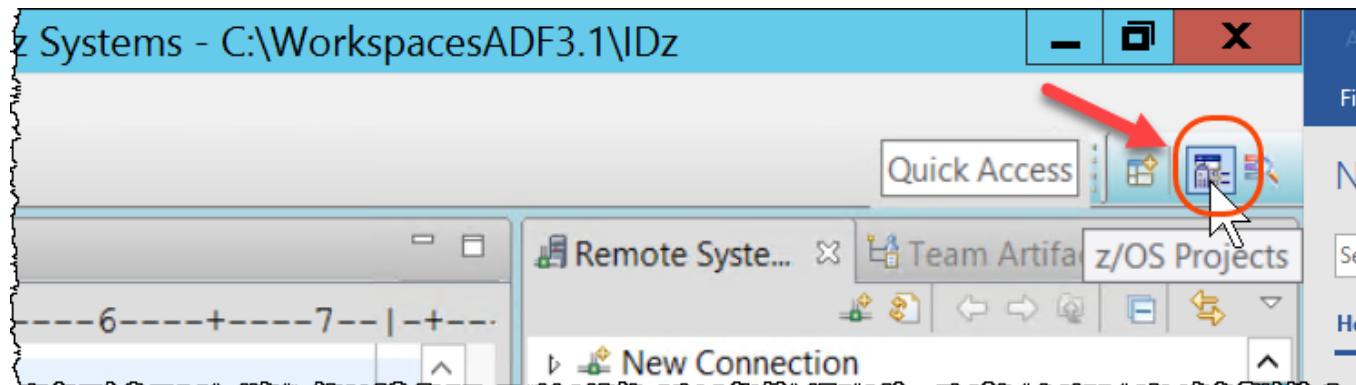


Notice – This breakpoint did not happen since the execution did not execute the 3 selected paragraphs.

4.2.10 ► Close the **Visual Debug perspective**.



4.2.11 ► Go back to the **z/OS Projects Perspective** by clicking on the icon in the top right corner.



4.3 Accessing the z/OS JES

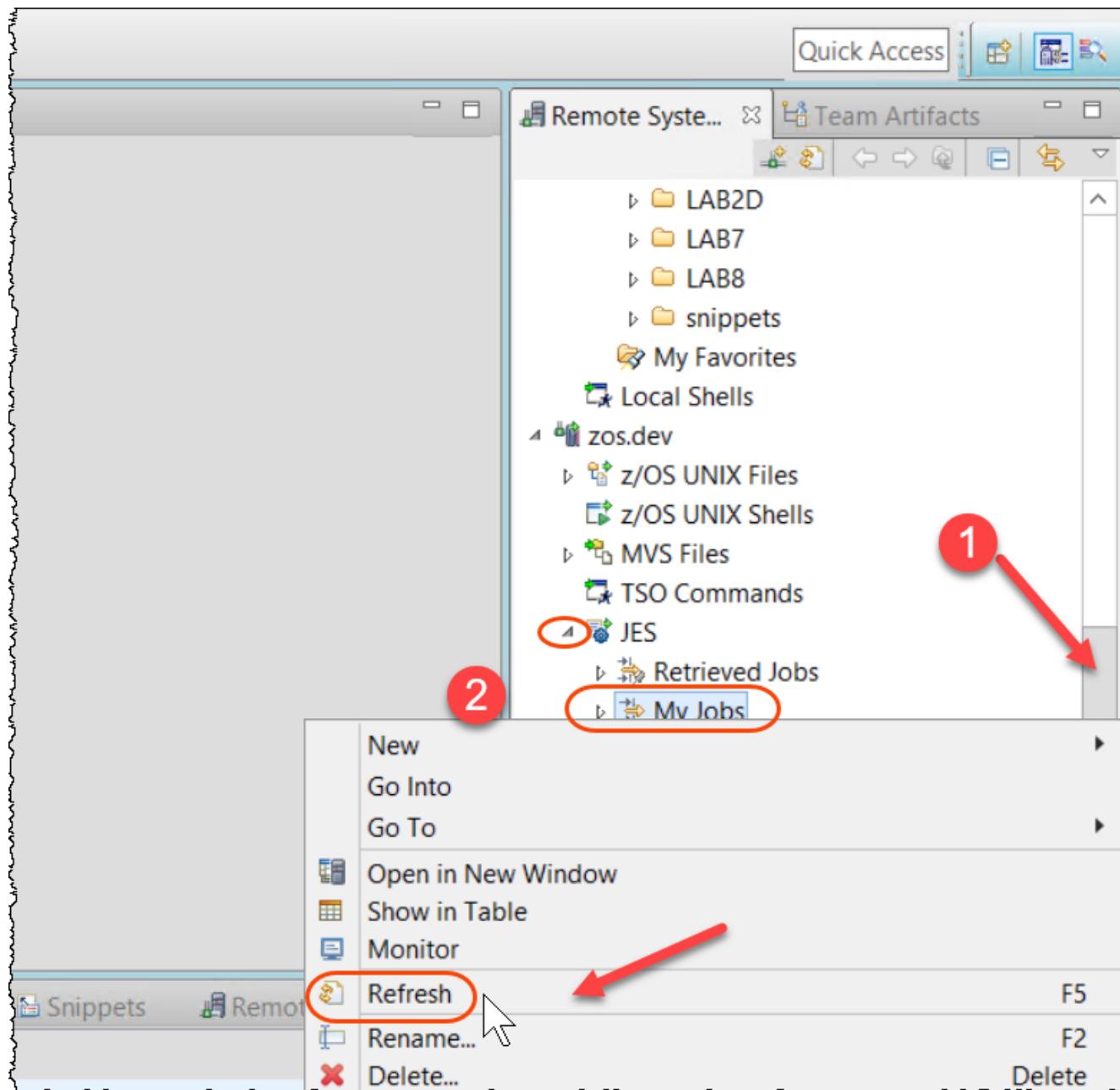
On ISPF many people use *SDSF* for this activity.

You will use the Job Monitor subsystem part of the host components.

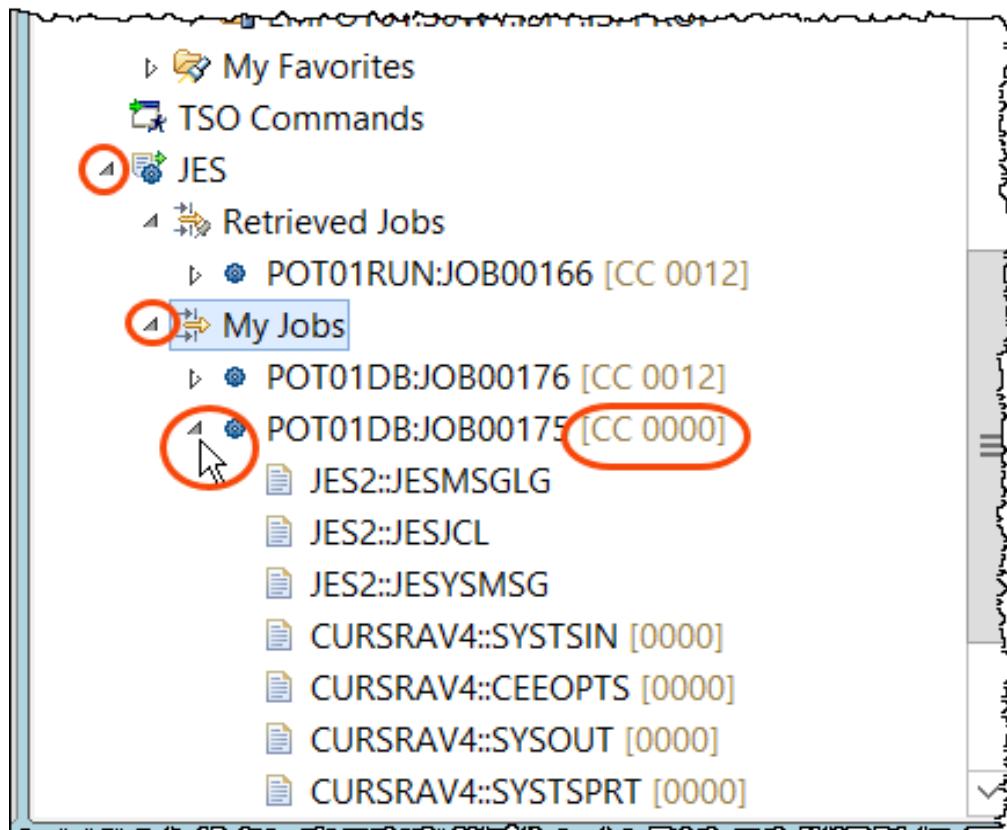
4.3.1 ► Use **Ctrl + Shift + F4** to close all opened editors. Do not save any changes.

Or just click on the of each opened editor.

4.3.2 ► Using the *Remote Systems* view scroll down, expand JES right click on **My Jobs** and select **Refresh**



4.3.3 ►| Expand **My Jobs** and the Job **POT01DB:JOB00xxx** that has the CC 0000
(it will be the latest if you did not make the optional step)



4.3.4 ►| On Remote Systems view, double click on the **POT01DB:JOB00xxx** that has the CC 0000

POT01DB:JOB00175.spool

```

-----+-----+-----+-----+-----+-----+-----+-----+
 1 | J E S 2   J O B   L O G   --   S Y S T E M   S O W 1   --   N O D E   S O W
 2 |
 3 09.39.41 JOB00175 ---- MONDAY, 12 AUG 2019 -----
 4 09.39.41 JOB00175 IRR0101 USERID EMPOT01 IS ASSIGNED TO THIS JOB.
 5 09.39.42 JOB00175 ICH70001I EMPOT01 LAST ACCESS AT 09:37:51 ON MONDAY, AUGUST 12, 2019
 6 09.39.42 JOB00175 $HASP373 POT01DB STARTED - INIT 1 - CLASS A - SYS SOW1
 7 09.39.43 JOB00175 +EQAP0018I Using DBM port 5336; timeout=300 seconds
 8 09.40.24 JOB00175 -
 9 09.40.24 JOB00175 -JOBNAME STEPNAME PROCSTEP RC EXCP TCB SRB CLOCK SERV E
10 09.40.24 JOB00175 -POT01DB 00 1240 ***** .00 .7 2251
11 09.40.24 JOB00175 -POT01DB ENDED. NAME- TOTAL TCB CPU TIME= .02
12 09.40.24 JOB00175 $HASP395 POT01DB ENDED - RC=0000
13 ----- JES2 JOB STATISTICS -----
14 12 AUG 2019 JOB EXECUTION DATE
15          27 CARDS READ
16          167 SYSOUT PRINT RECORDS
17          0 SYSOUT PUNCH RECORDS
18          12 SYSOUT SPOOL KBYTES
19          0.70 MINUTES EXECUTION TIME
20 1 //POT01DB JOB ,
21  // MSGCLASS=H,MSGLEVEL=(1,1),TIME=1440,REGION=70M,COND=(16,LT)
22  //*/ ZPICL Debug
23 2 //CURSRAV4 EXEC PGM=IKJEFT01,DYNAMNBR=20
24 3 // STEPLIB  DD DSN=EMPOT01.POT.LOAD,DISP=SHR
25 4 //          DD DSN=EMPOT01.ZPICL.LOAD,DISP=SHR
26  //          DD DISP=SHR,DSN=IBMUUSER.VER1.SEQAMOD
27  //          DD DISP=SHR,DSN=IBMUUSER.VER1.SEQAAUTH
  
```

Remote Systems

- EMPOT01.POT.COPYLIB
- EMPOT01.POT.DBRMLIB
- EMPOT01.POT.JCL
- EMPOT01.POT.LISTING
- EMPOT01.POT.LOAD
- EMPOT01.POT.OBJ
- EMPOT01.POT.PLI
- EMPOT01.POT.PLLISTING
- EMPOT01.POT.SP2.COBOL
- EMPOT01.SOW1.ISPF.ISPPROF
- My Favorites
- TSO Commands
- JES
 - Retrieved Jobs
 - POT01RUN:JOB00166 [CC 0012]
 - My Jobs
 - POT01DB:JOB00176 [CC 0012]
 - POT01DB:JOB00175 [CC 0000]**

4.3.5 ► Scroll down to see the various JOB steps. Notice that a small report has been printed.

POT01DB:JOB00175.spool

```
--+---1---+---2---+---3---+---4---+---5---+---6---+---7---+---8
91    CPU:   0 HR 00 MIN 01.66 SEC  SRB:   0 HR 00 MIN 00.03 SEC
92    VIRT: 660K SYS: 360K EXT: 8816K SYS: 10552K
93    ATB- REAL:          20K SLOTS:           0K
94    VIRT- ALLOC:      10M SHRD:          0M
95 IEF375I JOB/POT01DB /START 2019224.0939
96 IEF033I JOB/POT01DB /STOP 2019224.0940
97    CPU:   0 HR 00 MIN 01.66 SEC  SRB:   0 HR 00 MIN 00.03 SEC
98    TSOLIB ACTIVATE DA('DSNB10.SDSNLOAD')
99    DSN SYSTEM(DBBG)
100   RUN PROGRAM(DB2REGI) PLAN(DB2REGI) -
101     LIB('EMPOT.ZPICL.LOAD')
102 END
103   TEST(,,DBM)
104 *DEPT*      ACC
105 *PERF-AVG*  0000000
106 *PERF-MIN*  0008
107 *PERF-MAX*  0008
108 *HOURS-AVG* 0001599
109 *HOURS-MIN* 0001599
110 *HPOURS-MAX* 0001599
111 ACC      .00  8.00   8.00   15.99   15.99   15.99
112 *DEPT*      FIN
113 *PERF-AVG*  0000000
114 *PERF-MIN*  0003
115 *PERF-MAX*  0009
116 *HOURS-AVG* 0002269
117 *HOURS-MIN* 0003245
118 *HPOURS-MAX* 0000889
```

Remote Systems Team Artifacts

- ▶ EMP01.POT.COPYLIB
- ▶ EMP01.POT.DBRLIB
- ▶ EMP01.POT.JCL
- ▶ EMP01.POT.LISTING
- ▶ EMP01.POT.LOAD
- ▶ EMP01.POT.OBJ
- ▶ EMP01.POT.PLI
- ▶ EMP01.POT.PLI.LISTING
- ▶ EMP01.POT.SP2.COBOL
- ▶ EMP01.SOW1.ISPF.ISPPROF

My Favorites

TSO Commands

JES

- ▶ Retrieved Jobs
 - ▶ POT01RUN:JOB00166 [CC 0012]
- ▶ My Jobs
 - ▶ POT01DB:JOB00175 [CC 0000] 1
 - ▶ POT01DB:JOB00175 [CC 0000] 2

JES2:JESMSGMSG

JES2:JESJCL

JES2:JESYMSG

CURSRAV4:SYSTIN [0000]

CURSRAV4:CEEOPTS [0000]

CURSRAV4:SYSOUT [0000]

CURSRAV4:SYSPRT [0000]

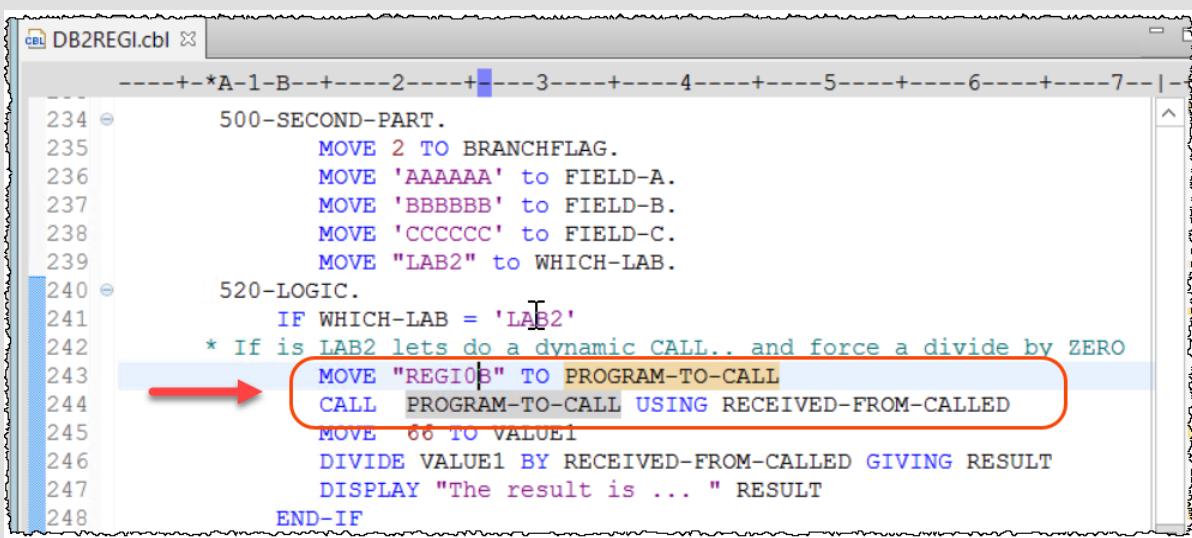
4.3.6 ► Close all opened editors. (**Ctrl + Shift + F4**). Click **No** if asked to save changes.
Or just click on the of each opened editor.

Section 5. Modify the COBOL code to fix the bug

You will work with a z/OS member using the editor and now we will be working with the assets located on the z/OS remote system.

What z/OS remote assets you will work with?

This is a batch program that reads DB2. In addition, this program does a Dynamic call to another COBOL program named **REG10B**.



```

DB2REGI.cbl

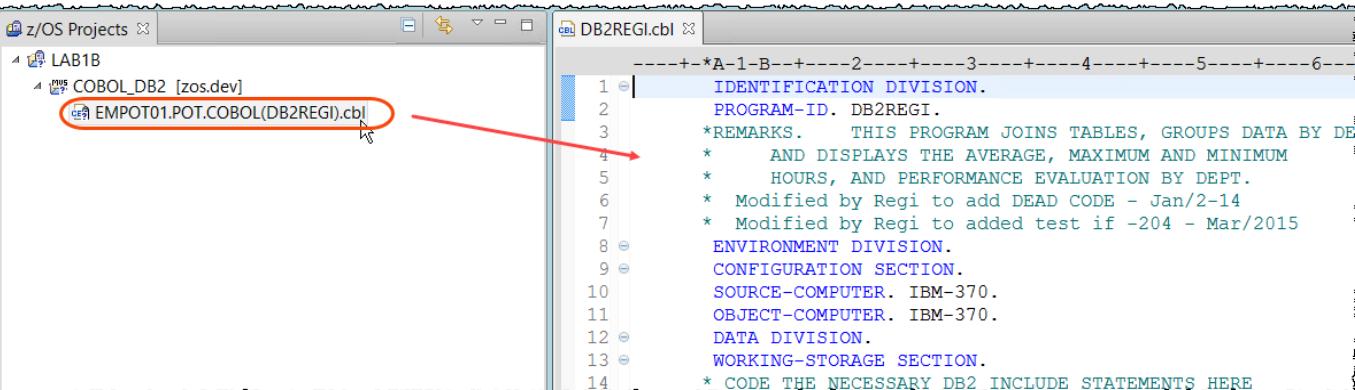
-----+*A-1-B-----2-----3-----4-----5-----6-----7-----|-
234      500-SECOND-PART.
235      MOVE 2 TO BRANCHFLAG.
236      MOVE 'AAAAAA' to FIELD-A.
237      MOVE 'BBBBBB' to FIELD-B.
238      MOVE 'CCCCCC' to FIELD-C.
239      MOVE "LAB2" to WHICH-LAB.
240      520-LOGIC.
241      IF WHICH-LAB = 'LAB2'
242      * If is LAB2 lets do a dynamic CALL.. and force a divide by ZERO
243      MOVE "REG10B" TO PROGRAM-TO-CALL
244      CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
245      MOVE 66 TO VALUE1
246      DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
247      DISPLAY "The result is ... " RESULT
248      END-IF

```

5.1 Using the Editor with z/OS COBOL programs

Before fixing the code let's explore some editor capabilities on the main DB2 COBOL program.

5.1.1 ► Using the z/OS Projects perspective and the z/OS Projects view, double click on **EMPOT01.POT.COBOL(DB2REGI)** to open the file using the editor.



```

z/OS Projects
  LAB1B
  COBOL_DB2 [zos.dev]
    EMPOT01.POT.COBOL(DB2REGI.cbl)

DB2REGI.cbl

-----+*A-1-B-----2-----3-----4-----5-----6-----|-
1      IDENTIFICATION DIVISION.
2      PROGRAM-ID. DB2REGI.
3      *REMARKS. THIS PROGRAM JOINS TABLES, GROUPS DATA BY DEPT.
4      * AND DISPLAYS THE AVERAGE, MAXIMUM AND MINIMUM
5      * HOURS, AND PERFORMANCE EVALUATION BY DEPT.
6      * Modified by Regi to add DEAD CODE - Jan/2-14
7      * Modified by Regi to added test if -204 - Mar/2015
8      ENVIRONMENT DIVISION.
9      CONFIGURATION SECTION.
10     SOURCE-COMPUTER. IBM-370.
11     OBJECT-COMPUTER. IBM-370.
12     DATA DIVISION.
13     WORKING-STORAGE SECTION.
14     * CODE THE NECESSARY DB2 INCLUDE STATEMENTS HERE.

```

5.1.2 ► Click on the **Outline** tab (bottom and left) to see the *Outline* view.

► Using the editor, browse the program and notice that the contents of the *Outline* view are synchronized with the COBOL source code and vice versa.

► Click on the **PROCEDURE DIVISION**. Notice that you can navigate using the *Outline* view.

The screenshot shows the z/OS Studio interface with the COBOL editor open. The left pane displays the project structure under 'z/OS Projects' and the current file 'DB2REGI.cbl'. The bottom-left pane has tabs for 'Properties' and 'Outline', with 'Outline' being the active tab. The main editor window shows COBOL code. A red arrow points from the 'PROCEDURE DIVISION' section in the outline to the same section in the code editor. The code editor highlights the 'PROCEDURE DIVISION' section, which contains the following text:

```

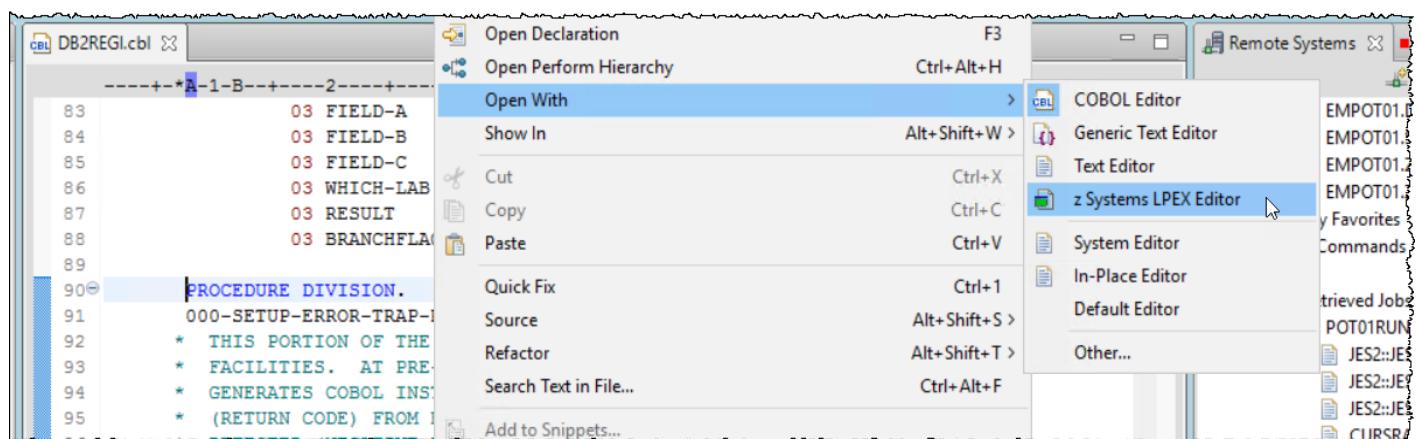
-----+*A-1-B---+----2----+---3---+---4---+---5---+---6---+---7---|+-
85          03 FIELD-C          PIC X(6).
86          03 WHICH-LAB        PIC X(4).
87          03 RESULT           PIC 99.
88          03 BRANCHFLAG       PIC 99.

90      PROCEDURE DIVISION.
91          000-SETUP-ERROR-TRAP-RTN.
* THIS PORTION OF THE PROGRAM ACTIVATES THE SQL ERROR TRAPPING
* FACILITIES. AT PRE-COMPILATION TIME, THE DB2 PRE-COMPILER
* GENERATES COBOL INSTRUCTIONS TO INTERROGATE THE SQLCODE
* (RETURN CODE) FROM EACH CALL. IF A SQLERROR CONDITION IS
* DETECTED (NEGATIVE RETURN CODE), EXECUTION WILL BRANCH TO THE
* 999-ERROR-TRAP-RTN TO DISPLAY AN APPROPRIATE ERROR MSG.
* SET UP YOUR ERROR HANDLING ROUTINES
000-MAINLINE-RTN.
* THE MAINLINE CONTAINS THE DRIVER CODE TO PERFORM OUR DATA
* BASE ACCESS AND DISPLAY ROUTINES.
          PERFORM 100-DECLARE-CURSOR-RTN THRU 100-EXIT.

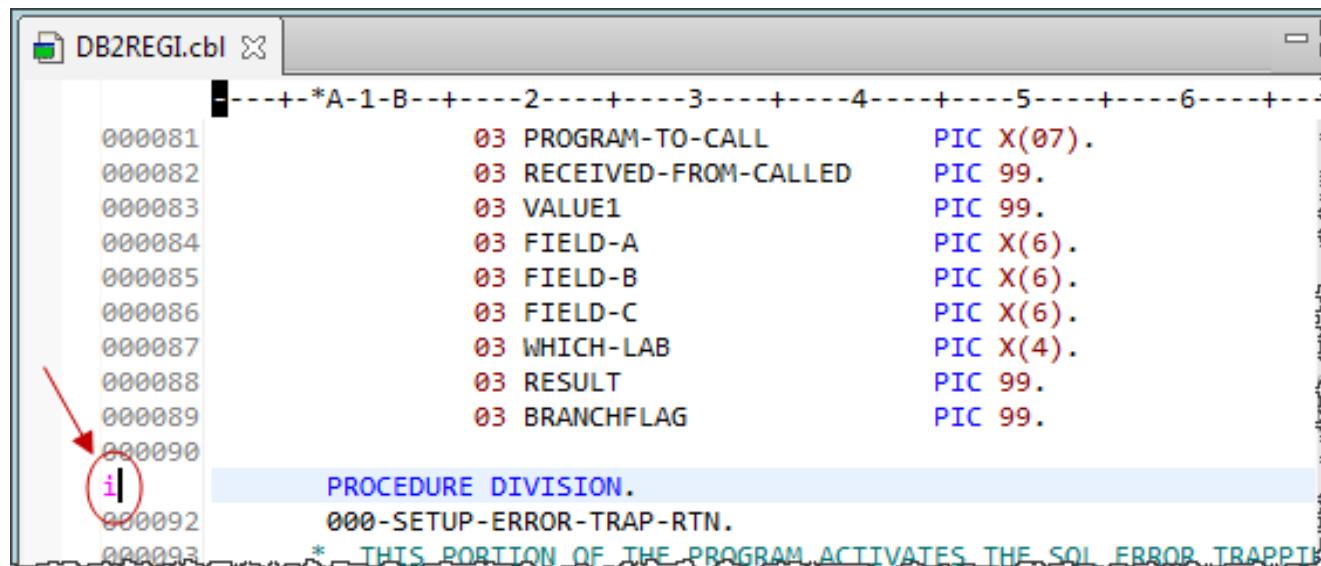
```

5.1.3 If you prefer the ISPF editor, you might use another IDz editor called "LPEX editor".

► To switch the editor, right click on the program and select **Open with > z Systems LPEX Editor**



5.1.4 ► Note that the column on left are similar as the prefix area of the ISPF editor..
 You may type commands to add lines etc..
 Like **I** to insert a line, **d** to delete, **m** to move, etc. **Try it.**



```

DB2REGI.cbl

-----+-----+-----+-----+-----+-----+
000081      03 PROGRAM-TO-CALL      PIC X(07).
000082      03 RECEIVED-FROM-CALLED  PIC 99.
000083      03 VALUE1                PIC 99.
000084      03 FIELD-A               PIC X(6).
000085      03 FIELD-B               PIC X(6).
000086      03 FIELD-C               PIC X(6).
000087      03 WHICH-LAB             PIC X(4).
000088      03 RESULT                PIC 99.
000089      03 BRANCHFLAG            PIC 99.
000090
000091      PROCEDURE DIVISION.
000092          000-SETUP-ERROR-TRAP-RTN.
000093          * THIS PORTION OF THE PROGRAM ACTIVATES THE SQL ERROR TRAPPI

```

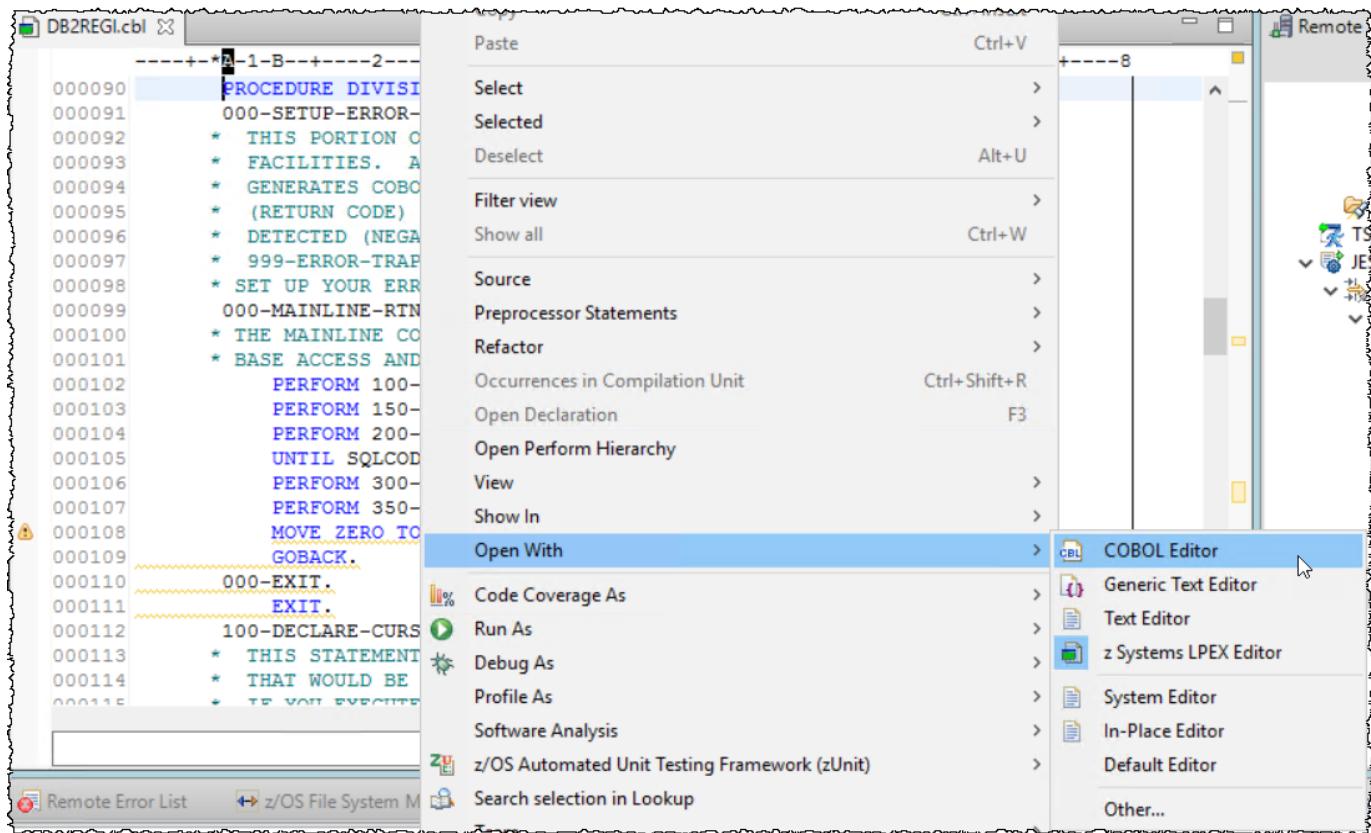
5.1.5 Switch back to **COBOL editor** (since the screen captures uses the COBOL editor)

► Right click on the program and select **Open with > COBOL Editor**

Click **No** for saving changes.

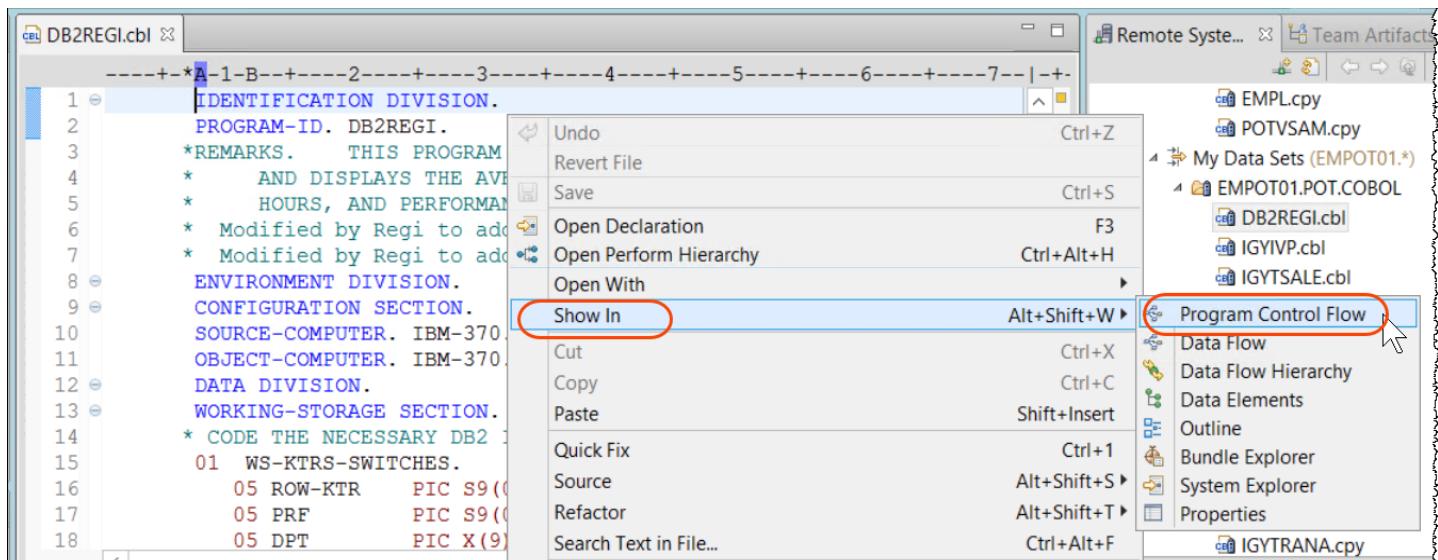
This editor is more like the Java Editor and gives you more capabilities. When you see all advantages, probably you will prefer this editor than the ISPF like editor (LPEX)..

Please keep this program opened as we will work on it later.



5.1.6 Let's see the program in more details..

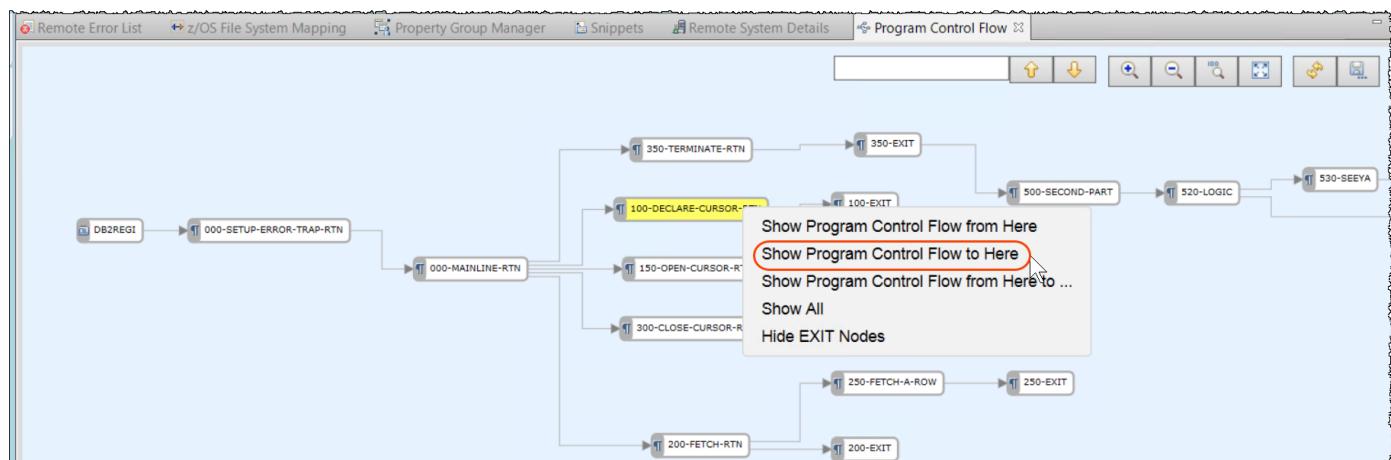
► Right click on the editor and select **Show In → Program Control Flow**



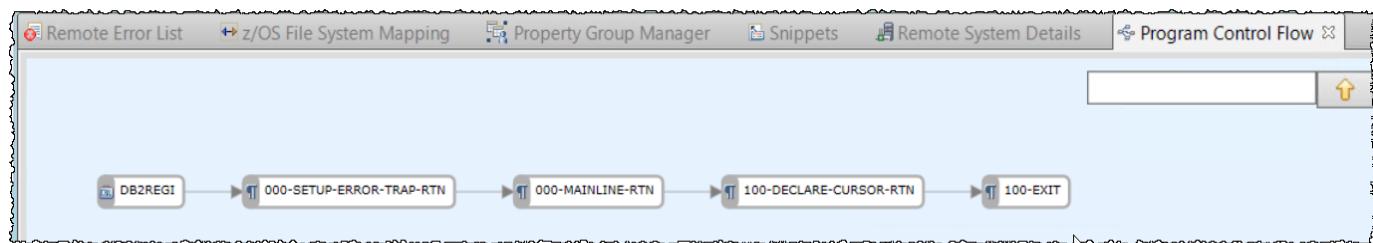
5.1.7 See the results under *Program Control Flow* view.

▶ Double click “**Program Control Flow**” title to have a bigger diagram (full page).

▶ Right-click on **100-DECLARE-CURSOR-RTN** box and select **Show Program Flow Control to here**.



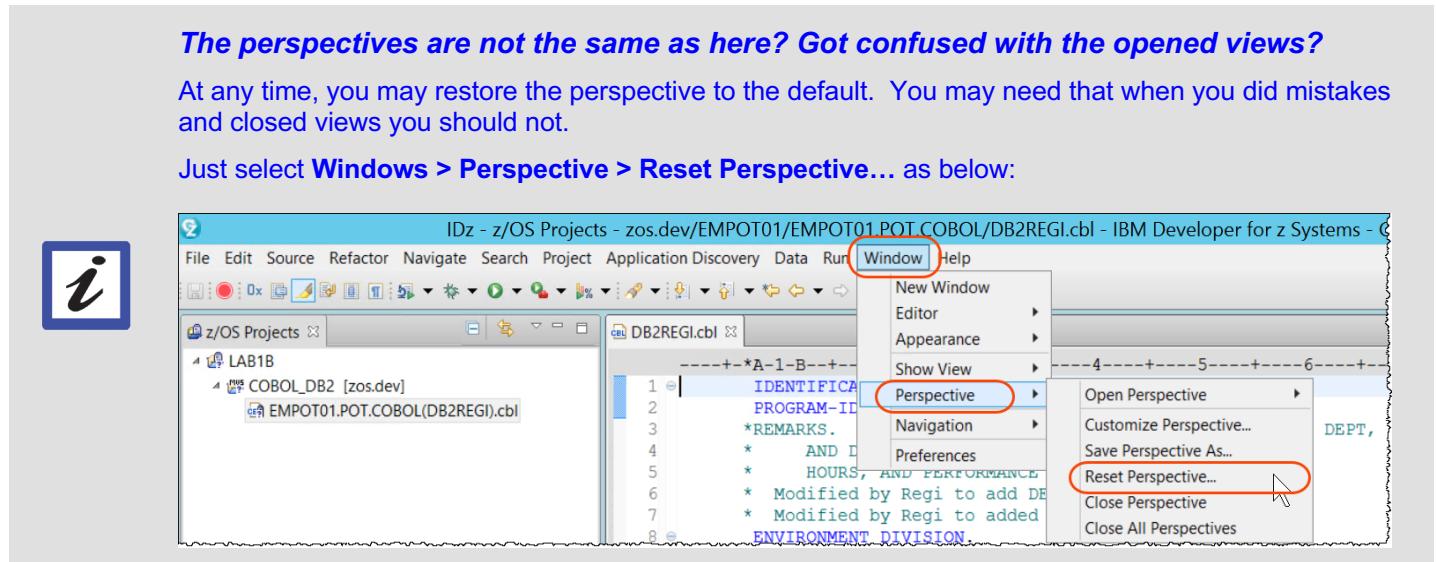
5.1.8 It shows the path to reach this paragraph.. Nice uh? That helps with extremely complex diagrams



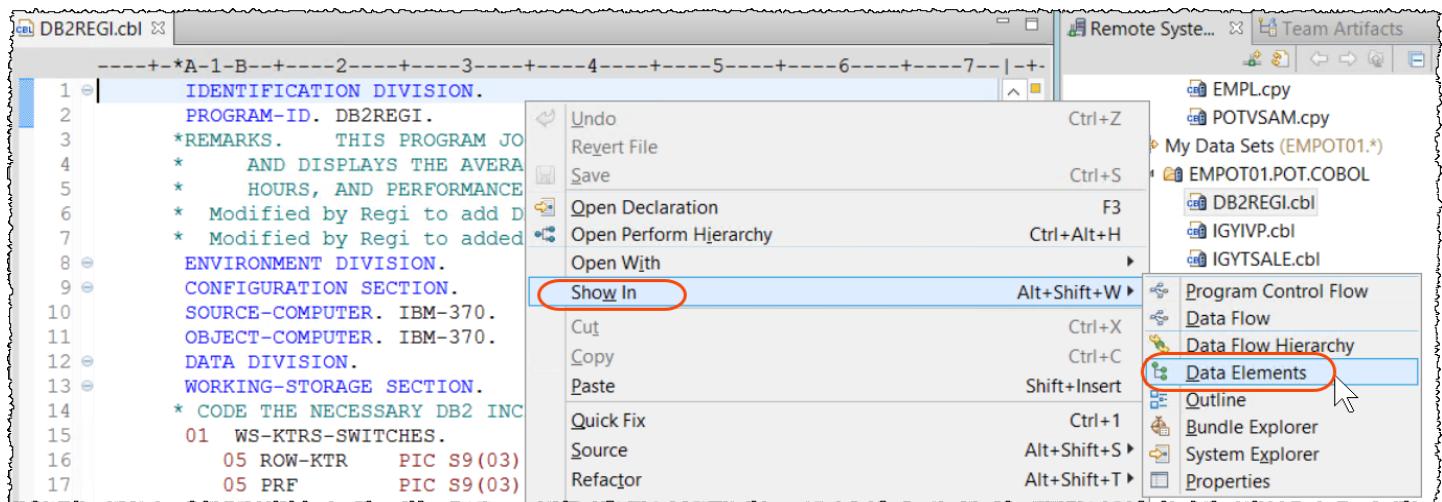
The perspectives are not the same as here? Got confused with the opened views?

At any time, you may restore the perspective to the default. You may need that when you did mistakes and closed views you should not.

Just select **Windows > Perspective > Reset Perspective...** as below:



- 5.1.9 ► Double click "Program Control Flow" title to have the diagram smaller.
 ► Right click again on the editor and select
Show In → Data Elements

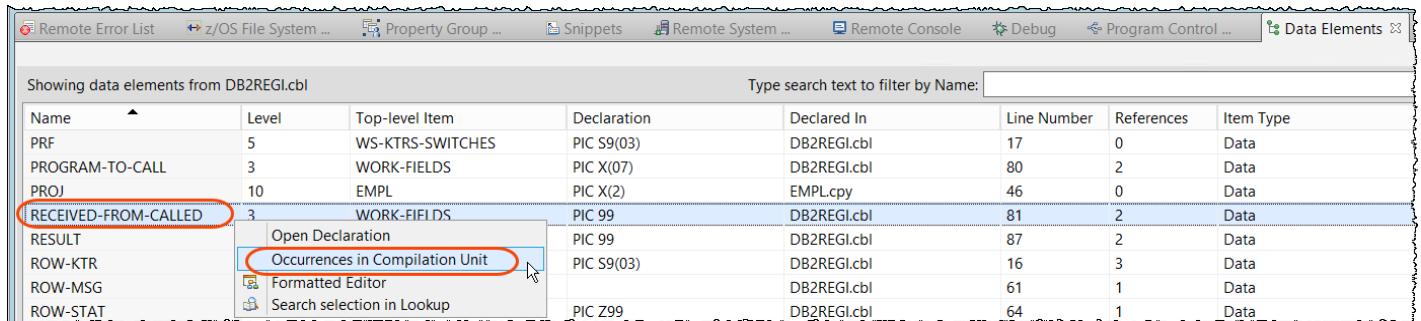


```

1  IDENTIFICATION DIVISION.
2    PROGRAM-ID. DB2REGI.
3    *REMARKS. THIS PROGRAM JO
4      * AND DISPLAYS THE AVERA
5      * HOURS, AND PERFORMANCE
6      * Modified by Regi to add D
7      * Modified by Regi to added
8  ENVIRONMENT DIVISION.
9  CONFIGURATION SECTION.
10 SOURCE-COMPUTER. IBM-370.
11 OBJECT-COMPUTER. IBM-370.
12 DATA DIVISION.
13 WORKING-STORAGE SECTION.
14 * CODE THE NECESSARY DB2 INC
15   01 WS-KTRS-SWITCHES.
16     05 ROW-KTR      PIC S9(03)
17     05 PRF          PIC S9(03)

```

- 5.1.10 ► On Data Elements view, resize the view, scroll down and right click on RECEIVED-FROM-CALLED and select Occurrences in Compilation Unit



Name	Level	Top-level Item	Declaration	Declared In	Line Number	References	Item Type
PRF	5	WS-KTRS-SWITCHES	PIC S9(03)	DB2REGI.cbl	17	0	Data
PROGRAM-TO-CALL	3	WORK-FIELDS	PIC X(07)	DB2REGI.cbl	80	2	Data
PROJ	10	EMPL	PIC X(2)	EMPL.cpy	46	0	Data
RECEIVED-FROM-CALLED	3	WORK-FIELDS	PIC 99	DB2REGI.cbl	81	2	Data
RESULT			PIC 99	DB2REGI.cbl	87	2	Data
ROW-KTR			PIC S9(03)	DB2REGI.cbl	16	3	Data
ROW-MSG				DB2REGI.cbl	61	1	Data
ROW-STAT			PIC Z99	DB2REGI.cbl	64	1	Data

- 5.1.11 ► Double click on line 246. It will show in the line that is abending.
 Notice the colors. When the variable is referenced, it is blue and when it is modified is brown.

The screenshot shows the Rational Developer for System z interface. The main window displays a CBL (COBOL) source code editor with the file 'DB2REGL.cbl'. The code contains several 'RECEIVED-FROM-CALLED' statements, which are highlighted in blue. A red arrow points from the bottom search results to the line '246: DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT' in the editor. The search results panel at the bottom lists three matches for 'RECEIVED-FROM-CALLED' in the compilation unit 'DB2REGL.cbl'. The third result, which is the one highlighted by the red circle, corresponds to the line in the editor.

```
-----+---A-1-B---+---2---+---3---+---4---+---5---+---6---+---7---+---8
241      IF WHICH-LAB = 'LAB2'
242      * If is LAB2 lets do a dynamic CALL.. and force a divide by ZERO
243          MOVE "REG10B" TO PROGRAM-TO-CALL
244          CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
245          MOVE 66 TO VALUE1
246          DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
247          DISPLAY "The result is ... " RESULT
248
249      END-IF
250      IF BRANCHFLAG > 1
251          CALL 'REGIOC' USING Input-name
252          DISPLAY "BRANCHFLAG GREATER THAN 1"
253          PERFORM 530-SEEYA
254      ELSE
255          DISPLAY "BRANCHFLAG <= 1 no STATIC CALL"
256          PERFORM 540-GOODBYE.
257
258      530-SEEYA.
259          DISPLAY "EXECUTED SEEYA PARAGRAPH".
260
261      540-GOODBYE.
```

Remote Error... z/OS File Syst... Property Grou... Snippets Remote Syste... Console Remote Cons... Program Cont...

'RECEIVED-FROM-CALLED' - 3 matches in compilation unit of 'DB2REGL.cbl'

- DB2REGL.cbl (3 matches)
 - 81 03 RECEIVED-FROM-CALLED PIC 99.
 - 244: CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
 - 246: DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT

You need to fix the called program named **REG10B** that is returning 0 on this variable as seen before with Fault Analyzer.

Exploring Data Flow

This capability is cool. *Program data flow* provides a graphical and hierarchical view of the data flow within a COBOL program. You can use this feature to examine how a data element is populated, modified, or written elsewhere.

5.1.12 ►► Double click on line 81. It will show where RECEIVED-FROM-CALLED is defined.

DB2REGI.cbl

```

-----+*A-1-B---+--2---+--3---+--4---+--5---+--6---+--7---+--8
      01 WORK-FIELDS.
      03 INPUT-NAME          Pic x(30).
      03 PROGRAM-TO-CALL    PIC X(07).
      03 RECEIVED-FROM-CALLED PIC 99.
      03 VALUE1              PIC 99.
      03 FIELD-A             PIC X(6).
      03 FIELD-B             PIC X(6).
      03 FIELD-C             PIC X(6).
      03 WHICH-LAB            PIC X(4).
      03 RESULT               PIC 99.
      03 BRANCHFLAG           PIC 99.

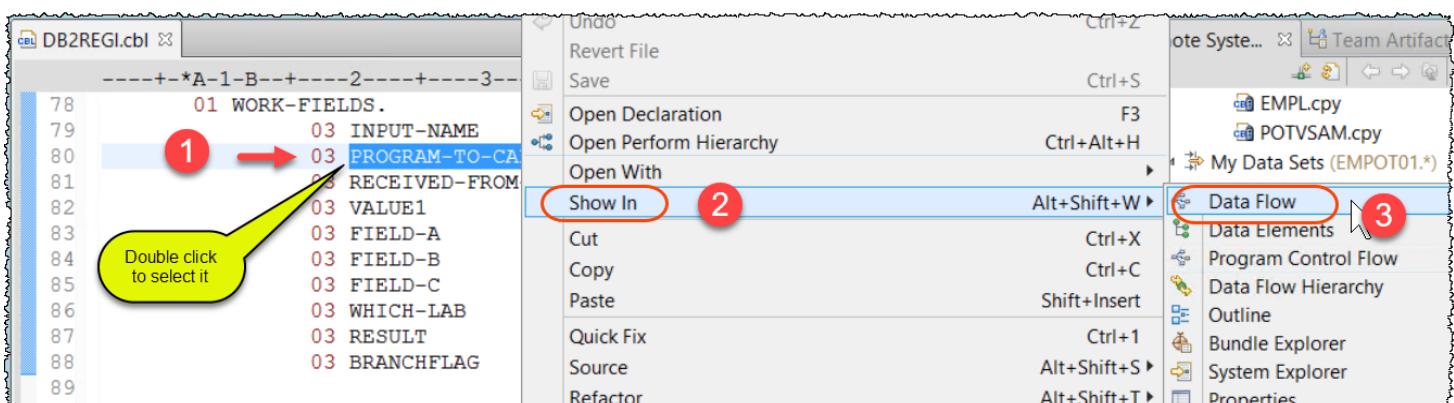
      PROCEDURE DIVISION.
      000-SETUP-ERROR-TRAP-RTN.
      * THIS PORTION OF THE PROGRAM ACTIVATES THE SQL ERROR TRAPPING
      * FACILITIES. AT PRE-COMPILATION TIME, THE DB2 PRE-COMPILER
      * GENERATES COBOL INSTRUCTIONS TO INTERROGATE THE SQLCODE
      * (RETURN CODE) FROM EACH CALL. IF A SQLERROR CONDITION IS
      * DETECTED (NEGATIVE RETURN CODE), EXECUTION WILL BRANCH TO THE
      * 999-ERROR-TRAP-RTN TO DISPLAY AN APPROPRIATE ERROR MSG.
      * SET UP YOUR ERROR HANDLING ROUTINES
      000-MAINLINE-RTN.
      * THE MAINLINE CONTAINS THE DRIVER CODE TO PERFORM OUR DATA
      * BASE ACCESS AND DISPLAY ROUTINES.
      000-MAINLINE-RTN.
      PERFORM 100-DECLARE-CURSOR-RTN  MUNIT 100  EXIT.

```

RECEIVED-FROM-CALLED - 3 matches in compilation unit of 'DB2REGI.cbl'

- 81: 03 RECEIVED-FROM-CALLED PIC 99.
- 244: CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
- 246: DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT

5.1.13 ➔ Select **PROGRAM-TO-CALL** (on line 80) double clicking on it, right click and select **Show In > Data Flow**. (the options order can differ from the screen capture here).



5.1.14 ►| Move the cursor to the line between **REGI0B** and **03 PROGRAM-TO-CALL**.

►| Clicking in this line shows the program statement where “**REGI0B**” is moved to **PROGRAM-TO-CALL**.

```

DB2REGL.cbl

240      520-LOGIC.
241          IF WHICH-LAB = 'LAB2'
242              * If is LAB2 lets do a dynamic CALL.. and force a divide by ZERO
243                  MOVE "REGI0B" TO PROGRAM-TO-CALL
244                      CALL PROGRAM-TO-CALL USING RECEIVED-FROM-CALLED
245                      MOVE 66 TO VALUE1
246                      DIVIDE VALUE1 BY RECEIVED-FROM-CALLED GIVING RESULT
247                      DISPLAY "The result is ... " RESULT
248      END-IF
249          IF BRANCHFLAG > 1
250              CALL 'REGI0C' USING Input-name
251              DISPLAY "BRANCHFLAG GREATER THAN 1"

```

5.1.15 ►| Close all opened editors if still opened. (**CTRL + Shift + F4**)

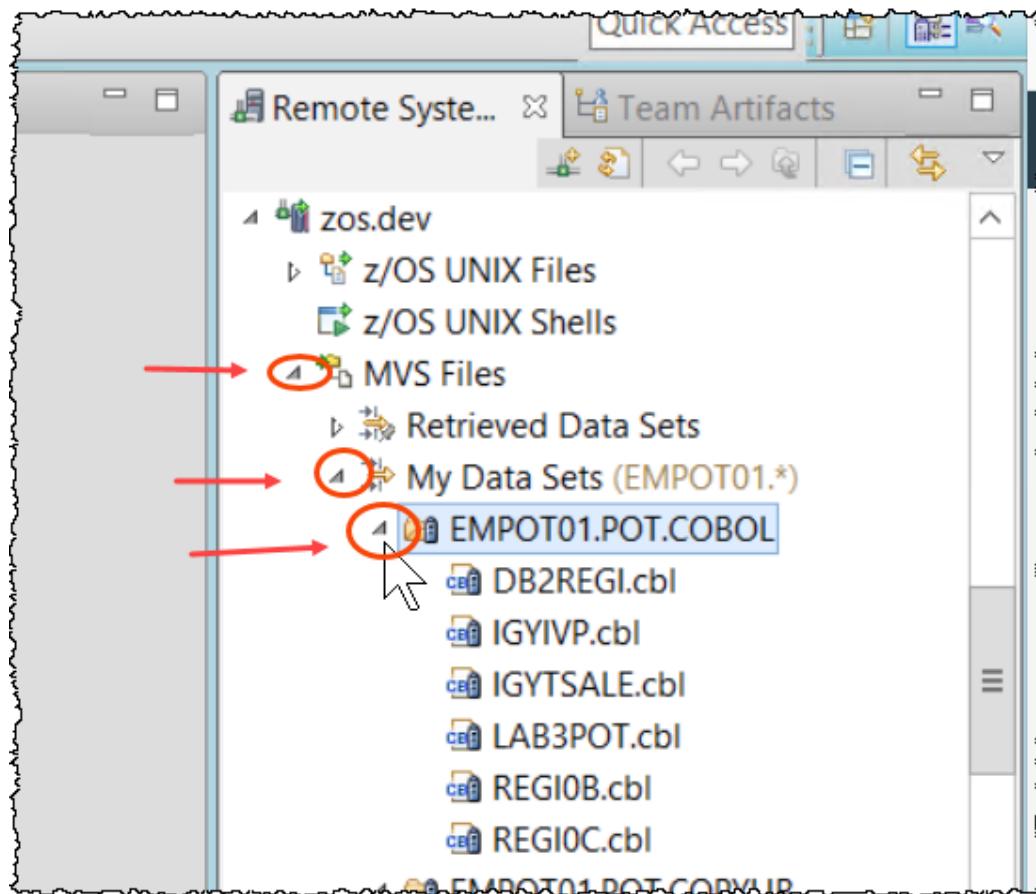
Or just click on the of each opened editor.

5.2 Fixing the program REGI0B

You need to fix the called program named **REGI0B** that is returning 0 on this variable as seen before with Fault Analyzer and the debugger.

This program is on your **PDS EMPOT01.POT.COBOL**.

5.2.1 ► Using **Remote Systems** view scroll back, expand **MVS Files**, **My Data Sets** and **EMPOT01.POT.COBOL**. if not already expanded.



5.2.4 ► Double click on the program **REGI0B.cbl** to edit it.

The screenshot shows the Rational Developer for z/OS interface. On the left is the COBOL editor window titled "REGI0B.cbl" containing the following code:

```

1  Identification Division.
2  Program-ID. "REGI0B".
3  ****
4  * This program is called.
5  * It returns a value (0)
6  * This will cause an error when this value is used in a division
7  ****
8  Data Division.
9  Working-Storage Section.
10 Linkage Section.
11 01 Recvd_Parms.
12    05 In-value      Pic 99.
13
14 Procedure Division using Recvd_Parms.
15   Move 0 to In-value.
16   Goback.
17

```

On the right is the "Remote System" browser pane showing a tree structure of files under "zos.dev". A red arrow points from the "REGI0B.cbl" entry in the tree to the editor window. The "REGI0B.cbl" file is highlighted with a red oval.

5.2.5 ► Move the cursor after the **In-value** and press **enter** to add a blank line

The screenshot shows the COBOL editor window with the same code as before. A yellow callout bubble points to the cursor position at the end of line 12, after the "In-value" field. The bubble contains the text: "This is the COBOL editor. Just press enter to add a new line."

5.2.6 You can now take advantage of the **editor content assist...**

► On the column 12 as shown below, type **m** and press **Ctrl + Space**

5.2.7 ► Select the statement **MOVE** (you can use the mouse double click or select and press enter)

The screenshot shows a COBOL editor interface. A code snippet is displayed:

```
Linkage Section.  
01 Recvd-Parms.  
    05 In-value      Pic 99.  
  
Procedure Division using Recvd-Parms.  
    Move 0 to In-value.
```

A context menu is open at the end of the line "Move 0 to In-value.", indicated by a red circle labeled "1". The menu contains several options:

- MERGE
- MOVE** (circled in red, indicated by a red circle labeled "2")
- MULTIPLY
- MULTIPLY - NOT ON SIZE ERROR - END-MULTIPLY
- MULTIPLY - ON SIZE ERROR - END-MULTIPLY
- MULTIPLY - ON SIZE ERROR - NOT ON SIZE ERROR - END-M

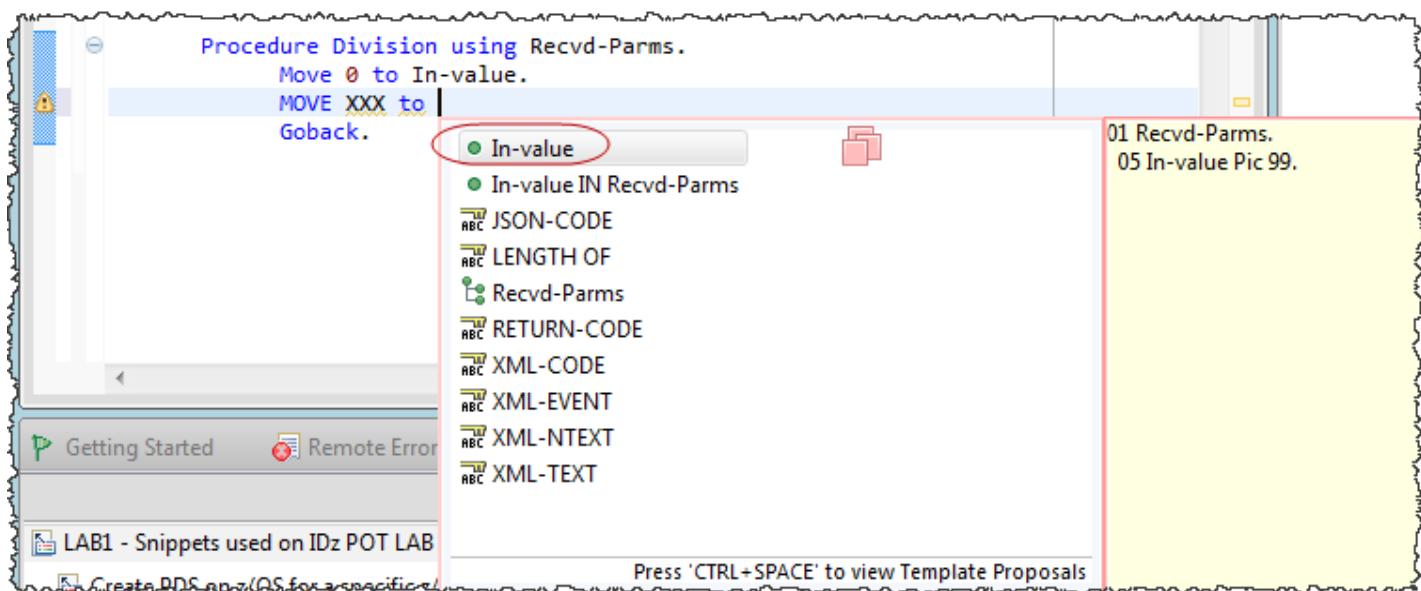
5.2.8 ► Type “XXX to”

The screenshot shows a COBOL editor interface. A code snippet is displayed:

```
Procedure Division using Recvd-Parms.  
    Move 0 to In-value.  
    MOVE XXX to  
    Goback.
```

The line "MOVE XXX to" has been typed and is highlighted with a red oval. The line "Goback." is also present below it.

5.2.9 ► Press **Ctrl + Space** to list the variables and select **In-value**.
 Notice that variables could be defined in copybooks.



The screenshot shows the z/OS COBOL smart editor interface. The code in the editor window is:

```
Procedure Division using Recvd-Parms.
  Move 0 to In-value.
  MOVE XXX to |
```

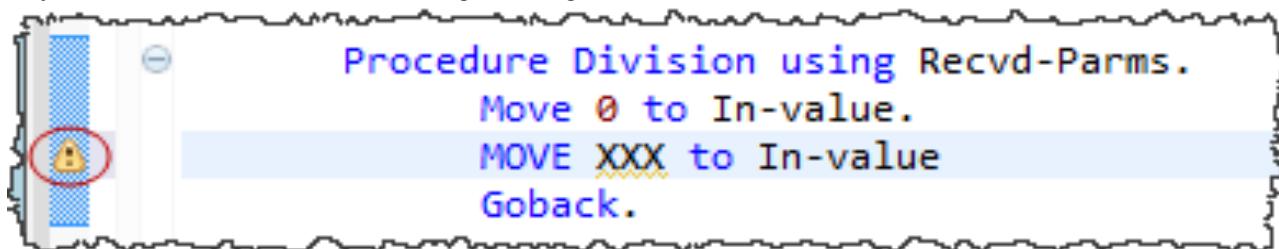
A red oval highlights the word "In-value" in the code. A dropdown menu is open at the cursor position, showing the following options:

- In-value
- In-value IN Recvd-Parms
- RBC JSON-CODE
- RBC LENGTH OF
- Recv-Parms
- RBC RETURN-CODE
- RBC XML-CODE
- RBC XML-EVENT
- RBC XML-NTEXT
- RBC XML-TEXT

A yellow box on the right side of the screen displays the message: "01 Recvd-Parms.
05 In-value Pic 99."

At the bottom of the editor window, there are tabs for "Getting Started" and "Remote Error". A status bar at the bottom says "Press 'CTRL+SPACE' to view Template Proposals".

A yellow mark  shows that something is wrong.

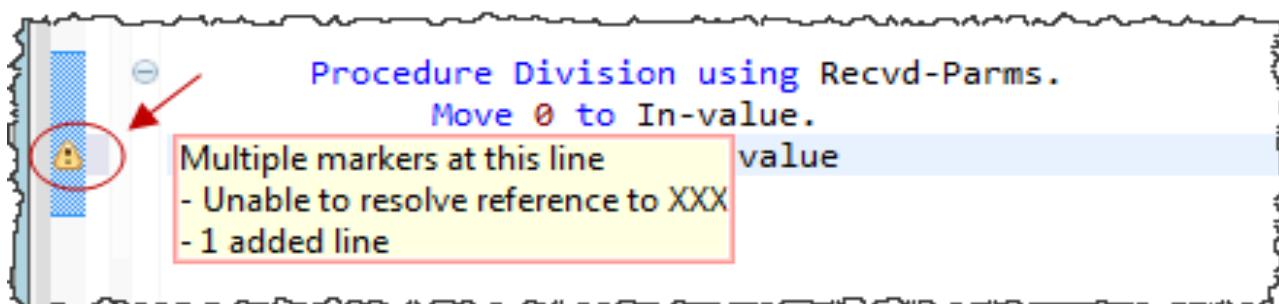


The screenshot shows the z/OS COBOL smart editor interface. The code in the editor window is:

```
Procedure Division using Recvd-Parms.
  Move 0 to In-value.
  MOVE XXX to In-value
  Goback.
```

A yellow warning icon  is placed next to the first "MOVE" statement. The rest of the code is in blue, indicating it is part of a template proposal.

5.2.10 ► Move the cursor to the yellow mark to see what the error is.
 You are finding this error without using the z/OS compilation. If you were using ISPF you would see that only after submitting this JOB to the z/OS COBOL compiler.
 Productivity and z/OS CPU saving. You are using the power of the smart editor.

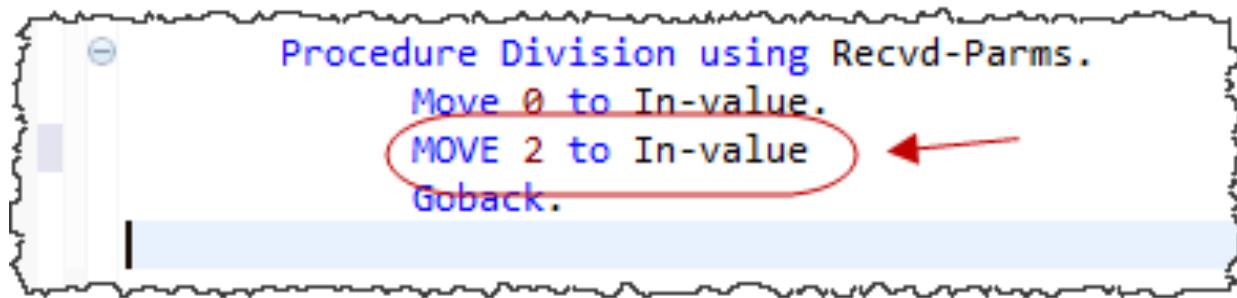


The screenshot shows the z/OS COBOL smart editor interface. A yellow tooltip box appears over a yellow warning icon  on the left margin. The tooltip contains the following text:

Multiple markers at this line value
 - Unable to resolve reference to XXX
 - 1 added line

An arrow points from the text "Multiple markers at this line" towards the yellow warning icon.

5.2.11 ►| Change from **XXX** to **2** as below. Note that the icon  will go away.



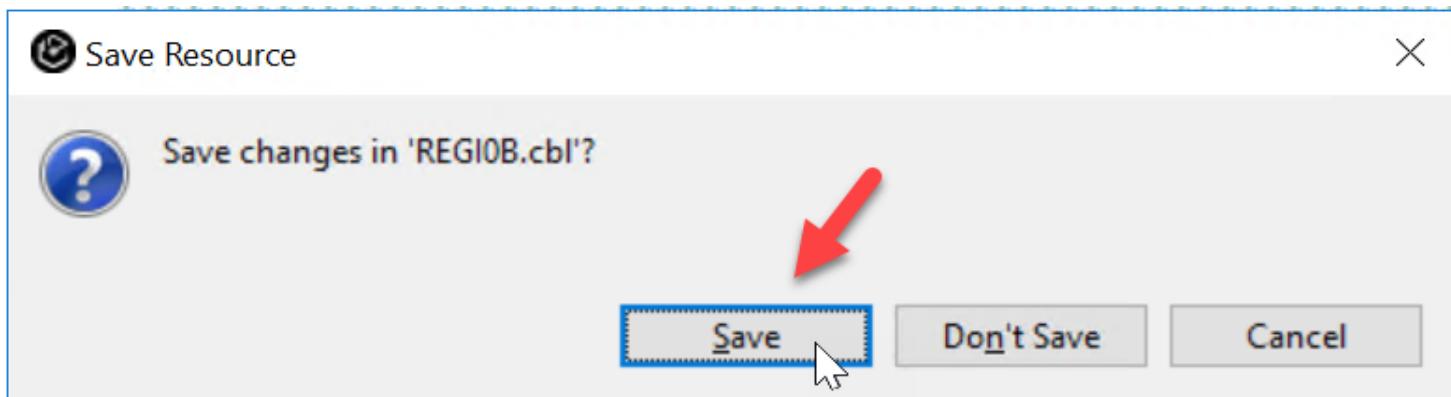
```
Procedure Division using Recvd_Parms.  
Move 0 to In-value.  
MOVE 2 to In-value ←  
Goback.
```

5.3 Compile and link REGI0B.

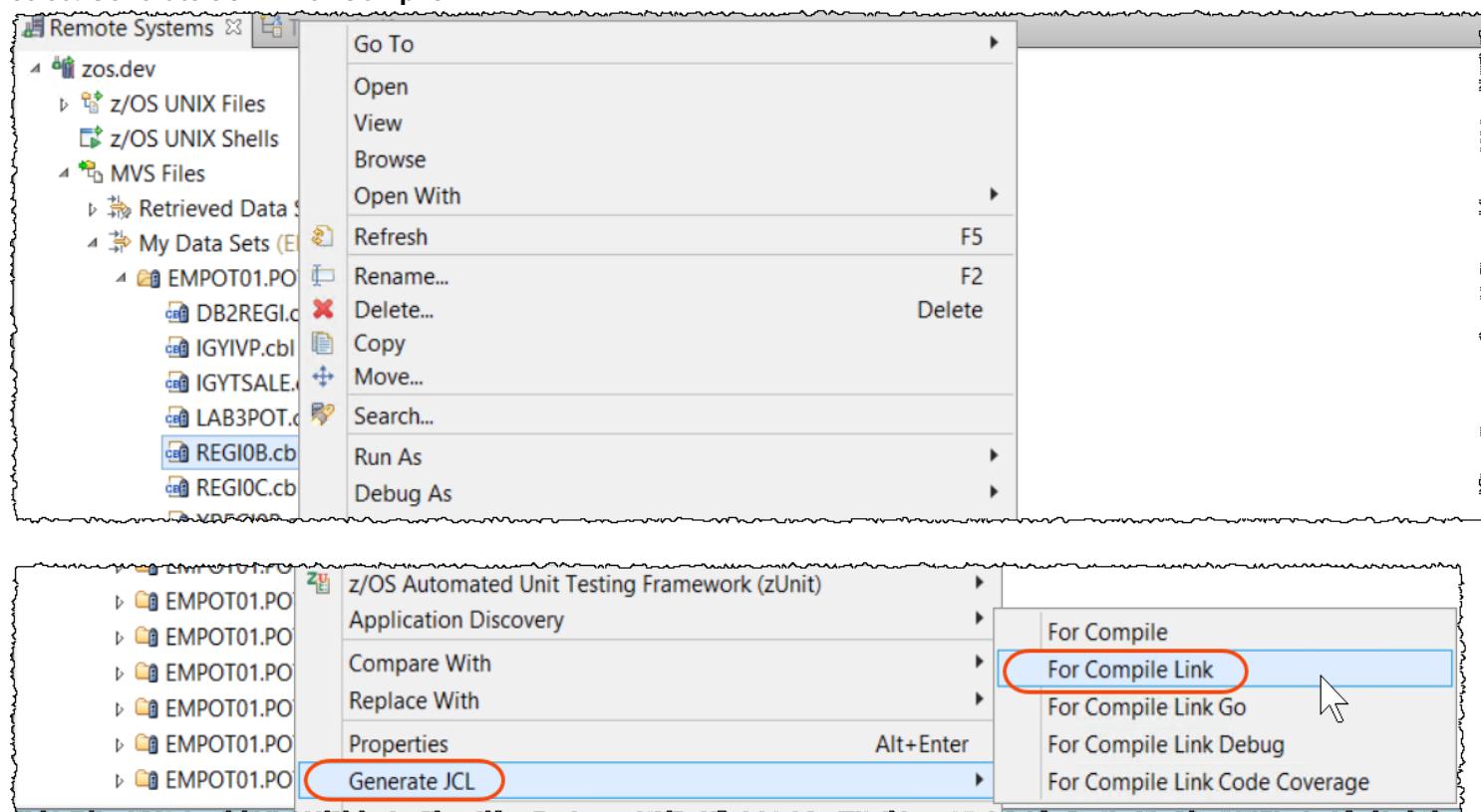
You need to compile and link the program that you just changed. You could use a JCL or let IDz generate a JCL for you. Once you have a Property Group associated to the COBOL code a JCL file could be generated on the fly when you need it. You have associated the property Group at step 7.2.1 above.

5.3.1 ►| Click **Ctrl + Shift + F4** to close the editors. Or just left click on the  of each opened editor.

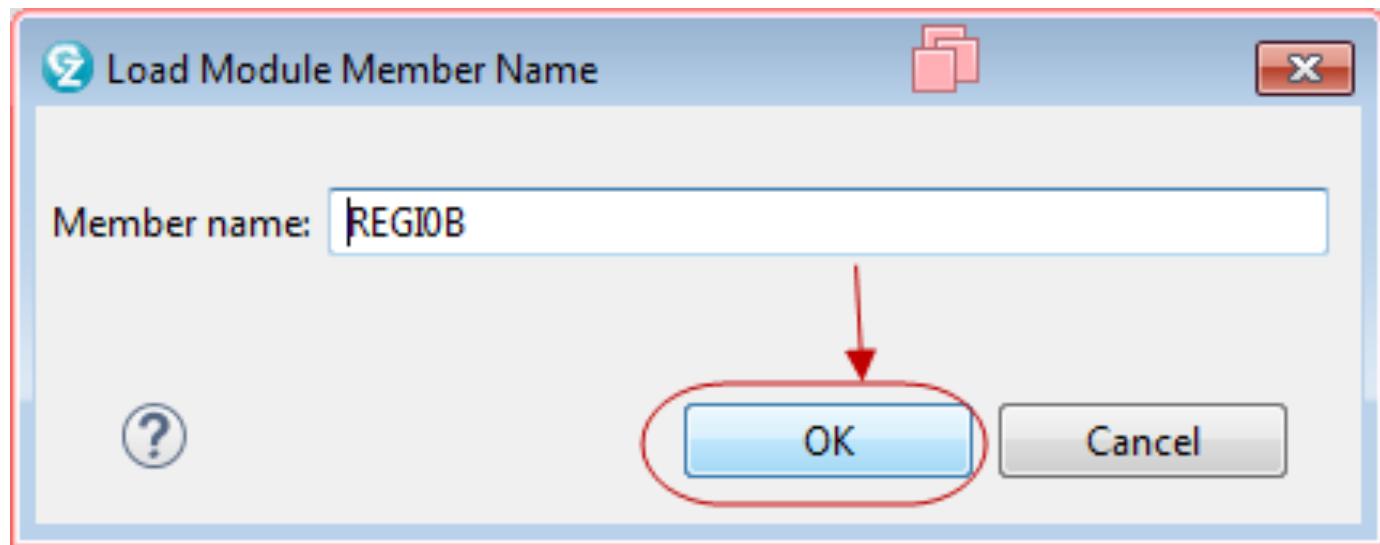
►| Click **Save** to save the changes.



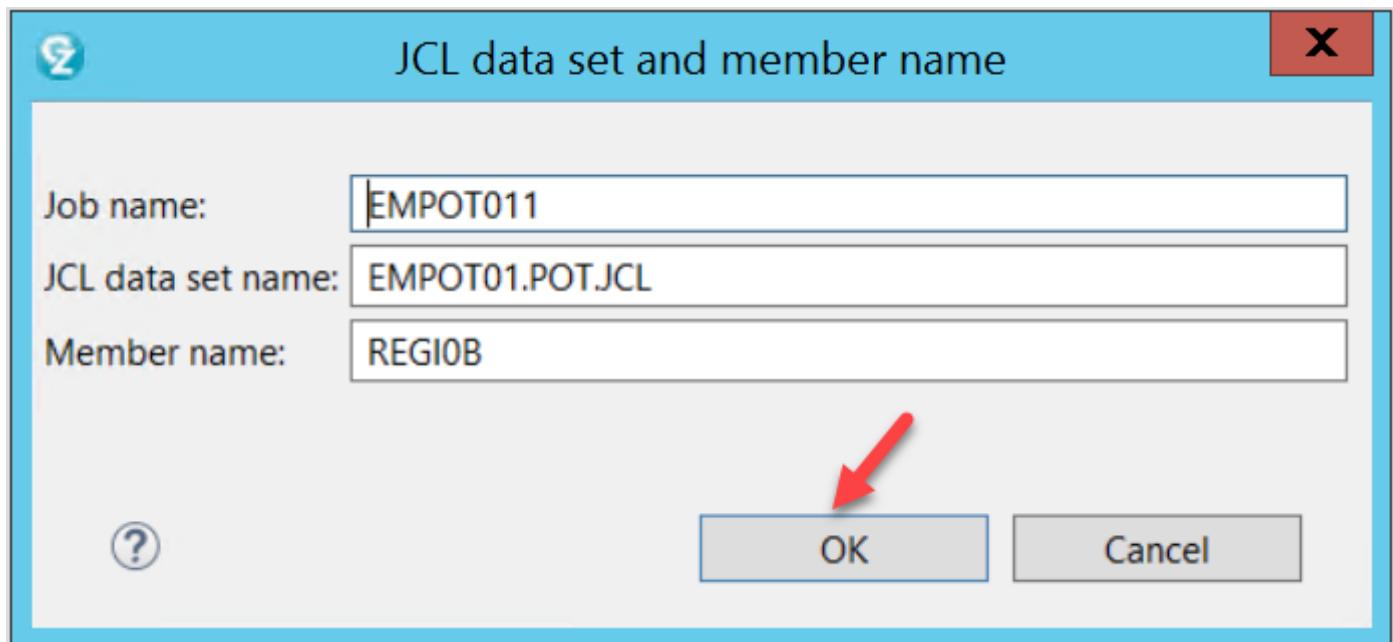
5.3.2 ► Using the Remote Systems view right click on **REGI0B.cbl** and select **Generate JCL > For Compile Link**.



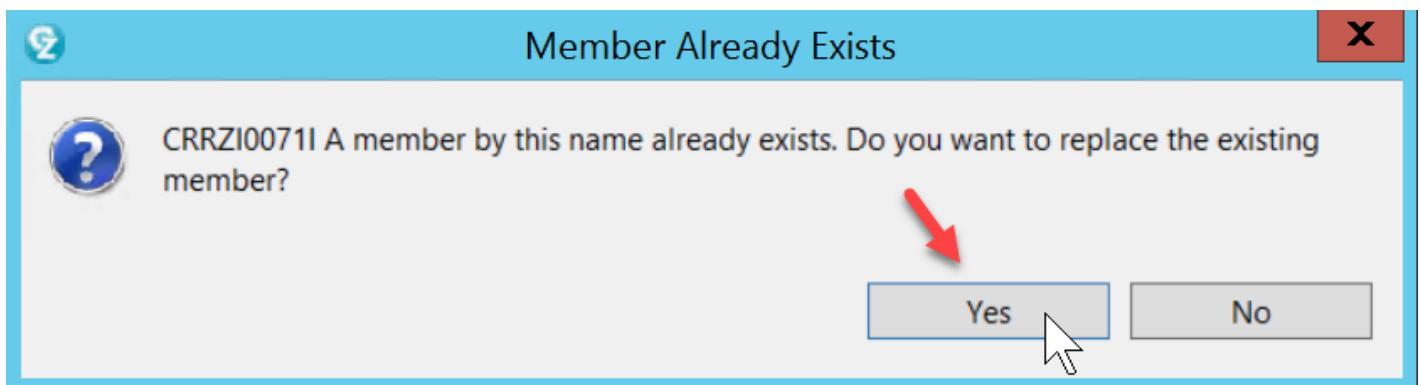
5.3.3 ► Accept the default member name and click **OK**



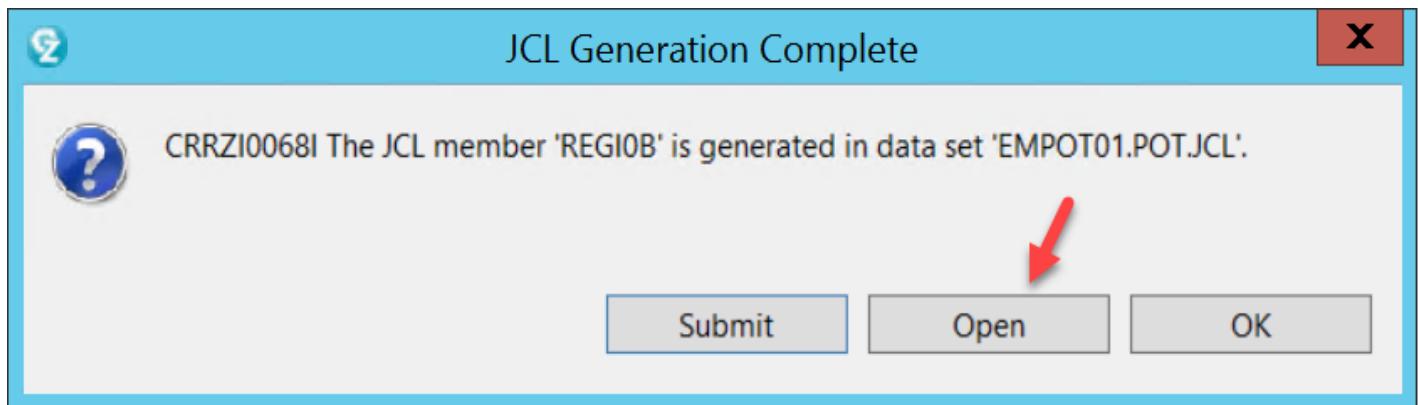
5.3.4 ► Accept the default and click **OK**



5.3.5 ► Click **Yes** if the member already exists



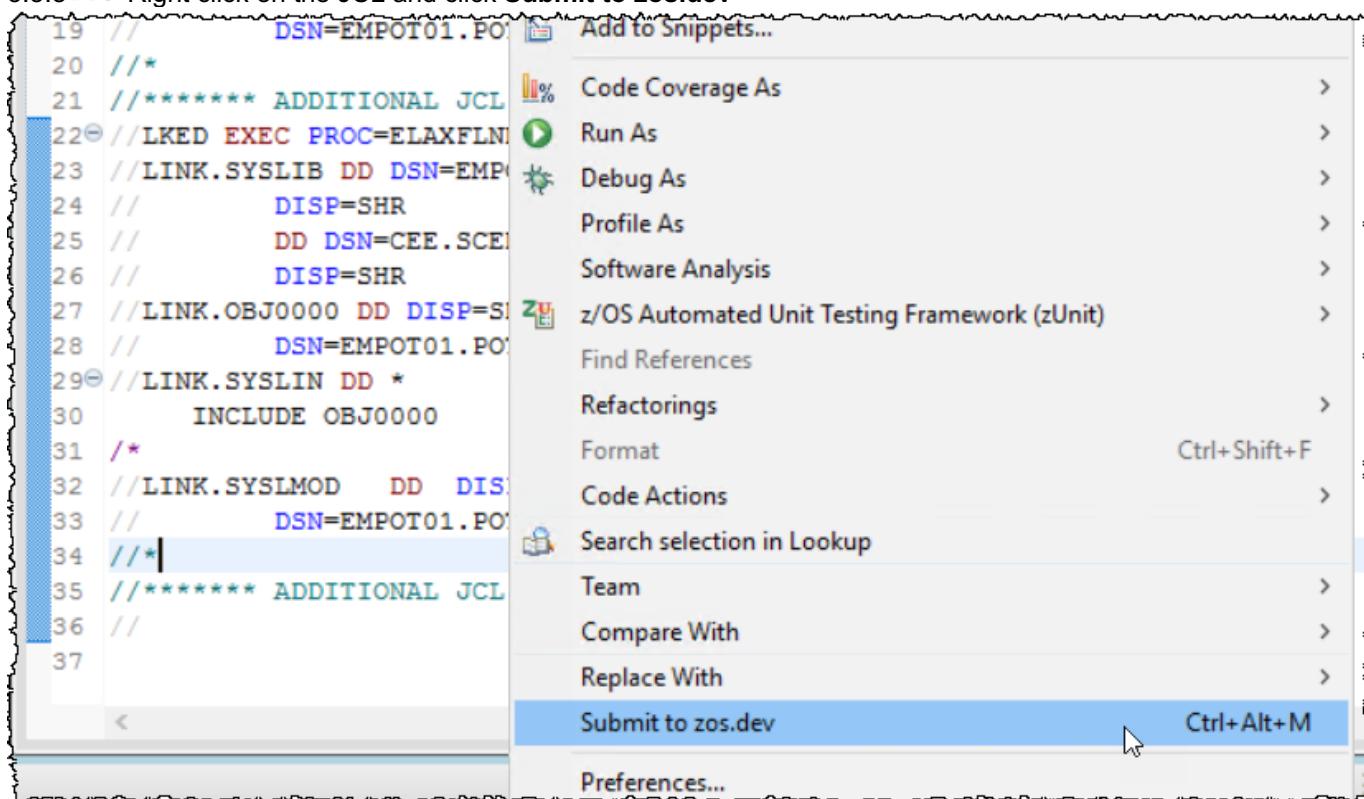
5.3.6 ► Click **Open** to verify the generated JCL



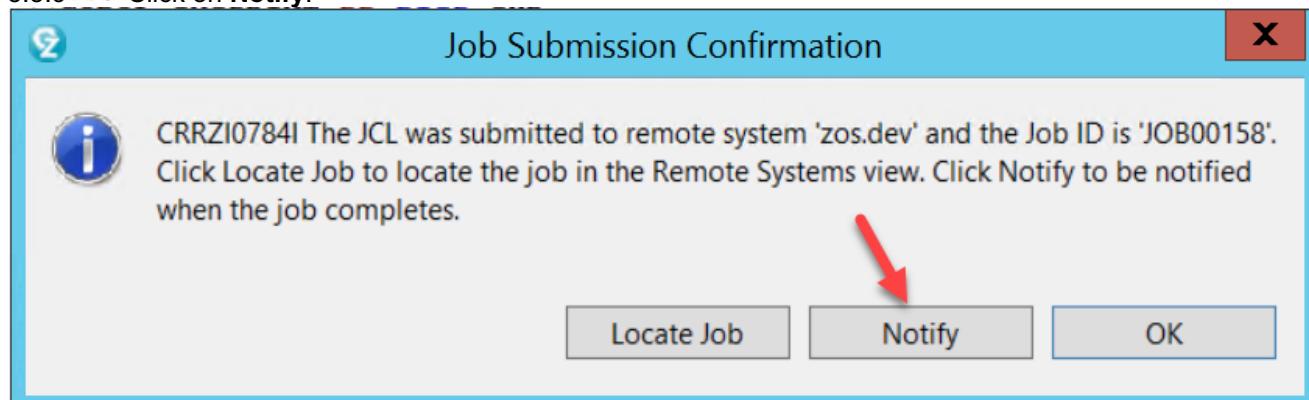
5.3.7 ► Scroll down and notice that the *Load module* will be created on the PDS: **EMPOT01 . POT . LOAD**.

```
REGI0B.jcl
14 //      DD DISP=SHR,
15 //      DSN=EMPOT.ZPICL.COPYLIB
16 // COBOL.SYSXMLSD DD DUMMY
17 // COBOL.SYSIN DD DISP=SHR,
18 //      DSN=EMPOT01.POT.COBOL (REGI0B)
19 //*
20 //***** ADDITIONAL JCL FOR COMPILE HERE *****
21 //LKED EXEC PROC=ELAXFLNK
22 //LINK.SYSLIB DD DSN=EMPOT01.POT.OBJ,
23 //      DISP=SHR
24 //      DD DSN=CEE.SCEELKED,
25 //      DISP=SHR
26 //LINK.OBJ0000 DD DISP=SHR,
27 //      DSN=EMPOT01.POT.OBJ (REGI0B)
28 //LINK.SYSLIN DD *
29     INCLUDE OBJ0000
30 /*
31 //LINK.SYSLMOD DD DISP=SHR,
32 //      DSN=EMPOT01.POT.LOAD (REGI0B) DSN=EMPOT01.POT.LOAD (REGI0B)
33 //*
34 //***** ADDITIONAL JCL FOR LINK HERE *****
35 //
```

5.3.8 ► Right click on the JCL and click **Submit to zos.dev**



5.3.9 ► Click on **Notify**.



5.3.10 You MUST have **000** as return code.

► Click on **EMPOT01:JOB00xxx**

```

20 //***** ADDITIONAL JCL FOR COMPILE HERE *****
21 //LKED EXEC PROC=ELAXFLNK
22 //LINK.SYSLIB DD DSN=EMPOT01.POT.OBJ,
23 //      DISP=SHR

```

[5/28/19, 10:34 AM] Job **JOB00158** is being submitted
[5/28/19, 10:34 AM] Job **EMPOT01:JOB00158** ended with completion code CC 0000

5.3.11 ► Expand **EMPOT01:JOB00xxx** and verify the results double clicking on **LKED:LINK:SYSPRINT**.

► Scroll down and verify that the load module **REGI0B** invoked by your DB2 COBOL program is now created in **EMPOT01.POT.LOAD**.

```

298 SAVE OPERATION SUMMARY:
299
300 MEMBER NAME          REGI0B
301 LOAD LIBRARY         EMPOT01.POT.LOAD
302 PROGRAM TYPE        PROGRAM OBJECT (FORMAT 3)
303 VOLUME SERIAL        C2DBAR
304 DISPOSITION          ADDED NEW
305 TIME OF SAVE         09.33.57 MAY 28, 2019
306
307
308 SAVE MODULE ATTRIBUTES:
309
310 AC                  000
311 AMODE               31
312 COMPRESSION          NONE
313 DC                  NO
314 EDITABLE             YES
315 EXCEEDS 16MB         NO
316 EXECUTABLE           YES
317 LONGPARM             NO
318 MIGRATABLE           NO

```

REGI0B is now on EMPOT01.POT.LOAD

5.3.10 ► Use **Ctrl + Shift + F4** to close all editors.

Or just click on the of each opened editor

5.4 Execute the COBOL/DB2 program

On this step, you will explore some capabilities of the JCL editor and modify the JCL for execution.

5.4.1 ► Using *Remote System View*, scroll back to locate the file **pot01run.jcl** under the folder **Local/Local Files/ADF_POT/empot01** and **double click** to edit.

```

JCL pot01run.jcl
-----+-----+-----+-----+-----+-----+-----+
1 //POT01RUN JOB ,                         |-----+-----+-----+-----+-----+-----+-----+-----+
2 // MSGCLASS=H,MSGLEVEL=(1,1),TIME=1440,REGION=70M,COND=(16,LT) |-----+-----+-----+-----+-----+-----+-----+-----+
3 //*      ZPICL Debug |-----+-----+-----+-----+-----+-----+-----+-----+
4 //CURSRAV4 EXEC PGM=IKJEFT01,DYNAMNBR=20 |-----+-----+-----+-----+-----+-----+-----+-----+
5 //STEPLIB   DD DSN=EMPOT01.POT.LOAD,DISP=SHR |-----+-----+-----+-----+-----+-----+-----+-----+
6 //          DD DSN=EMPOT.ZPICL.LOAD,DISP=SHR |-----+-----+-----+-----+-----+-----+-----+-----+
7 //*          DD DISP=SHR,DSN=IBMUUSER.VER1.SEQAMOD |-----+-----+-----+-----+-----+-----+-----+-----+
8 //*          DD DISP=SHR,DSN=IBMUUSER.VER1.SEQAAUTH |-----+-----+-----+-----+-----+-----+-----+-----+
9 //          DD DSN=DSNB10.SDSNLOAD,DISP=SHR |-----+-----+-----+-----+-----+-----+-----+-----+
10 //         DD DSN=DSNB10.DBDBG.RUNLIB.LOAD,DISP=SHR |-----+-----+-----+-----+-----+-----+-----+-----+
11 //         DD DISP=SHR,DSN=FEK910.SFEKAUTH |-----+-----+-----+-----+-----+-----+-----+-----+
12 //SYSPRINT DD SYSOUT=* |-----+-----+-----+-----+-----+-----+-----+-----+
13 //SYSOUT    DD SYSOUT=* |-----+-----+-----+-----+-----+-----+-----+-----+
14 //SYSTSPRT  DD SYSOUT=* |-----+-----+-----+-----+-----+-----+-----+-----+
15 //SYSTSIN   DD * |-----+-----+-----+-----+-----+-----+-----+-----+
16 TSOLIB ACTIVATE DA('DSNB10.SDSNLOAD') |-----+-----+-----+-----+-----+-----+-----+-----+
17 DSN SYSTEM(DBDBG) |-----+-----+-----+-----+-----+-----+-----+-----+
18 RUN PROGRAM(DB2REGI) PLAN(DB2REGI) - |-----+-----+-----+-----+-----+-----+-----+-----+
19 LIB('EMPOT.ZPICL.LOAD') |-----+-----+-----+-----+-----+-----+-----+-----+
20 END |-----+-----+-----+-----+-----+-----+-----+-----+
21 /* |-----+-----+-----+-----+-----+-----+-----+-----+
22 //
```

5.4.2 Notice on bottom left corner the same *Outline* view that you have for the COBOL program.
It will help to navigate on large JCL members.

► Expand **CURSRAV4 EXEC PGM=** and click on **STEPLIB**

5.4.3 ► Notice that your PDS where **REG10B** is created is already on the STEPLIB.

The screenshot shows the z/OS Studio interface with two main windows:

- z/OS Projects** window (left):
 - Project: LAB1B
 - File: COBOL_DB2 [zos.dev]
 - File: EMPOT01.POT.COBOL(DB2REGI).cbl
- JCL pot01run.jcl** window (right):


```

1 //POT01RUN JOB ,
2 // MSGCLASS=H,MSGLEVEL=(1,1),TIME=1440,REGION=70M,COND=(16,LT)
3 //*
4 //CURSRAV4 EXEC PGM=IKJEFT01,DYNAMNBR=20
5 //STEPLIB DD DSN=EMPOT01.POT.LOAD,DISP=SHR
6 // DD DSN=EMPOT.ZPICL.LOAD,DISP=SHR
7 // DD DSN=DSNB10.SDSNLOAD,DISP=SHR
8 // DD DSN=DSNB10.DBDBG.RUNLIB.LOAD,DISP=SHR
9 // DD DISP=SHR,DSN=FEK910.SFEKAUTH
10// DD DISP=SHR,DSN=DSNB10.DBDBG.RUNLIB.LOAD,DISP=SHR
11// DD DISP=SHR,DSN=FEK910.SFEKAUTH
12// SYSPRINT DD SYSOUT=*
13// SYSOUT DD SYSOUT=*
14// SYSTSPPRT DD SYSOUT=*
15// SYSTSIN DD *
16 TSOLIB ACTIVATE DA('DSNB10.SDSNLOAD')
17 DSN SYSTEM(DBDBG)
18 RUN PROGRAM(DB2REGI) PLAN(DB2REGI) -
19 LIB('EMPOT.ZPICL.LOAD')
20 END
21 /*
22 //
      
```

Annotations in the screenshot:

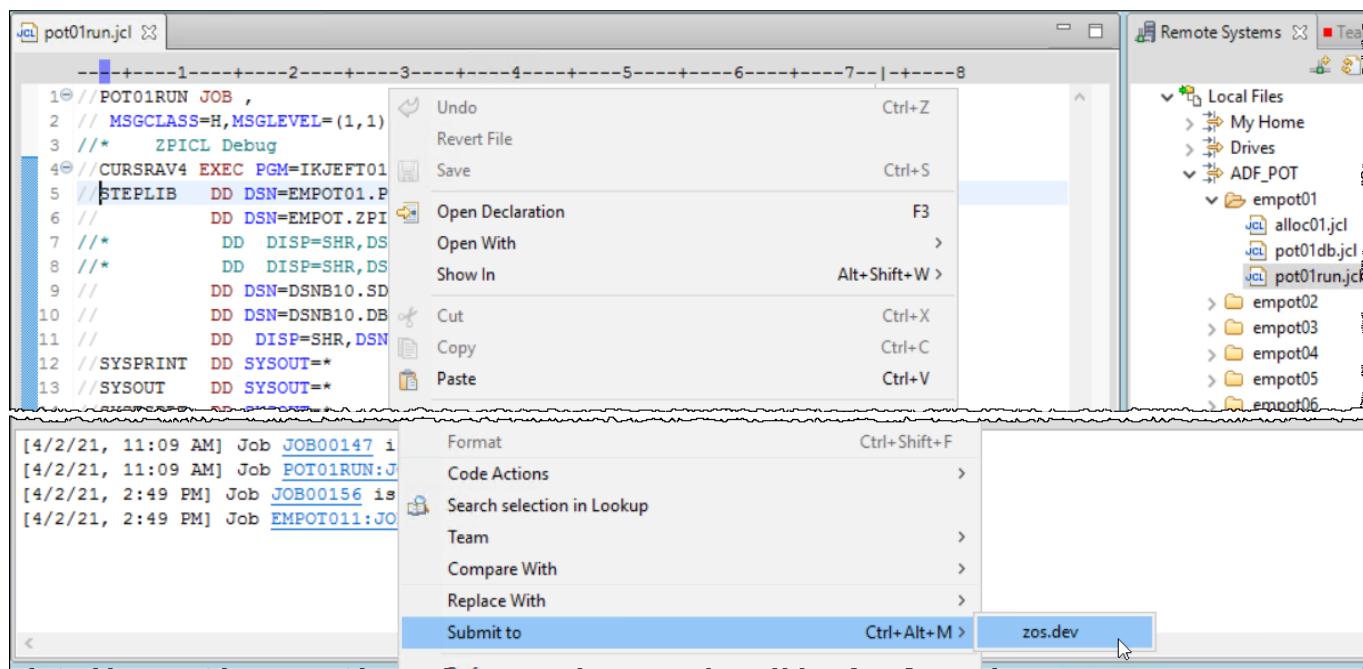
- A red arrow points from the line `5 //STEPLIB DD DSN=EMPOT01.POT.LOAD,DISP=SHR` in the JCL editor to the corresponding entry in the **Properties** view.
- A red circle with the number **1** is placed over the **Properties** tab in the bottom-left corner.
- A red circle with the number **2** is placed over the `STEPLIB DD DSN=EMPOT01.POT.LOAD,DISP=SHR` entry in the **Properties** view.

Output window (bottom right):

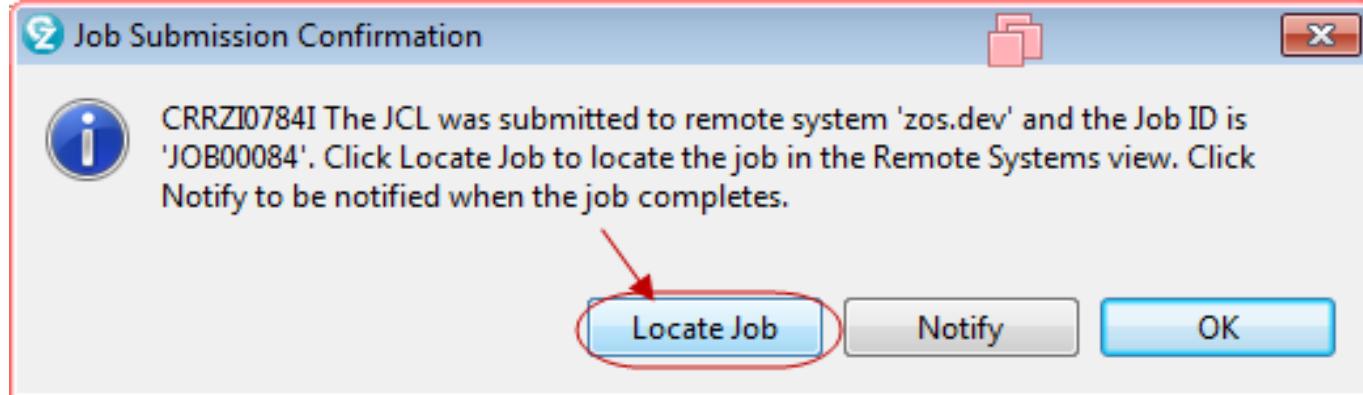
```

[5/28/19, 10:34 AM] Job JOB00158 is being submitted
[5/28/19, 10:34 AM] Job EMPOT011:JOB00158 ended with completion code
      
```

5.4.4 ► Right click and select **Submit to > zos.dev**



5.4.5 ► Click on **Locate Job**. The job submitted will be shown under JES folder under Remote Systems view



5.4.6 You MUST have the return code 0000 and the bug will be corrected.

► Using *Remote Systems* view, verify the results double expanding **POT01RUN:JOB00xxx** (where **00xxx** can be any number) and clicking on **CURSRAV4:SYSOUT**. Now you MUST have 000 as return code
Tip: You may need to refresh Retrieved Jobs to see it go from active to finished

The screenshot shows a Rational Application Developer interface. On the left is a code editor window titled 'pot01run.jcl' containing COBOL code. The right side shows a 'Remote Systems' view with a file tree.

```

33 *DEPT*      REG
34 *PERF-AVG*  0000000
35 *PERF-MIN*  0009
36 *PERF-MAX*  0009
37 *HOURS-AVG* 0002675
38 *HOURS-MIN* 0002675
39 *HOURS-MAX* 0002675
40 REG    .00   9.00   9.00   26.75   26.75   26.75
41 *DEPT*      N/A
42 *PERF-AVG*  0000000
43 *PERF-MIN*  0000
44 *PERF-MAX*  0000
45 *HOURS-AVG* 0003545
46 *HOURS-MIN* 0003545
47 *HOURS-MAX* 0003545
48 N/A    .00   .00   .00   35.45   35.45   35.45
49 *** END - OF - DATA ***
50 * * *      ROWS READ --> 06
51 The result is ... 33
52 Thanks to ????????????????????? attending this
53 BRANCHFLAG GREATER THAN 1
54 EXECUTED SEEYA PARAGRAPH
55 EXECUTED SEEYA PARAGRAPH
56 EXECUTED GOODBYE PARAGRAPH

```

A red box highlights the output from lines 51 to 56. Red numbers 1, 2, and 3 point to specific items in the file tree:

- 1 points to 'POT01RUN:JOB00113 [CC 0000]' which is circled.
- 2 points to 'CURSRAV4::SYSOUT [0000]' which is circled.
- 3 points to 'CURSRAV4::SYSTSPRT [0000]' which is circled.

The file tree includes:

- LAB2C
- LAB2D
- LAB7
- LAB8
- snippets
 - To_COPY_and_PASTE.txt
- My Favorites
- Local Shells
- zos.dev
 - z/OS UNIX Files
 - z/OS UNIX Shells
 - MVS Files
 - TSO Commands
 - JES
 - Retrieved Jobs
 - POT01RUN:JOB00113 [CC 0000] (circled)
 - JES2::JESMSGLG
 - JES2::JESJCL
 - JES2::JESYMSG
 - CURSRAV4::SYSTSIN [0000]
 - CURSRAV4::SYSOUT [0000] (circled)
 - CURSRAV4::SYSTSPRT [0000]
 - POT01CC:JOB00092 [CC 0000]

5.4.7 ► Use **Ctrl + Shift + F4** to close all editors. Or just click on the of each opened editor

What have you done so far?

You used **Fault Analyzer** to identify why the program was abending.

You used the **Debugger** to temporarily fix the error caused by the called program REGI0B.

You used IDz to change the program REGI0B to return other value than zero.

You compiled and linked REGI0B creating another version on your load library.

You executed again the JOB but now using the REGI0B that you created and verified that the abend is gone.



Section 6. Using the Code coverage

Code coverage analyzes a running program and generates a report of statements that were executed, compared to the total number of executable lines.

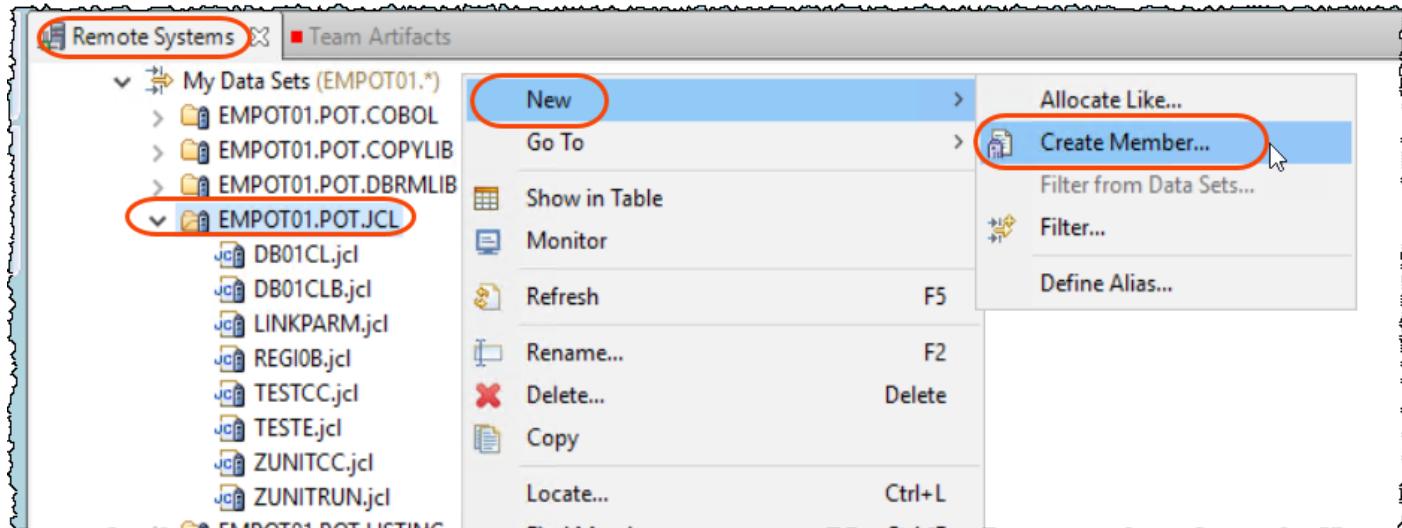
Developer for z Systems Host Utilities provides two ways to invoke Code coverage in batch mode, A sample JCL procedure, to process a single program run, and a set of scripts to start and stop a permanently active Code coverage collector that can process multiple program runs.

You can run code coverage for any application you can debug. You can generate code coverage reports that you can view in the workbench or save the results to your file system for future analysis. We will show a simple batch mode invocation.

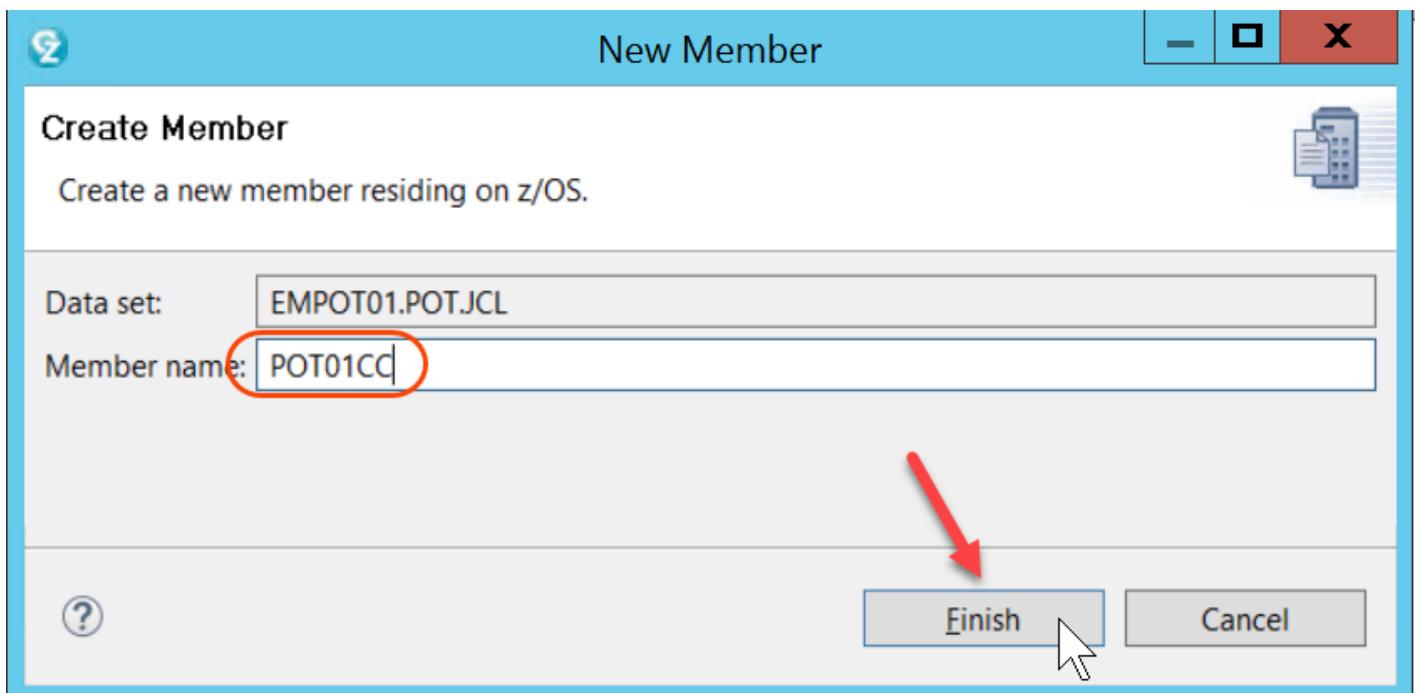
6.1 Creating a JCL to run the code coverage

You will create a JCL file to run Code Coverage

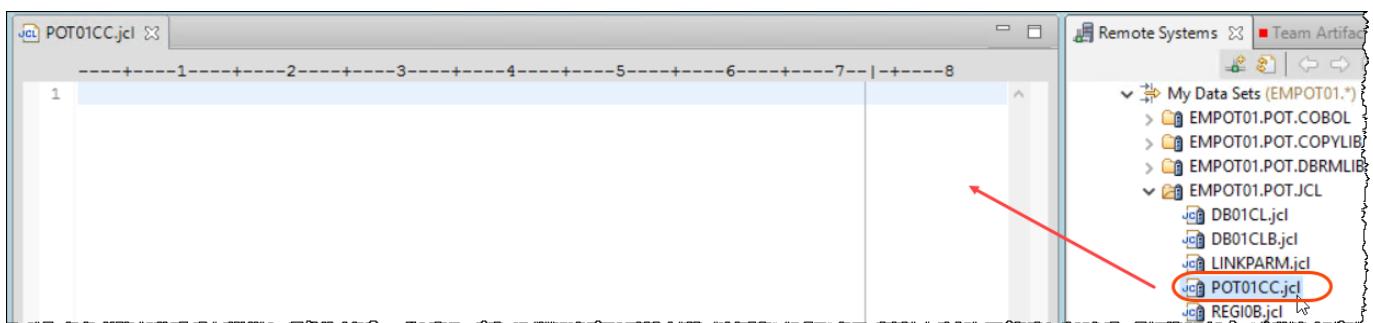
6.1.1 ► Using Remote Systems view, right click on **EMPOT01.POT.JCL** and select **New > Create Member...**



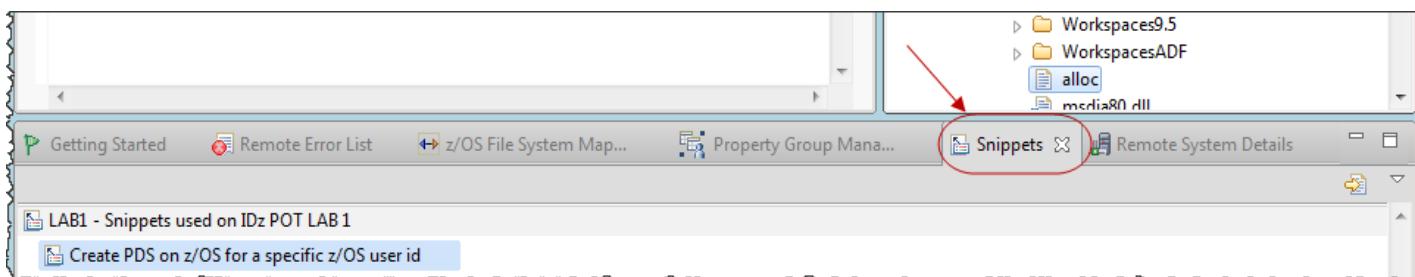
6.1.2 ► Name it as **POT01CC** and click **Finish**



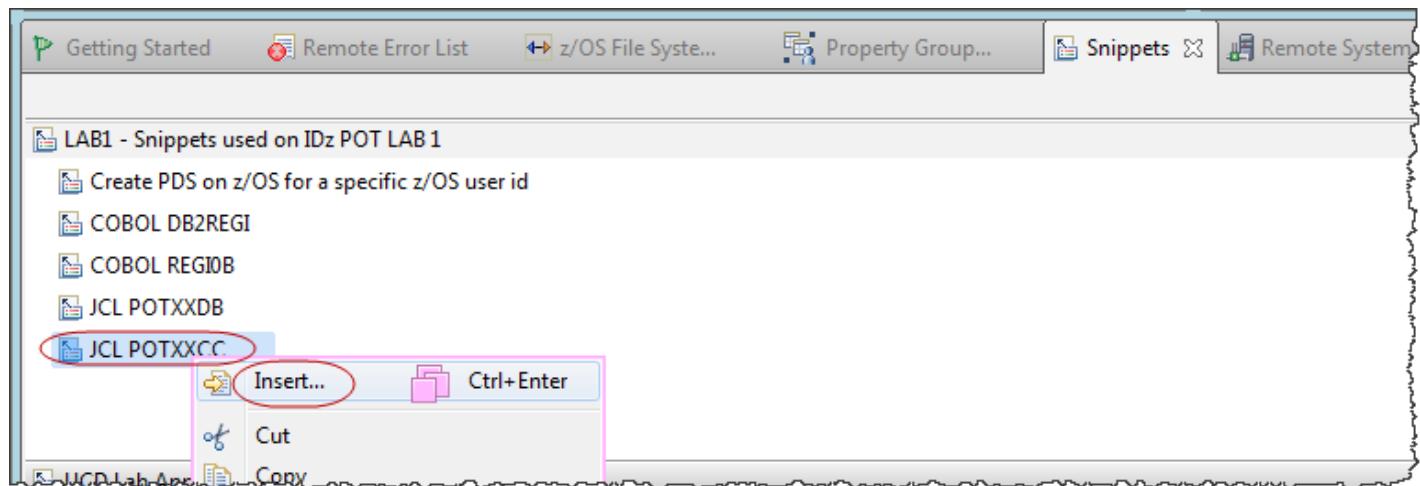
6.1.3 ► Double click on **POT01CC** to edit it



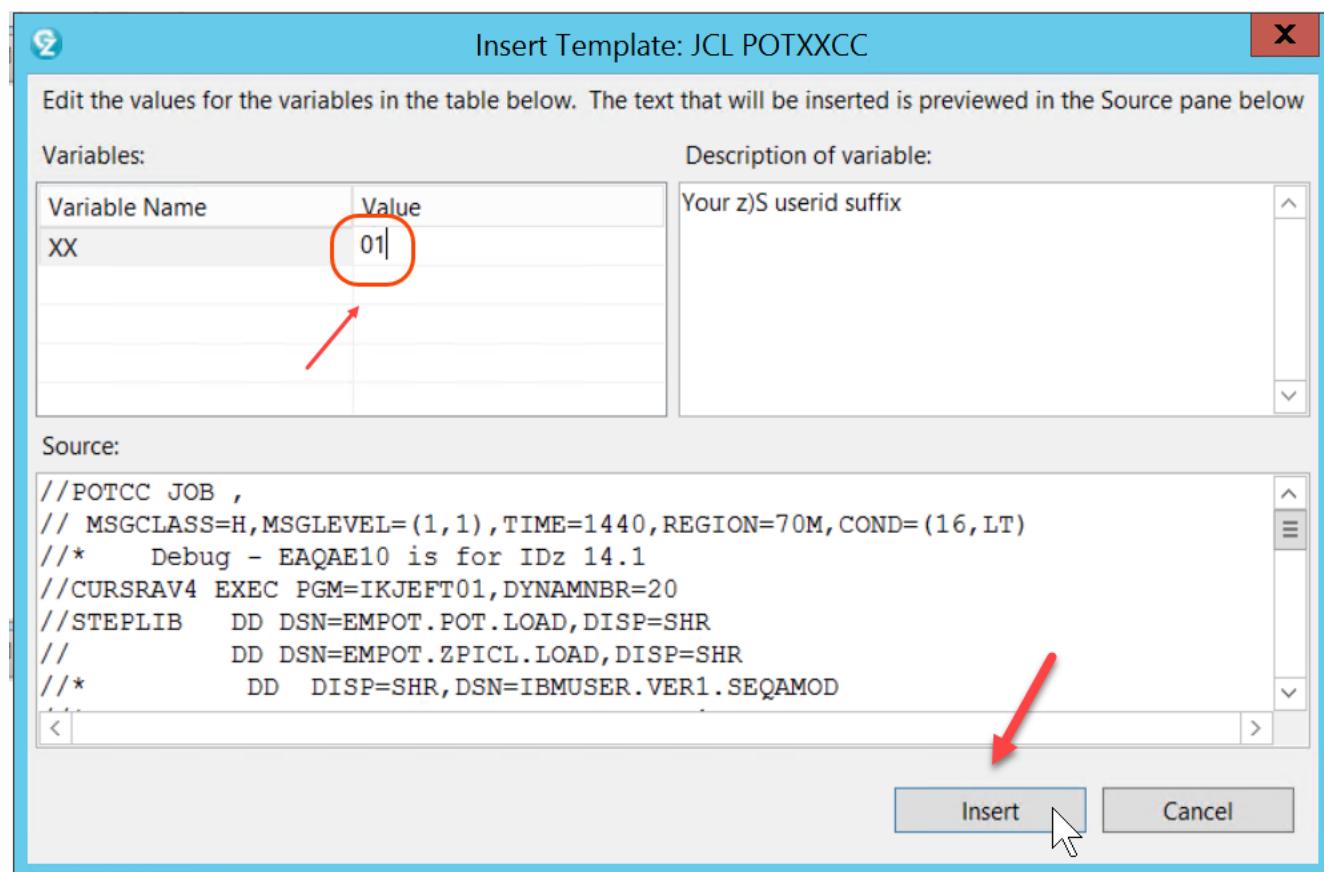
6.1.4 ► Click on **Snippets** tab on the bottom...



6.1.5 ► Right click on **JCL POTXXCC** and select **Insert...**



6.1.6 ► When the *Insert* dialog opens, type **01** in the *Value* and click **Insert**



6.1.7 . Notice that the Snippets variables will be replaced.

►| Use **Ctrl + S** to save the changes.

```

jcl *POT01CC.jcl X
-----+---1---+---2---+---3---+---4---+---5---+---6---+---7---
1 //POT01CC JOB ,
2 // MSGCLASS=H,MSGLEVEL=(1,1),TIME=1440,REGION=70M,COND=(16,LT)
3 //*      Debug - EAQAE10 is for IDz 16
4 //CURSRAV4 EXEC PGM=IKJEFT01,DYNAMNBR=20
5 //STEPLIB   DD DSN=EMPOT01.POT.LOAD,DISP=SHR
6 //           DD DSN=EMPOT.ZPICL.LOAD,DISP=SHR
7 //           DD DISP=SHR,DSN=IBMUSER.VER1.SEQAMOD
8 //           DD DISP=SHR,DSN=IBMUSER.VER1.SEQAAUTH
9 //
10 //          DD DSN=DSNC10.SDSNLOAD,DISP=SHR
11 //          DD DSN=DSNC10.DBCG.RUNLIB.LOAD,DISP=SHR
12 //          DD DISP=SHR,DSN=FEK910.SFEKAUTH
13 //          DD DISP=SHR,DSN=IDZEE.V16R0.IDBG.SEQAMOD
14 //SYSPRINT DD SYSOUT=*
15 //SYSOUT    DD SYSOUT=*
16 //SYSTSPRT DD SYSOUT=*
17 //SYSTSIN   DD *
18     TSOLIB ACTIVATE DA('DSNC10.SDSNLOAD')
19     DSN SYSTEM(DBCG)
20     RUN PROGRAM(DB2REGI) PLAN(DB2REGI) -
21       LIB('EMPOT.ZPICL.LOAD')
22 END
23 /*
24 //***** Launch Debug Tool - COMMENTS IS FOR IDz 16
25 //CEEOPTS DD *
26   TEST(,,,DBMDT:*)
27   ENVAR("AQE_STARTUP_KEY=CC,DB2REGI")
28 /*    TEST(ALL,,PROMPT,DBMDT:*)
29 /*    ENVAR("EQA_STARTUP_KEY=CC")
30 /*

```

6.2 Submit the JCL for z/OS execution

You will now execute the fixed batch COBOL/DB2 on z/OS

6.2.1 ►| Right click on the JCL being edited and select **Submit to zos.dev**

The screenshot shows the IBM Rational Developer for z/OS interface. The main window displays a COBOL source file named 'POT01CC.jcl'. The code includes various JCL statements like //POT01CC JOB, //MSGCLASS=H, //CURSRAV4 EXEC PGM=IKJEFT01, and several DD statements. A context menu is open over the code, listing options such as Undo, Save, Open Declaration, Show In, Cut, Copy, Paste, Quick Fix, Source, Search Text in File..., Add to Snippets..., Code Coverage As, Coverage As, Run As, Debug As, Profile As, Software Analysis, Team, Compare With, Replace With, Submit to zos.dev, and Preferences... . The 'Submit to zos.dev' option is highlighted in blue. At the bottom of the interface, there is a toolbar with icons for Console, Problems, z/OS Debugger Profiles, and Code Coverage. Below the toolbar, a 'Workspace Results' section shows a single entry: 'DB2REGI_2024_04_23_101658_0792'.

6.2.2 ➔ When the dialog pops up, select **Switch**

POT01CC.jcl DB2REGI.DB2REGI.cob X

```

1 1 IDENTIFICATION DIVISION.
2 2 PROGRAM-ID. DB2REGI.
3 3 *REMARKS. THIS PROGRAM JOINS TABLES, GROUPS DATA BY DEPT,
4 4 * AND DISPLAYS THE AVERAGE, MAXIMUM AND MINIMUM
5 5 * HOURS, AND PERFORMANCE EVALUATION BY DEPT.
6 6 * Modified by Regi to add DEAD CODE - Jan/2-14
7 7 * Modified by Regi to added test if -204 - Mar/2015
8 8 ENVIRONMENT DIVISION.
9 9 CONFIGURATION SECTION.
10 10 SOURCE-COMPUTER. IBM-370.
11 11 OBJECT-COMPUTER. IBM-370.
12 12 DATA DIVISION.
13 13 WORKING-STORAGE SECTION.
14 14 * CODE THE NECESSARY DB2 INCLUDE STATEMENTS HERE
15 15 01 WS
16 16 05
17 17 05
18 18 05
19 19 05
20 20 * -----
21 21 01 W-
22 22 * MODIF
23 23 * PROG
24 24 *
25 25 * Remember my decision
26 26 *
27 27 * YOU W
28 28 * VALUES IN THE EMPL TABLE; ONE FOR DEPT, AND ONE FOR PERF.
29 29 * QUESTION . .
30 30 * EXEC SQL INCLUDE SQLCA END-EXEC.
31 31 01 SQLCA GLOBAL VOLATILE.

```

Confirm Perspective Switch

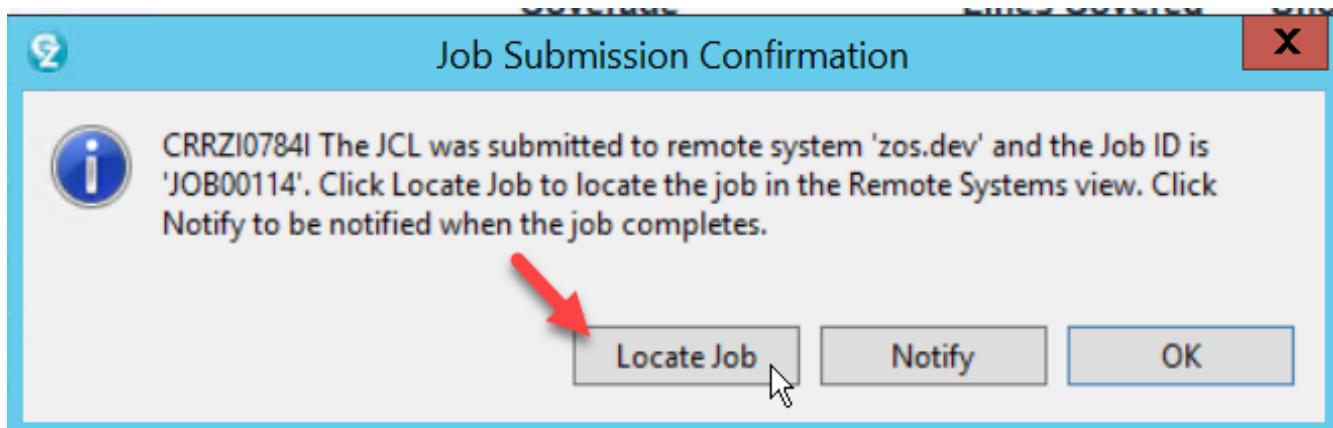
This kind of launch is configured to open the Debug perspective when it suspends.
This Debug perspective supports application debugging by providing views for displaying the debug stack, variables and breakpoints.

Switch to this perspective?

Remember my decision

Switch No

6.2.3 ► When the dialog pops up, select **Locate Job**



6.2.4 Debug will be start and the COBOL PGM will show up your right. See on your left **DB2REGI:01 Thread**.

The screenshot shows the IBM Rational Application Developer interface. On the left, the Project Explorer displays a 'DB2REGI [Incoming Remote Debug Session]' node, which contains a 'Thread:1 (Runnable)' node. A red arrow points to this 'Thread:1' node. Below it, the process ID 'Process: 537413824 Program: DB2REGI' is listed. The main workspace shows a COBOL source code editor with the file 'DB2REGI.DB2REGI.cob'. The code is as follows:

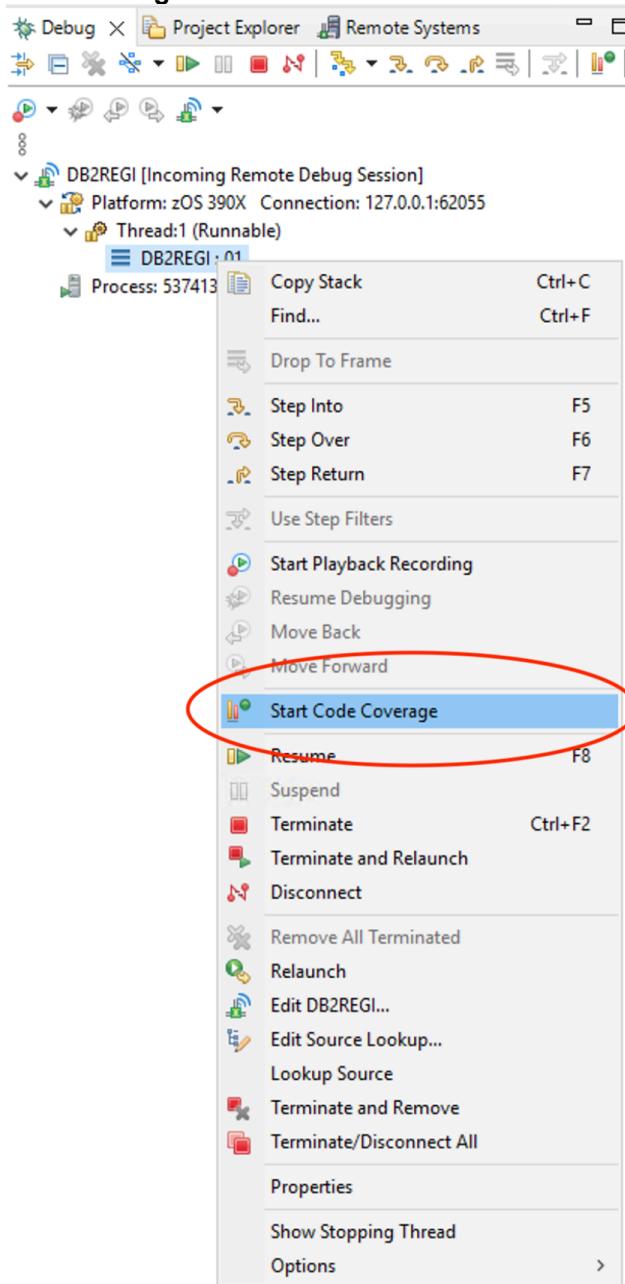
```

1 IDENTIFICATION DIVISION.
2 PROGRAM-ID. DB2REGI.
3 *REMARKS. THIS PROGRAM JOINS TABLES, GROUPS DATA BY DEPT,
4 * AND DISPLAYS THE AVERAGE, MAXIMUM AND MINIMUM
5 * HOURS, AND PERFORMANCE EVALUATION BY DEPT.
6 * Modified by Regi to add DEAD CODE - Jan/2-14
7 * Modified by Regi to added test if -204 - Mar/2015
8 ENVIRONMENT DIVISION.
9 CONFIGURATION SECTION.
10 SOURCE-COMPUTER. IBM-370.
11 OBJECT-COMPUTER. IBM-370.
12 DATA DIVISION.
13 WORKING-STORAGE SECTION.
14 * CODE THE NECESSARY DB2 INCLUDE STATEMENTS HERE
15 01 WS-KTRS-SWITCHES.
16    05 ROW-KTR    PIC S9(03) COMP-3 VALUE +0.
17    05 PRF        PIC S9(03) VALUE +0.
18    05 DPT        PIC X(9).
19    05 HOURS      PIC S9(03) VALUE +0.
20 * ---- Added by REGI
21 01 W-SQLCA      PIC S9(9) COMP.
22 * MODIFY THE TABLE-ROW PICTURE CLAUSES FOR THE HOST
23 * PROGRAM VARIABLES - LOOK AT THE TABLE/COLUMN DEFINITIONS
24 * IN YOUR MANUAL (APPENDIX A FROM THE SQL PORTION)
25 * AND THE SQL TO COBOL DATATYPES IN CHAPTER 3
26 *
27 * YOU WILL NEED TWO NULL INDICATORS - BASED ON POTENTIAL NULL
28 * VALUES IN THE EMPL TABLE; ONE FOR DEPT, AND ONE FOR PERF.
29 * QUESTION . .
30 * EXEC SQL INCLUDE SQLCA END-EXEC.
31 01 SQLCA GLOBAL VOLATILE.
32    05 SQLCAID PIC X(8).
33    05 SQLCABC PIC S9(9) COMP-5.
34    05 SQLCODE PIC S9(9) COMP-5.
35    05 SQLERRM.
36      49 SQLERRML PIC S9(4) COMP-5.
37      49 SQLERRMC PIC X(70).

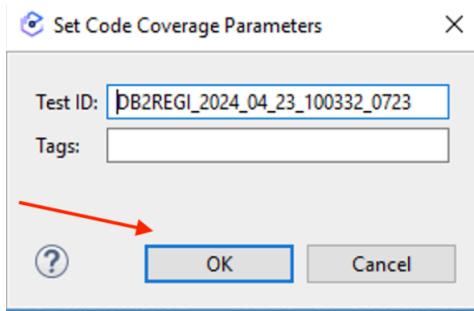
```

At the bottom of the interface, there are tabs for 'Console', 'Registers', 'Debug Console', 'Problems', 'z/OS Debugger Profiles', and 'Memory'. The 'Registers' tab is currently selected.

6.2.5 ► Right Click on DB2REGI:01 Thread and select Start Code Coverage.



6.2.6 Set Code Coverage Parameters window will show up. Just Click OK.



6.2.7 ► Click on icon (or Press F5). You will see a RED Bars changing to GREEN Bars. It means the code is coverage. As you can see the error routine on the picture above is not tested..
Imagine how bad if you go to production and never test this condition

```

POT01CC.jcl DB2REGI.DB2REGI.cob
000-SETUP-ERROR-TRAP-RTN.
* THIS PORTION OF THE PROGRAM ACTIVATES THE SQL ERROR TRAPPING
* FACILITIES. AT PRE-COMPILATION TIME, THE DB2 PRE-COMPILER
* GENERATES COBOL INSTRUCTIONS TO INTERROGATE THE SQLCODE
* (RETURN CODE) FROM EACH CALL. IF A SQLERROR CONDITION IS
* DETECTED (NEGATIVE RETURN CODE), EXECUTION WILL BRANCH TO THE
* 999-ERROR-TRAP-RTN TO DISPLAY AN APPROPRIATE ERROR MSG.
* SET UP YOUR ERROR HANDLING ROUTINES
000-MAINLINE-RTN.
* THE MAINLINE CONTAINS THE DRIVER CODE TO PERFORM OUR DATA
* BASE ACCESS AND DISPLAY ROUTINES.
PERFORM 100-DECLARE-CURSOR-RTN THRU 100-EXIT.
PERFORM 150-OPEN-CURSOR-RTN THRU 150-EXIT.
PERFORM 200-FETCH-RTN THRU 200-EXIT
UNTIL SQLCODE = +100.
PERFORM 300-CLOSE-CURSOR-RTN THRU 300-EXIT.
PERFORM 350-TERMINATE-RTN THRU 350-EXIT.
MOVE ZERO TO RETURN-CODE.
GOBACK.
000-EXIT.
EXIT.
100-DECLARE-CURSOR-RTN.
* THIS STATEMENT CREATES AN "ACTIVE SET", A GROUP OF ROWS
* THAT WOULD BE THE OUTPUT FROM THE EXECUTION OF THE STATEMENT
* IF YOU EXECUTED IT INTERACTIVELY.
* ==> CODE THE SQL STATEMENT TO JOIN THE EMPL AND PAY TABLES
* ==> GROUP THEM BY DEPT AND DISPLAY THE DEPT AND:
* ==> AVERAGE, MINIMUM AND MAXIMUM - HOURS AND PERF BY DEPT
* ==> Diagnostic Codes optional
* %regi
EXEC SQL
DECLARE C1 CURSOR FOR
SELECT DEPT, MIN(PERF), MAX(PERF), AVG(PERF),
MIN(HOURS), MAX(HOURS), AVG(HOURS)
FROM RBAROSA.EMPL E, RBAROSA.PAY P
WHERE E.NBR = P.NBR
* AND PERF > :PERF

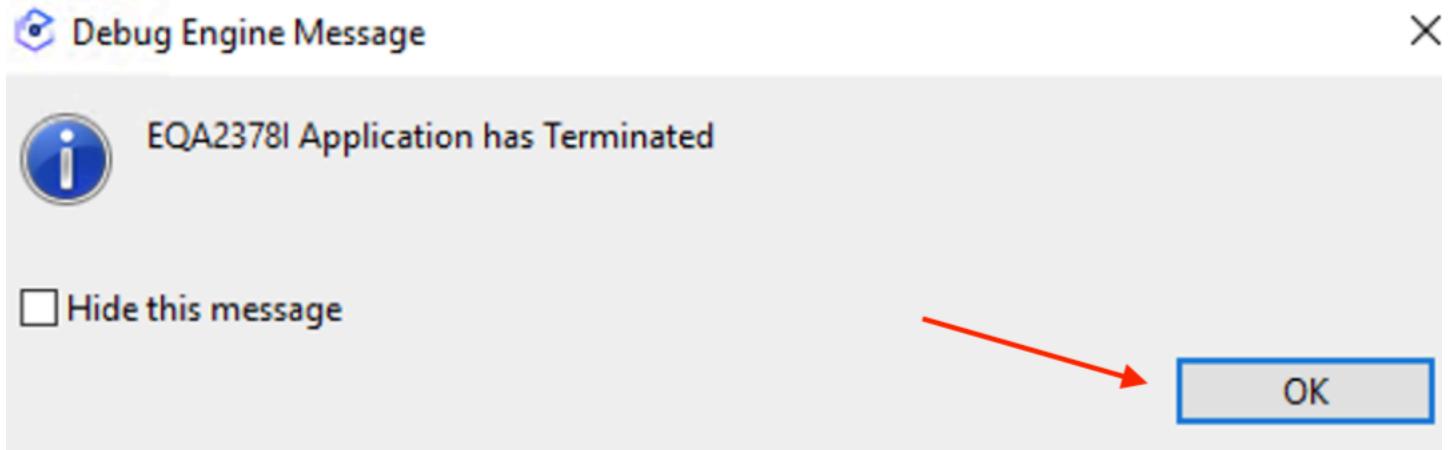
```

6.2.8 Take a look at on your last window (**Variables**). You can follow the values of the variables.

The screenshot shows the IBM Rational Developer for z/OS IDE interface. On the left, the code editor displays a COBOL program named DB2REGI.DB2REGI.cob. The code includes various MOVE statements and SQL EXEC statements. A specific line of code is highlighted: `EXEC SQL FETCH C1 INTO`. To the right of the code editor is a variable viewer window titled '(x)= Varia...'. This window lists several variables with their current values, some of which are highlighted with a red border:

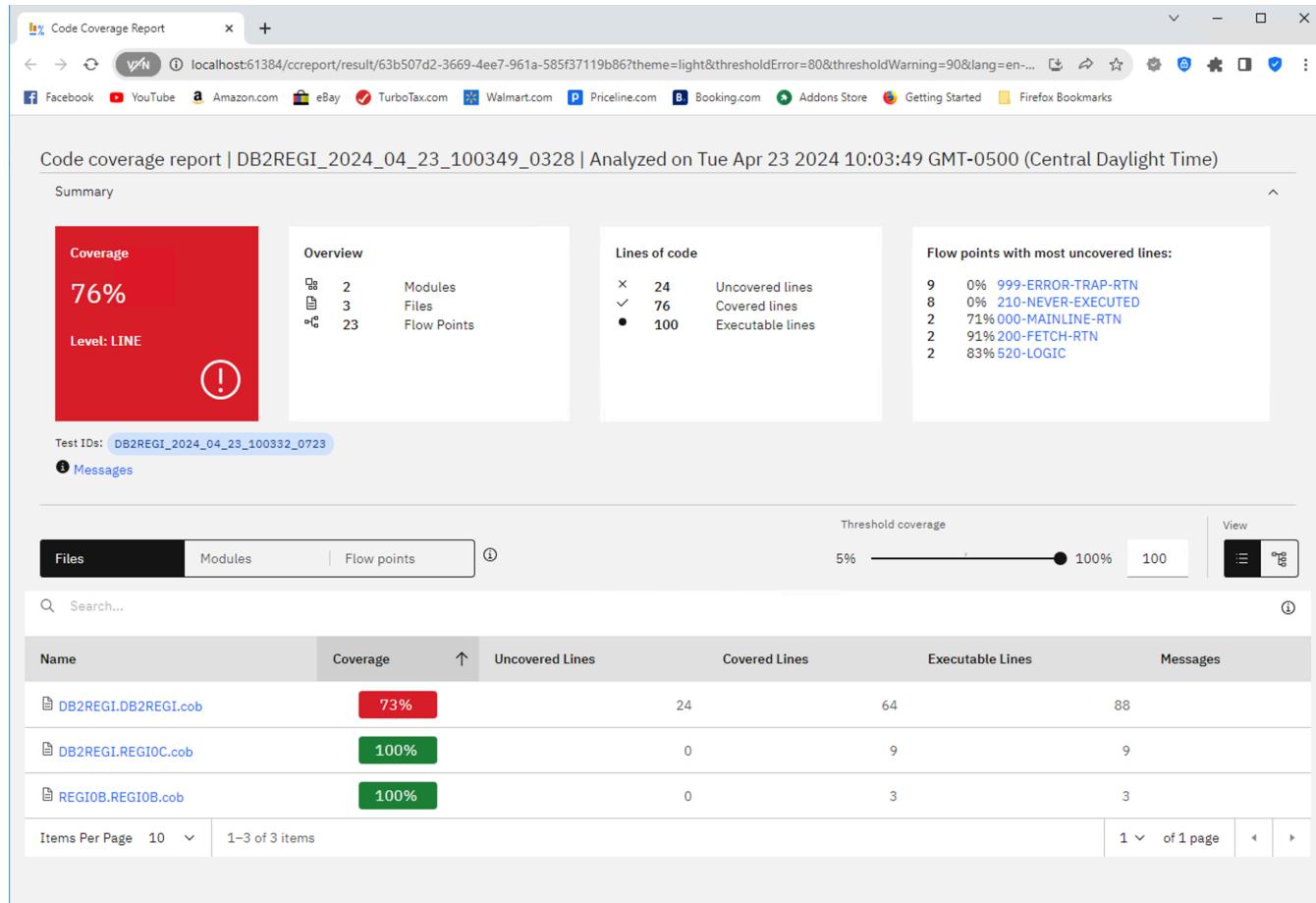
Name	Value
DEPT-NULL OF NULL-GROUP	-00001
DEPT-TBL OF TABLE-ROW	'N/A'
HOURS-NULL OF NULL-GROUP	+00000
HOURS-TBL- AVG OF TABLE-ROW	+00035.45
HOURS-TBL-MAX OF TABLE-ROW	+00035.45
HOURS-TBL-MIN OF TABLE-ROW	+00035.45
PERF-NULL OF NULL-GROUP	-00001
PERF-TBL- AVG OF TABLE-ROW	+00000.00
PERF-TBL-MAX OF TABLE-ROW	+00000
PERF-TBL-MIN OF TABLE-ROW	+00000
SQLCA	

6.2.9 ► Click on or press F8 until you see Application has Terminated window and 9 ► Click OK.



6.2.10 The code coverage report will be created. You can see that this test covered **76%**. Take a look at the programs which have coverage. The percentage of executable lines that are covered.

- **Coverage:** The percentage of executable lines that are covered. The coverage level can either be LINE or FLOWPOINT.
- **Overview:** Shows the numbers of modules, files, and flow points.
- **Lines of code:** Shows the numbers of uncovered lines, covered lines, and executable lines
- **Flow points with most uncovered lines:** The lowest-level flow points with most uncovered lines are listed. You can click the flow point to open the source or listing in an editor.

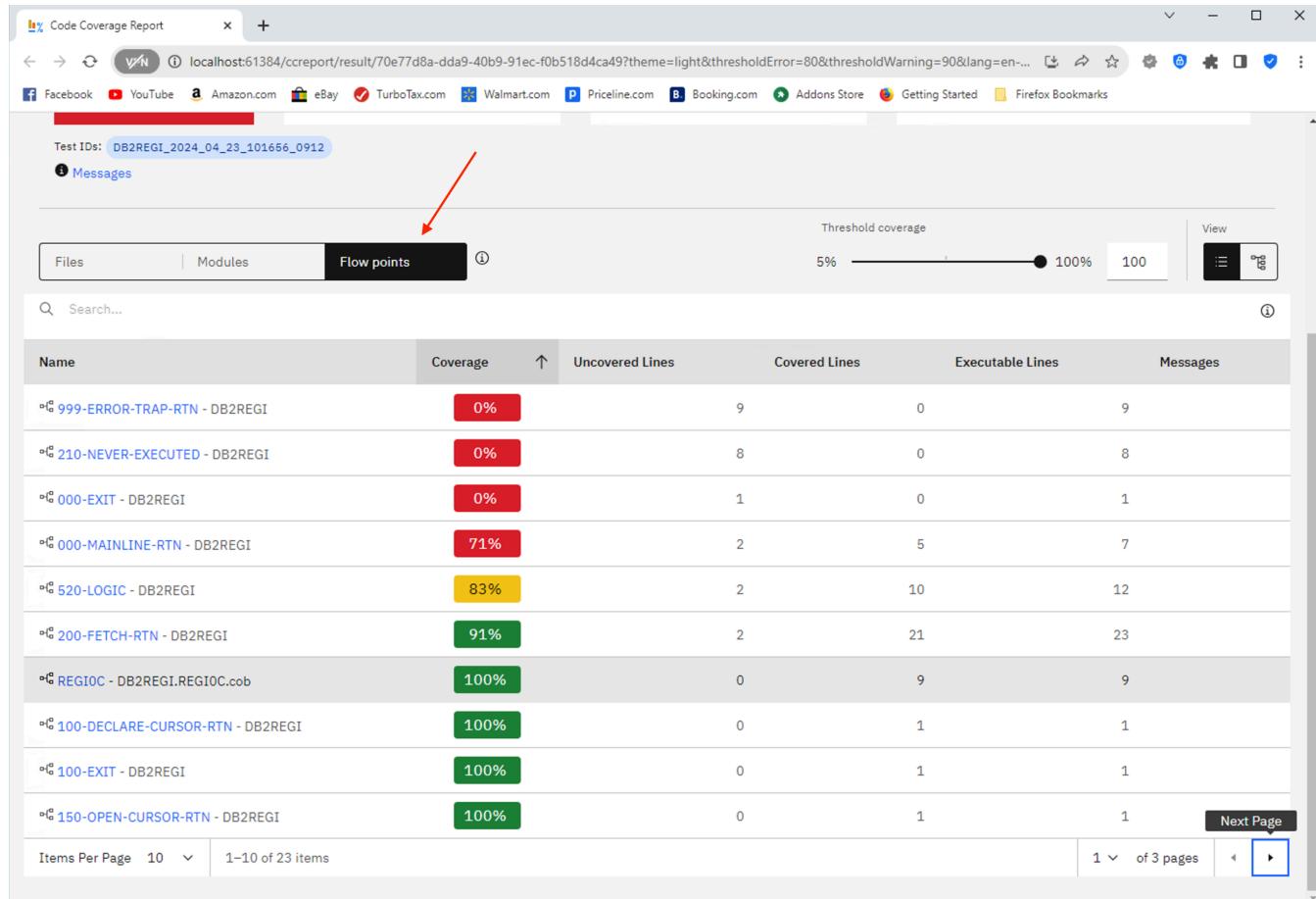


6.2.11 In the code coverage report, Click on **Flow Points** icon.

The coverage percentage is highlighted with a colored bar to indicate the status according to the thresholds you specified on the File Analysis Defaults preference page that can be opened by clicking the drop-down menu icon in the Code Coverage Results view.

- Passed (green bar): the coverage percentage is greater than or equal to the warning threshold.
- Warning (yellow bar): the coverage percentage is smaller than the warning threshold, but larger than or equal to the failure threshold. By default, the warning threshold is 90.
- Failed (red bar): the coverage is smaller than the failure threshold. By default, the failure threshold is 80.

*Note You can explore the Coverage Report, navigating on the page.

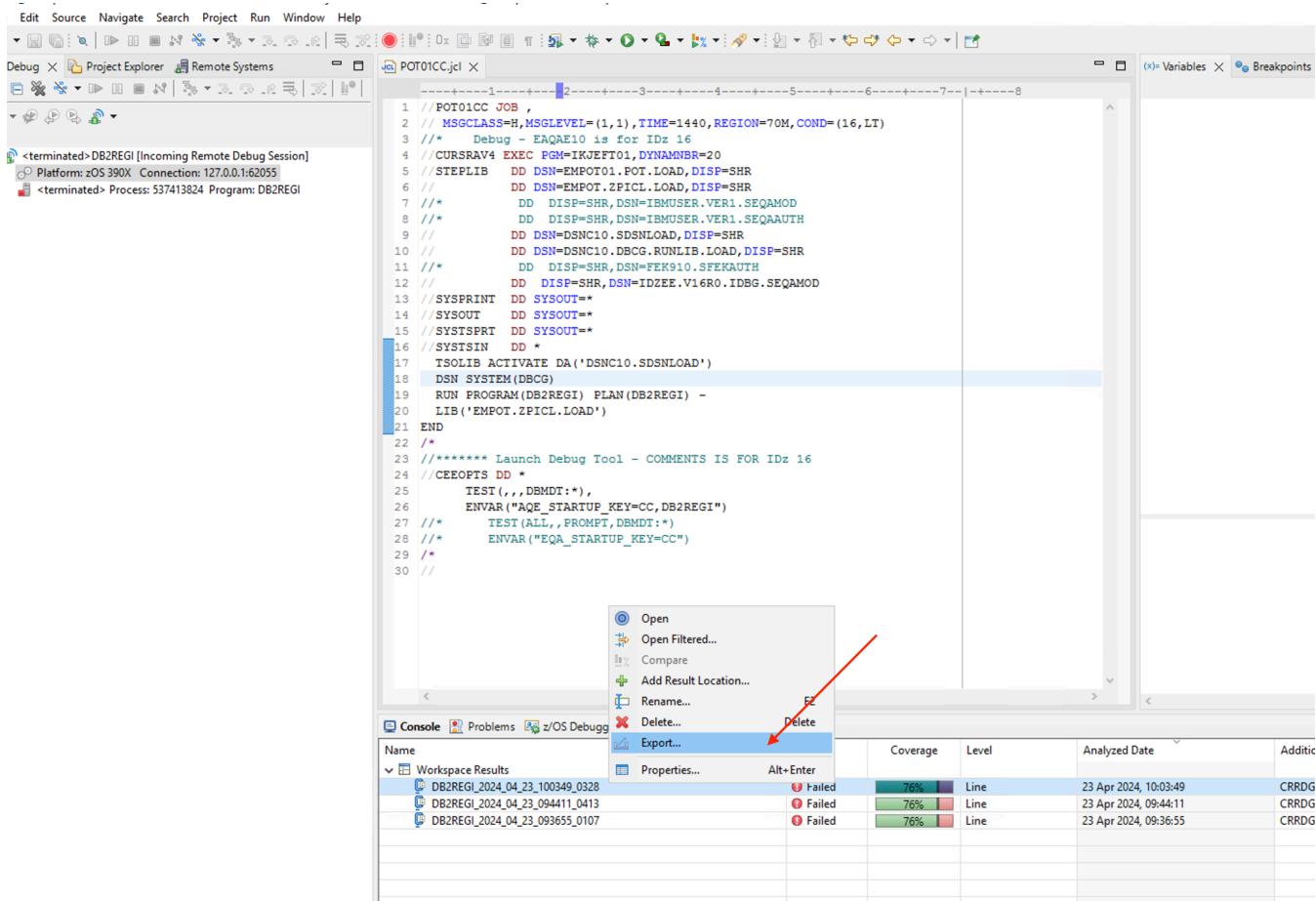


Name	Coverage	Uncovered Lines	Covered Lines	Executable Lines	Messages
999-ERROR-TRAP-RTN - DB2REGI	0%	9	0	9	
210-NEVER-EXECUTED - DB2REGI	0%	8	0	8	
000-EXIT - DB2REGI	0%	1	0	1	
000-MAINLINE-RTN - DB2REGI	71%	2	5	7	
520-LOGIC - DB2REGI	83%	2	10	12	
200-FETCH-RTN - DB2REGI	91%	2	21	23	
REGIOC - DB2REGI.REGI0C.cob	100%	0	9	9	
100-DECLARE-CURSOR-RTN - DB2REGI	100%	0	1	1	
100-EXIT - DB2REGI	100%	0	1	1	
150-OPEN-CURSOR-RTN - DB2REGI	100%	0	1	1	

6.2.12 Back to IDZ, on **Debug Perspective**. On the **Code Coverage Results** tab, Right Click on the result and select **Export**. You could also create a PDF report as documentation or export this data to other products (like ADDI or SonarQube)

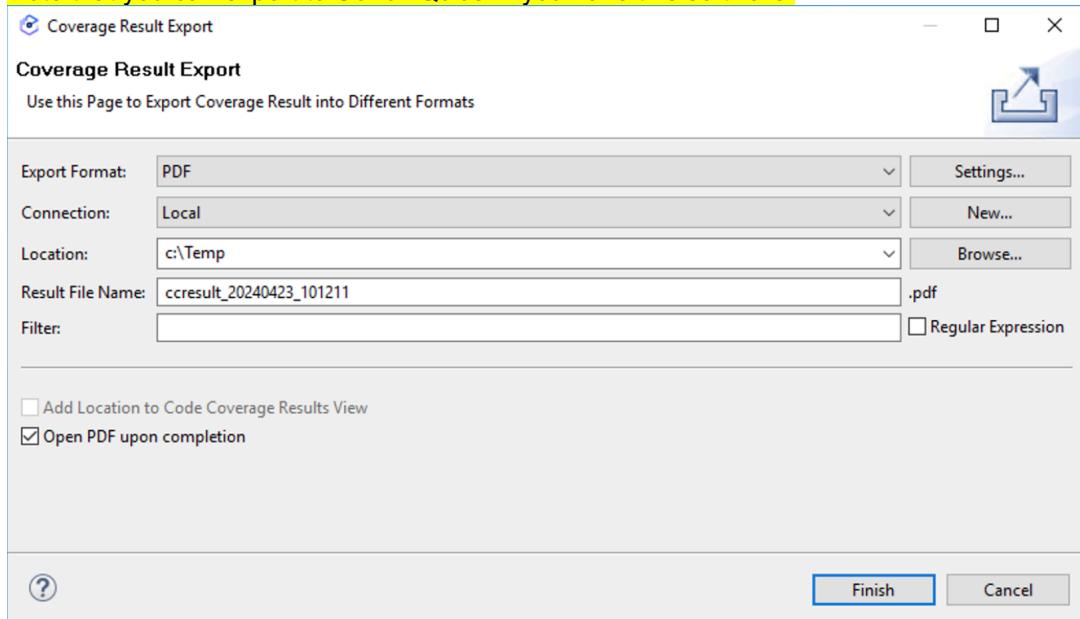
In the Code Coverage Results view, you can check the status of a result:

- If the coverage percentage is smaller than the failure threshold, the Failed icon is displayed in the Status column.
- If the coverage percentage level is smaller than the warning threshold, but greater than or equal to the failure threshold, the Warning icon is displayed in the Status column.
- If the coverage percentage is greater than or equal to the warning threshold, the Passed icon is displayed.



6.2.13 Select **PDF**, **C:\Temp**, **Open PDF upon completion** and **Finish**.

Note that you can export to **Sonar Qube** if you have this software.



6.2.14 A PDF report will be generated..

File Overall Summary

Report Info

Report: ccresult_20240423_101211.pdf

Type: LINE

Generation Date: 23 Apr 2024, 10:12:30

Filter String:

Overall Summary

Code Coverage Summary

Total Number of Files: 3	File Coverage: 100%
Total Number of Functions: 24	Function Coverage: 88%
Total Number of Lines: 100	Line Coverage: 76%
Total Number of Statements: 100	Statement Coverage: 76%

Define Properties

lineMarker: false

Import File

Type

Analysis Date

C:\Users\Administrator\idz_workspace\.metadata \plugins\com.ibm.debug.pdt.codecoverage.core \DB2REGI_2024_04_23_100349_0328.cczip (Current Result)	Merged result	23 Apr 2024, 10:03:49
--	---------------	-----------------------

File and Message

DB2REGI_2024_04_23_100349_0328

CRRDG9201W Generated in a debug session.

6.2.15  Close all editors pressing **Ctrl + Shift + F4** Or just click on the  of each opened editor.

Congratulations! You have completed the Lab 1.

LAB 2 - Create UrbanCode Deploy infrastructure and deploy to z/OS

Updated June,2024 (revised by Carlos Hirata/Wilbert Kho, Reviewed by Joe Huang and Frank Hernandez)

Abstract:

You will create and deploy an existing COBOL CICS application using UrbanCode Deploy to z/OS

This Lab has 3 parts.

Part 1 – Create the UrbanCode Deploy application infrastructure.

In this section, you will design the actual deployment process; you need to prepare the application infrastructure in UrbanCode Deploy (UCD). First, you will create a component and add component version, then you will define resources and map them to an environment, and finally you will create an application and add environment and component to it.

Part 2 - Create the UrbanCode Deploy deployment processes.

On this part you will create a deployment process for your component and the application process that uses the component process to deploy the component.

Part 3 – Deploy the application to z/OS CICS

On this part you will deploy the Application to the z/OS CICS.



Each time you see the symbol ►► it means that you have to “do” something on your computer – not merely read the document.

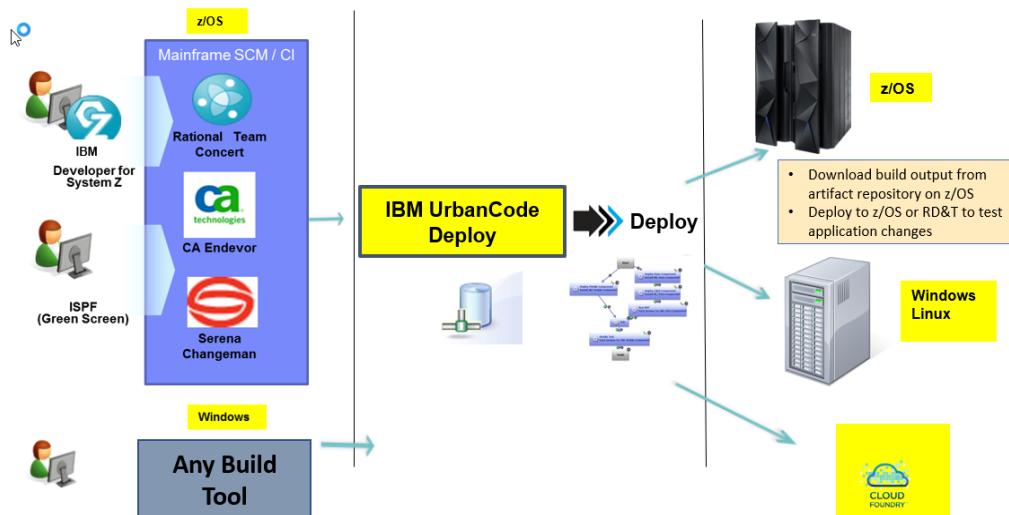
Lab Architecture and proposed scenario

The architecture used in this lab consists of.

1. The **z/OS Server** running on the ZDT that has CICS running and the UCD z/OS agent.
2. The **UrbanCode Deploy Server** that is running on Linux and on the same machine that has the z/OS (ZDT)

Below the architecture of each cloud instance

Below is a picture that describes UCD:



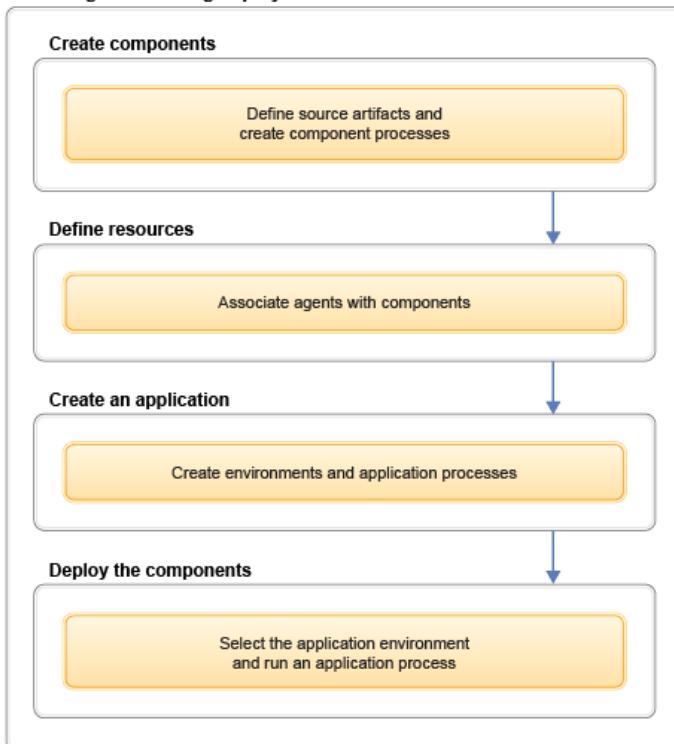
PART 1 – Create the UrbanCode application infrastructure

In this section, you will design the actual deployment process; you need to prepare the application infrastructure in UCD.

First, you will create a component and add component version, then you will define resources and map them to an environment, and finally you will create an application and add environment and component to it

The main steps are as follows:

Creating and running deployments

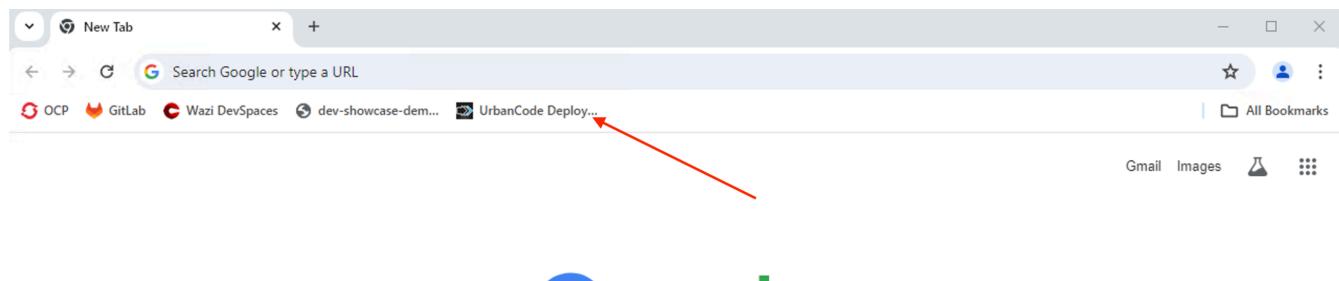


Before start UCD lab, let's verify UCD agent (zOS) is connect to UCD server (Linux Server).

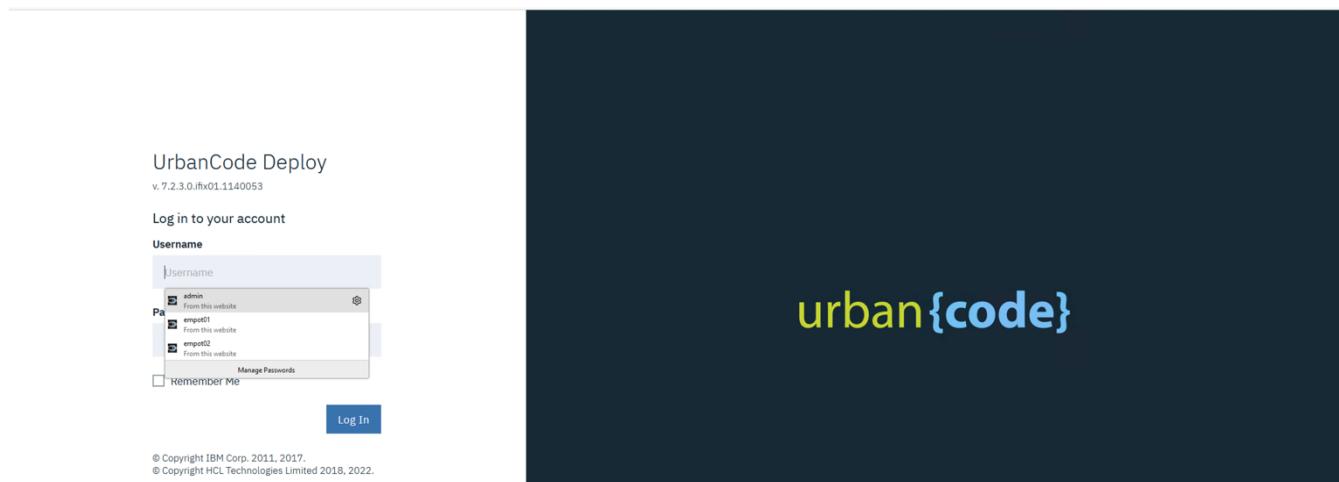
0.1.1 Open firefox browser.



0.1.2 Look at Bookmark toolbar and select **UrbanCode Deploy**



0.1.3. Login with **admin** and password **admin**



0.1.4 On Urbancode page, locate **Resources** tab. Look at the agent (zos.dev2) and check the status.

If the status is **Online**, it is good and skip to Task 1 in this lab. If the status is **Offline**. You must start the agent on zos.

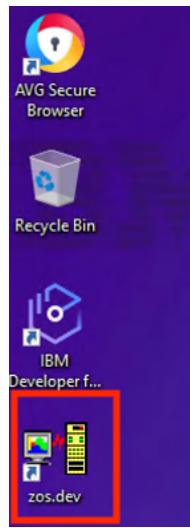
Resources

Resource Tree Resource Templates Agents Agent Configuration Templates Agent Relays Agent Pools Cloud Connections

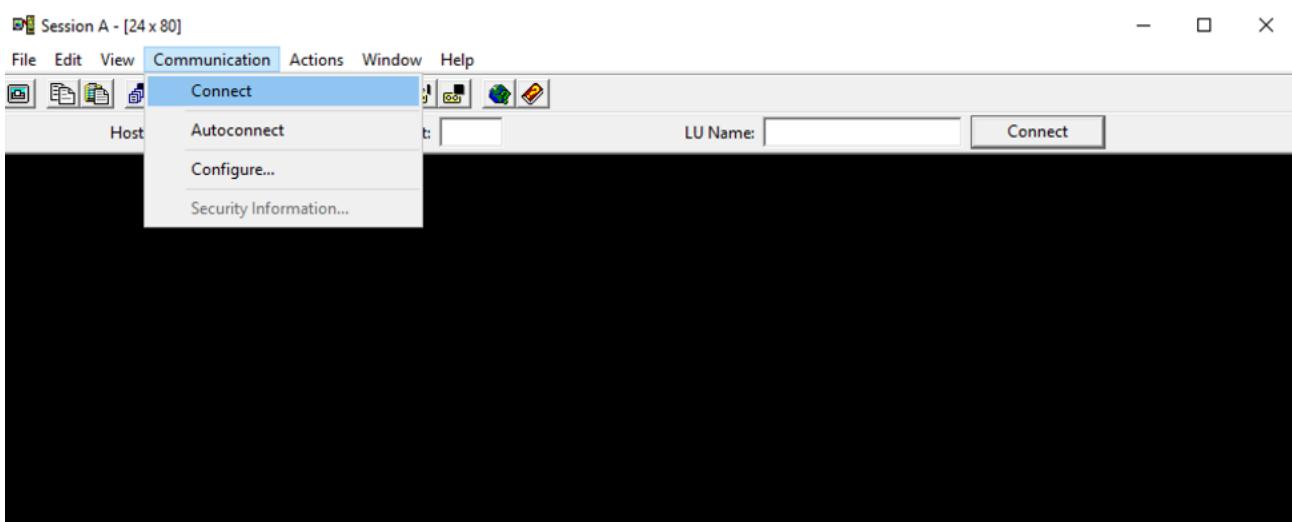
Show ▾ Bulk Actions ▾ Select All... ▾ Create Top-Level Group ▾ Customize Display ▾

Name	Tags	Inventory	Status	Description
zOS Environments				...
Dev				...
zos.dev2 (View Agent)			Offline	...
CBSA-backend (View Component)		version-375 (+ 27 more)		...
Genapp-backend (View Component)		version-458 (+ 15 more)		...
J05MortgagePOT (View Component)		20240617-185833 (+ 1 more)		...

0.1.5 On Windows Desktop, click zos.dev icon and open PCMM application.



0.1.6 Click on Communication → Connect



0.1.7 The z/OS main screen will show up and enter logon ibmuser (it can be lower case).

```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
[Icons] Connect [Icons]
Host: dev-showcase-demo-z Port: 23 LU Name: [ ] Disconnect
z/OS V2R5 LVLI PUT2112/RSU2112 IP Address = 10.1.1.1
                                         VTAM Terminal = TCP00004

Application Developer System

        // 0000000  SSSSS
        // 00      00 SS
zzzzzz // 00      00 SS
zz    // 00      00 SSSS
zz    // 00      00      SS
zz    // 00      00      SS
zzzzzz // 0000000  SSSS

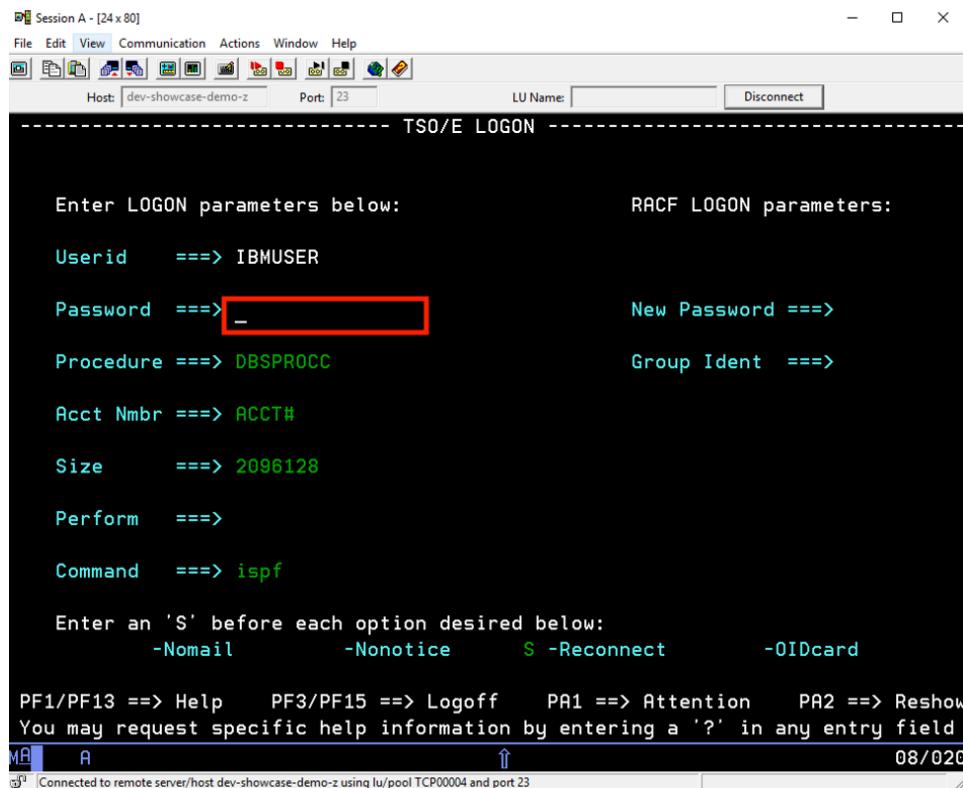
System Customization - RDCD.Z25A.~

====> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
====> Enter L followed by the APPLID
====> Examples: "L TSO", "L CICSTS56", "L CICSTS55", "L IMS15"

logon ibmuser_
MA A 24/014
Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00004 and port 23

```

0.1.8 The userid is IBMUSER and the password is **sys1**. Press **ENTER** and **ENTER**.



Session A - [24x80]

File Edit View Communication Actions Window Help

Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect

TSO/E LOGON

Enter LOGON parameters below:

Userid ===> IBMUSER

Password ===> [REDACTED]

Procedure ===> DBSPROCC

Acct Nmbr ===> ACCT#

Size ===> 2096128

Perform ===>

Command ===> ispf

RACF LOGON parameters:

New Password ===>

Group Ident ===>

Enter an 'S' before each option desired below:

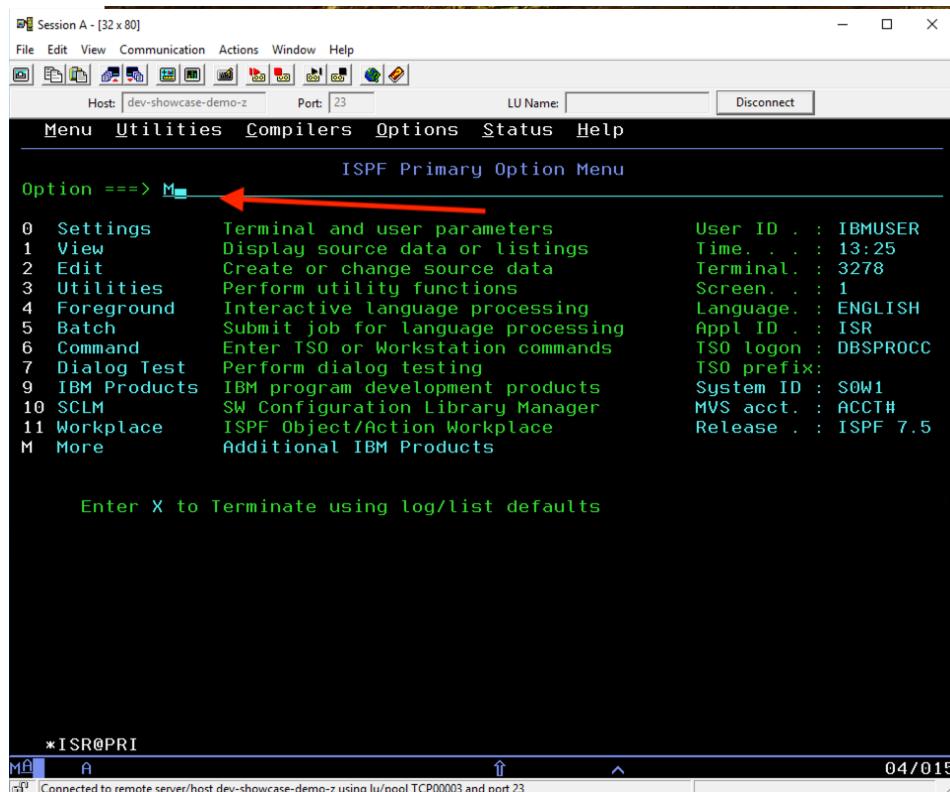
-Nomail -Notice S -Reconnect -OIDcard

PF1/PF13 ==> Help PF3/PF15 ==> Logoff PA1 ==> Attention PA2 ==> Reshow
You may request specific help information by entering a '?' in any entry field

MA A ↑ 08/020

Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00004 and port 23

0.1.9 On ISPF Primary Menu. Enter the letter **M** on **Option** field.Press **ENTER**.



Session A - [32x80]

File Edit View Communication Actions Window Help

Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect

Menu Utilities Compilers Options Status Help

ISPF Primary Option Menu

Option ===> M [Red arrow pointing to the M option]

0 Settings	Terminal and user parameters	User ID . : IBMUSER
1 View	Display source data or listings	Time. . . : 13:25
2 Edit	Create or change source data	Terminal. : 3278
3 Utilities	Perform utility functions	Screen. . : 1
4 Foreground	Interactive language processing	Language. : ENGLISH
5 Batch	Submit job for language processing	Appl ID . : ISR
6 Command	Enter TSO or Workstation commands	TSO logon : DBSPROCC
7 Dialog Test	Perform dialog testing	TSO prefix:
9 IBM Products	IBM program development products	System ID : SOW1
10 SCLM	SW Configuration Library Manager	MVS acct. : ACCT#
11 Workplace	ISPF Object/Action Workplace	Release . : ISPF 7.5
M More	Additional IBM Products	

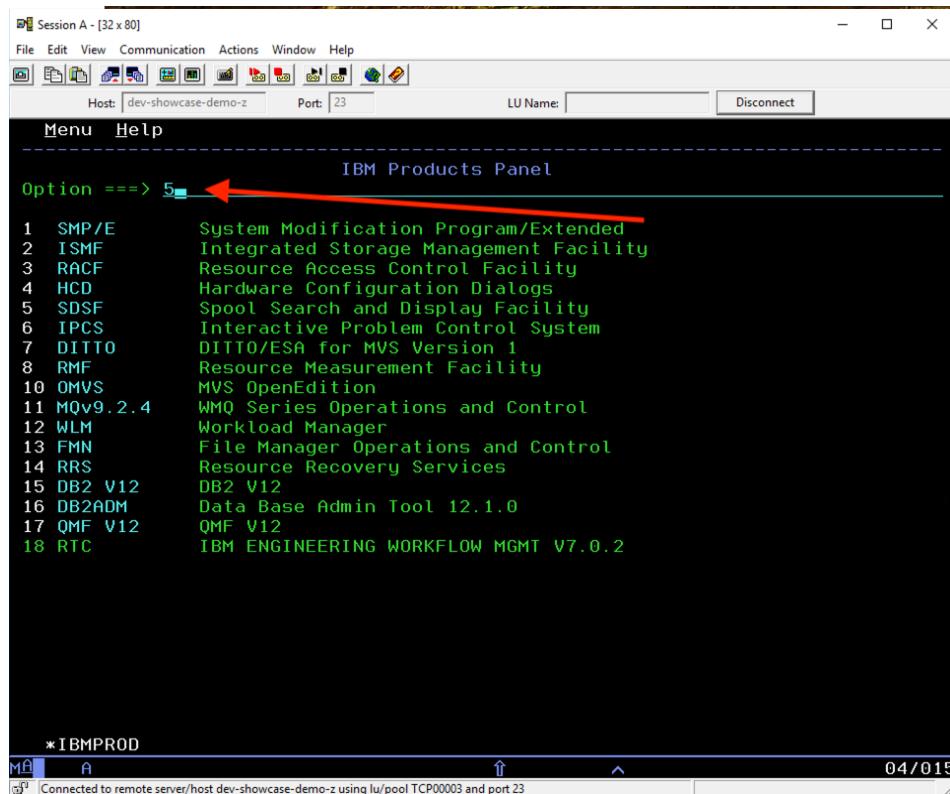
Enter X to Terminate using log/list defaults

* ISR@PRI

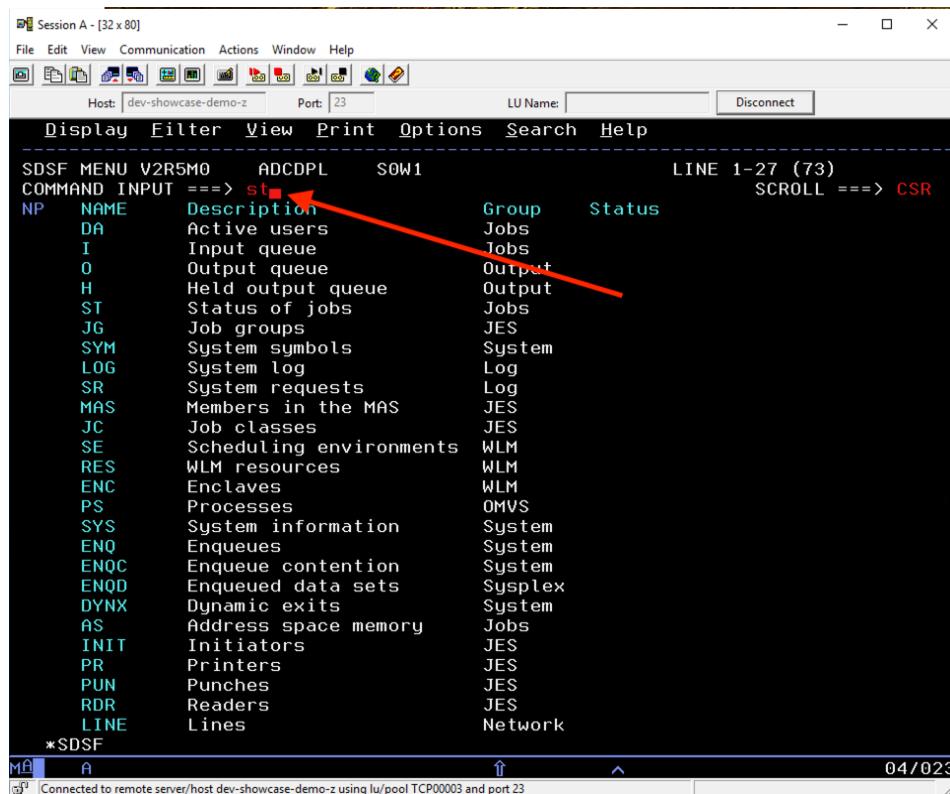
MA A ↑ ^ 04/015

Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00003 and port 23

0.1.10 On IBM Products Panel. Enter the number **5** to go to **Spool Search and Display Facility**.

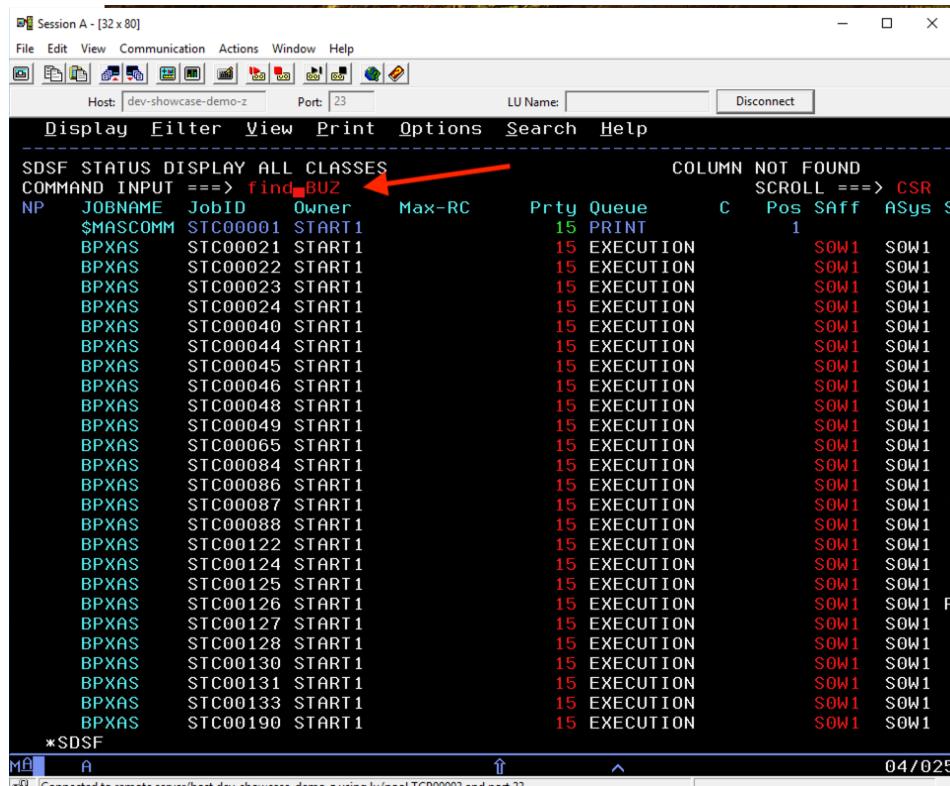


0.1.11 On **SDSF Menu**. Type **st** command on **COMMAND INPUT** and press **ENTER**.



SDSF MENU V2R5MO ADCDPL SOW1
 LINE 1-27 (73) SCROLL ==> CSR
 COMMAND INPUT ==> st [red arrow pointing here]
 NP NAME Description Group Status
 DA Active users Jobs
 I Input queue Jobs
 O Output queue Output
 H Held output queue Output
 ST Status of jobs Jobs
 JG Job groups JES
 SYM System symbols System
 LOG System log Log
 SR System requests Log
 MAS Members in the MAS JES
 JC Job classes JES
 SE Scheduling environments WLM
 RES WLM resources WLM
 ENC Enclaves WLM
 PS Processes OMVS
 SYS System information System
 ENQ Enqueues System
 ENQC Enqueue contention System
 ENQD Enqueued data sets Sysplex
 DYNX Dynamic exits System
 AS Address space memory Jobs
 INIT Initiators JES
 PR Printers JES
 PUN Punches JES
 RDR Readers JES
 LINE Lines Network
 *SDSF

0.1.12 On SDSF STATUS DISPLAY ALL CLASSES, type find buz (3 letters is enough) and press ENTER.



SDSF STATUS DISPLAY ALL CLASSES COLUMN NOT FOUND
 COMMAND INPUT ==> find BUZ [red arrow pointing here]
 SCROLL ==> CSR
 NP JOBNAME JobID Owner Max-RC Prty Queue C Pos SAff ASys S
 \$MASCOMM STC00001 START1 15 PRINT 1
 BPXAS STC00021 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00022 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00023 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00024 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00040 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00044 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00045 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00046 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00048 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00049 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00065 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00084 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00086 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00087 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00088 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00122 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00124 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00125 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00126 START1 15 EXECUTION SOW1 SOW1 P
 BPXAS STC00127 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00128 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00130 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00131 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00133 START1 15 EXECUTION SOW1 SOW1
 BPXAS STC00190 START1 15 EXECUTION SOW1 SOW1
 *SDSF

0.1.13 BUZAGNT agent might be started, but it is not connected to the UCD server. It is necessary to cancel this JOB and start it again. On the column NP, type letter c the front of BUZAGNT job and press ENTER.

Session A - [32 x 80]

File Edit View Communication Actions Window Help

Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect

Display Filter View Print Options Search Help

SDSF STATUS DISPLAY ALL CLASSES LINE 38-63 (118)

COMMAND INPUT ==> SCROLL ==> CSR

NP	JOBNAME	JobID	Owner	Max-RC	Prtv	Queue	C	Pos	Saff	ASys	S
c	BUZAGNT	STC00464	START1		15	EXECUTION			SOW1	SOW1	
	CFZCIM	STC00027	CFZSRV		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00175	CICSUSER		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00028	CICSUSER	CC 0001	1	PRINT		11			
	CNMPROC	STC00068	START1		15	EXECUTION			SOW1	SOW1	
	CSF	STC00016	START1		15	EXECUTION			SOW1	SOW1	
	DBB	STC00074	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBBS	STC00075	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBCADM1	STC00055	START1		15	EXECUTION			SOW1	SOW1	
	DBCDBM1	STC00050	START1		15	EXECUTION			SOW1	SOW1	
	DBCGDIST	STC00054	START1		15	EXECUTION			SOW1	SOW1	
	DBCGLRLM	STC00041	START1		15	EXECUTION			SOW1	SOW1	
	DBCGMSTR	STC00029	START1		15	EXECUTION			SOW1	SOW1	
	DBGMGR	STC00026	START1		15	EXECUTION			SOW1	SOW1	
	EQAPROF	STC00077	STCEQA		15	EXECUTION			SOW1	SOW1	
	EQARMTD	STC00078	START1		15	EXECUTION			SOW1	SOW1	
	GITLADR	STC00076	IBMUSER	CC 3840	1	PRINT		9			
	HTTPD1	STC00085	WEBSRV		15	EXECUTION			SOW1	SOW1	
	IBMUSER	TSU00463	IBMUSER		15	EXECUTION			SOW1	SOW1	
	IMSGNJCL	JOB00064	START1	CC 0000	1	PRINT	A	5			
	IMS1CTL	STC00033	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DLI	STC00051	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DRC	STC00052	START1		15	EXECUTION			SOW1	SOW1	
	IMS1ERE	JOB00061	START1	CC 0000	1	PRINT	A	2			
	IMS1ERE	JOB00062	START1	CC 0008	1	PRINT	A	3			
	IMS1ERE	JOB00063	START1	CC 0008	1	PRINT	A	4			

*SDSF

MB A 06/003

Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00003 and port 23

0.1.14 BUZAGNT is canceled and now start the BUZAGNT. On COMMAND INPUT, type /s BUZAGNT. Press ENTER.

Session A - [32 x 80]

File Edit View Communication Actions Window Help

Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect

Display Filter View Print Options Search Help

SDSF STATUS DISPLAY ALL CLASSES LINE 38-63 (118)

COMMAND INPUT ==> /s buzagnt 1

NP	JOBNAME	JobID	Owner	Max-RC	Prtv	Queue	C	Pos	Saff	ASys	S
	BUZAGNT	STC00464	START1			CANCELED			37		
	CFZCIM	STC00027	CFZSRV		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00175	CICSUSER		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00028	CICSUSER	CC 0001	1	PRINT		11			
	CNMPROC	STC00068	START1		15	EXECUTION			SOW1	SOW1	
	CSF	STC00016	START1		15	EXECUTION			SOW1	SOW1	
	DBB	STC00074	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBBS	STC00075	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBCADM1	STC00055	START1		15	EXECUTION			SOW1	SOW1	
	DBCDBM1	STC00050	START1		15	EXECUTION			SOW1	SOW1	
	DBCGDIST	STC00054	START1		15	EXECUTION			SOW1	SOW1	
	DBCGLRLM	STC00041	START1		15	EXECUTION			SOW1	SOW1	
	DBCGMSTR	STC00029	START1		15	EXECUTION			SOW1	SOW1	
	DBGMGR	STC00026	START1		15	EXECUTION			SOW1	SOW1	
	EQAPROF	STC00077	STCEQA		15	EXECUTION			SOW1	SOW1	
	EQARMTD	STC00078	START1		15	EXECUTION			SOW1	SOW1	
	GITLADR	STC00076	IBMUSER	CC 3840	1	PRINT		9			
	HTTPD1	STC00085	WEBSRV		15	EXECUTION			SOW1	SOW1	
	IBMUSER	TSU00463	IBMUSER		15	EXECUTION			SOW1	SOW1	
	IMSGNJCL	JOB00064	START1	CC 0000	1	PRINT	A	5			
	IMS1CTL	STC00033	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DLI	STC00051	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DRC	STC00052	START1		15	EXECUTION			SOW1	SOW1	
	IMS1ERE	JOB00061	START1	CC 0000	1	PRINT	A	2			
	IMS1ERE	JOB00062	START1	CC 0008	1	PRINT	A	3			
	IMS1ERE	JOB00063	START1	CC 0008	1	PRINT	A	4			

*SDSF

MB A 04/031

Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00003 and port 23

0.1.15 BUZAGNT agent is connect.

Session A - [32x80]

File Edit View Communication Actions Window Help

Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect

Display Filter View Print Options Search Help

SDSF STATUS DISPLAY ALL CLASSES LINE 38-63 (119)

COMMAND INPUT ==> CSR

NP	JOBNAME	JobID	Owner	Max-RC	Prtv	Queue	C	Pos	Saff	ASys	S
	BUZAGNT	STC00465	START1		15	EXECUTION			SOW1	SOW1	
	BUZAGNT	STC00464	START1	CANCELED	1	PRINT			37		
	CFZCIM	STC00027	CFZSRV		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00175	CICSUSER		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00028	CICSUSER	CC 0001	1	PRINT		11			
	CNMPROC	STC00068	START1		15	EXECUTION			SOW1	SOW1	
	CSF	STC00016	START1		15	EXECUTION			SOW1	SOW1	
	DBB	STC00074	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBBS	STC00075	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBCADM	STC00055	START1		15	EXECUTION			SOW1	SOW1	
	DBCDBM1	STC00050	START1		15	EXECUTION			SOW1	SOW1	
	DBC DIST	STC00054	START1		15	EXECUTION			SOW1	SOW1	
	DBC GIRLM	STC00041	START1		15	EXECUTION			SOW1	SOW1	
	DBC GMSTR	STC00029	START1		15	EXECUTION			SOW1	SOW1	
	DBG MGR	STC00026	START1		15	EXECUTION			SOW1	SOW1	
	EQAPROF	STC00077	STCEQA		15	EXECUTION			SOW1	SOW1	
	EQARMTD	STC00078	START1		15	EXECUTION			SOW1	SOW1	
	GITLABR	STC00076	IBMUSER	CC 3840	1	PRINT		9			
	HTTPD1	STC00085	WEBSRV		15	EXECUTION			SOW1	SOW1	
	IBMUSER	TSU00463	IBMUSER		15	EXECUTION			SOW1	SOW1	
	IMSGNJCL	JOB00064	START1	CC 0000	1	PRINT	A	5			
	IMS1CTL	STC00033	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DLI	STC00051	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DR	STC00052	START1		15	EXECUTION			SOW1	SOW1	
	IMS1ERE	JOB00061	START1	CC 0000	1	PRINT	A	2			
	IMS1ERE	JOB00062	START1	CC 0008	1	PRINT	A	3			

*SDSF

MB A 04/021

Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00003 and port 23

0.1.16 Logout the PCOMM session. On COMMAND INPUT, type =x press ENTER. and type logoff and ENTER.

Session A - [32x80]

File Edit View Communication Actions Window Help

Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect

Display Filter View Print Options Search Help

SDSF STATUS DISPLAY ALL CLASSES CHARS 'BUZ' FOUND

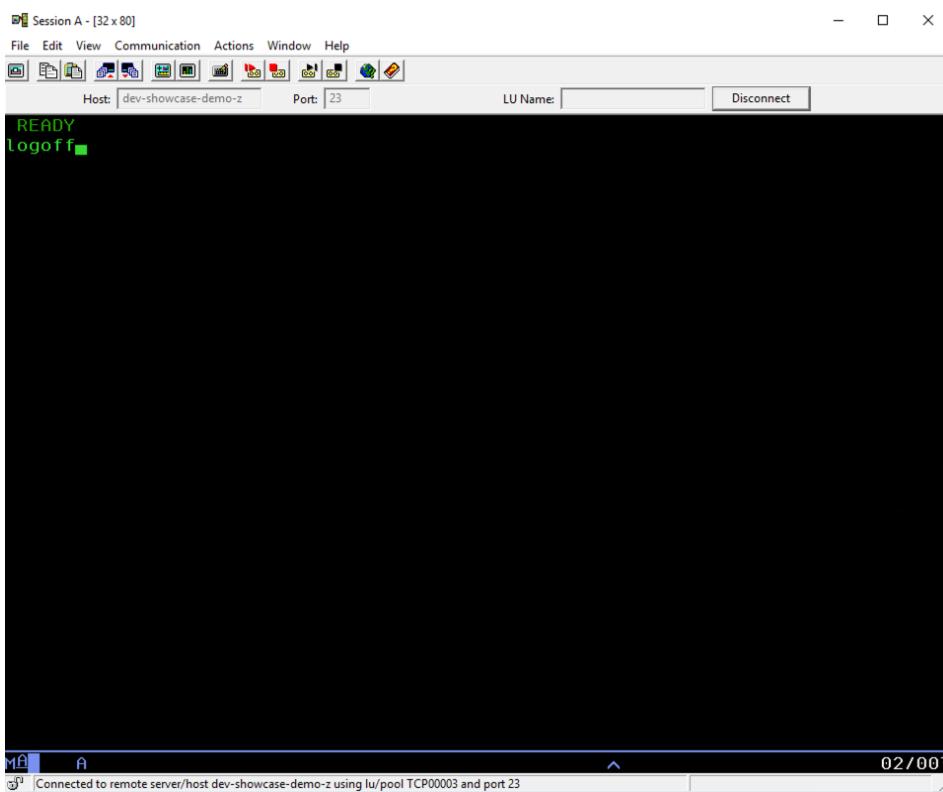
COMMAND INPUT ==> CSR

NP	JOBNAME	JobID	Owner	Max-RC	Prtv	Queue	C	Pos	Saff	ASys	S
	BUZAGNT	STC00465	START1		15	EXECUTION			SOW1	SOW1	
	BUZAGNT	STC00464	START1	CANCELED	1	PRINT			37		
	CFZCIM	STC00027	CFZSRV		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00175	CICSUSER		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00028	CICSUSER	CC 0001	1	PRINT		11			
	CNMPROC	STC00068	START1		15	EXECUTION			SOW1	SOW1	
	CSF	STC00016	START1		15	EXECUTION			SOW1	SOW1	
	DBB	STC00074	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBBS	STC00075	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBCADM	STC00055	START1		15	EXECUTION			SOW1	SOW1	
	DBCDBM1	STC00050	START1		15	EXECUTION			SOW1	SOW1	
	DBC DIST	STC00054	START1		15	EXECUTION			SOW1	SOW1	
	DBC GIRLM	STC00041	START1		15	EXECUTION			SOW1	SOW1	
	DBC GMSTR	STC00029	START1		15	EXECUTION			SOW1	SOW1	
	DBG MGR	STC00026	START1		15	EXECUTION			SOW1	SOW1	
	EQAPROF	STC00077	STCEQA		15	EXECUTION			SOW1	SOW1	
	EQARMTD	STC00078	START1		15	EXECUTION			SOW1	SOW1	
	GITLABR	STC00076	IBMUSER	CC 3840	1	PRINT		9			
	HTTPD1	STC00085	WEBSRV		15	EXECUTION			SOW1	SOW1	
	IBMUSER	TSU00468	IBMUSER		15	EXECUTION			SOW1	SOW1	
	IMSGNJCL	JOB00064	START1	CC 0000	1	PRINT	A	5			
	IMS1CTL	STC00033	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DLI	STC00051	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DR	STC00052	START1		15	EXECUTION			SOW1	SOW1	
	IMS1ERE	JOB00061	START1	CC 0000	1	PRINT	A	2			
	IMS1ERE	JOB00062	START1	CC 0008	1	PRINT	A	3			

*SDSF

MB A 04/023

Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00005 and port 23



0.1.17 Back to Firefox browser on UCD page, locate **Resource** and check **zos.dev2** is **Online**.

Name	Tags	Inventory	Status	Description
zos.dev2 (View Agent)		version-375 (+ 27 more)	Online	
CBSA-backend (View Component)		version-458 (+ 15 more)		
Genapp-backend (View Component)		20240617-185833 (+ 1 more)		
J05MortgagePOT (View Component)				

Now go to Task 1

Task 1 – Logon to UCD server running on Linux and verify that the UrbanCode agents are running

You will now connect to the UrbanCode Deploy (UCD) server running on a LINUX (same Linux that ZDT is running) and that each student have access to it

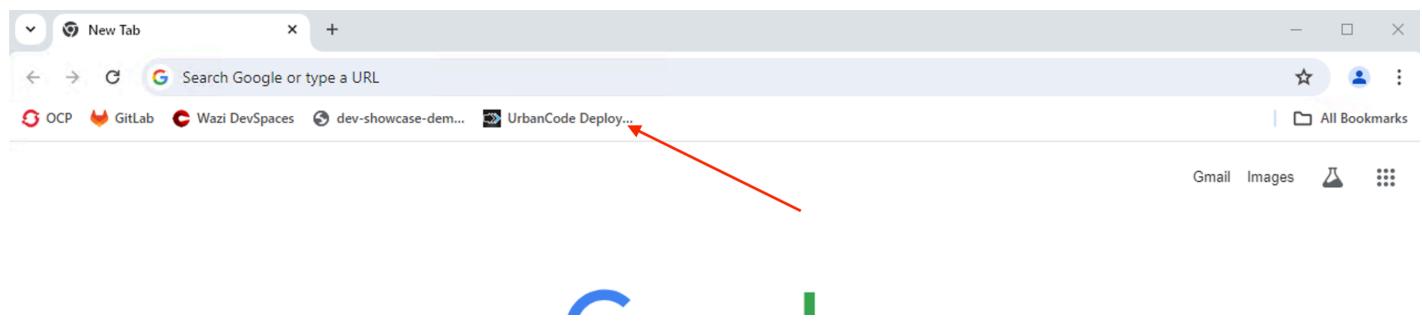
Here you will verify the UCD agents that the UCD server has access to.

You will use the user id **empot02** and password **empot02**. Remember that here the Userid and password is **lower case**.

- 1.1  Start the Firefox browser to go to UCD server on Linux You can use the icon on the base of your screen Move the mouse to the base of your screen if you do not see this bar.



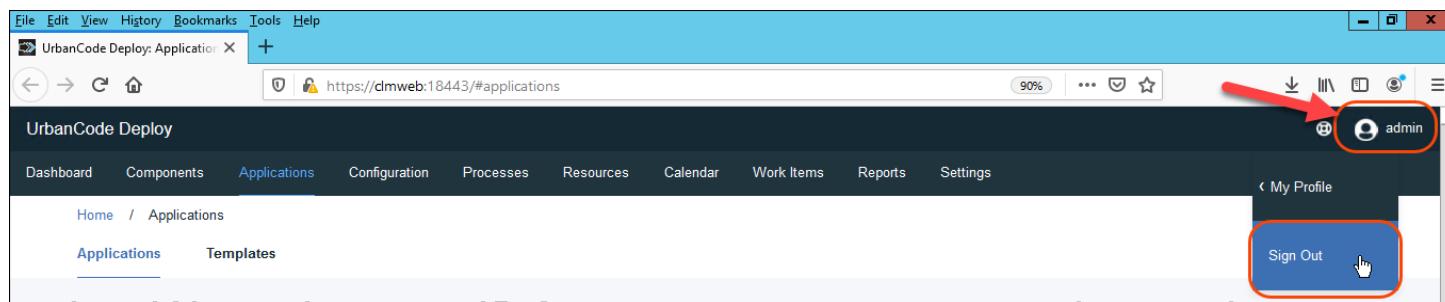
-  Look at Bookmark toolbar and select **UrbanCode Deploy**



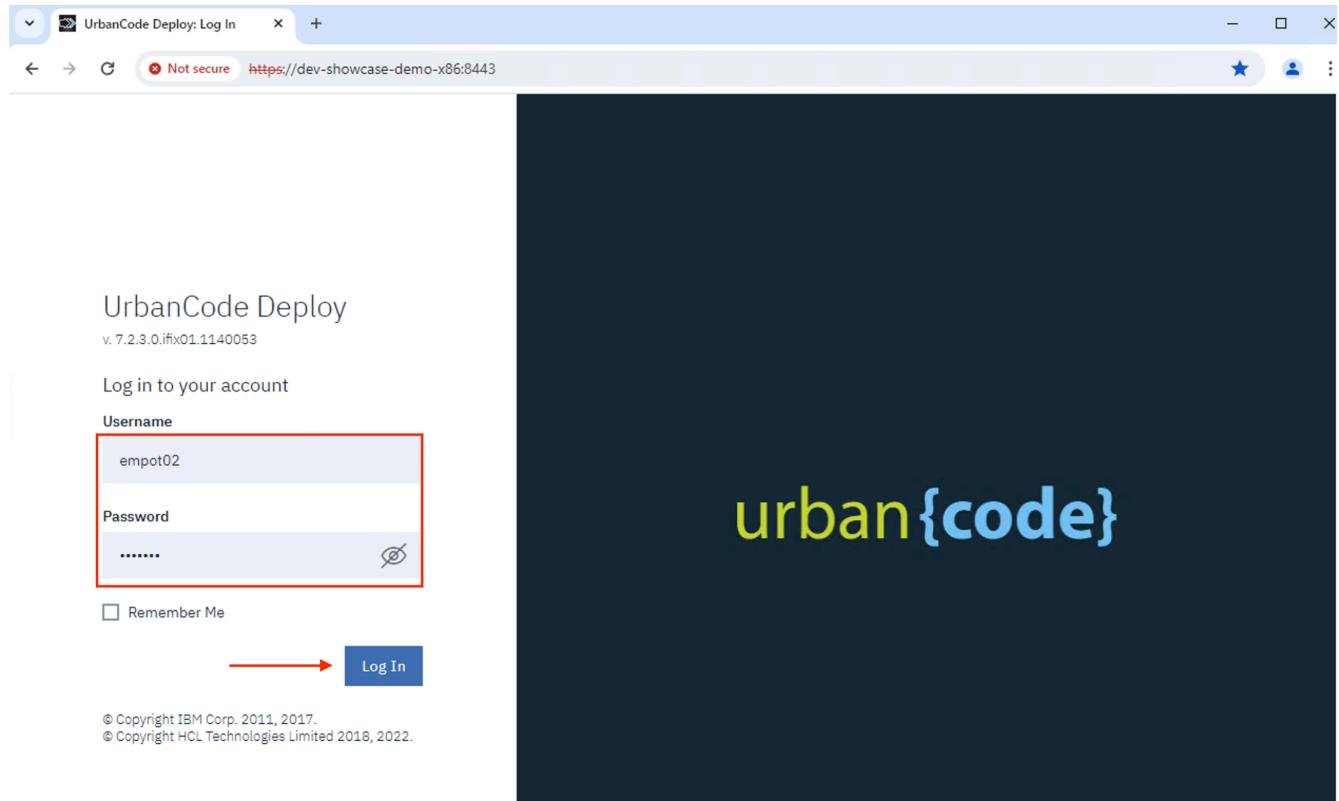
Or use: <https://dev-showcase-demo-x86:8443/>

- 1.2. If you are already logged on, you must first logoff and logon with your right user id.

-  Click on **admin** (or whatever user id is logged in) and select **Sign Out**



- 1.3. ►| Login with **empot02** and password **empot02**
Remember that this is case sensitive here and lower case



- 1.4. ►| Verify that the UCD agents are running by clicking **Resources > Agents**.

Check agent **zos.dev2** agent is running (**online**)

On this lab you will use only the z/OS V7.2 agent (**zos.dev2**)

UrbanCode Deploy: Components

Not secure https://dev-showcase-demo-x86:8443/#components/components

UrbanCode Deploy Components Applications Processes Resources Calendar Work Items More empot02

Components Components Templates

View By: Component Bulk Actions Import Components Create Component Customize Display

Name	Tags	Latest Import	Latest Version	Template	Description	Created	By
CBSA-backend			version-718	Mainframe Deploy		3/6/2023, 3:55 AM	admin
Genapp-backend			version-1089	Mainframe Deploy		3/3/2023, 4:27 AM	admin
MYCOMP			20230216-125003			2/3/2023, 3:39 PM	admin

Items per page: 10 | 1-3 of 3 items

1 of 1 pages < Previous 1 Next > Refresh

UrbanCode Deploy: Resources

Not secure https://dev-showcase-demo-x86:8443/#resources/agents

UrbanCode Deploy Dashboard Components Applications Processes Resources Calendar Work Items More empot02

Resources Resource Tree Resource Templates Agents Agent Configuration Templates Agent Relays Agent Pools Cloud Connections

View By: Agent Bulk Actions Select All... Discover Available Network Hosts Install New Agent Customize Display

Name	Tags	Description	Status	Date Created	Last Contact	License Type	Licensed	Version
x86-agent		Error	2/21/2023, 11:15 AM	3/18/2024, 10:40 AM	Authorized	false	7.2.3.0.ifix01	
zos.dev2		Online	3/16/2023, 6:22 PM	4/23/2024, 3:46 PM	Unlicensed	false	7.2.0.0.11087	

Items per page: 10 | 1-2 of 2 items

1 of 1 pages < Previous Refresh

Task 2 – Create a UCD component

Components are groups of deployable artifacts that make up an application. You will later create your UCD application named “**J02Mortgage**” that will include only one component that you will create here.



UCD: What is a Component?

A component is a logical representation of deployable artifact along with user-defined processes that operate on them. The physical artifacts are specified in a Component Version. A component has one or more component versions. Each component version has one or more files.

In UrbanCode Deploy, components represent deployable items along with user-defined processes that operate on them. Deployable items, or artifacts, can be files, PDS members, images, databases, etc. In our example, component will include **PDS** members **J02CMORT** and **J02MPMT**.



z/OS: What is a PDS?

A partitioned data set or PDS consists of a *directory* and *members*. The directory holds the address of each member, or “file”, and thus makes it possible for programs or the operating system to access each member directly. Each member, however, consists of sequentially stored records.

Partitioned data sets are often called *libraries*. Programs, or other content, are stored as members of partitioned data sets. Generally, the operating system loads the members of a PDS into storage sequentially, but it can access members directly when selecting a program for execution.

A PDS also contains a directory. The directory contains an entry for each member in the PDS with a reference (or pointer) to the member. Member names are listed alphabetically in the directory, but members themselves can appear in any order in the library. The directory allows the system to retrieve a particular member in the data set.

2.1 ► On the **Components** page, click **Create Component**:

The screenshot shows the 'Components' page in the UrbanCode Deploy interface. A red circle labeled '1' highlights the 'Components' tab in the top navigation bar. Another red circle labeled '2' highlights the 'Create Component' button in the top right corner of the main content area, which is enclosed in a red oval.

Name	Tags	Latest Import	Latest Version	Template	Description	Created	By
CBSA-backend		version-718	Mainframe Deploy			3/6/2023, 3:55 AM	admin
Genapp-backend		version-1089	Mainframe Deploy			3/3/2023, 4:27 AM	admin
MYCOMP		20230216-125003				2/3/2023, 3:39 PM	admin

2.2 Provide values for the required fields on the **Create Component** dialog.

- Select **MVSCOMPONENT** in the **Template** field (Templates serve as models for various groups of resources. MVSCOMPONENT template, preinstalled with UCD, includes basic z/OS deploy processes and other z/OS related settings)
- Specify component Name as **J02Mortgage** (Be careful since the name is case sensitive).
- Scroll down and click **Save**.

Create Component ⓘ

X

Component Template

MVSCOMPONENT

1

▼

Template Version

Always Use Latest

▼

Name *

J02Mortgage

2

Description

3

Cancel

Save



UCD: What is a Component Template?

A component template, stores component processes and properties so you can create components from them; template-based components inherit the template's properties and process.

2.3 The component **J02Mortgage** is created

The screenshot shows the UrbanCode Deploy web interface. The top navigation bar includes links for Dashboard, Components, Applications, Processes, Resources, Calendar, Work Items, Reports, and Settings. Below the navigation is a breadcrumb trail: Home / Components / J02Mortgage. The main title 'Component: J02Mortgage' is highlighted with a red oval. Below the title is a 'Show details' link. A horizontal menu bar below the title contains links for Dashboard, Usage, Configuration, Calendar, Versions, Processes, and Changes.

The MVSCOMPONENT is not in the drop down?
Your UCD server must have this template defined there. If NOT call the instructor..

1. Logon ad **admin/admin**
2. Click **Settings > Teams** (*under Security*)
3. For EACH team perform as below..

3.1 – Click Team 01 under Team Object Mappings/View: select **Component Template**

3.2 – Click Add and select MVSCOMPONENT

Name	Type
DB2 Bind	Standard Component Template
MVSCOMPONENT	Standard Component Template

Task 3 – Create a ship list file and z/OS component version

The z/OS **ship list file** specifies which files from the build to include in the new component version to deploy. You must create a ship list file before you run z/OS deployment tools.

Ship list files are XML files that contain a list of files to be deployed on z/OS. For more information, refer to the UCD documentation:

http://www.ibm.com/support/knowledgecenter/SS4GSP_6.2.6/com.ibm.udeploy.doc/topics/zos_shiplistfiles.html

To create a ship list file for the component to be deployed you could use many ways, including ISPF.
To make this easy you will use IDz.

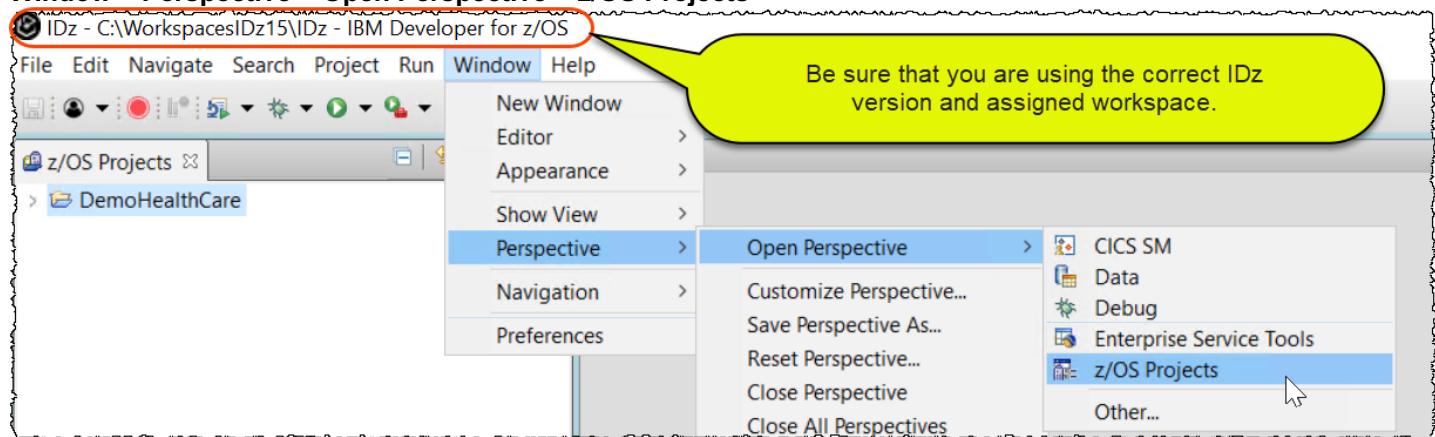
3.1 Start *IBM Developer for z Systems version 16* if it is not already started

►► Using the desktop double click on **IDz V16** icon.

►► Verify that the message indicates that it is Version **16**

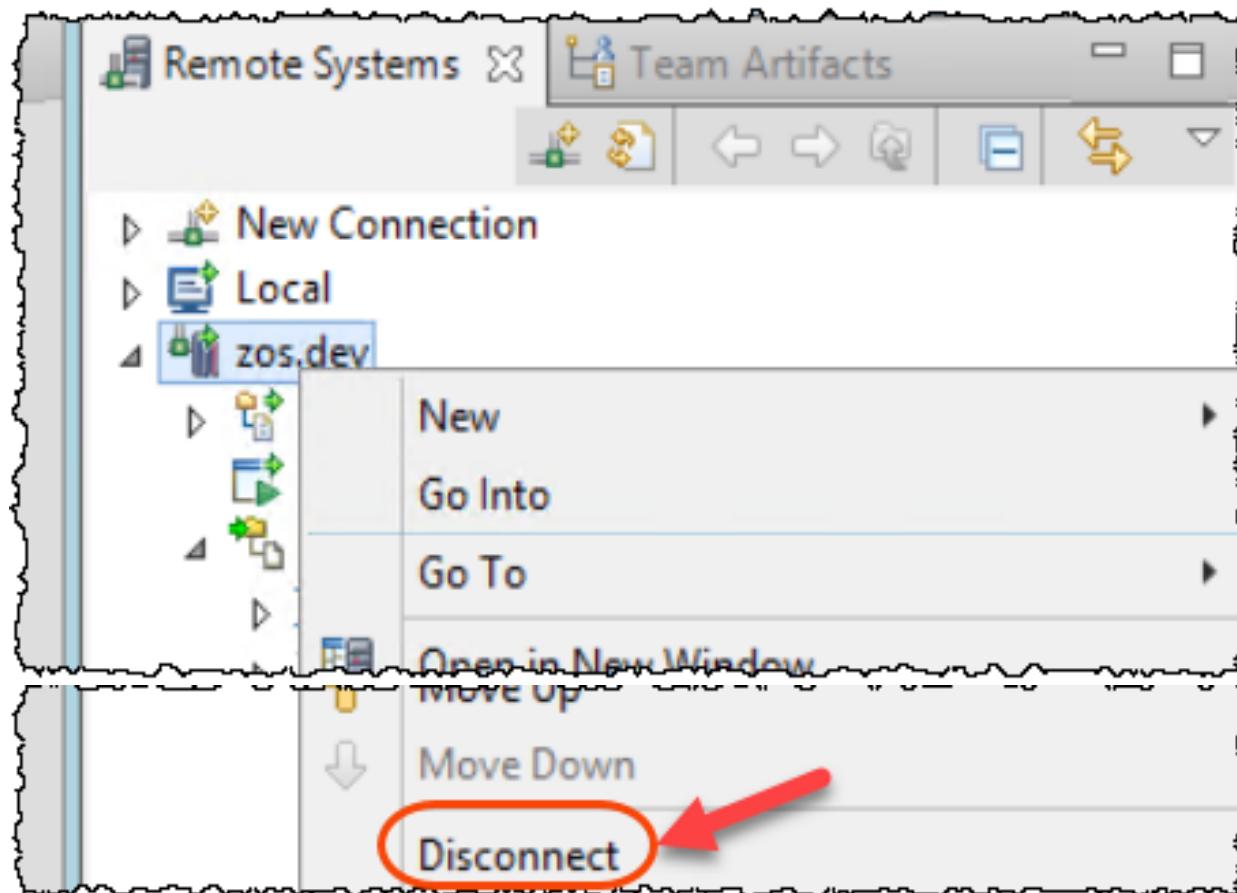
IMPORTANT -> This icon will start an eclipse workspace that has already some definitions required for this lab.
PLEASE DO NOT start IDz using other way than this icon.

3.2 ► Open the z/OS Projects perspective by selecting
Window > Perspective > Open Perspective > z/OS Projects

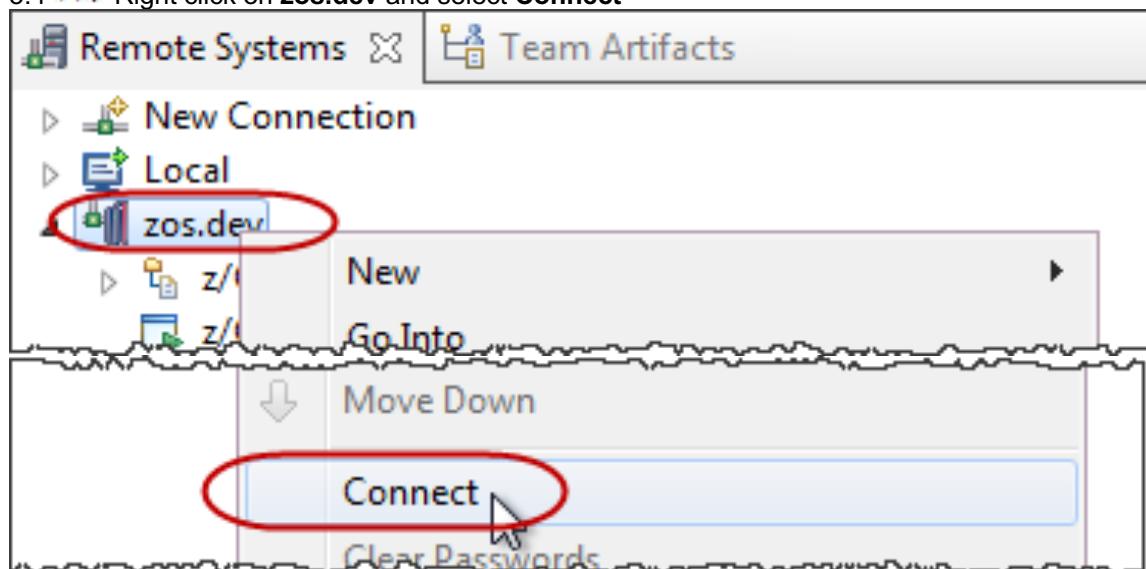


3.3 Before you used `empot01`, `empot05` or `ibmuser` and you need to disconnect. Your new userid now will be `empot02`.

► Right click on `zos.dev` and select **Disconnect**



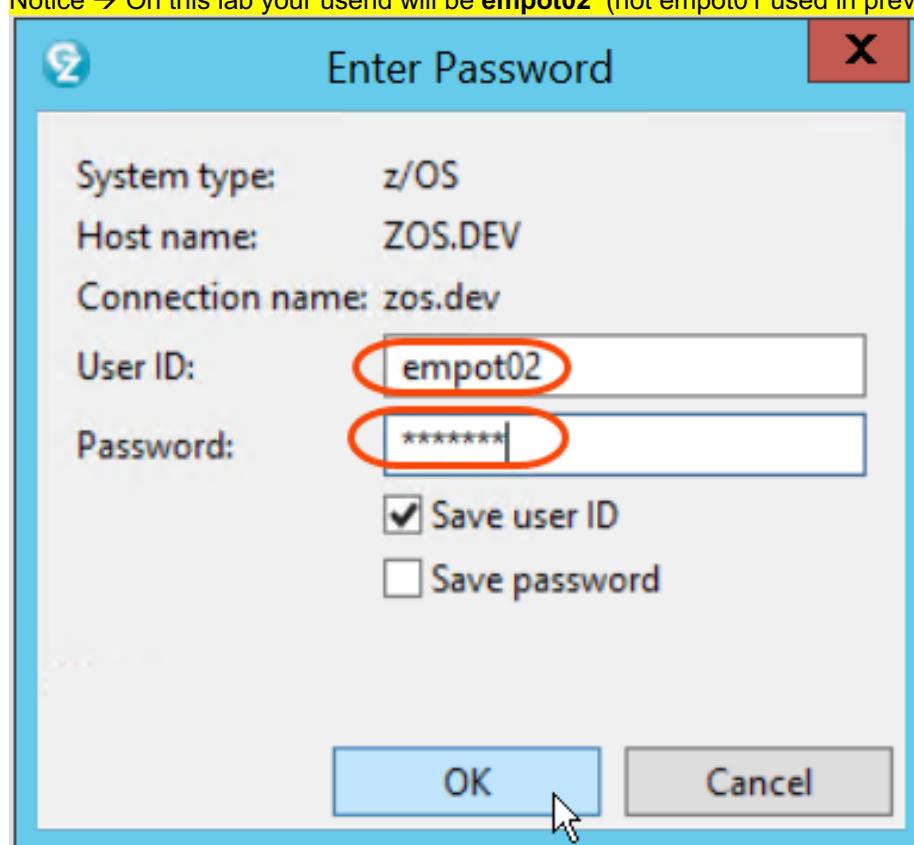
3.4 ► Right click on **zos.dev** and select **Connect**



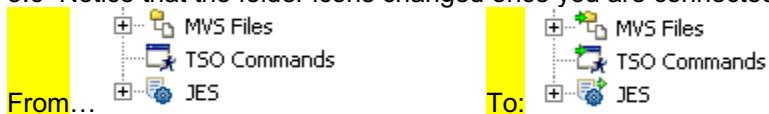
3.5 ► Type **empot02** as userid and **empot02** as password.

Click **OK** to connect to z/OS.

Notice → On this lab your userid will be **empot02** (not empot01 used in previous labs)



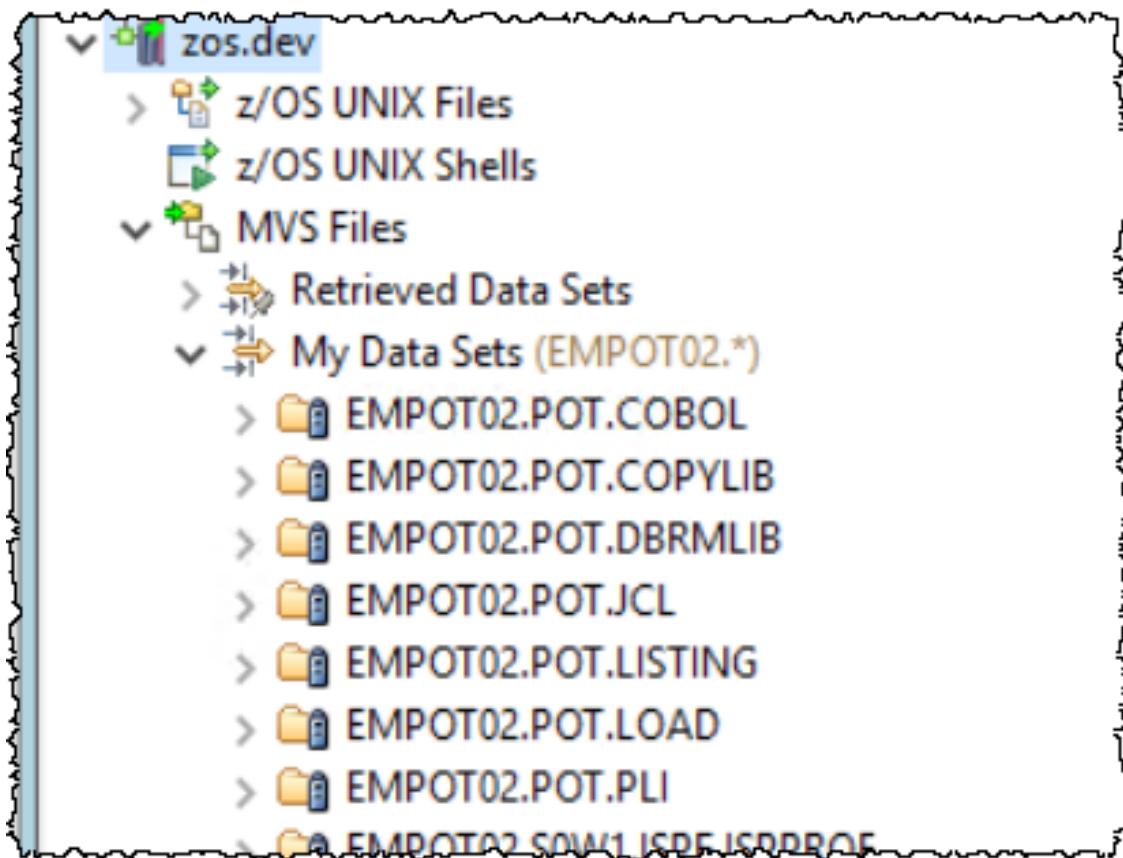
3.6 Notice that the folder icons changed once you are connected to z/OS. A small green arrow is added.,,



3.7 ► Expand MVS Files and My Data Sets (to see all your MVS data sets

Note that you do not have the same PDSs shown below (EMPOT02.*) this is just an example..

Depending on which ID you are using you may have no data sets, the IDs are reused and we have no control what is there.



You are connected to a z/OS remote system. Now you will create a Ship List and submit JCL to create the executables to be deployed.

3.8 You now can create the UCD ship list. An XML file that has the list of the load modules that will be deployed. This xml could be created from an SCM build tool like RTC, Endevor etc.. For this lab we will create it manually. .

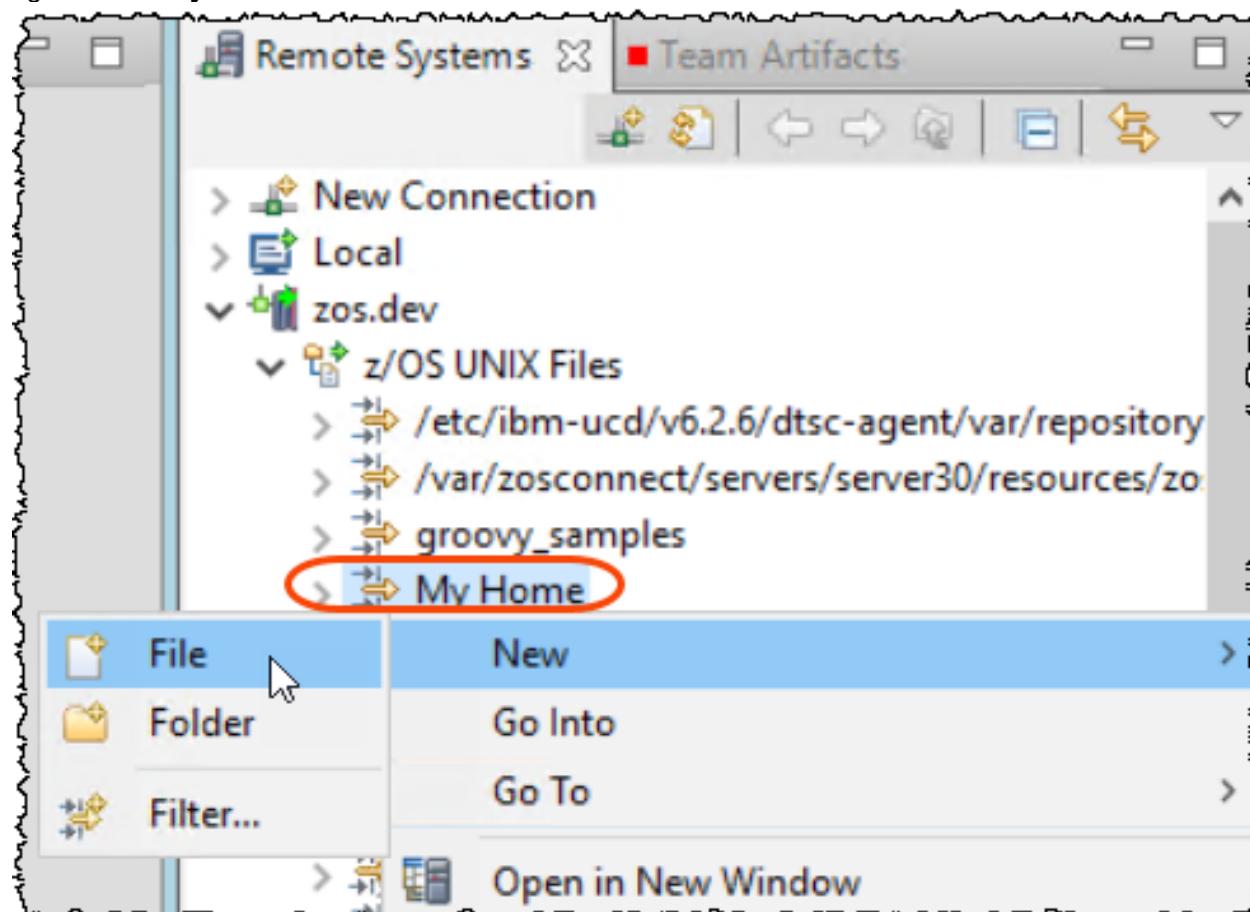
What is the Ship list file?

The ship list file specifies which files from the build to include in the new component version to deploy. You must create a ship list file before you run the IBM® z/OS® deployment tools.

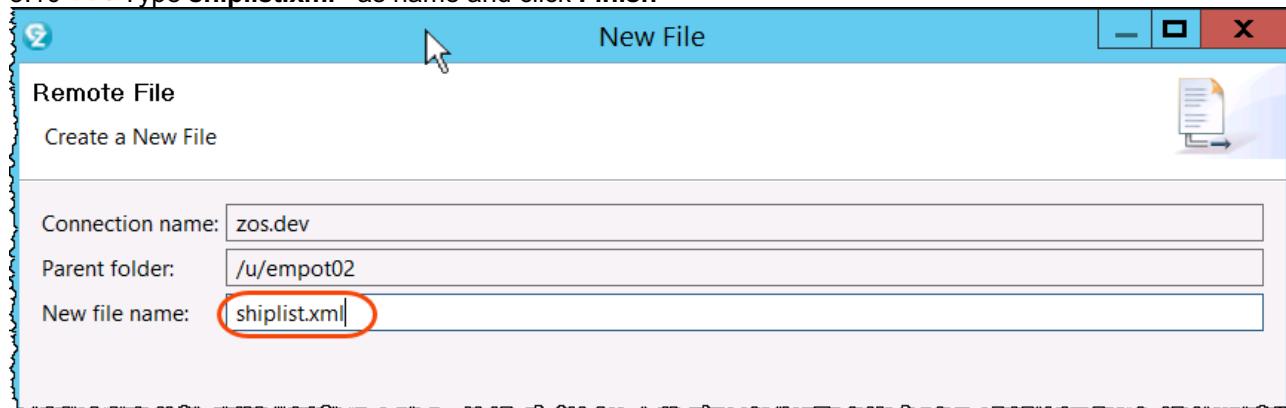


Ship list files are XML files that contain a list of files. The container type that is used to identify partitioned data set (PDS) resources is PDS and the resource type is *PDSmember*. You can use an asterisk (*) as a wildcard for the resource name, if you want all members in a partitioned data set (PDS) to be included in a package. Typically, you write a script that works with your build engine to create a ship list file from the build output.

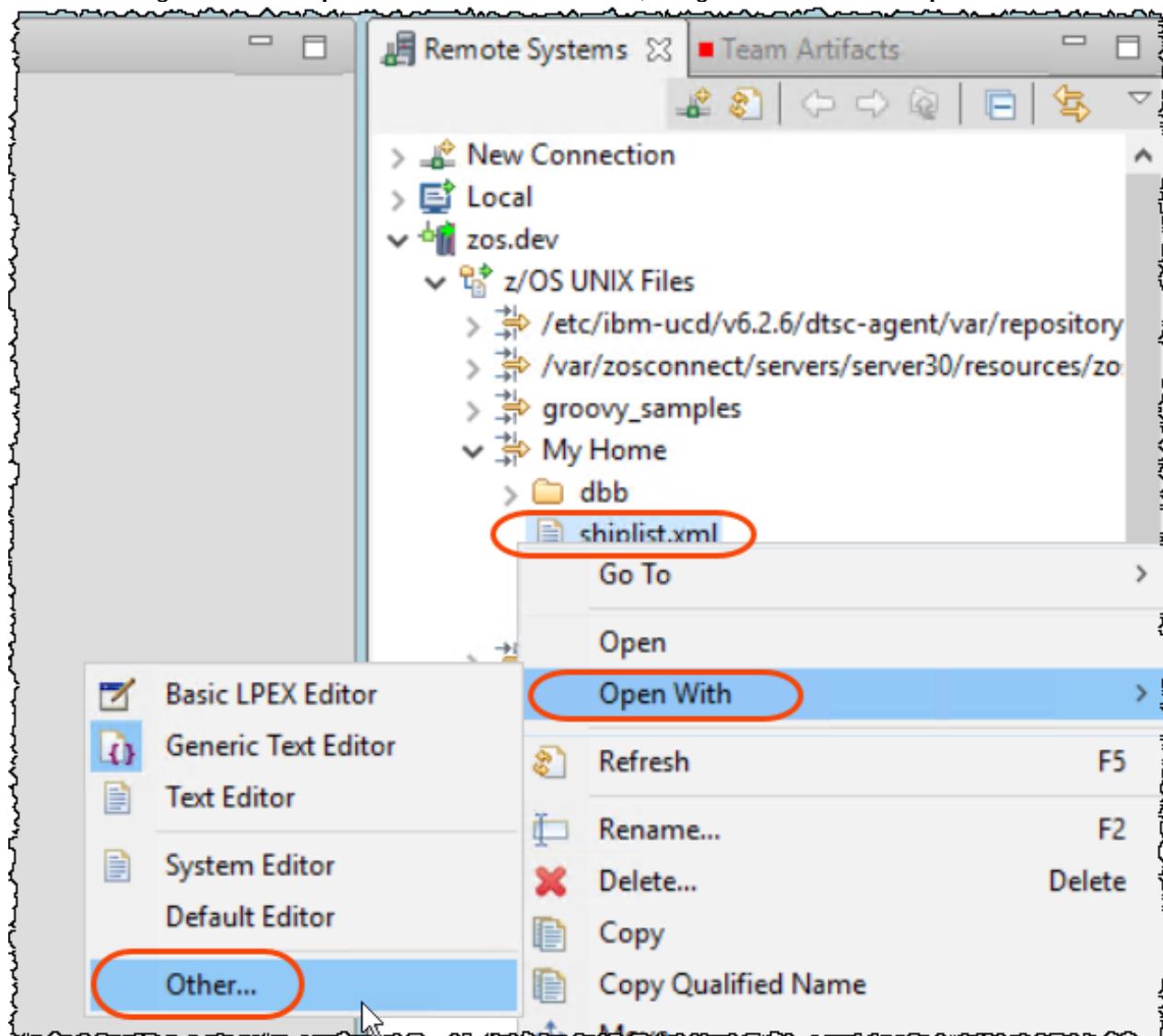
- 3.9 ► Using the *Remote Systems* view (on top right), expand **z/OS UNIX Files**, right click on **My Home** → **New** → **File**



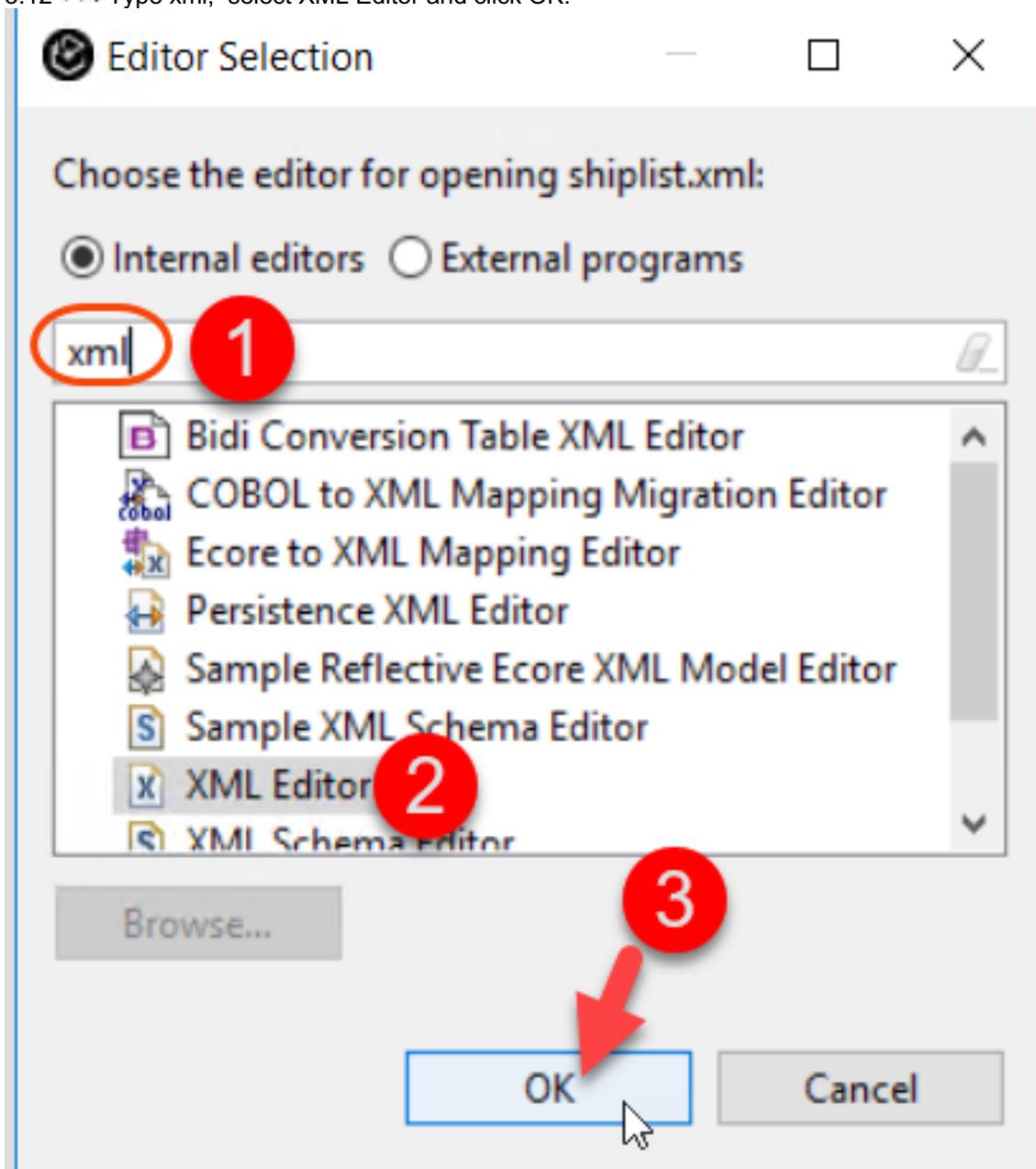
3.10 ► Type **shiplist.xml** as name and click **Finish**



3.11 ► Right click on **shiplist.xml** and select it to edit, or right click and select **Open With Other...**



3.12 ► Type xml, select XML Editor and click OK.



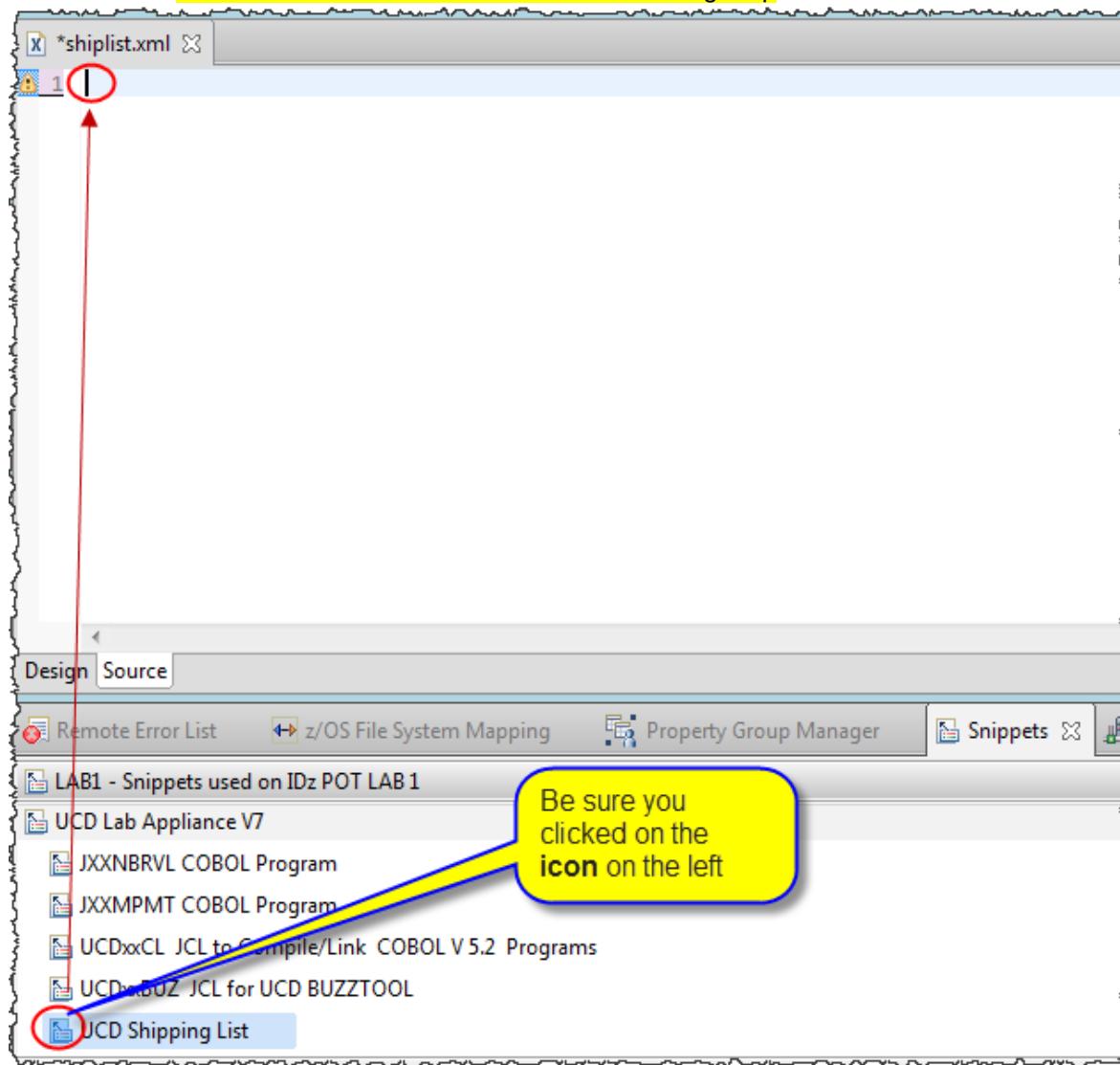
3.13 ► Click on **Source** tab



3.14 ►| Click on **Snippets** tab on the bottom.

►| Scroll down and use the **UCD Shipping List** snippets to drag and drop to the **FIRST position** of the file

IMPORTANT: You must click on the icon  to be able to drag/drop



3.15 ►| When the dialog opens type **02** ..

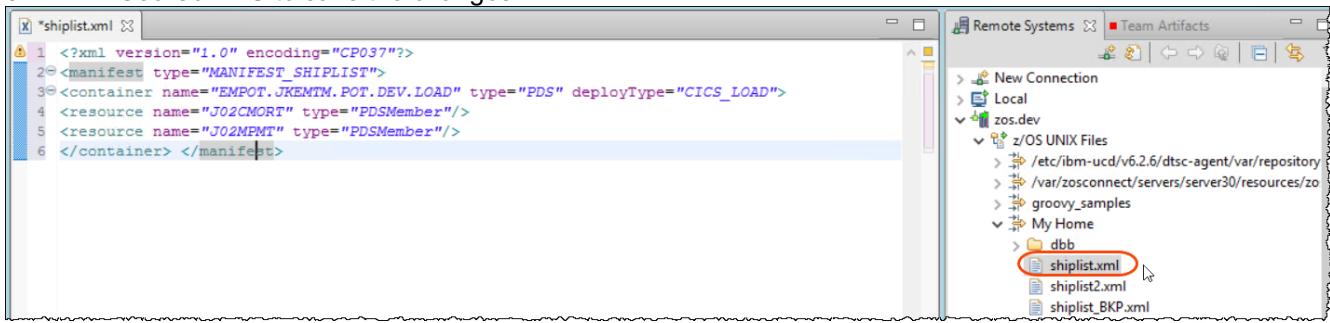
3.16 ►| Click **Insert** to insert the lines. Notice that the variables are replaced

```

1 //UCD02BUZ JOB ,MSGLEVEL=(1,1),NOTIFY=&SYSUID,REGION=0M
2 /**
3  /* 1. This job executes the BUZTOOL to create a VERSION
4  /* 2. Edit the Shipping List to see what will be packed
5  /* on this example is at /u/empot02/shiplist.xml
6  /* 3. Modify the version name and submit to z/OS execution
7  /* 4. When completed verify the USS version folder is created at
8  /* /etc/ibm-ucd/v6.2.6/dtsc-agent/var/repository/J02Mortgage
9  /* 5. Go to URBAN CODE and run the DEPLOY at DEV level
10 /**
11 //BUZTOOL EXEC PGM=BPXBATCH,REGION=0M
12 //STDOUT DD SYSOUT=*
13 //STDERR DD SYSOUT=*
14 //STDPARM DD *
15 SH /var/ucd/agent72/bin/buztool.sh createzosversion
16     "-c" "J02Mortgage"
17     "-s" "/u/empot02/shiplist.xml"
18     "-v" "20240423"
19 /*
20 /* yyyyymmdd will be the name of your version
21 //

```

3.17 ➔ Use Ctrl + S to save the changes



Task 4 - Create the UCD component version from JCL

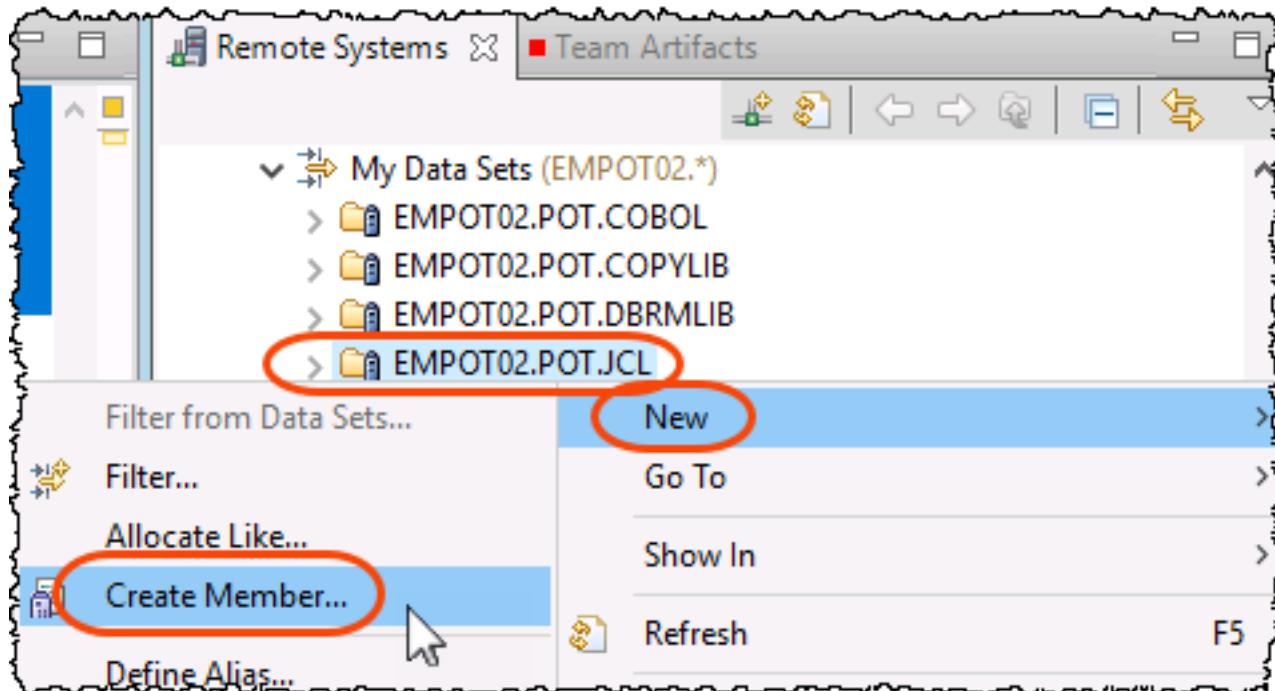
On this task, you will now use the z/OS UCD tool named “**BUZTOOL**” to create a version of the modules that need to be deployed. The UCD Toolkit installed on z/OS used the UCD Client to communicate to the UCD

server.

In this lab the UCD code station on z/OS is kept in the USS Files or on UCD server (installation decision).

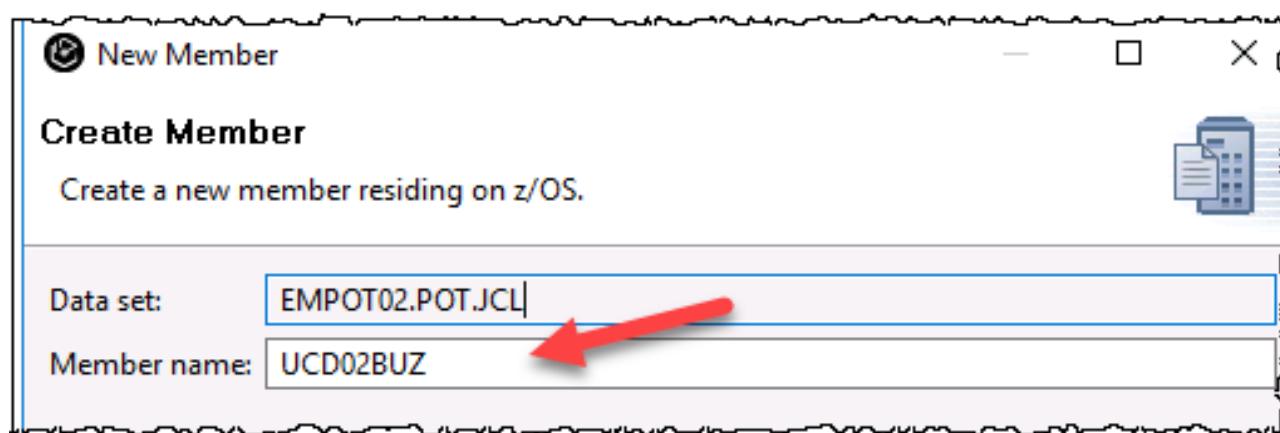
- 4.1 ► Using Remote System view, expand **My Data Sets**, right click on **EMPOT02.POT.JCL** and create a member.

If you don't have this PDS call the instructor.

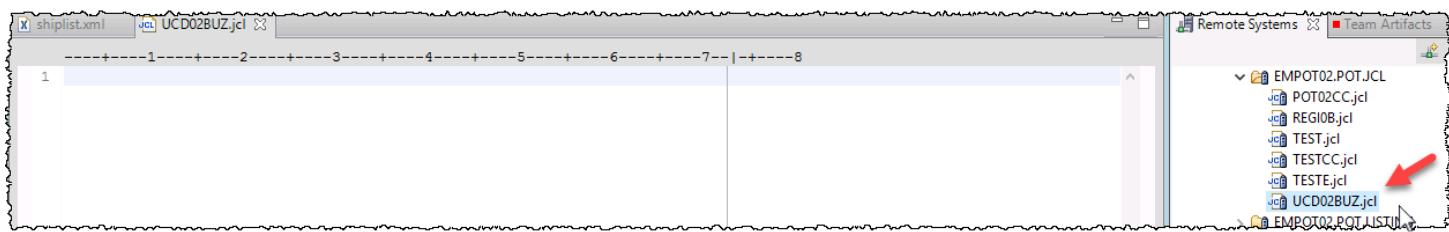


- 4.2 ► Name it **UCD02BUZ** and click **Finish**

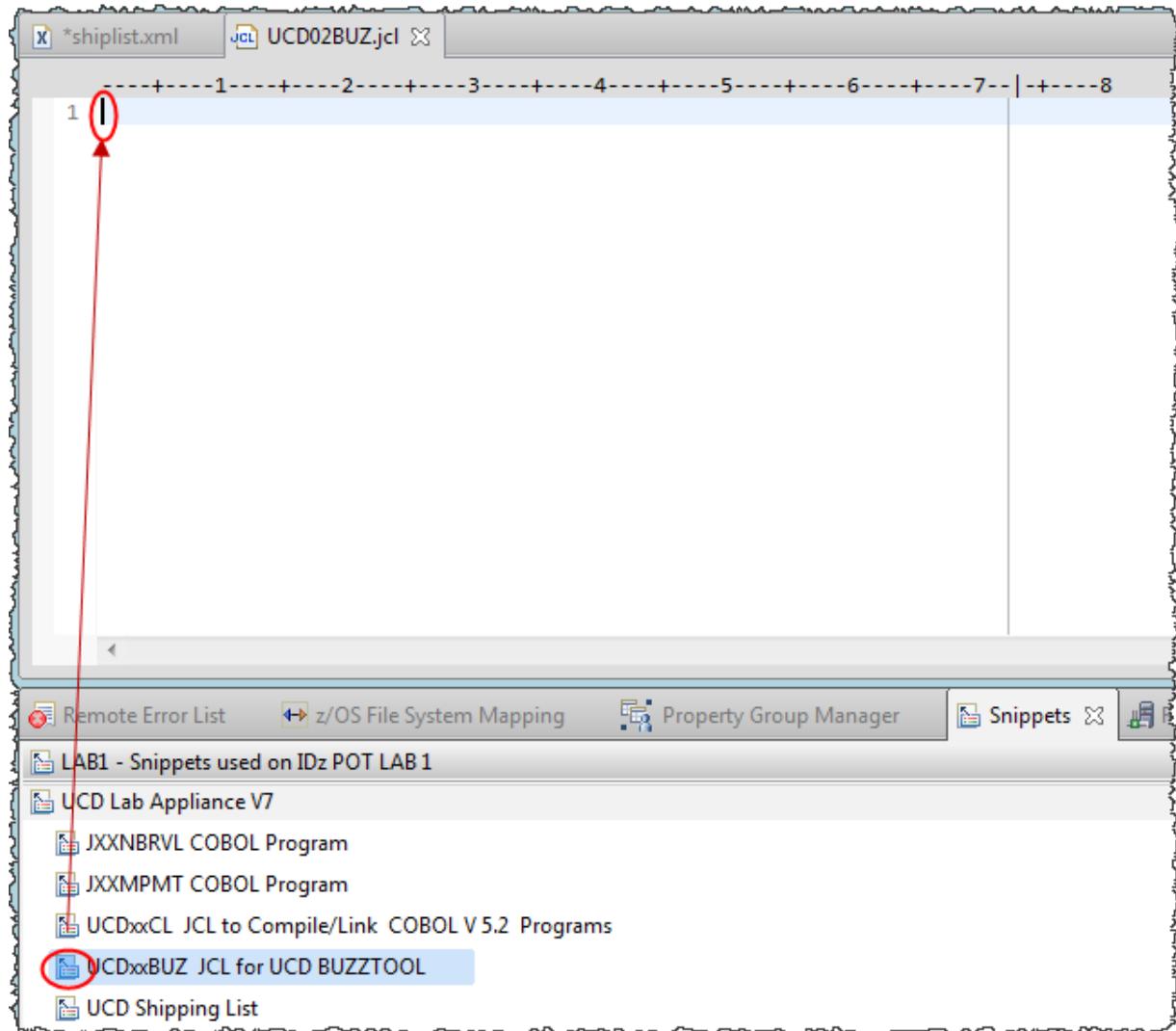
NOTE: If you have already this member there, right click and delete it.



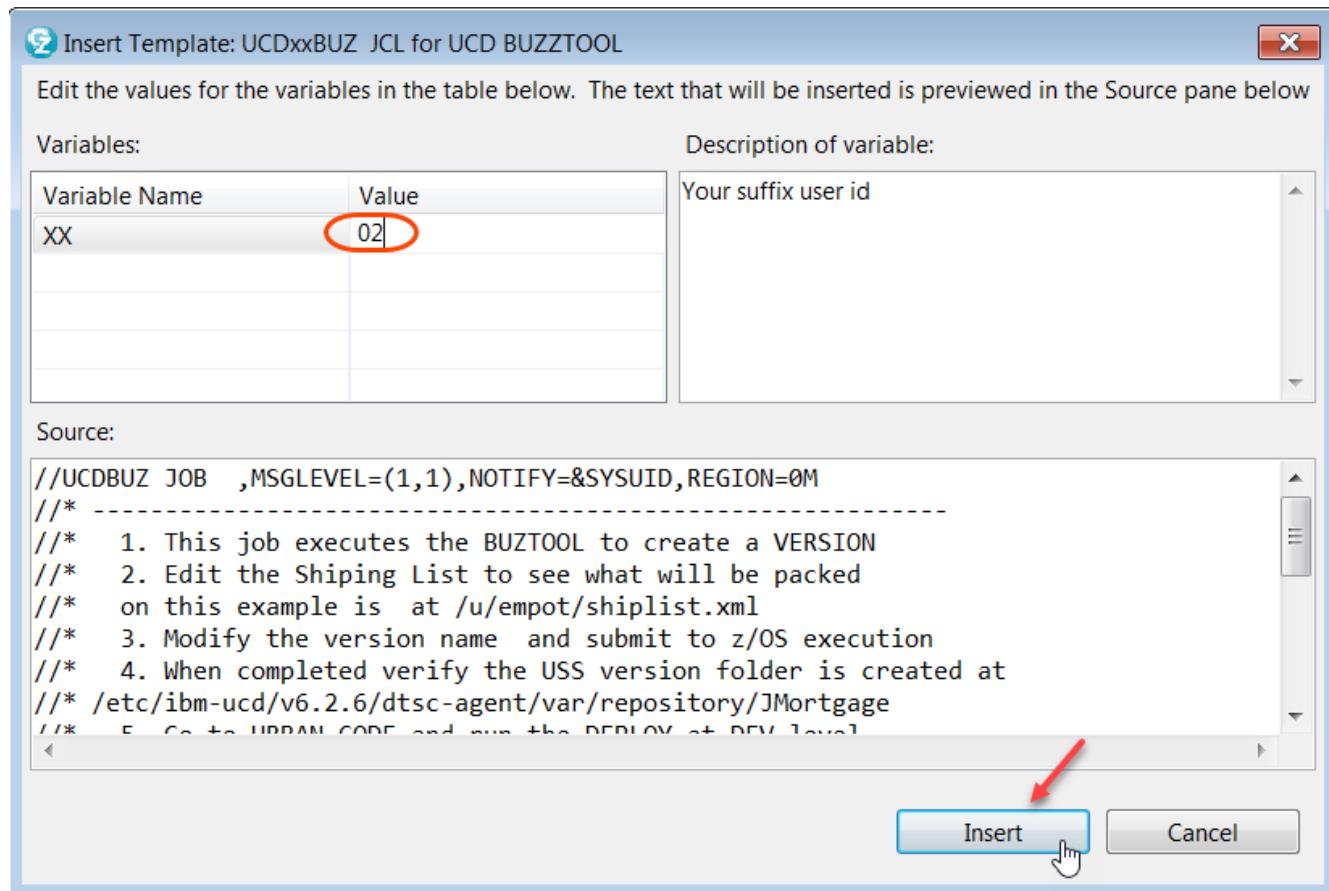
- 4.3 ► Double click on **UCD02BUZ.jcl** to edit.



- 4.4 ►| Use the **UCDxxBUZ** snippets to drag and drop to the **FIRST POSITION** of the file
IMPORTANT: You must click on the icon to be able to drag/drop

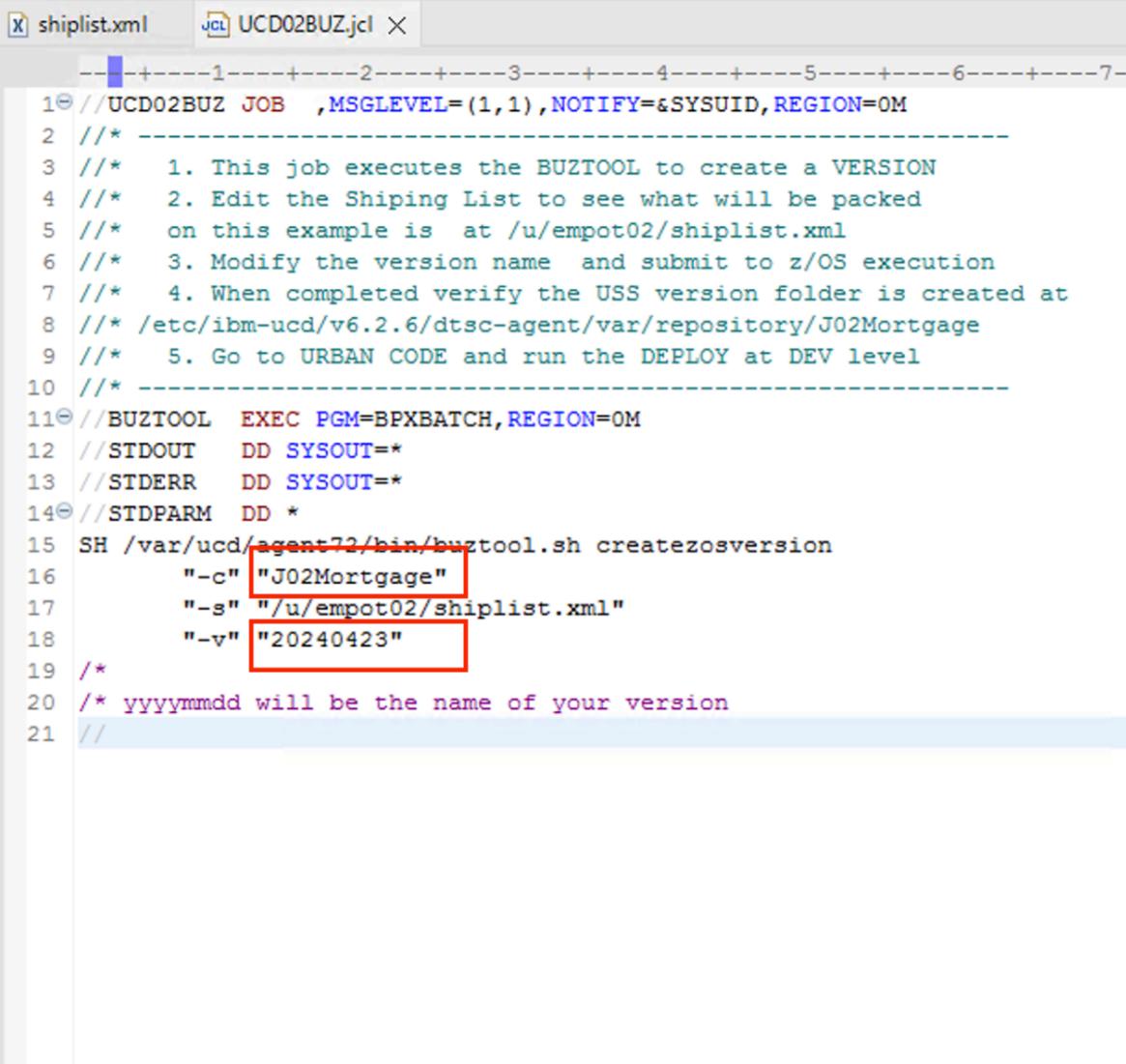


- 4.5 ►| When the dialog opens type **02**. ..
►| Click **Insert** to insert the lines. Note that the variables are replaced



This job uses the **buztool createzosversion** command to create the component version.

4.6 ➤ Scroll down and understand the parameters passed to the *BUZTOOL*



```

1④ //UCD02BUZ JOB ,MSGLEVEL=(1,1),NOTIFY=&SYSUID,REGION=0M
2 /**
3   1. This job executes the BUZTOOL to create a VERSION
4   2. Edit the Shipping List to see what will be packed
5   3. Modify the version name and submit to z/OS execution
6   4. When completed verify the USS version folder is created at
7   5. Go to URBAN CODE and run the DEPLOY at DEV level
8 /**
9 /**
10 /**
11④ //BUZTOOL EXEC PGM=BPXBATCH,REGION=0M
12 //STDOUT DD SYSOUT=*
13 //STDERR DD SYSOUT=*
14④ //STDPARM DD *
15 SH /var/ucd/agent72/bin/buztool.sh createzosversion
16     "-c" "J02Mortgage"
17     "-s" "/u/empot02/shiplist.xml"
18     "-v" "20240423"
19 /*
20 /* yyyyymmdd will be the name of your version
21 //

```

Possible arguments for the buztool [createzosversion](#) command



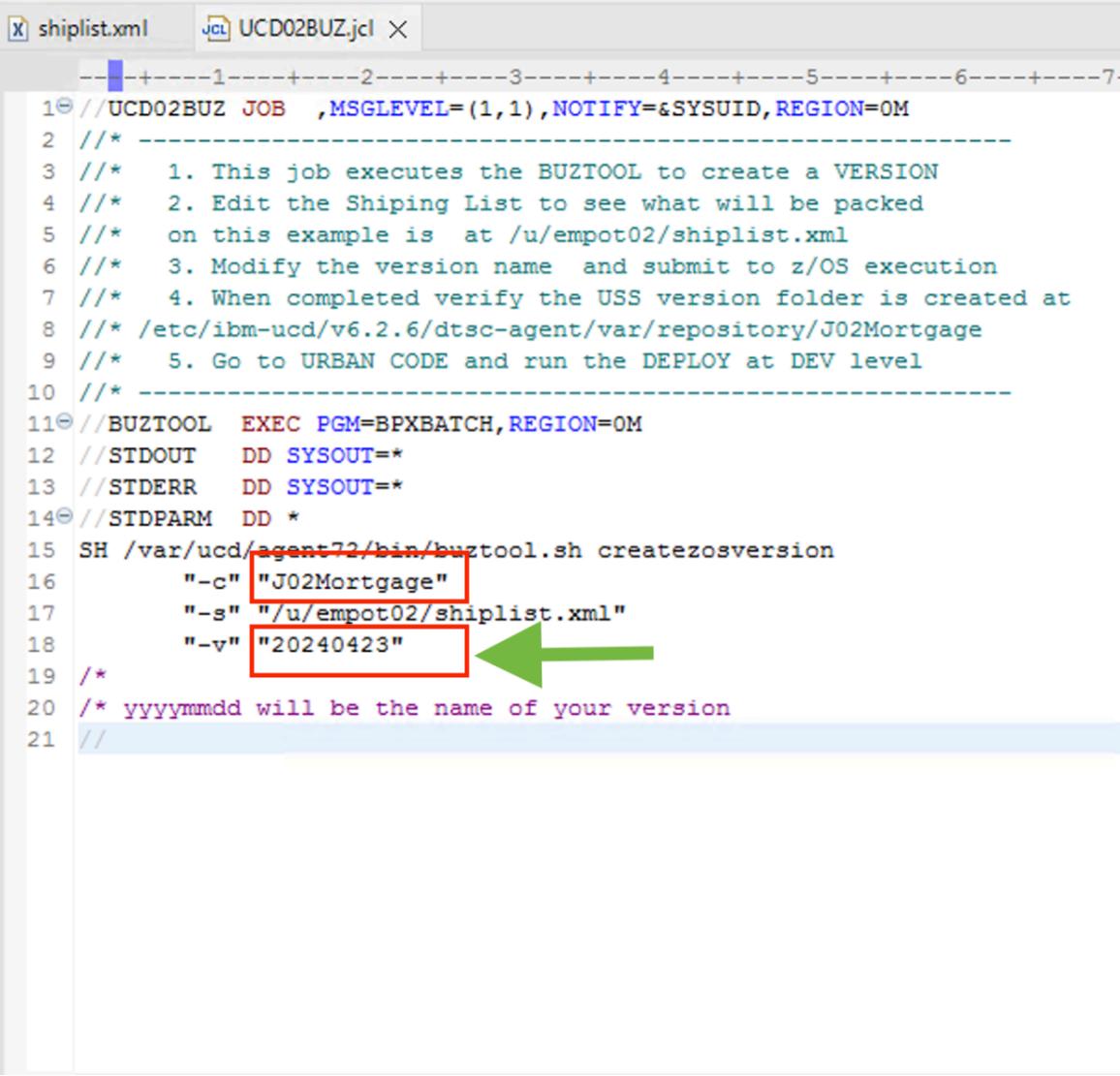
Parameter	Required	Description
-c	true	The name of the component in IBM UrbanCode Deploy. The component name can contain only letters, numbers, and spaces.
-v	false	The name of the version to create. If a version is not specified, a version name is generated from the current time stamp. The version name can contain only letters, numbers, and spaces.

	-s	true	The location of the ship list file.
	-verb	false	To display a trace log, set this parameter to true.

4.7 ►| Name a version for this component:

Modify from "yyyyymmdd" to today's date in the format of **year, month and day**

Example: I used **20240423**. This version name must be unique for this component to avoid errors of being already created on the code station.



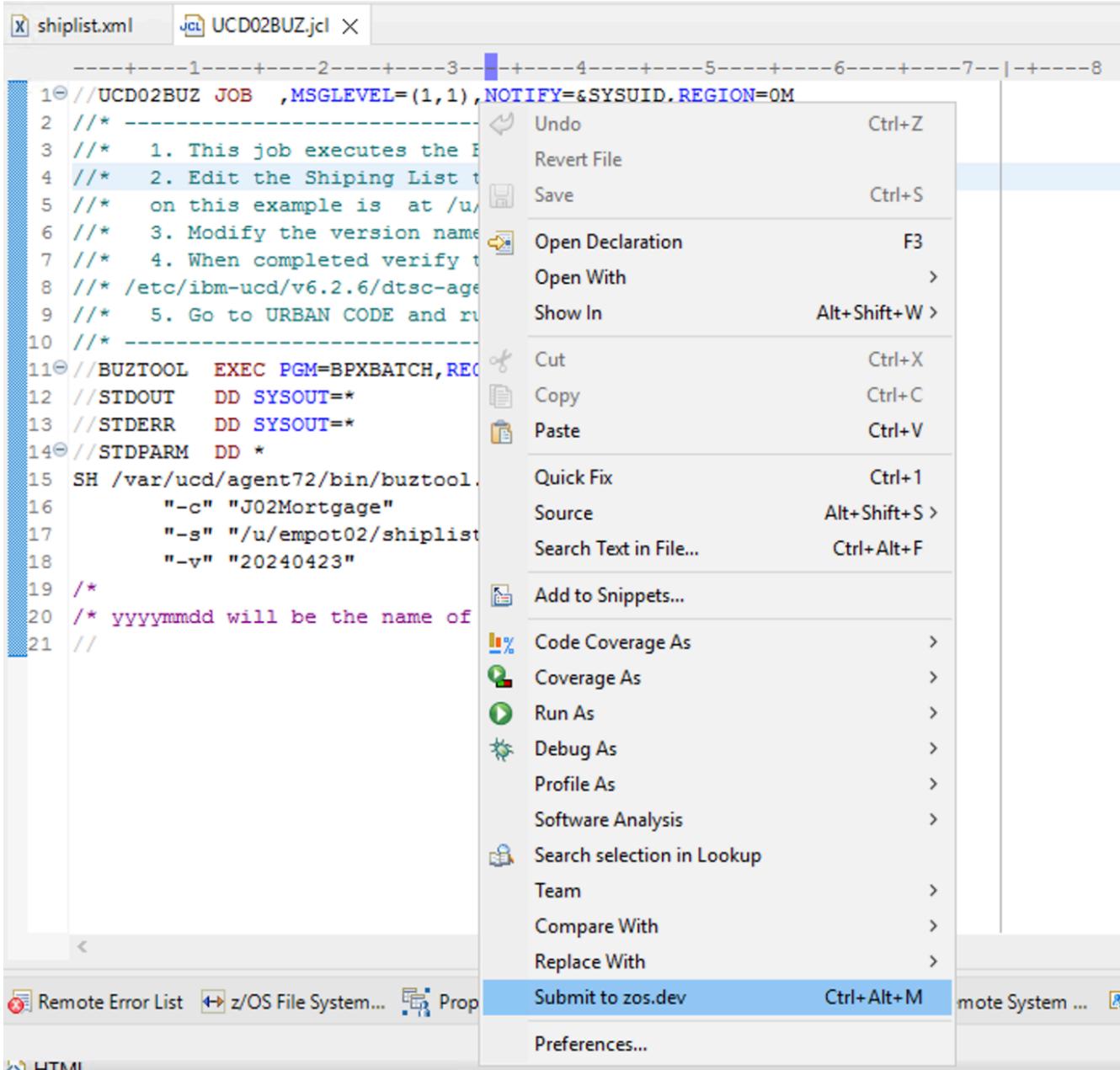
```

shiplist.xml UCD02BUZ.jcl X
-----1-----2-----3-----4-----5-----6-----7-----8
1 //UCD02BUZ JOB ,MSGLEVEL=(1,1),NOTIFY=&SYSUID,REGION=0M
2 /**
3 1. This job executes the BUZTOOL to create a VERSION
4 2. Edit the Shipping List to see what will be packed
5 3. Modify the version name and submit to z/OS execution
6 4. When completed verify the USS version folder is created at
7 //etc/ibm-ucd/v6.2.6/dtsc-agent/var/repository/J02Mortgage
8 /**
9 5. Go to URBAN CODE and run the DEPLOY at DEV level
10 /**
11 //BUZTOOL EXEC PGM=BPXBATCH,REGION=0M
12 //STDOUT DD SYSOUT=*
13 //STDERR DD SYSOUT=*
14 //STDPARM DD *
15 SH /var/ucd/agent72/bin/buztool.sh createzosversion
16     "-c" "J02Mortgage"
17     "-s" "/u/empot02/shiplist.xml"
18     "-v" "20240423" ←
19 /*
20 /* yyyyymmdd will be the name of your version
21 //

```

4.8 ►| Use **Ctrl + S** to save

►| Right click on the editor and select **Submit to zos.dev**



```

1 //UCD02BUZ JOB ,MSGLEVEL=(1,1),NOTIFY=&SYSUID.REGION=0M
2 /*
3  * 1. This job executes the BuzTool
4  * 2. Edit the Shipping List to
5  *    on this example is at /u/...
6  * 3. Modify the version name
7  * 4. When completed verify the
8  *    /etc/ibm-ucd/v6.2.6/dtsc-age
9  * 5. Go to URBAN CODE and run
10 /*
11 //BUZTOOL EXEC PGM=BPXBATCH,REC
12 //STDOUT DD SYSOUT=*
13 //STDERR DD SYSOUT=*
14 //STDPARM DD *
15 SH /var/ucd/agent72/bin/buztool.
16      "-c" "J02Mortgage"
17      "-s" "/u/empot02/shiplist"
18      "-v" "20240423"
19 /*
20 /* yyyyymmdd will be the name of
21 //

```

The context menu options include:

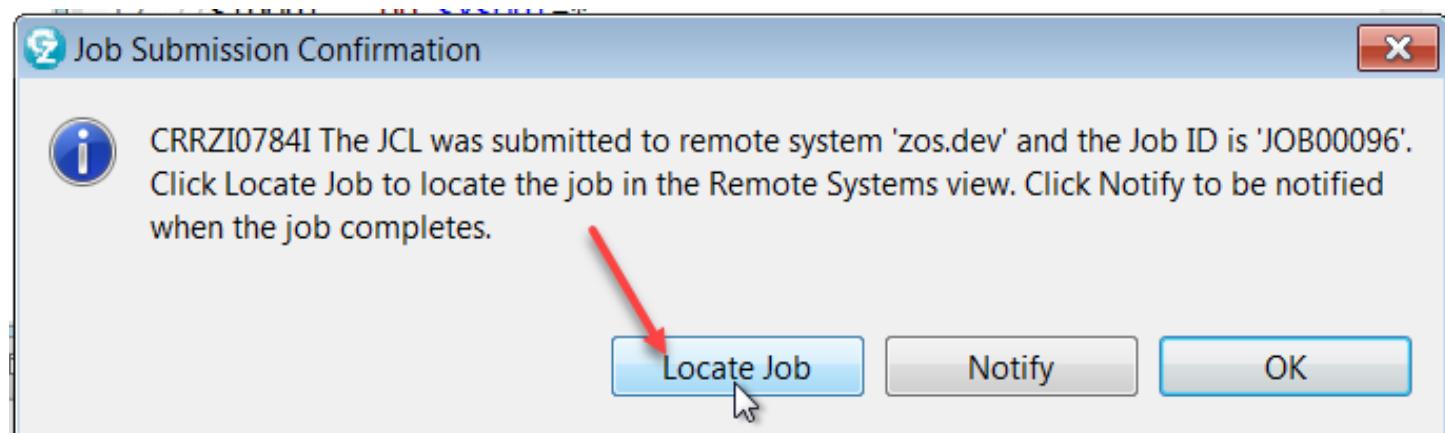
- Undo (Ctrl+Z)
- Revert File
- Save (Ctrl+S)
- Open Declaration (F3)
- Open With >
- Show In Alt+Shift+W >
- Cut (Ctrl+X)
- Copy (Ctrl+C)
- Paste (Ctrl+V)
- Quick Fix (Ctrl+1)
- Source Alt+Shift+S >
- Search Text in File... Ctrl+Alt+F
- Add to Snippets...
- Code Coverage As >
- Coverage As >
- Run As >
- Debug As >
- Profile As >
- Software Analysis >
- Search selection in Lookup
- Team >
- Compare With >
- Replace With >
- Submit to zos.dev** Ctrl+Alt+M
- Preferences...



You do not find the option **Submit?**...

Be sure you are using the correct editor. Either LPEX or JCL editors will provide this capability when right clicking. Be sure you pasted the content on line 1 and column 1. If you still don't see, you can right click on the editor content and select **Open with → Other -> JCL Editor** and try again..

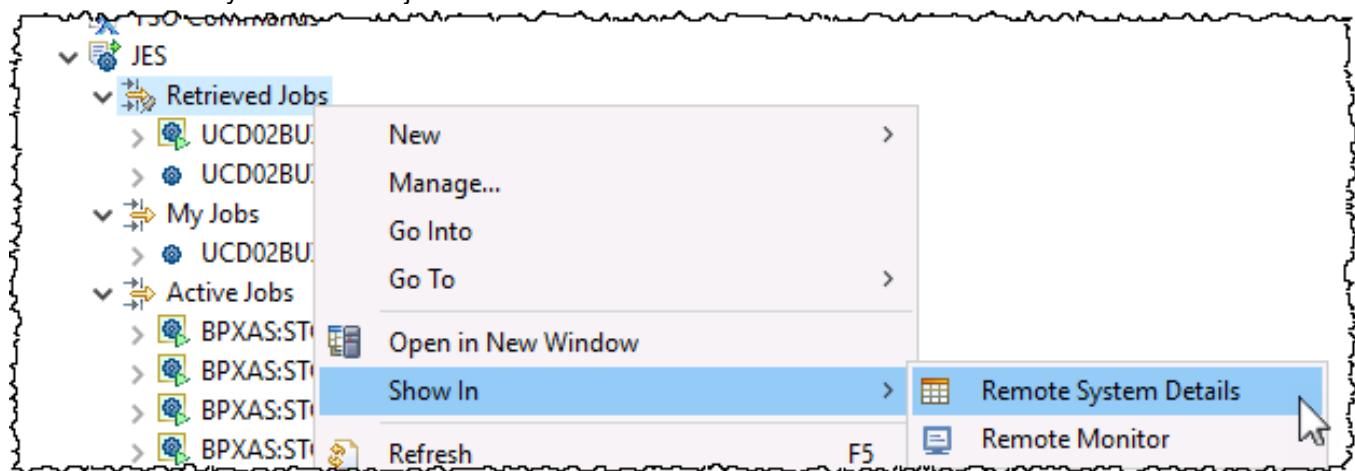
4.9 ►| Click **Locate Job** to get the job results



This job may take as long as 1 or 2 minutes.. Remember that your z/OS instance on cloud has limited power resources and is very slow.

- 4.10 ► On the right under *Remote System* view right click on **Retrieve Jobs** and select **Show in > Remote System Details**.

This is a better way to review the job execution



- 4.11 ► Use the **refresh icon** to see the job being executed. Once execution is completed you must have a *User Return Code*

Resource	Job ID	Job Name	Job Own...	Job Entr...	Return C...	Return In...	System Ret...	User Ret...	Return Stat...	Queue Pos...	System Na...
UCD02BUZ:JOB00096	JOB00096	UCD02B...	EMPOT02	2018/07...	CC 0000	NORMAL		000	COMPLE...	10	

The return code **must be 0** and the version must be created on UCD Server

Note: you might receive return code 512. To continue your Lab, **disconnect and connect** change as **IBMUSER** and password is **sys1**.

On Remote Systems. Open **UCD02BUZ.jcl** on **EMPOT02.POT.JCL** folder, redo items **4.8,4.9 ,and 4.10.**

Resource	Job ID	Job Name	Job Owner	Job Entry Date	Return Code	Return Info	System Ret...	User Ret...	Return Stat...	Queue Pos...	System Na...
UCD02BUZ:JOB00205	JOB00205	UCD02BUZ	IBMUSER	2024/04/23 16:58:42	CC 0000	NORMAL		000	COMPLE...	35	A
UCD02BUZ:JOB00200	JOB00200	UCD02BUZ	EMPOT02	2024/04/23 16:53:02	CC 0512	NORMAL		200	COMPLE...	34	A

- 4.12 ► Double click and scroll down to see the results.

```

[X] shiplist.xml [JCL UCD02BUZ.jcl [UCD02BUZ:JOB00218.spool X
-----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+-----6-----+-----7-----+-----8
45 IEF142I UCD02BUZ BUZTOOL - STEP WAS EXECUTED - COND CODE 0000
46 IEF285I IBMUSER._UCD02BUZ.JOB00218.D0000102.?           SYSOUT
47 IEF285I IBMUSER._UCD02BUZ.JOB00218.D0000103.?           SYSOUT
48 IEF285I IBMUSER._UCD02BUZ.JOB00218.D0000101.?           SYSIN
49 IEF373I STEP/BUZTOOL /START 2024115.0832
50 IEF032I STEP/BUZTOOL /STOP 2024115.0832
51          CPU:    0 HR 00 MIN 00.00 SEC   SRB:    0 HR 00 MIN 00.00 SEC
52          VIRT: 168K SYS: 240K EXT: 132K SYS: 10068K
53          ATB- REAL: 1256K SLOTS:          OK
54          VIRT- ALLOC: 14M SHRD: 0M
55 IEF375I JOB/UCD02BUZ/START 2024115.0832
56 IEF033I JOB/UCD02BUZ/STOP 2024115.0832
57          CPU:    0 HR 00 MIN 00.00 SEC   SRB:    0 HR 00 MIN 00.00 SEC
58 SH /var/ucd/agent72/bin/buztool.sh createzosversion
59     "-c" "J02Mortgage"
60     "-s" "/u/empot02/shiplist.xml"
61     "-v" "20240424"
62 zOS toolkit config : /var/ucd/agent72/ (7.2.0.0,20210527-0014)
63 zOS toolkit binary : /var/ucd/agent72/ (7.2.0.0,20210527-0014)
64 zOS toolkit data set : BUZ720 (7.2.0.0,20210615-0431)
65 Reading parameters:
66 ....Command : createzosversion
67 ....Component : J02Mortgage
68 ....Version : 20240424
69 ....Shiplist file : /u/empot02/shiplist.xml
70 Verifying version
71 ....Repository location : /var/ucd/agent72/var/repository/J02Mortgage/20240424
72 Pre-processing shiplist:
73 ....Shiplist after processing :/var/ucd/agent72/var/repository/J02Mortgage/20240424/shiplist.xml
74 Packaging data sets:
75 ....Location to store zip : /var/ucd/agent72/var/repository/J02Mortgage/20240424
76 ....Zip name : package.zip
77 ....EMPO.T.JKEMTM.POT.DEV.LOAD.bin
78 ....Elapsed time for data set package or deploy operation : 0.386642
79 Create version and store package:
80 ....Version created in UCD server
81 ....Version:20240424 created
82 Elapsed time: 4.0 seconds.
83

```

4.13 Close or minimize IDz, you will not need any more on this lab.

Notice that during version creating process, the *BUZTOOL* communicates with your UCD server to obtain component information and to store version artifacts. If component version was created successfully, you can verify version information using UCD server interface.

4.14 Now Using the UCD browser, click **Components -> J02Mortgage**

UrbanCode Deploy Dashboard Components Applications Processes Resources Calendar Work Items More empot02

Components Components Templates

View By: Component Bulk Actions Import Components Create Component Customize Display

Name	Tags	Latest Import	Latest Version	Template	Description	Created	By
CBSA-backend			version-718	Mainframe Deploy		3/6/2023, 3:55 AM	admin
Genapp-backend			version-1089	Mainframe Deploy		3/3/2023, 4:27 AM	admin
J02Mortgage			20240423	MVSCOMPONENT		4/23/2024, 3:49 PM	empot02
MYCOMP			20230216-125003			2/3/2023, 3:39 PM	admin

Items per page: 10 | 1-4 of 4 items 1 of 1 pages < Previous 1 Next Refresh

4.15 ► Click , ① Versions and the ② version that you created (yyymmdd) .

UrbanCode Deploy Dashboard Components Applications Processes Resources Calendar Work Items More empot02

Components / J02Mortgage

Component: J02Mortgage

Created By empot02
Created On 4/23/2024, 3:49 PM
Template MVSCOMPONENT

Versions History Usage Configuration Calendar Processes Changes

Customize Display

Version	Statuses	Type	Created By	Date	Is Importing	Description
20240423		Incremental	admin	4/23/2024, 4:58 PM	false	

Items per page: 50 | 1-1 of 1 item 1 of 1 pages < Previous 1 Next Refresh

4.16 You can verify that the repository is on UCD Server Code Station (on Linux, not z/OS USS Files)

The screenshot shows the UrbanCode Deploy interface with the 'Components' tab selected. A breadcrumb navigation bar at the top indicates: Components / J02Mortgage / Versions / Version: 20240423. Below this, the page title is 'Version: 20240423'. On the left, there's a table with columns: 'Created By' (admin), 'Created On' (4/23/2024, 4:58 PM), and 'Repository Type' (CodeStation, highlighted with a red box). To the right, there's a 'Links' section with a 'Create' button and a message 'No links found.' with a link 'View all links'.

4.17 ►► Expand **EMPOT.JKEMT.POT.DEV.LOAD** and you should be able to see the list load modules that are packaged and stored in the Code Station (from the JCL that you submitted before).

Optionally the Code Station could be on the z/OS under USS folders.

In our example the Code Station is on the UCD server (LINUX):

The screenshot shows a table with columns: Name, Artifact Type, Deploy Type, Inputs, Last Modified, and Properties. There are four rows. The first row is expanded, showing three sub-items: J02CMORT and J02MPMT, both under the artifact type [PDS,ADD] and deploy type CICS_LOAD. The entire expanded row is highlighted with a red box.

Name	Artifact Type	Deploy Type	Inputs	Last Modified	Properties
▼ EMPOT.JKEMTM.POT.DEV.LOAD	[PDS,ADD]	CICS_LOAD			
J02CMORT		CICS_LOAD			
J02MPMT		CICS_LOAD			

Task 5 - Create UCD Resources

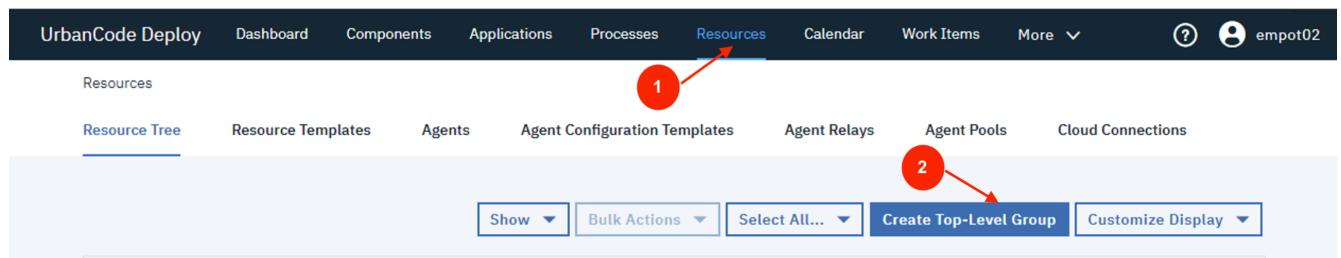
On this task you will create the UCD resources that is a user-defined construct.



UCD : What is a resource?

A resource is a logical deployment target that typically resolves to an agent and a user-defined construct that is based on the architectural model of UCD. Resources can contain other resources in a hierarchical tree structure (called also resource groups or folders) to represent complex targets. Resources are assigned to environments.

5.1 ► On the **Resources** tab, click **Create Top-Level Group**:



5.2 ► Type **zOS Resource Group 02** and click **Save**.

Create Resource

Name

Description

Teams [Add Teams](#)

Impersonation [Cancel](#) [Save](#)

You should have:

⋮	Name	Tags	Inventory	Status	Description	⋮
⋮	zOS Environments					⋮

5.3 ► Move the mouse to **ZOS Resource Group**, click the **...** (three dots) and using the drop-down menu click **Add Agent**:

The screenshot shows the 'Resource Tree' tab selected in the navigation bar. Below it is a table with columns: Name, Tags, Inventory, Status, and Description. Two rows are visible: 'zOS Resource Group 02' and 'zOS Environments'. A context menu is open over the 'zOS Resource Group 02' row, with the 'Add Agent' option highlighted. Red numbered callouts point to the three-dot menu icon (1) and the 'Add Agent' option (2).

5.4 ► In the **Agent** drop down dialog, select **zdtagent** and click **Save**

The screenshot shows the 'Create Resource' dialog. The 'Agent' dropdown is set to 'zos.dev2'. The 'Name' field contains 'zos.dev2'. The 'Description' field is empty. The 'Inherit Teams From Parent' checkbox is checked. The 'Teams' section shows 'Team 02'. The 'Save' button at the bottom is highlighted with a red arrow.

5.5 ► In the row **zdtagent**, click the **...** (three dots) and using the drop-down menu click **Add Component**:

The screenshot shows the IBM Software interface with the 'Resource Tree' tab selected. The main area displays a table with columns: Name, Tags, Inventory, Status, and Description. The table contains three rows: 'zOS Resource Group_02', 'zos.dev2 (View Agent)', and 'zOS Environments'. The 'zos.dev2' row is highlighted with a blue border. A context menu is open over this row, with two red circles numbered 1 and 2 pointing to specific options:

- Circle 1: 'Add Component' (highlighted with a red circle)
- Circle 2: 'Add Group' (highlighted with a red circle)

The context menu also includes other options: Edit, Delete, Add Group, Add Component Tag, Add From Template, Compare or Synchronize, Define New Template, and Synchronize With Template.

5.6 ► Select the component created earlier (**J02Mortgage**) and click **Save**.

The screenshot shows the 'Create Resource' dialog box. The 'Component' dropdown is set to 'J02Mortgage'. The dialog includes fields for 'Name' (set to 'J02Mortgage') and 'Description'. A checkbox 'Inherit Teams From Parent' is checked, and a 'Teams' section shows 'Team 02'. At the bottom, there are 'Cancel' and 'Save' buttons, with a red arrow pointing from the 'Save' button in the dialog to the 'Save' button in the context menu from the previous step.

You should have:

The screenshot shows the UrbanCode Deploy application management interface. At the top, there is a navigation bar with tabs: UrbanCode Deploy, Dashboard, Components, Applications (highlighted with a red arrow), Processes, Resources, Calendar, Work Items, Reports, and Settings. Below the navigation bar is a toolbar with buttons: View By: Application, Bulk Actions, Import Applications, Create Application (highlighted with a red arrow), and Customize Display. The main area displays a table of applications with columns: Name, Tags, Template, Description, Created, and By. Two applications are listed: 'CBSA' and 'Genapp'. At the bottom of the table, it says 'Items per page: 10 | 1-2 of 2 items'. Below this table is another section titled 'Resource Tree' with tabs: Resource Tree, Resource Templates, Agents, Agent Configuration Templates, Agent Relays, Agent Pools, and Cloud Connections. The 'Resource Tree' tab is selected. It shows a hierarchical tree structure with nodes like 'zOS Resource Group 02', 'zos.dev2 (View Agent)', and 'J02Mortgage (View Component)'. The node 'J02Mortgage (View Component)' is highlighted with a red box. Below the tree is a toolbar with buttons: Show, Bulk Actions, Select All..., Create Top-Level Group (highlighted with a red arrow), and Customize Display.

Task 6 - Create an UrbanCode Application

On this task you will create an UCD application.

Applications are responsible for bringing together all the components that must be deployed together.

UCD : What is a UCD Application?

Application is a module that contains the following elements:



- Components to be deployed
- Environments target and the resources to deploy the application.
- A process definition (note that application process runs on the server while component process runs on agent)
- Properties

UCD: Properties

Properties can be set in UCD for many different things, including components, environments,

processes, and applications. You can also set global properties for the system.

You can refer to a property by scope:

`${p:scope/propertyName}` for example: `${p:environemnt/DBconnect}`
or without scope:

`${p:propertyName}`

You define objects (Application, component, process, resource, etc) properties in the object's Configuration Tab of UCD browser interface.

For more details, refer to the full documentation on Properties:

http://www-01.ibm.com/support/knowledgecenter/SS4GSP_6.2.6/com.ibm.udeploy.reference.doc/topics/ud_properties.html

6.1 ► On the Applications tab, click Create Application:

The screenshot shows the UrbanCode Deploy application management interface. The top navigation bar includes links for Dashboard, Components, Applications (which is highlighted with a blue underline), Processes, Resources, Calendar, Work Items, Reports, and Settings. A user icon for 'empot02' is also present. Below the navigation, there are two tabs: 'Applications' (selected) and 'Templates'. A red circle labeled '1' points to the 'Applications' tab. A red circle labeled '2' points to the 'Create Application' button in the toolbar below the search/filter area. The main content area displays a table of applications with columns for Name, Tags, Template, Description, Created, and By. Two entries are listed: 'CBSA' and 'Genapp'. At the bottom, there are pagination controls for 'Items per page' (set to 10), '1 of 1 pages', and navigation buttons for 'Previous', 'Next', and page number '1'.

Name	Tags	Template	Description	Created	By
CBSA				3/6/2023, 3:54 AM	admin
Genapp				3/3/2023, 4:26 AM	admin

- 6.2 ► Provide the value for the *Name* field in the *Create Application* dialog
 Name it **J02Mortgage COBOL** and click **Save**

Create Application

Name *

J02Mortgage COBOL

Description

Teams

Team 02

Add Teams

Notification Scheme

None

Cancel Save

The result should be:

Applications / J02Mortgage COBOL

Application: J02Mortgage COBOL

Created By empot02
Created On 4/24/2024, 8:49 AM

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Task 7 - Create environment

On this task you will create The UCD environment that is a user-defined collection of resources that hosts an UCD application.

When you create an environment, you map resources to it that define where the parent application can run deployments.



UCD : What is a UCD Application Environment?

An environment is the application's mechanism for bringing together components with the agent that deploys them. Environments are typically modeled on some stage of the software project lifecycle, such as development, test, or production.

When you create an environment, you map resources to it and the application components that will be deployed.

You can also define environment specific properties.

7.1 ➡ Click Create Environment:

Applications / J02Mortgage COBOL

Application: J02Mortgage COBOL

Created By: empot02
Created On: 4/24/2024, 8:49 AM

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Drag environments by their names to re-order them. 0 Environments

Compare Environments Create Environment

Search by Name or Search by Blueprint

No Environments Found!

7.2 ➡ Provide the value for the Name field in the Create Environment dialog (for example, **Test**), choose a color, and click **Save**:

Create Environment

Name* 1

Description

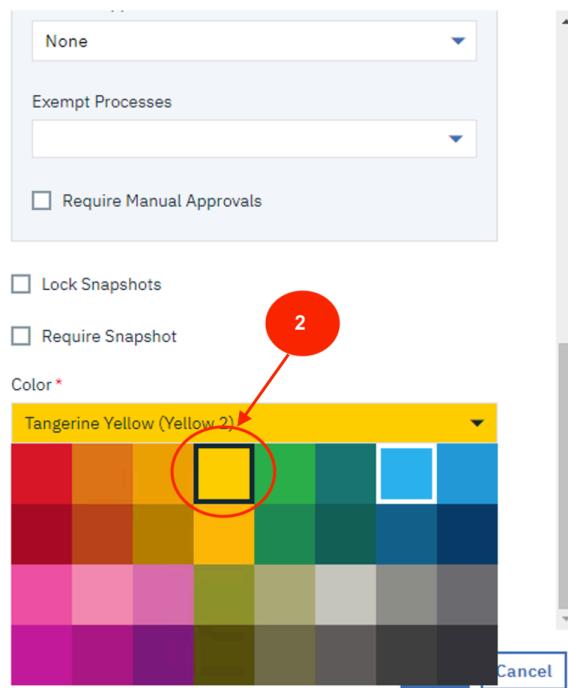
Blueprint

Teams
 Team 02

Deployment Approvals
Setting an approval will restrict deployments unless the approval process is satisfied. Two types of approval can be configured: Manual Approval Processes and External Approval Process.

And **scroll-down** and select the color (**Orange**) for your test and ➡ Click **Save**.

Create Environment



7.3 ► Click the environment name (**Test** in our example) to open environment properties:

Application: J02Mortgage COBOL

Created By empot02
Created On 4/24/2024, 8:49 AM

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Drag environments by their names to re-order them. 1 Environment

[Compare Environments](#)

[Create Environment](#)

Search by Name

or Search by Blueprint

[Collapse All](#)

> [Test](#)

Snapshot: None

Inventory: 0 / 0

7.4 ► Click Add Base Resources:

Environment: Test

Resources History Calendar Configuration Changes

No Desired Inventory

Show Bulk Actions Select All... Add Base Resources Customize Display

Name	Tags	Inventory	Status	Description
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Name"/>	<input type="button" value="Tags"/>	<input type="button" value="Versions"/>	<input type="button" value="Filter"/>	

No resources found.

7.5 ►| Expand zOS Resource Group 02 and os.dev2.

►| Select Component **J02Mortgage**: and click **Save**:

Note: In order for a component to be deployed by an application, it must be added to the application and also mapped to an agent-type resource. A component that is added to an application but not mapped to an agent resource, cannot be deployed by that application.

Similarly, a component that is mapped to an agent resource but not added to an application, cannot be deployed by that application.

Add Resource to Environment

Customize Display

Name	Tags	Inventory	Description
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="button" value="Name"/>	<input type="button" value="Tags"/>	<input type="button" value="Versions"/>	<input type="button" value="Filter"/>

zOS Environments

zOS Resource Group 02

zos.dev2

J02Mortgage

Refresh

1

2

Cancel Save

You should have this:

The screenshot shows the 'Environment: Test' section of the UrbanCode Deploy web interface. At the top, there are tabs for 'Resources', 'History', 'Calendar', 'Configuration', and 'Changes'. Below the tabs, a message says 'No Desired Inventory'. A large table lists components. One row for 'zos.dev2' is highlighted with a red box. The table has columns for Name, Tags, Inventory, Status, Description, and a more options menu. The 'Name' column shows 'zos.dev2' with sub-options like 'View Agent' and 'J02Mortgage'.

PART 2 - Create the UrbanCode deployment processes.

On this part you will you create a deployment process for your component and the application process that uses the component process to deploy the component.

Processes are automated tasks that run on agents.

There are three types of processes:



- **Generic processes** run outside the context of components or applications. Not used on this lab.
- **Application processes** run within the context of applications. In many cases, application processes call component processes. For example, an application process can call the component processes that deploy those components.
- **Component processes** run tasks on a single component, such as deploying it, uninstalling it, or running configuration tasks on it.

Task 8 - Create a components process

A component process is a succession of commands that are called steps. Steps can manipulate files, run system commands, set properties, pass information to other steps, and run programs. Steps are provided by automation plug-ins. Processes are designed with the drag-and-drop process editor where you drag plug-in steps onto the design editor and configure them as you go. Several plug-ins come with the product and others are available, which work with many different types of software. In this lab you use z/OS plug-ins. A component can have any number of processes defined for it, but a component must have at least one process.

In this task you create a deployment process for your component. Later, you create an application process that uses the component process to deploy the component.

We are going to create a **component process** for this example. This process will automate deployment of the new version of our **J02Mortgage COBOL** application.

8.1 ► Click on tab **Components**, find the component created earlier (**J02Mortgage**) and click on it

Name	Tags	Latest Import	Latest Version	Template	Description	Created	By
CBSA-backend			version-718	Mainframe Deploy		3/6/2023, 3:55 AM	admin
Genapp-backend			version-1089	Mainframe Deploy		3/3/2023, 4:27 AM	admin
J02Mortgage			20240424	MVSCOMPONENT		4/23/2024, 3:49 PM	empot02
MYCOMP			20230216-125003			2/3/2023, 3:39 PM	admin

8.2 ► Click the **Processes** tab

Process	Description
Deploy (MVSCOMPONENT)	Deploy data sets. Version artifacts are fetched from code station located in the same z/OS.
Deploy - get artifacts from CodeStation (MVSCOMPONENT)	Deploy data sets. Version artifacts are fetched from UrbanCode Deploy server CodeStation.
Deploy - get artifacts using FTP (MVSCOMPONENT)	Deploy data sets. Version artifacts are fetched from code station using FTP
Remove all versions (MVSCOMPONENT)	Remove all versions in an environment including the backup created during version deployment. Use this process when you want to start next round of development with a clean environment. Audit history is available even if versions have been removed from the environment.

You will notice that there is a small collection of processes that was created automatically for your component (**Deploy**, **Remove all versions**, **Uninstall**, and so on). Remember that those processes were

created from a template.

These processes can be used by themselves to perform simple tasks or can be used as starting points for more complex processes.

We will next create our deployment process. We could slightly simplify this task by reusing the existing process called **Deploy**. However, for the purposes of this example, we are going to create a brand new one.

- 8.3 ➡ Click **Create Process** button to get started.

Process	Description	⋮
Deploy (MVS COMPONENT)	Deploy data sets. Version artifacts are fetched from code station located in the same z/OS.	⋮
Deploy - get artifacts from CodeStation (MVS COMPONENT)	Deploy data sets. Version artifacts are fetched from UrbanCode Deploy server CodeStation.	⋮
Deploy - get artifacts using FTP (MVS COMPONENT)	Deploy data sets. Version artifacts are fetched from code station using FTP	⋮
Remove all versions (MVS COMPONENT)	Remove all versions in an environment including the backup created during version deployment. Use this process when you want to start next round of development with a clean environment. Audit history is available even if versions have been removed from the environment.	⋮
Remove redundant versions (MVS COMPONENT)	Remove redundant versions in an environment. Redundant versions are these versions which are replaced completely by later deployed versions.	⋮

- 8.4 ➡ Enter Process Name (**Deploy JKE to CICS**)

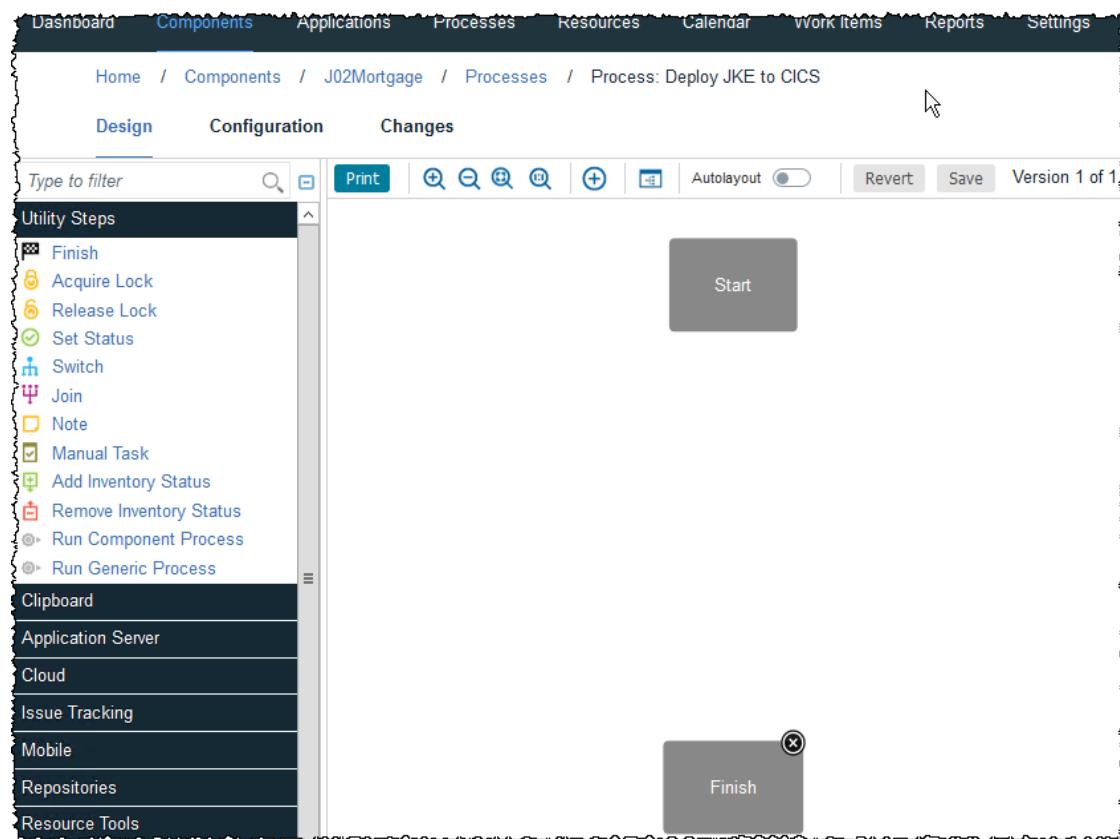
Leave all other options at their default values and click **Save**:

The screenshot shows a 'Create Process' dialog box. At the top left is the title 'Create Process'. On the left side, there are three input fields: 'Name *' with the value 'Deploy JKE to CICS', 'Description' (empty), and 'Process Type *' with the value 'Deployment'. Below these is a 'Inventory Status *' dropdown set to 'Active'. At the bottom right are two buttons: 'Cancel' and 'Save', with a red arrow pointing to the 'Save' button.

8.5 The **Process Editor** will open.

Here, you can organize the steps of a process, specify their properties, and connect them to each other. As usual, you can refer to UCD documentation for detailed information on editing processes. If you have internet access you can go to the link below for more details::

https://www.ibm.com/support/knowledgecenter/SS4GSP_7.0.2/com.ibm.udeploy.doc/topics/comp_workflow.html

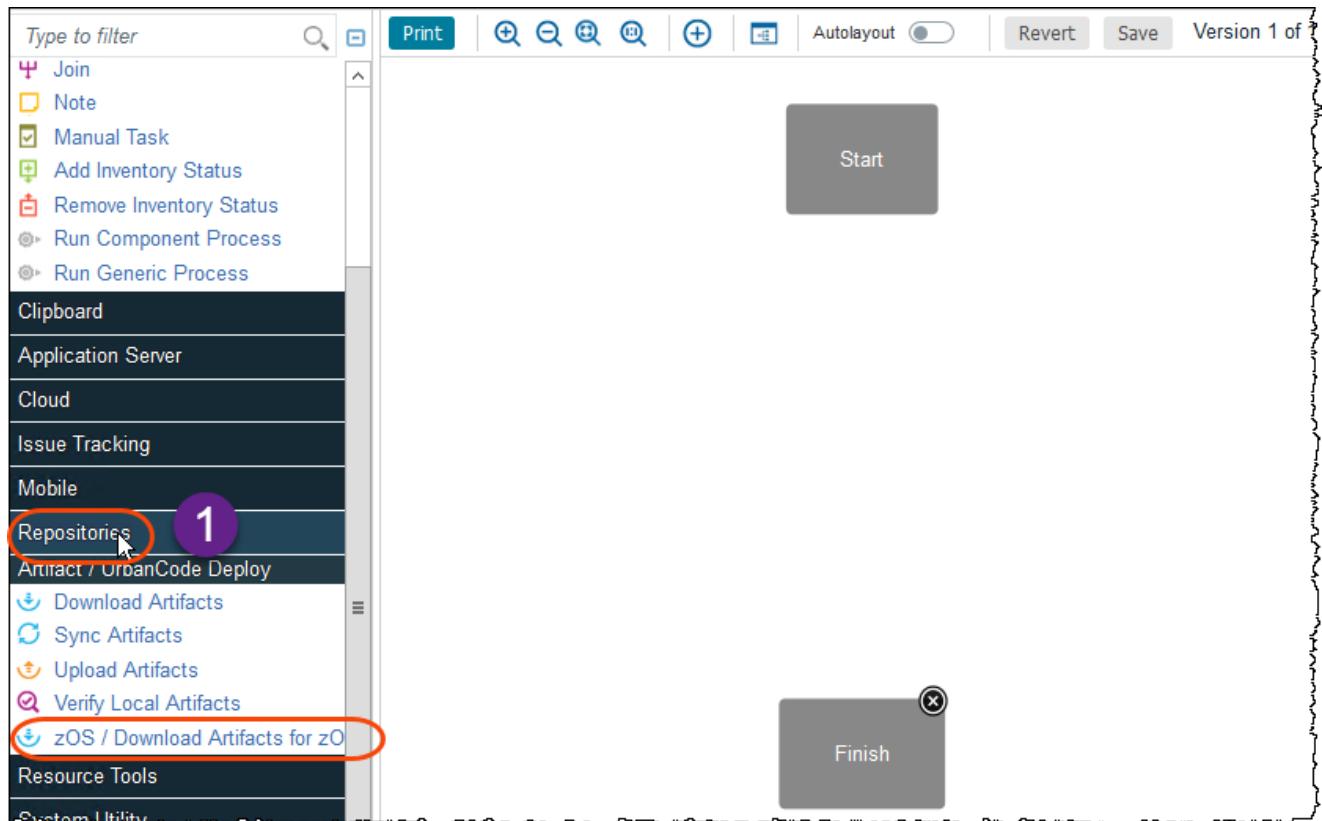


8.6 We are going to create a simple process for updating an existing CICS application.

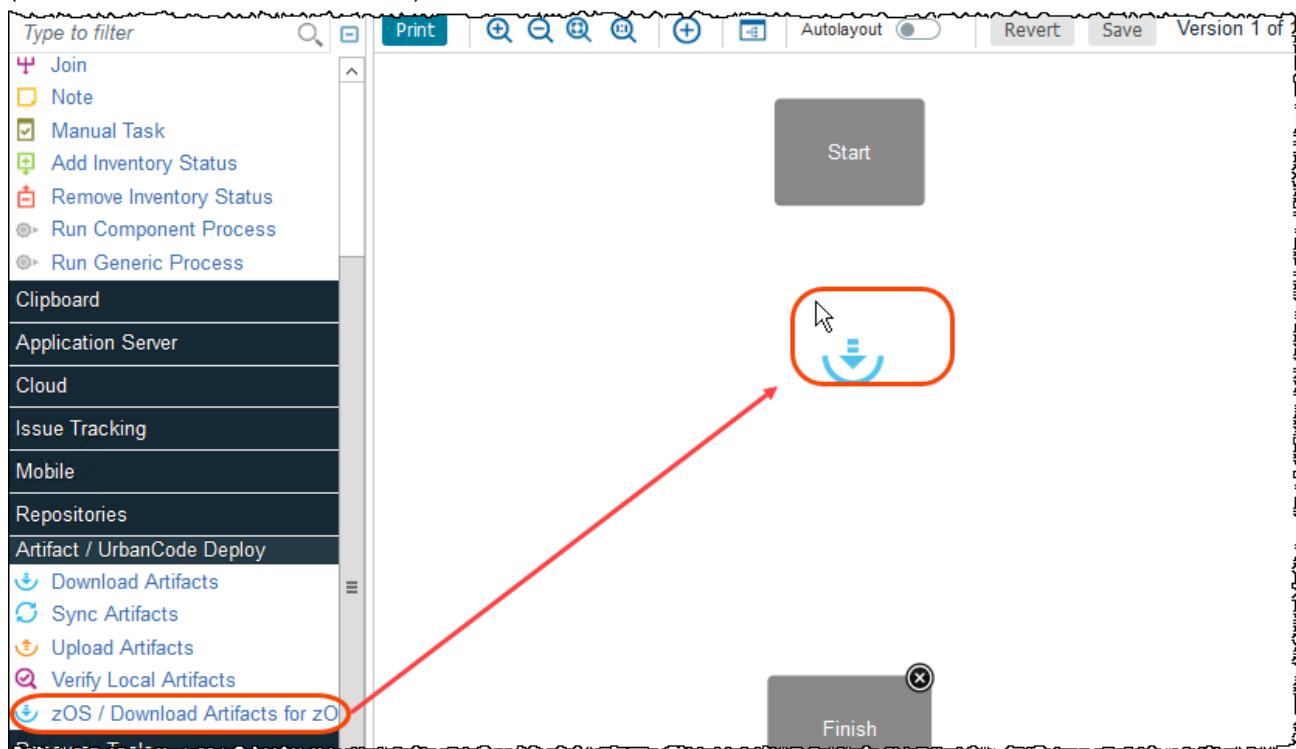
► On the left-hand side of the editor you will find an extensive **Palette** of process steps.

► Scroll down and expand **Repositories**.

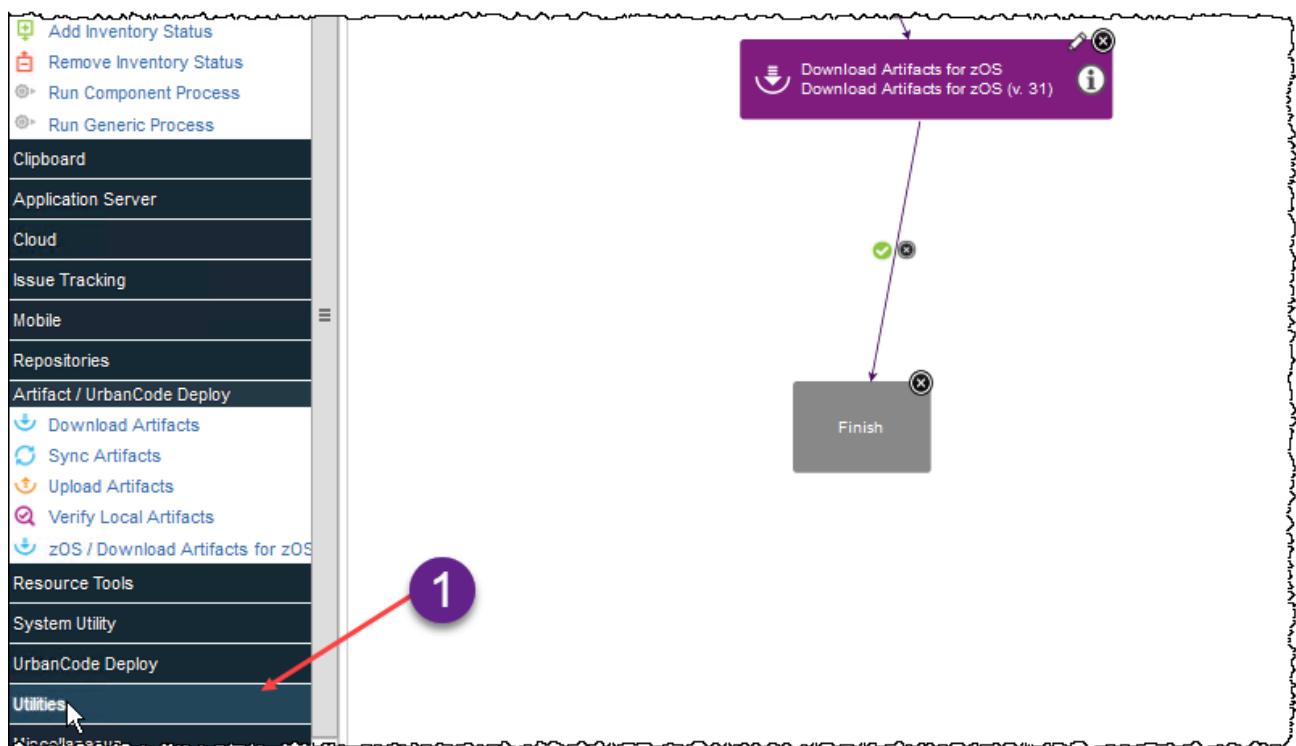
First you are going to use the step **zOS /Download Artifacts for zOS**:



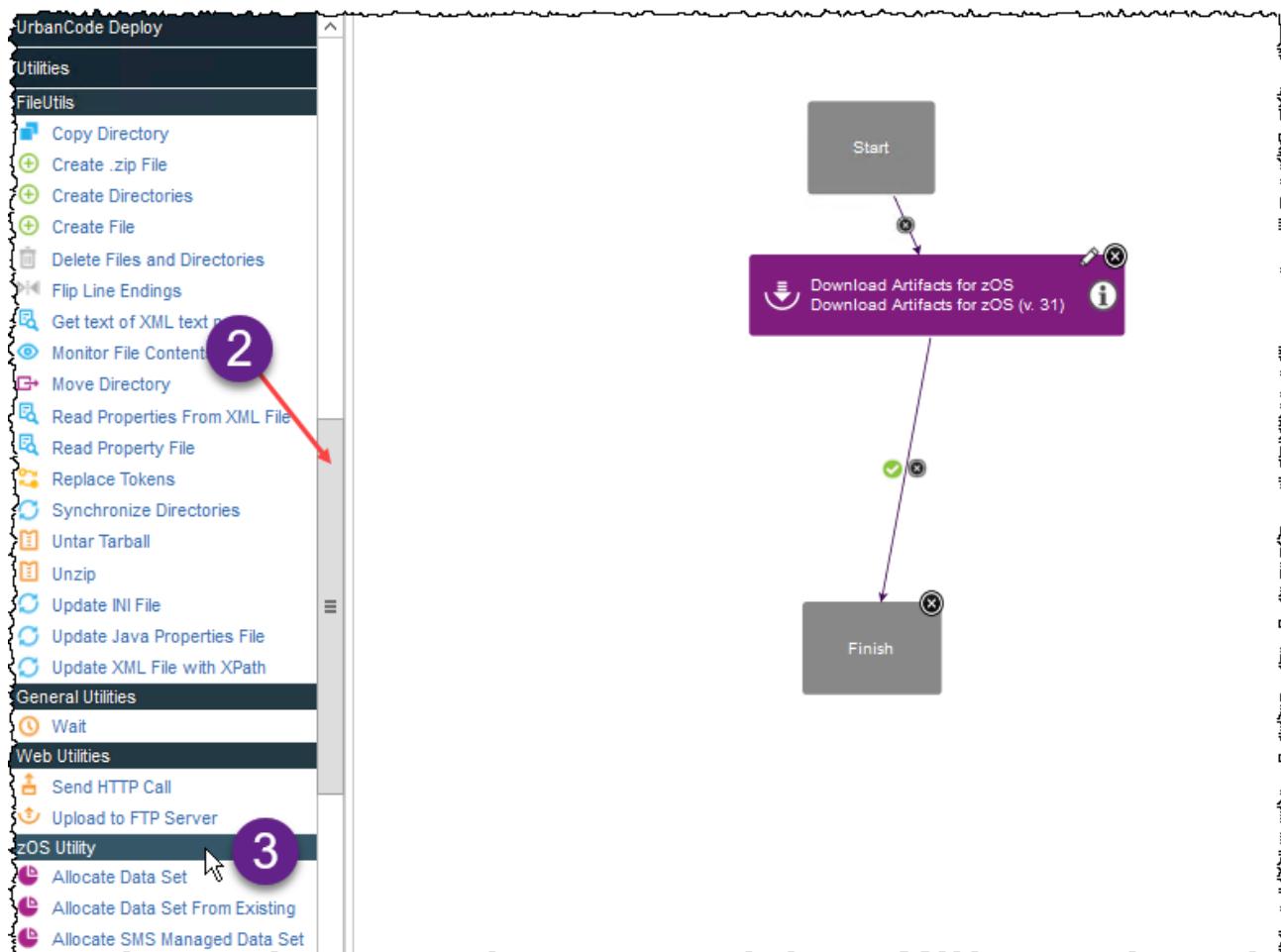
- 8.7 ► Drag and drop the first step, **zOS /Download Artifacts for zOS**, onto the design space (somewhere under the **Start** block).



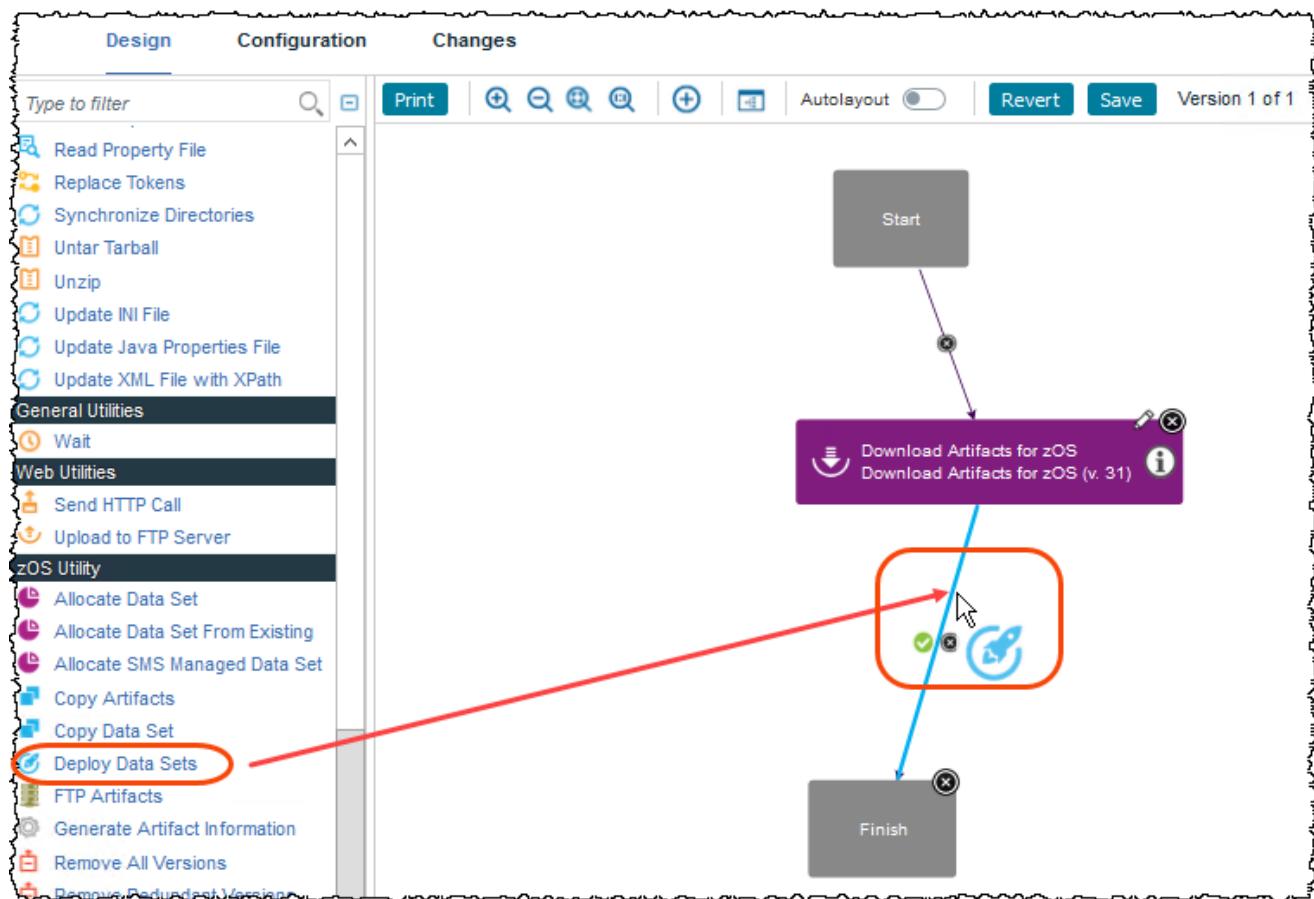
- 8.8 ► 1 Using the Design palette on left expand **Utilities**, clicking on it



- ▶ 2 Scroll down and 3 expand zOS Utility: The Deploy Datasets step will also be needed to load component artifacts from the z/OS repository and to bring them into z/OS work area for later deploy.



8.9 ► Scroll down and drag the **Deploy Data Sets** step onto the line that connects to the *Finish* as below



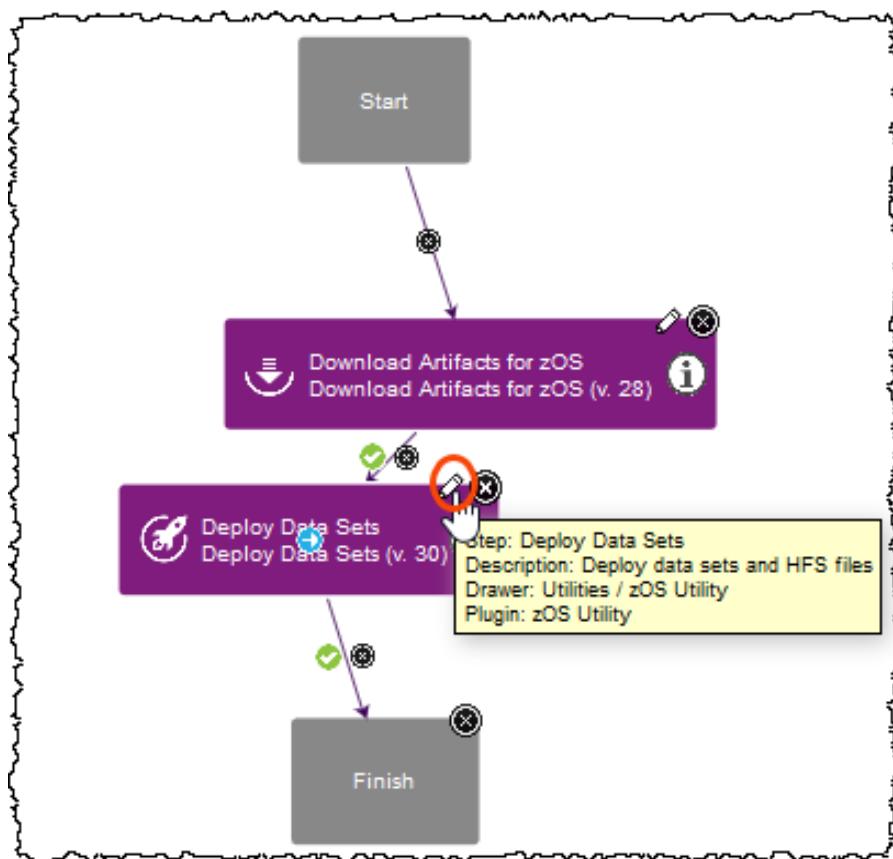
Properties

Properties can be set in UCD for many different things, including components, environments, processes, and applications. You can also set global properties for the system.

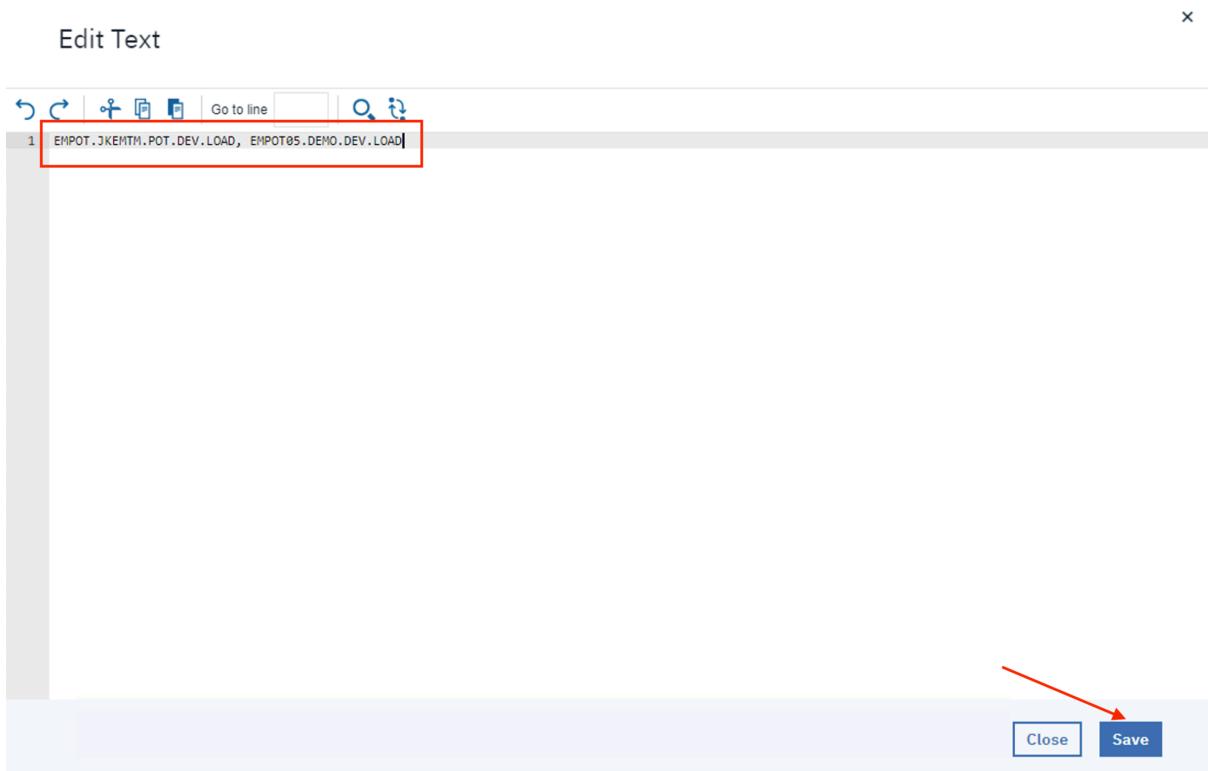
You can refer to a property by scope:
 `${p:scope/propertyName}`
 or without scope:
 `${p:propertyName}`

For more details, refer to the full documentation on Properties:
http://www-01.ibm.com/support/knowledgecenter/SS4GSP_6.1.1/com.ibm.udeploy.reference.doc/topics/ud_properties.html

- 8.10 ► Click on the pencil icon  on Deploy Data Sets to change the UCD properties for this step.



- 8.11 ► Enter the value for the *Data Set Mapping* property:
`EMPOT.JKEMTM.POT.DEV.LOAD, EMPOT05.DEMO.DEV.LOAD`



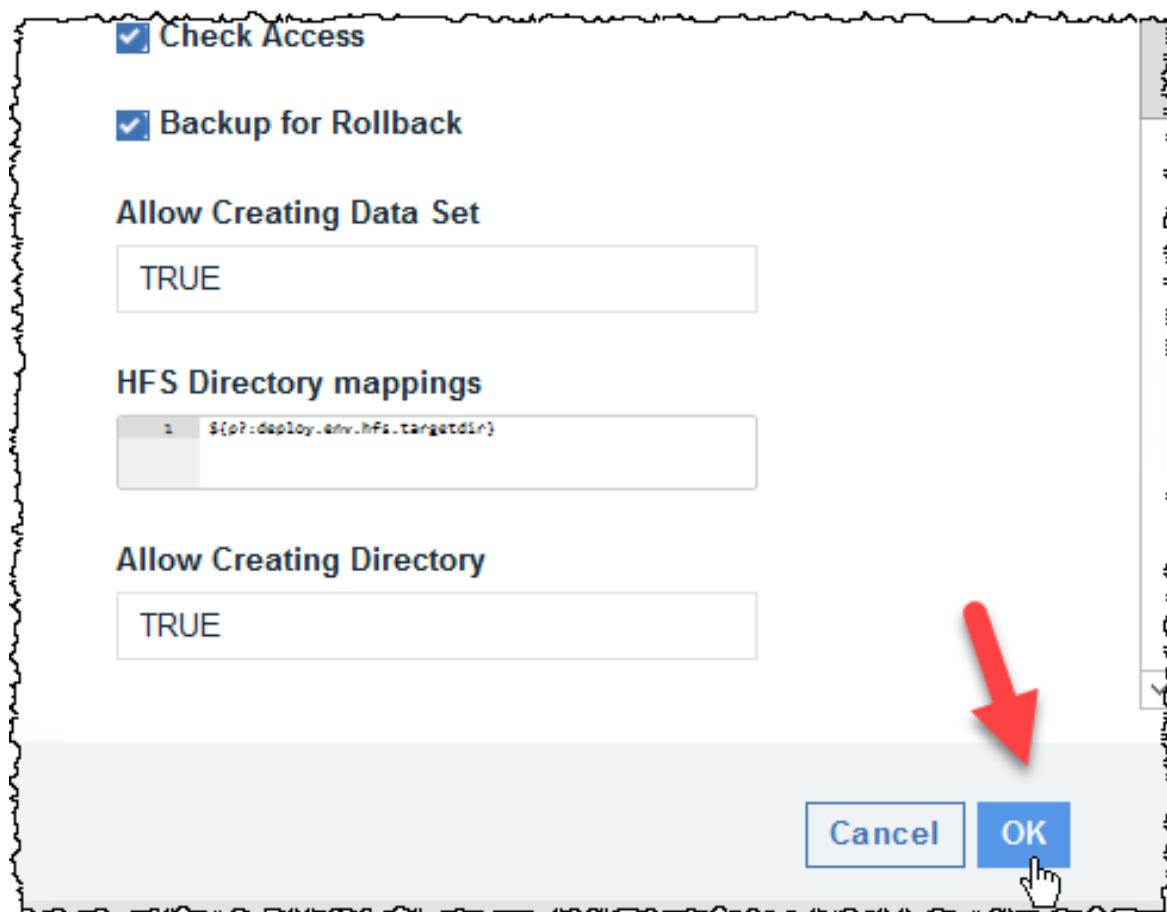
Data Set Mapping specifies which PDS members packaged with the component to deploy, and where to deploy them.

A mapping rule should follow format "From PDS, To PDS":

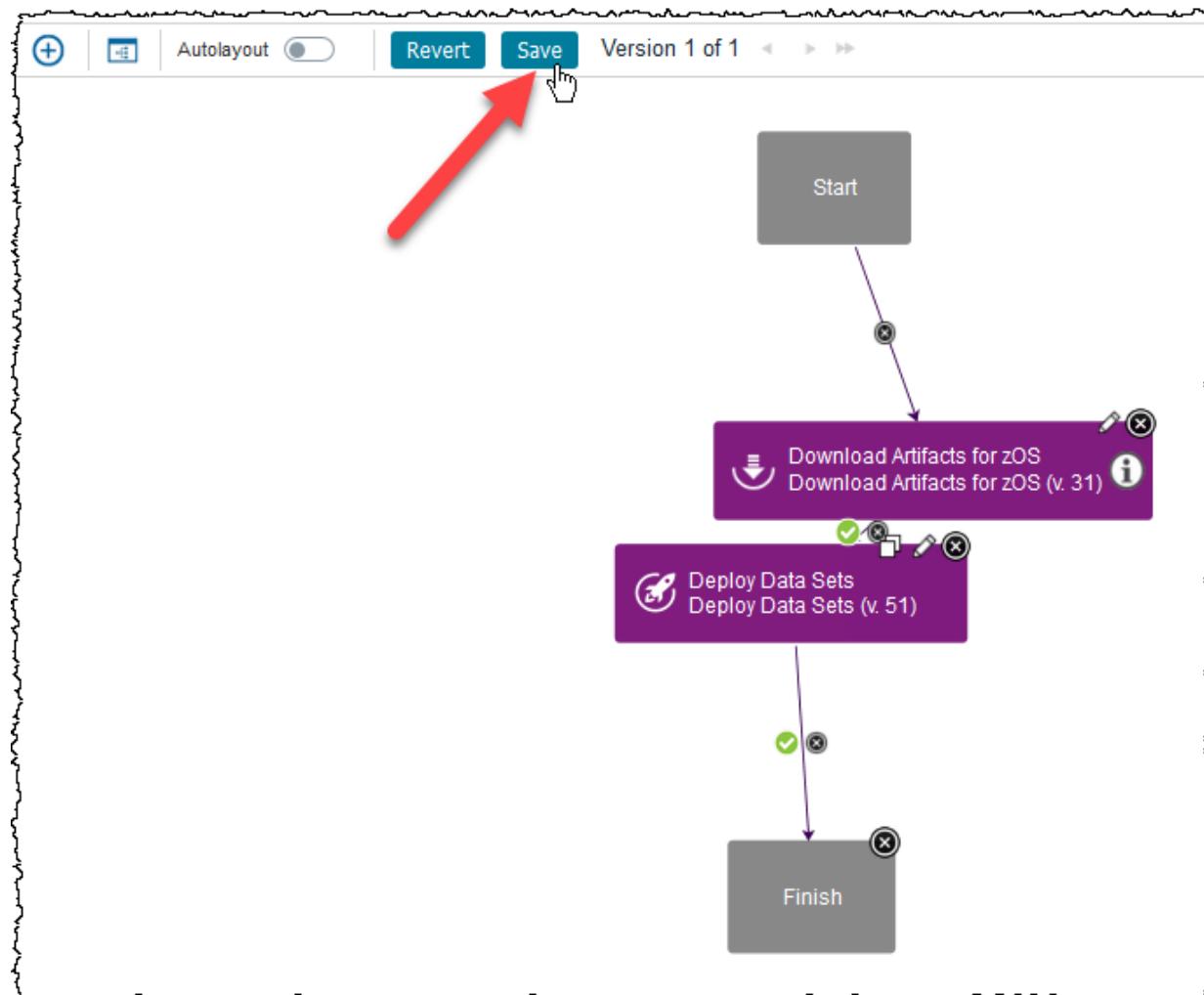
We could have that as variable but we will fix here to save time.

The image contains two screenshots. The top screenshot is titled 'Edit Properties for Deploy Data Sets'. It has fields for 'Name' (set to 'Deploy Data Sets') and 'Data Set Mapping'. The 'Data Set Mapping' field contains a table with one row: '1 \${p7:deploy.env.pds.mapping}' (with a question mark icon). A red circle highlights this row. The bottom screenshot is titled 'Edit Text' and shows the same mapping entry ('1 \${p7:deploy.env.pds.mapping}') in a text input field. A red arrow points to the 'Save' button at the bottom of this dialog. Both dialogs have a toolbar at the top with standard edit and search functions.

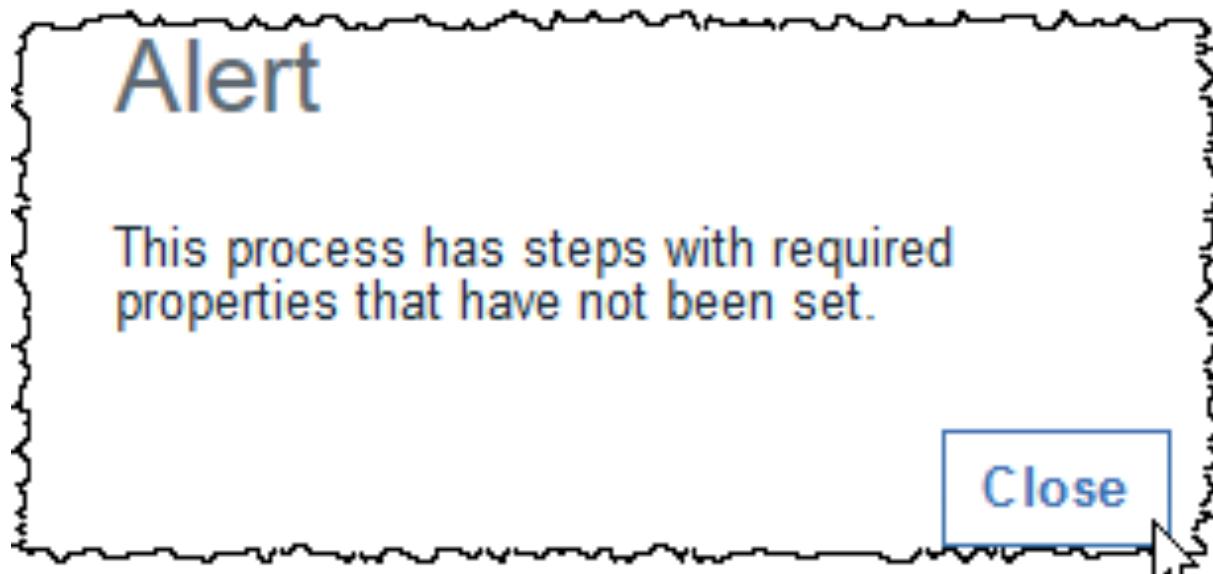
8.12 ► Click **OK** to save the properties updated



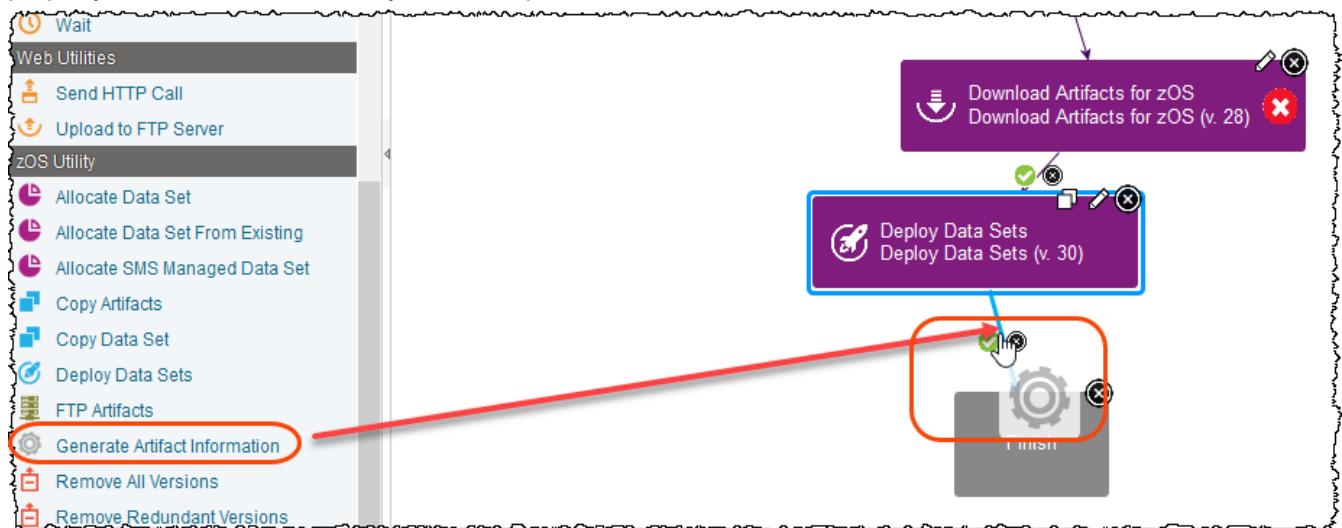
8.13 ► Click the **Save** button to save what you had done so far.



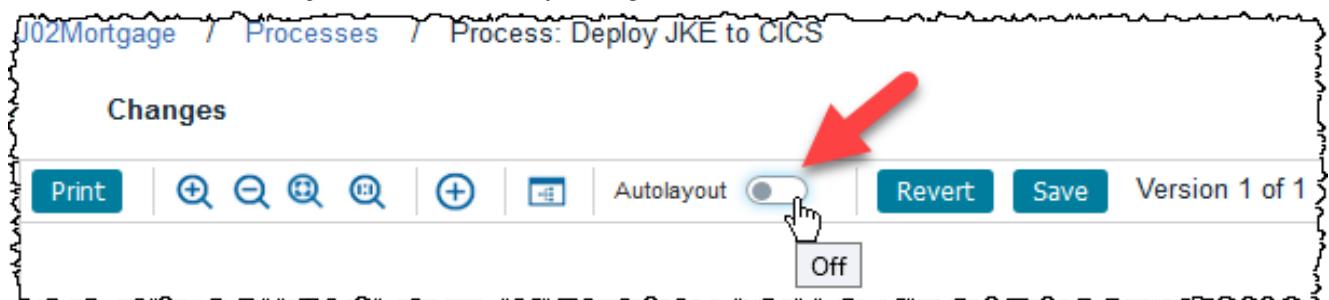
8.14 ► If there is a warning about missing property ignore it, clicking **Close**. You will fix that later.



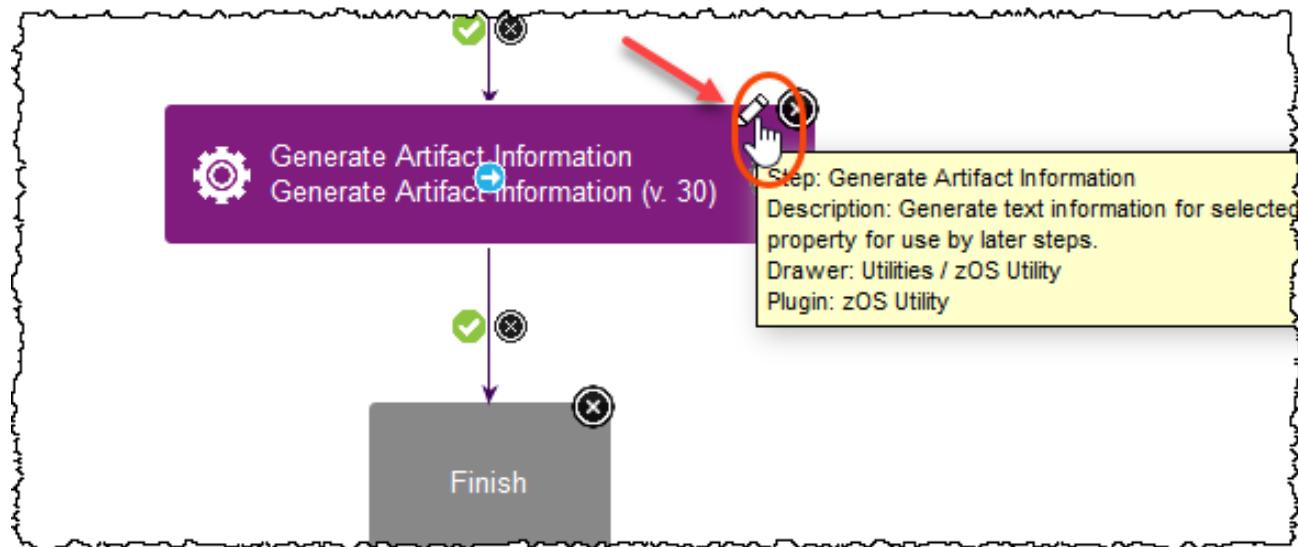
8.15 ► Scroll down on left pallet and drag and drop **Generate Artifact Information** on the line before Finish. You will need to specify a list of the load modules that CICS needs to do the *Newcopy*. This step generate text information for selected version artifacts. Information is put to output property 'text' to be consumed by a later step



8.16 ► Click on **Autolayout** to have the layout organized



- 8.17 ►| Click on the pencil icon  on **Generate Artifact Information** to change the UCD properties for this step.



- 8.18 ►| ① Change from *Generate Artifact Information* to **Generate Program List** (must respect upper/lower case and spaces)

- | ② Add **CICS_LOAD** to *Deploy Type Filter*
- | ③ Scroll down to add another property

Edit Properties for Generate Artifact Information

Name *

Generate Program List

1

For Each *

PDS Member

Order By *

ASC Order

3

Container Name Filter

Target Data Set Name Filter

Resource Name Filter

Deploy Type Filter

CICS_LOAD

2

Custom Properties Filter

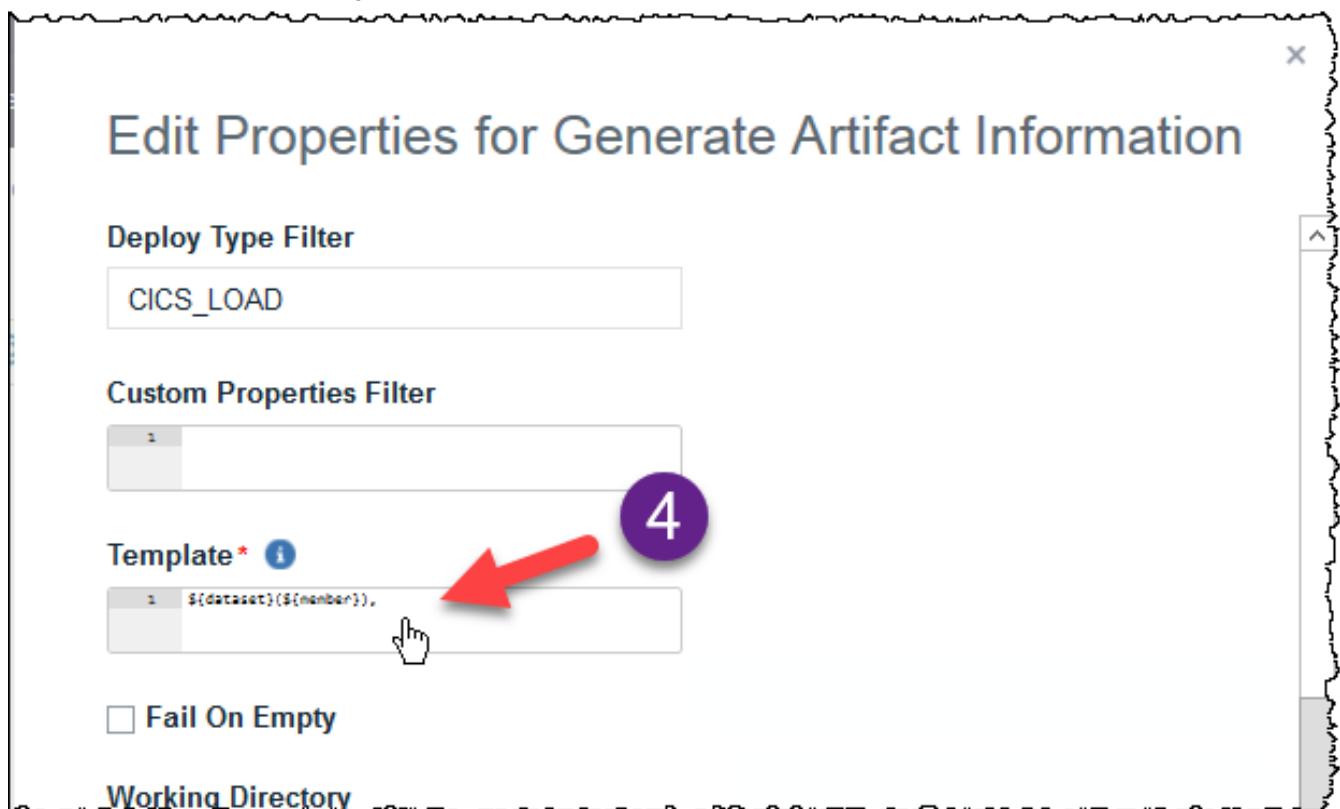
1

Template *

1 \$number,

Fail On Empty

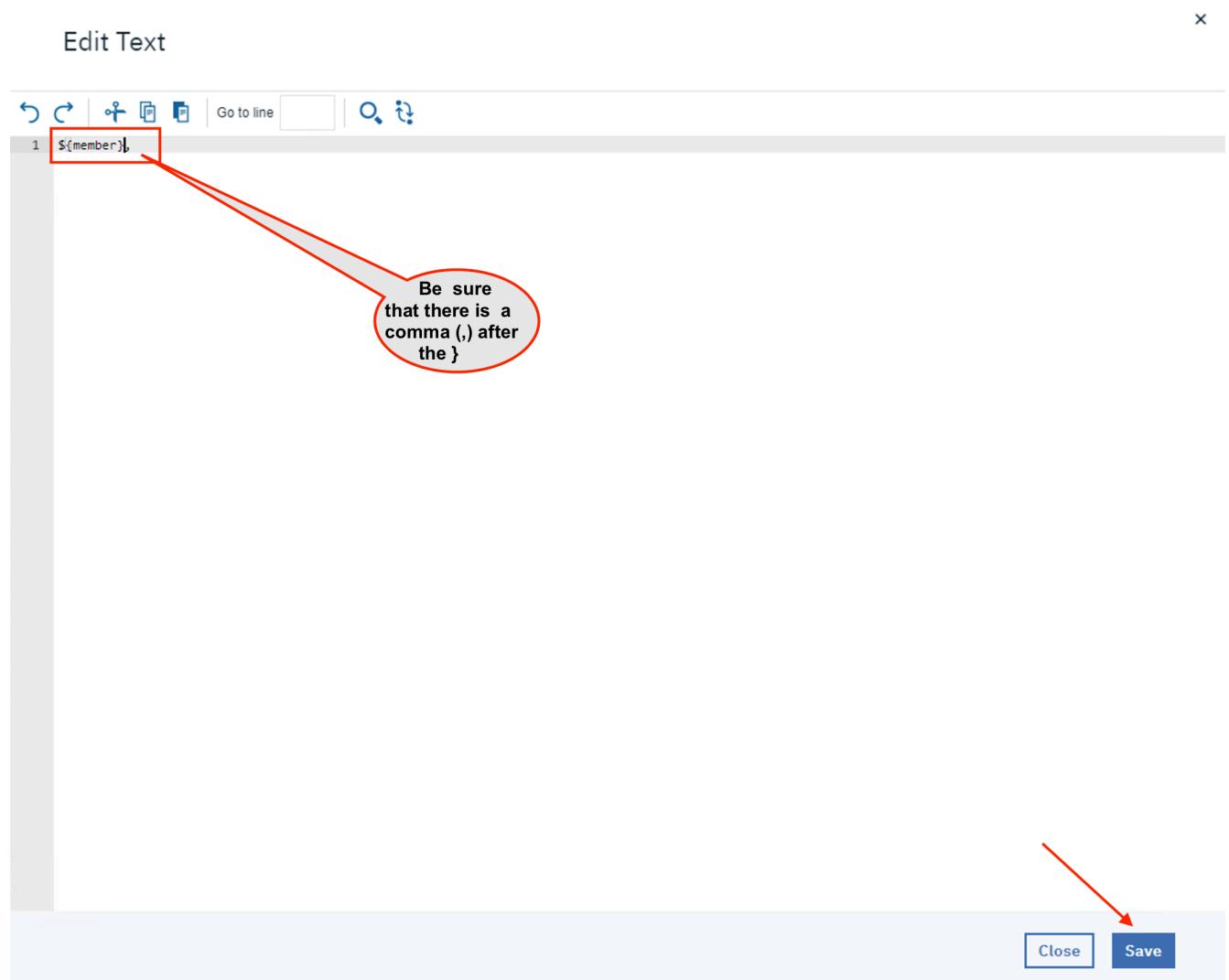
8.19 ► 4 Click on Template



► Change the value from "

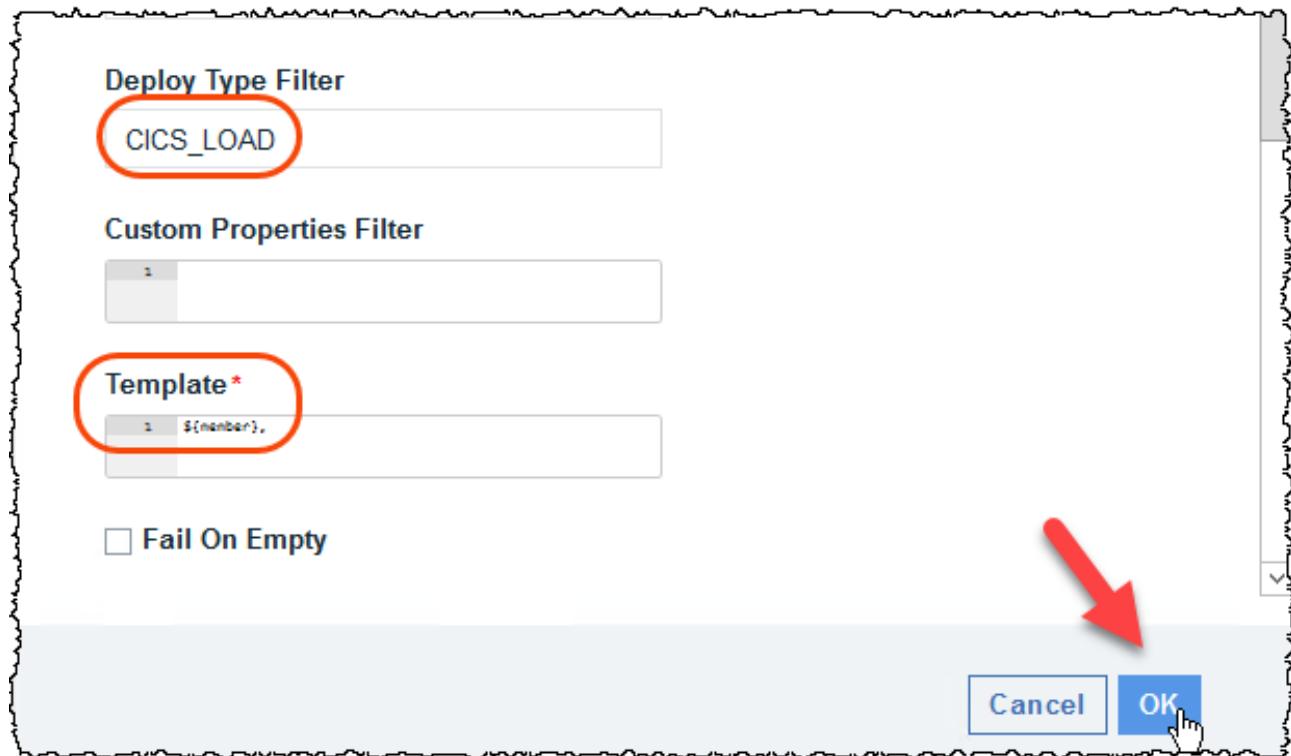
\${dataset}(\${member})," to **\${member},**

IMPORTANT ➔ Be sure that you add a , (comma) after the \${member},



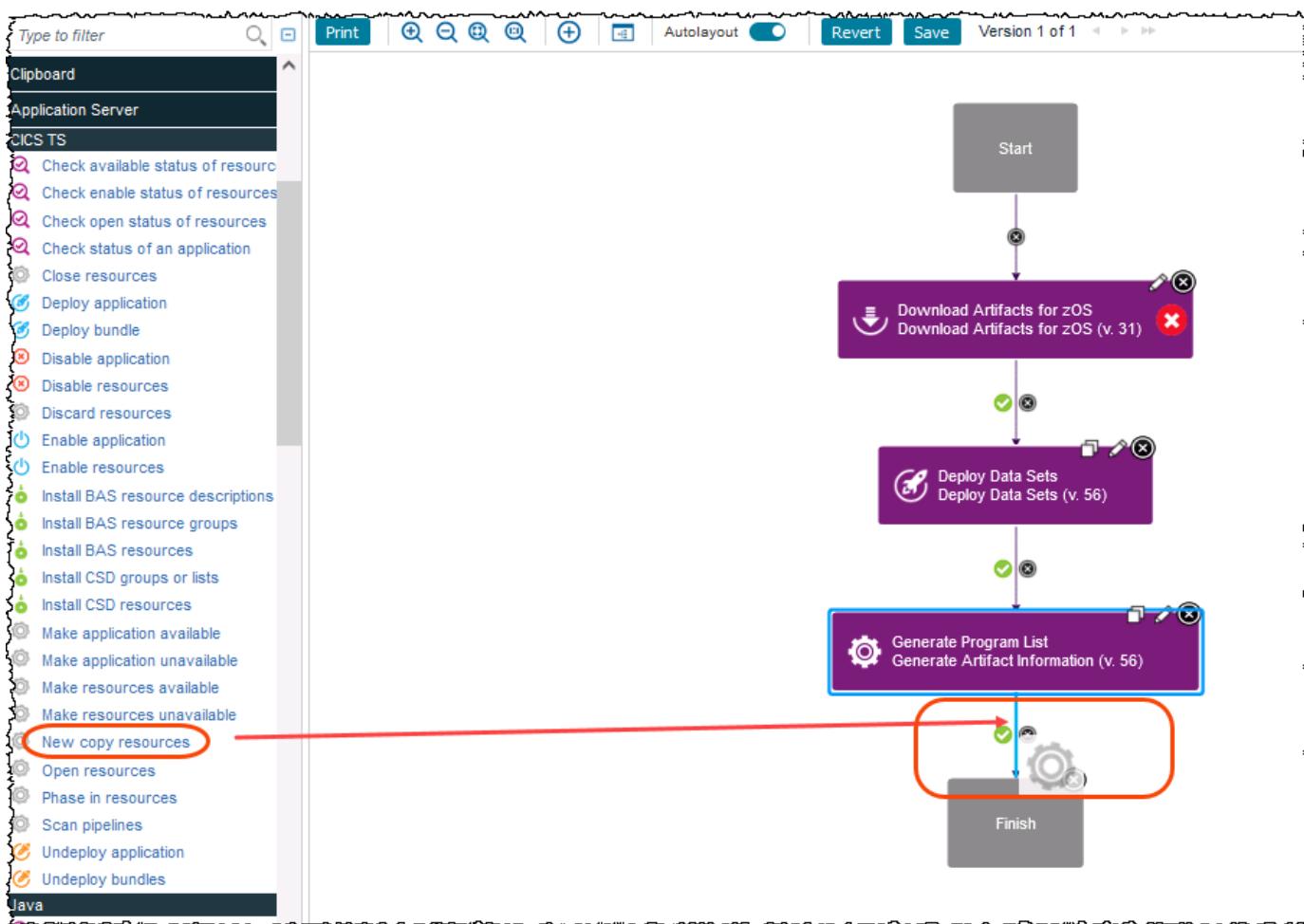
▶| Click **Save** to save the modified value

8.20 ▶| Click **OK** to close the dialog

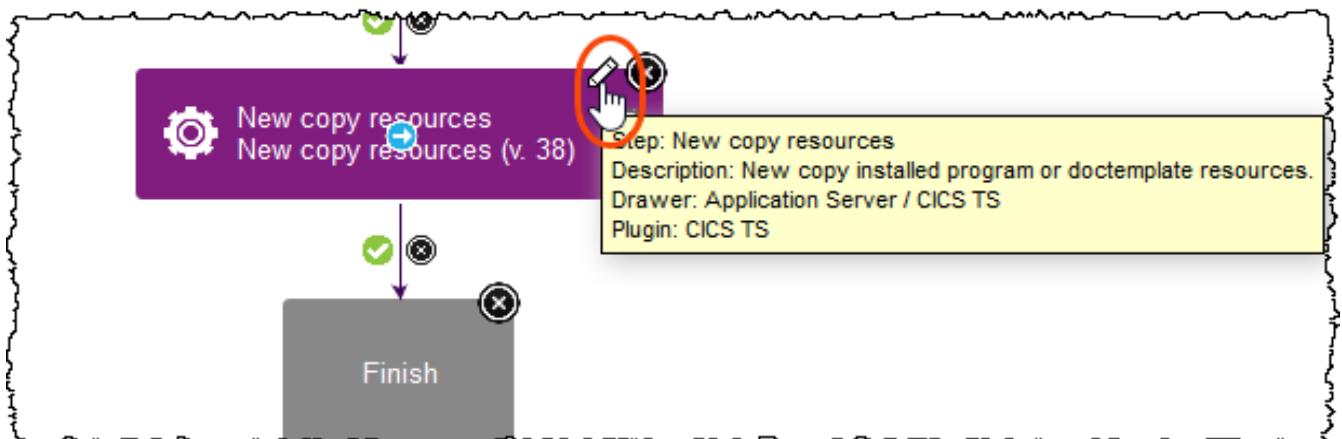


8.21 ► On the Palette (left side) expand **Application Server** (scroll up) and **CICS TS**.

► Drag and drop **New copy resources** to the line that connects to the *Finish* box..



8.22 ► Click on the pencil icon on **New copy resources** to change the UCD properties for this step.



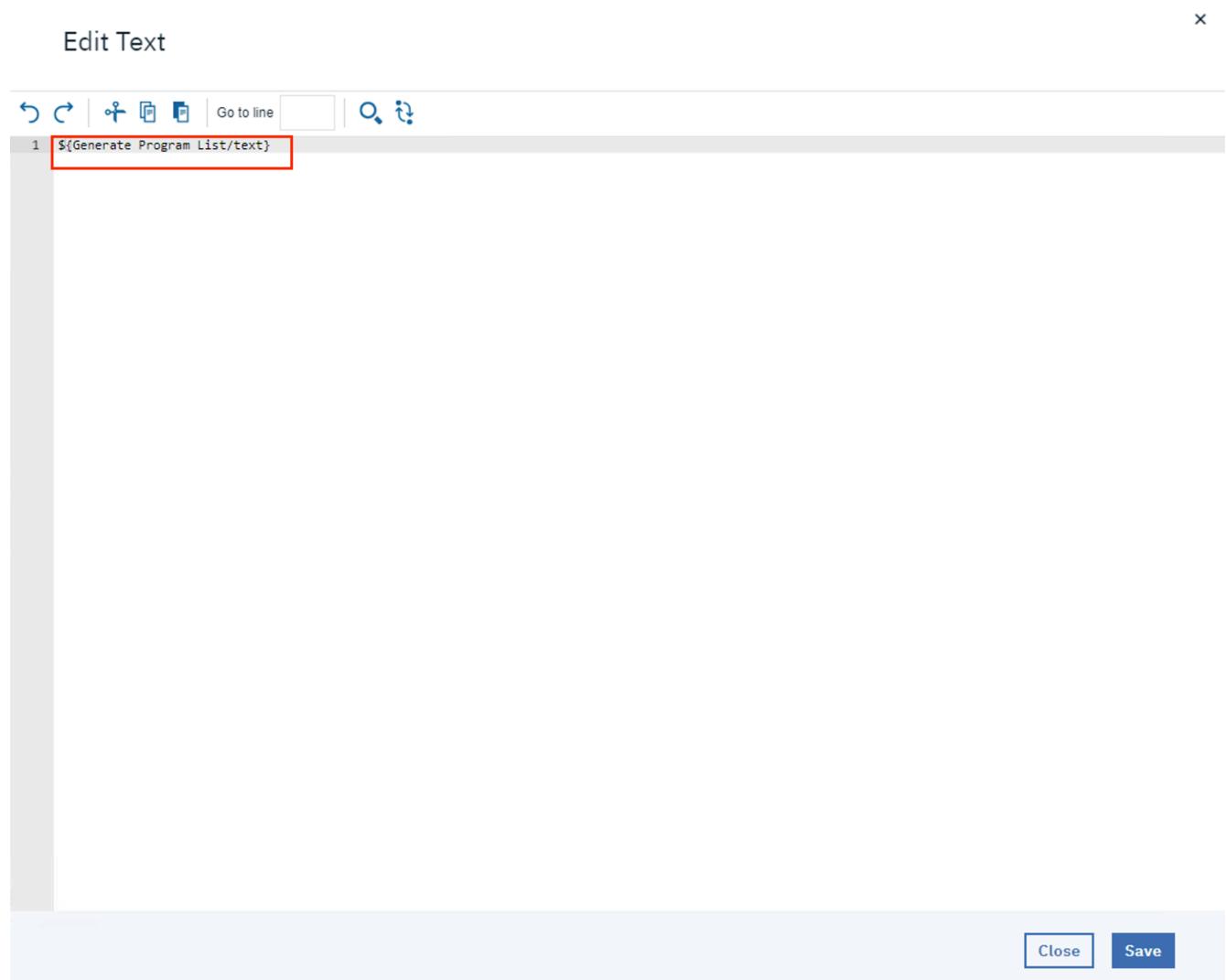
8.23 You need to specify a list of the load modules that CICS needs to do the *Newcopy*.

► Click on **Resource Name List**

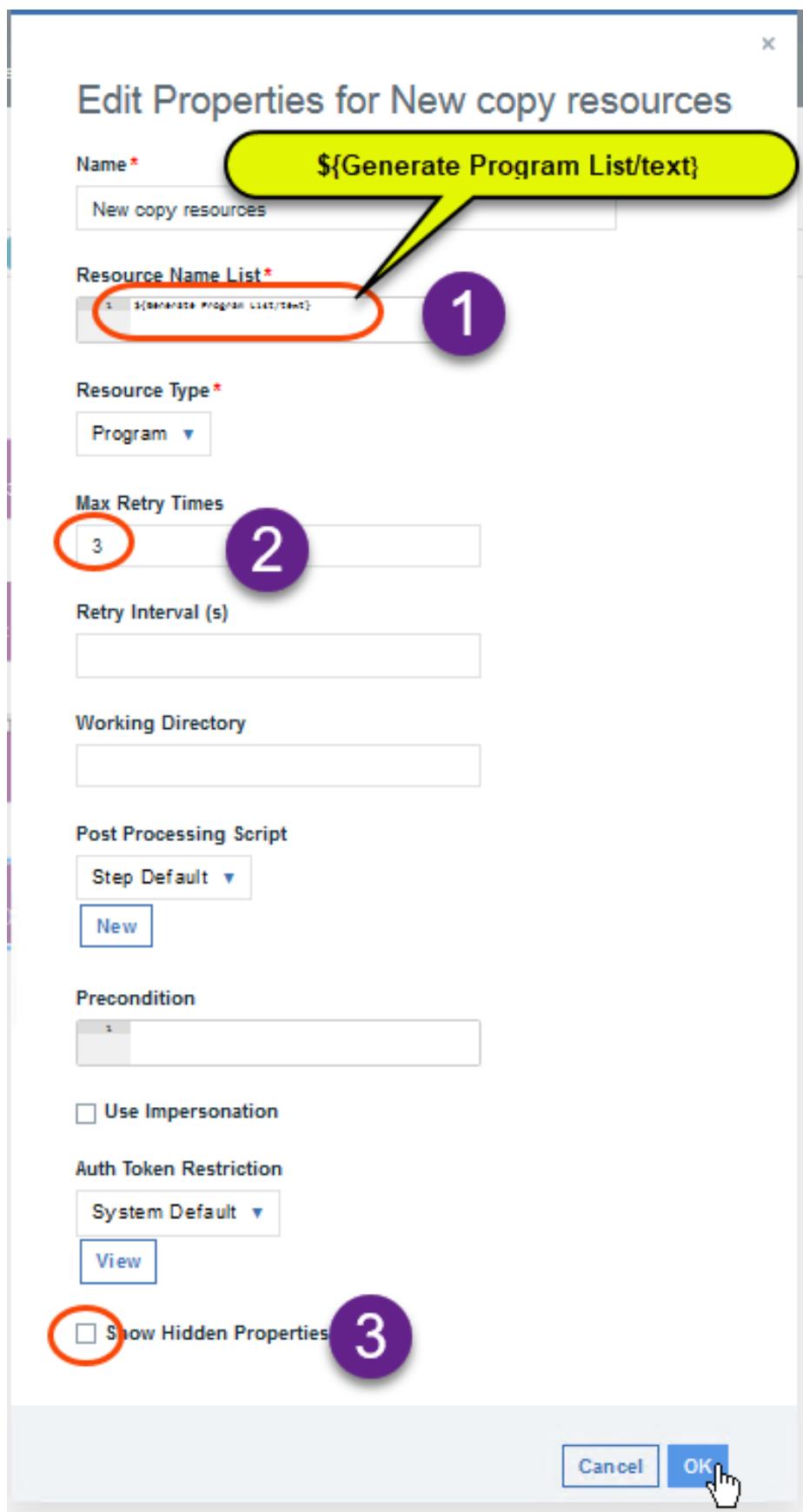
The screenshot shows a dialog box titled "Edit Properties for New copy resources". It contains the following fields:

- Name *: A text input field containing "New copy resources".
- Resource Name List *: A dropdown menu showing "1" and a hand cursor icon, which is circled in red.
- Resource Type *: A dropdown menu showing "Program".
- Max Retry Times: A text input field.

► Type `${Generate Program List/text}` (This is the step name that you created on 8.18) and click **Save**



- ▶▶ Type 3 for Max Retry Times
- ▶▶ Scroll down and click **Show Hidden properties**.



8.24 Scroll down to see those properties that need to be filled. Later we will do that at the **Test** level

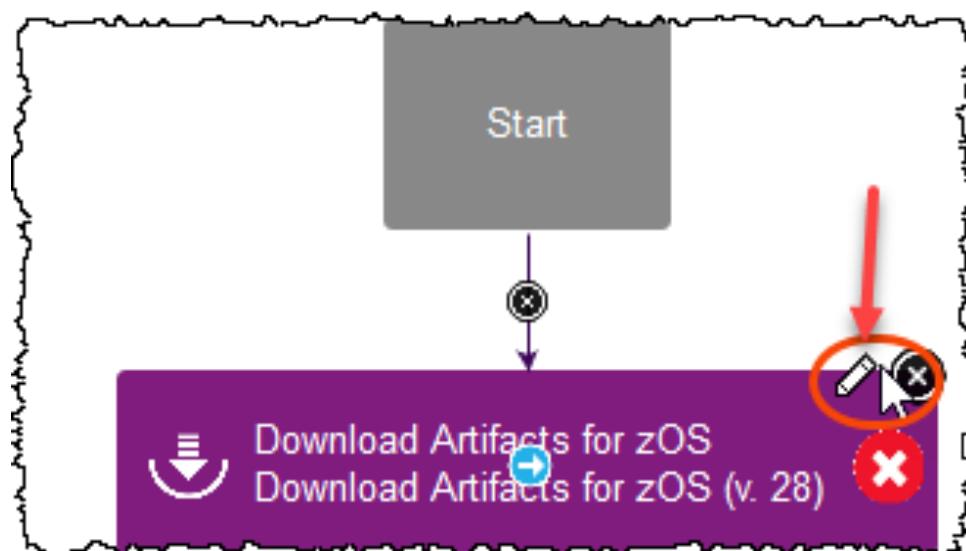
Edit Properties

Show Hidden Properties	<input checked="" type="checkbox"/>
Host *	<input type="text" value="\${p:cics.host}"/>
Port *	<input type="text" value="\${p:cics.cmciport}"/>
CICSPlex	<input type="text" value="\${p?:cics.cicsplex}"/>
Scope	<input type="text" value="\${p?:cics.scope}"/>
Username	<input type="text" value="\${p?:cics.username}"/>
Password	<input type="password" value="*****"/>
Enable SSL	<input type="text" value="\${p?:cics.ssl}"/>
Keystore Location	<input type="text" value="\${p?:cics.kslocation}"/>
Keystore Type	<input type="text" value="\${p?:cics.kstype}"/>
Keystore Password	<input type="password" value="*****"/>
Truststore Location	<input type="text" value="\${p?:cics.tslocation}"/>
Truststore Type	<input type="text" value="\${p?:cics.tstype}"/>
Truststore Password	<input type="password" value="*****"/>

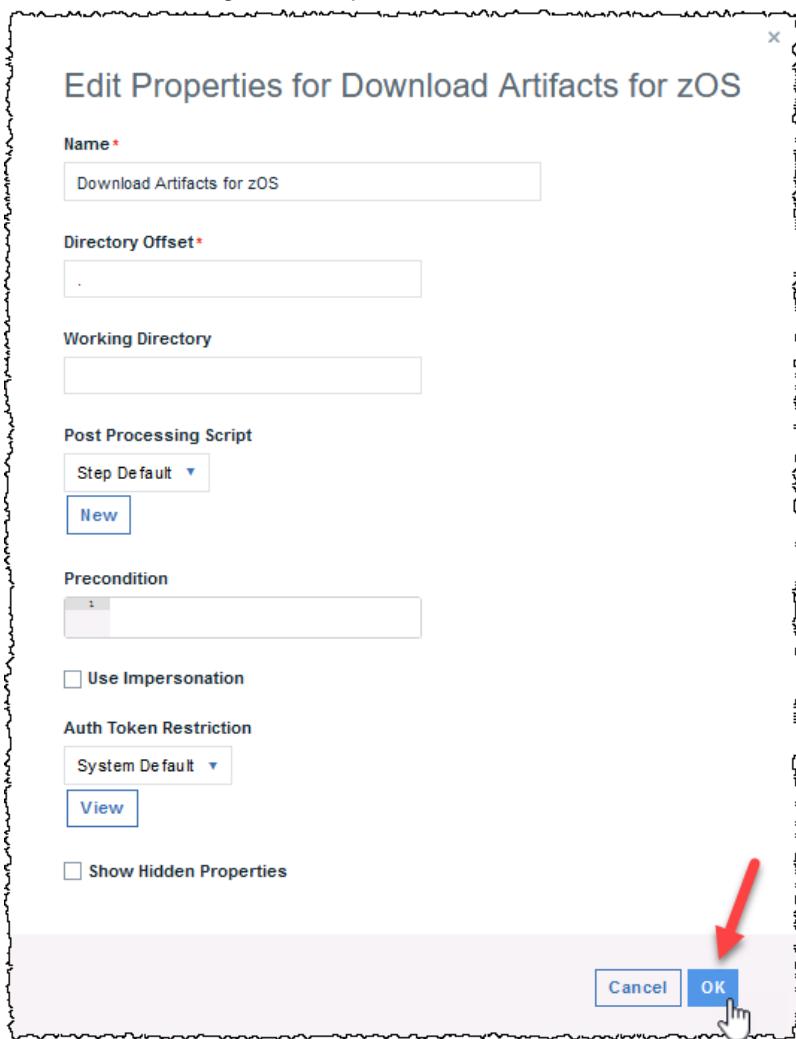
▶ Click **OK** to save it

8.25 You still need to fix a warning on first step (see the icon).

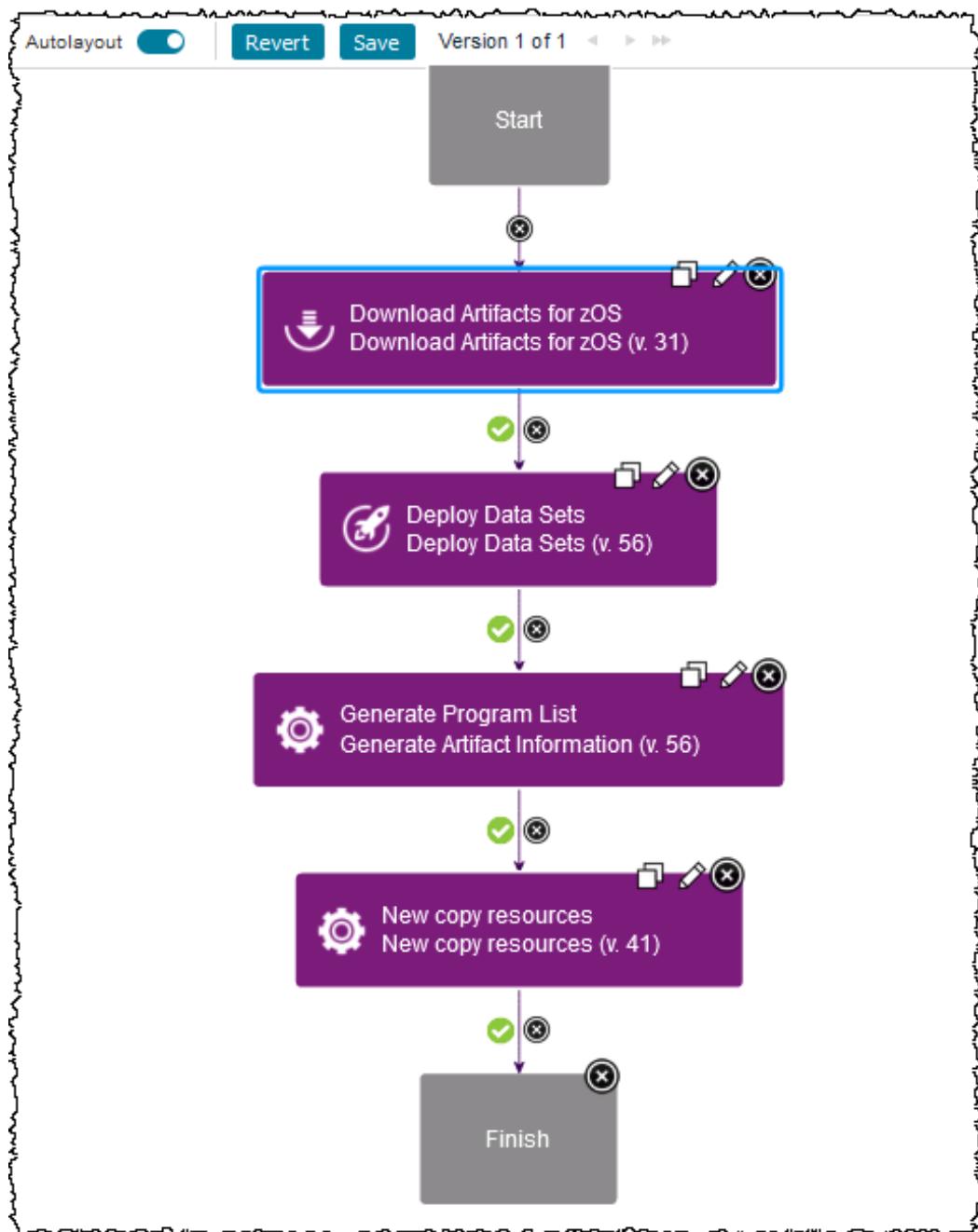
▶ Click on the **pencil** icon on **Download Artifacts for zOS** to possibly change the properties.



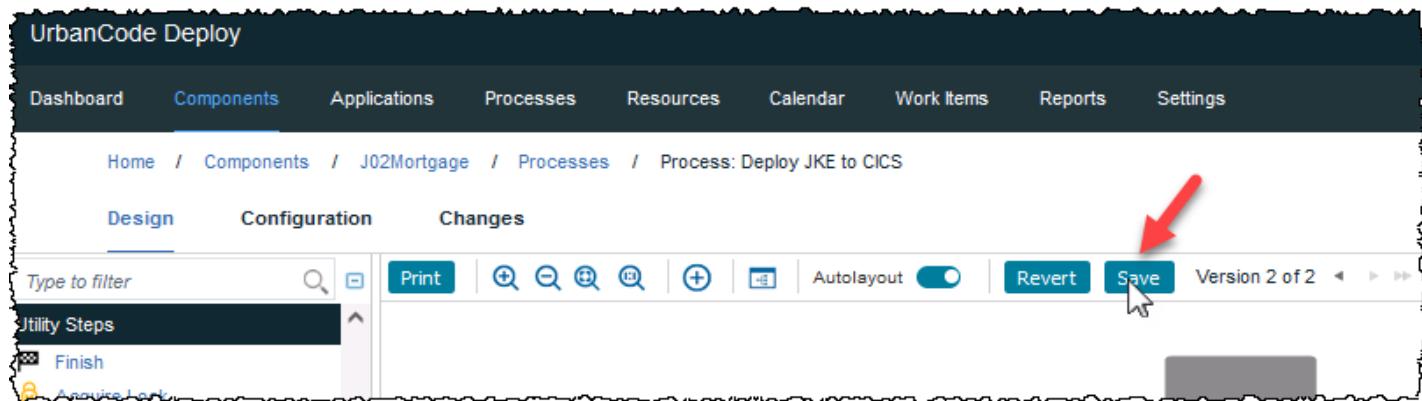
8.26 ► No changes are required here and click **OK**



8.27 The final result should look similar to this:



8.28 ►► Finally, click the **Save** button located in the upper left portion of the process editor to save the process:



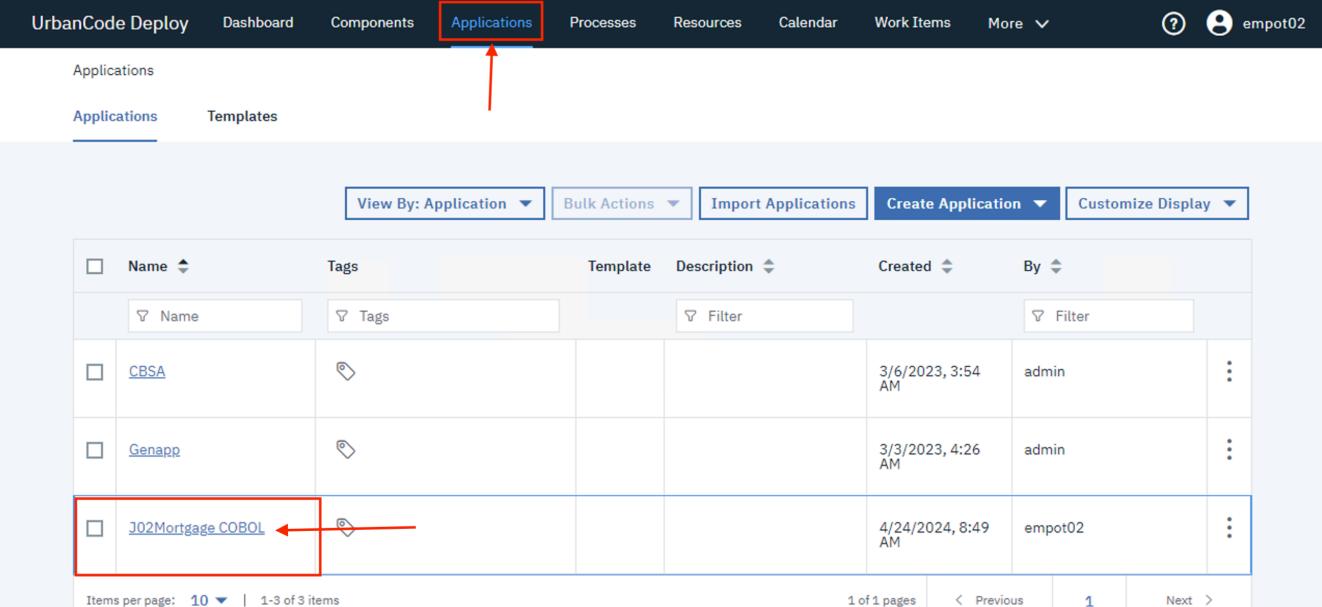
A message will indicate that the process is saved.

Task 9 - Create an application process

Application processes direct underlying component processes and orchestrate multi-component deployments. An application process, like a component process, consists of steps that are configured with the process editor.

In this task, you create an application process that installs the UCD application defined .

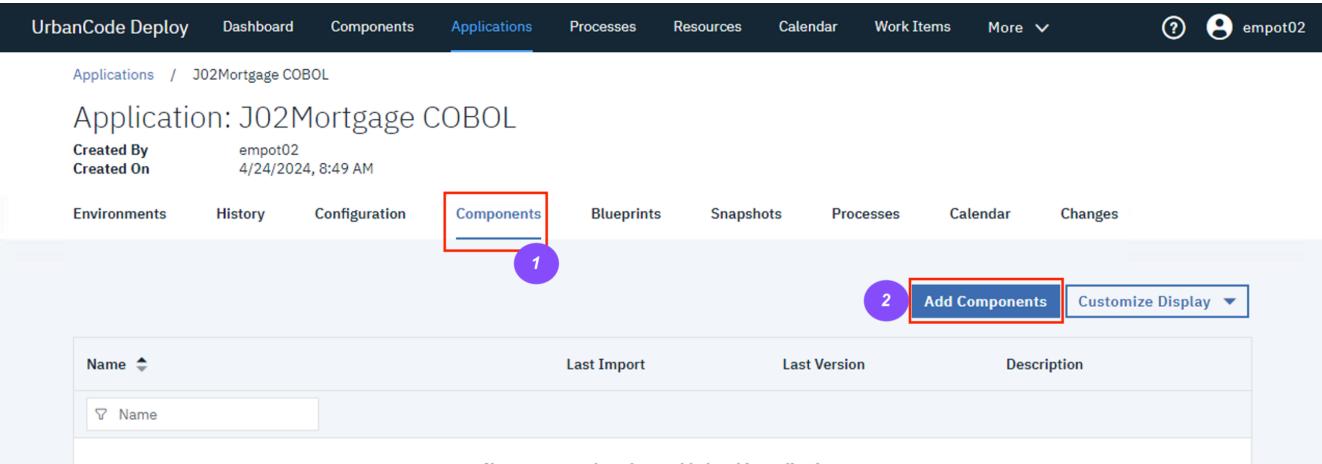
- 9.1 ► Go to **Applications** page, select your application (**J02 Mortgage - COBOL**),



The screenshot shows the UrbanCode Deploy interface with the Applications tab selected. A red box highlights the 'Applications' tab in the top navigation bar. A red arrow points from the 'Applications' tab to the 'J02Mortgage COBOL' row in the list below. The list includes columns for Name, Tags, Template, Description, Created, and By. The 'J02Mortgage COBOL' row is selected, indicated by a red border around its entire row. The row contains the name 'J02Mortgage COBOL', a tag icon, and other details.

Name	Tags	Template	Description	Created	By
CBSA				3/6/2023, 3:54 AM	admin
Genapp				3/3/2023, 4:26 AM	admin
J02Mortgage COBOL				4/24/2024, 8:49 AM	empot02

- 9.2 ► Click **Components** tab and click **Add Component**:



The screenshot shows the UrbanCode Deploy interface for the 'J02Mortgage COBOL' application. The 'Components' tab is selected, indicated by a red box and a purple circle with the number '1'. A red box highlights the 'Add Components' button in the top right corner, which is also circled in purple with the number '2'. The page displays basic application metadata and a table for managing components. A message at the bottom states 'No components have been added to this application.'

Name	Last Import	Last Version	Description
<input type="text"/>			

9.3 ➔ Select your UCD component created previously (**J02Mortgage**) and click **Save**:

9.4 ➔ Click the **Processes** tab and click **Create Process**:

9.5 ➔ Give a name to your process, like **Deploy J02 to CICS** and click **Save**:

Application: J02Mortgage COBOL

Created By
Created Onempot02
4/24/2024, 8:49 AM

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Drag environments by their names to re-order them. 1 Environment

Compare Environments

Create Environment

Search by Name

or Search by Blueprint

Collapse All

> Test

Snapshot: None

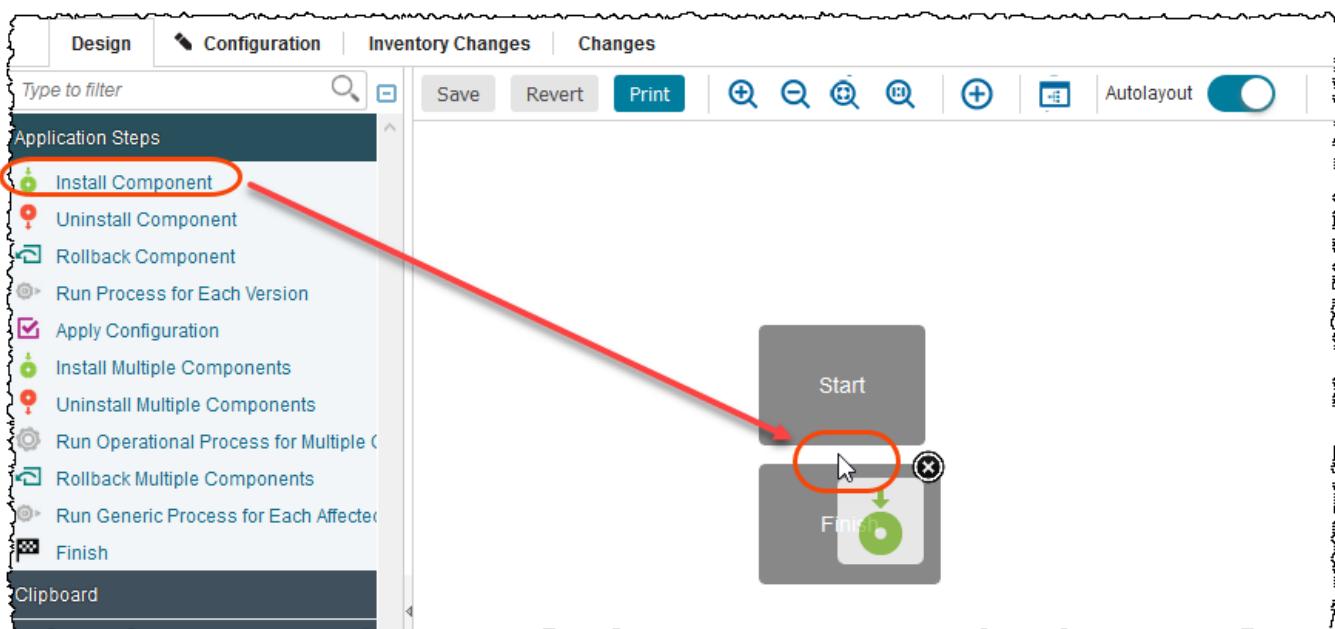
● Inventory: 0 / 0

Request Process

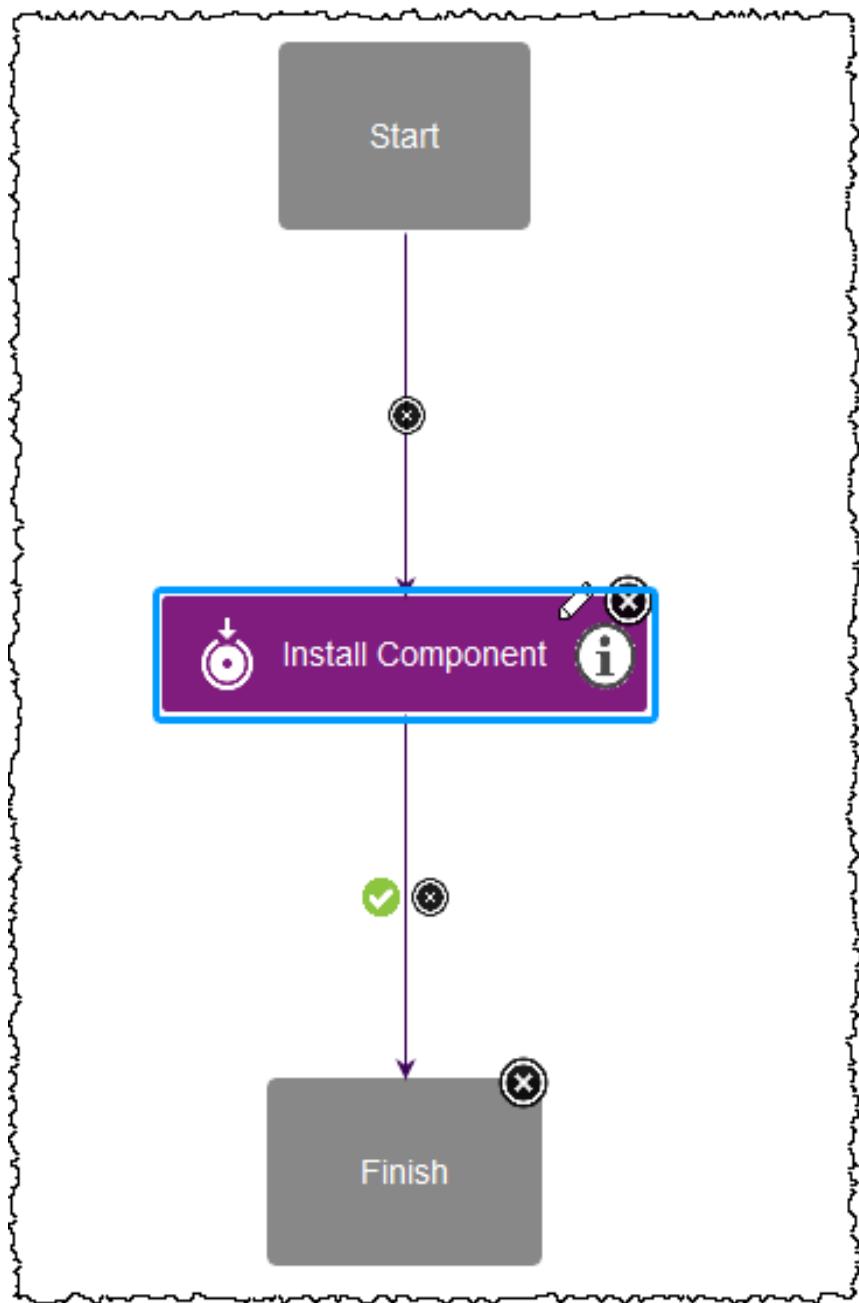


⋮

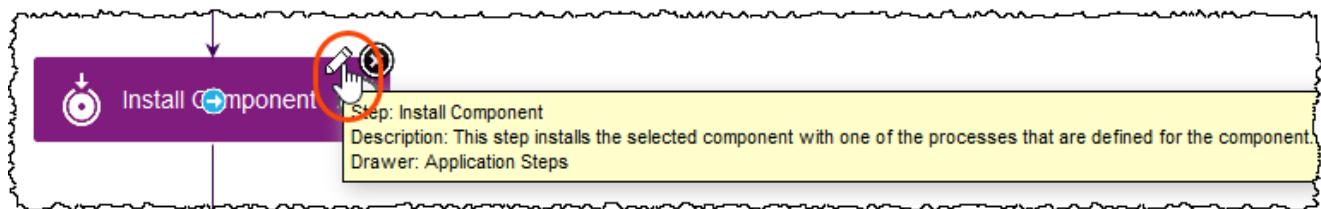
- 9.6 ► Find **Install Component...** step in the Design Palette, and drag it onto the space between *Start* and *Finish* boxes.



9.7 The result will be as below:



9.8 ➡ Click on the pencil icon on **Install Component** to possibly change the properties



9.9 Notice that the properties of this step are pre filled for you.

▶ Click OK:

Edit Properties for Install Component

Name*
Install: "J02Mortgage"

Component*
J02Mortgage

Use Versions Without Status*
Active ▾

Component Process*
Deploy JKE to CICS ▾

Limit to Resource Tag

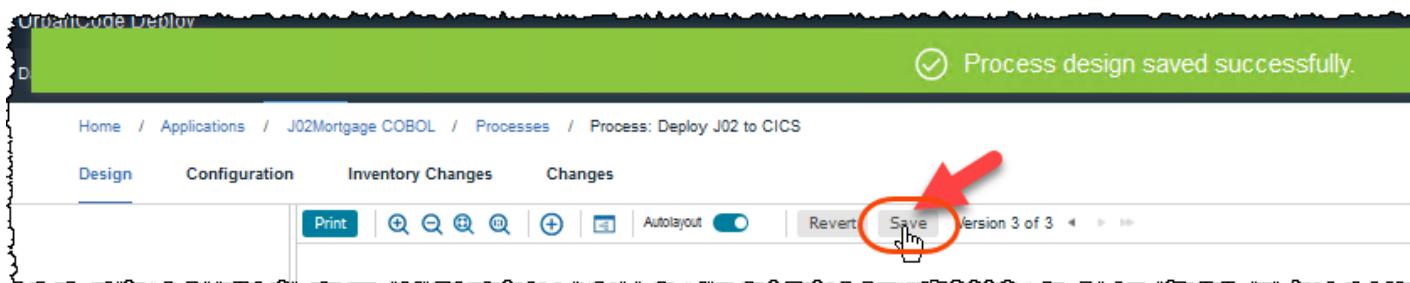
Maximum number of concurrent processes*
-1

Fail Fast

Precondition

Cancel OK

9.10 ▶ Click Save

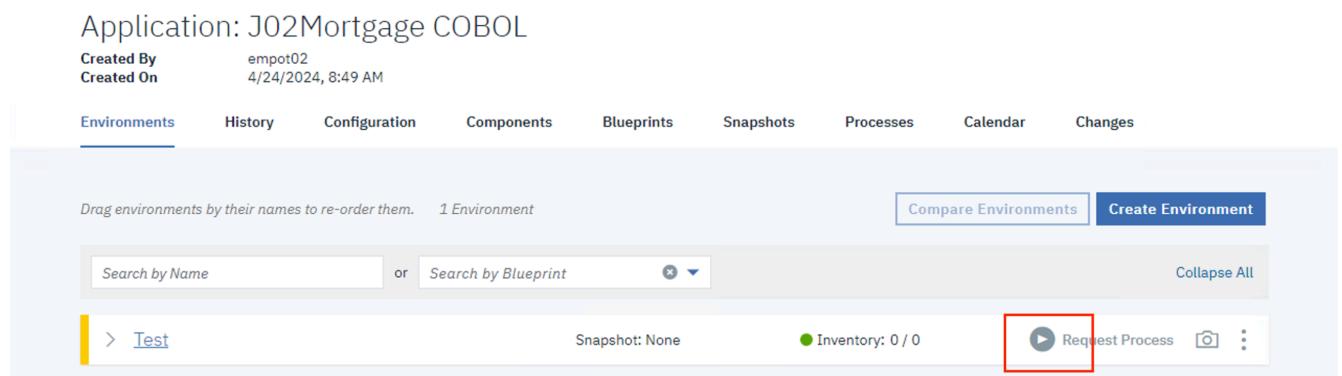


Task 10 – Environment properties configuration

On the task 7 you created the **Test** environment. For this lab we have just one environment, but usually we have few environments. Example the **Test** environment could reflect the Test LPAR and **Prod** environment that would reflect other LPAR. Each environment may have different values used on the deploy. Example the CICS test environment uses port 30090, the prod environment uses port 30091, etc..

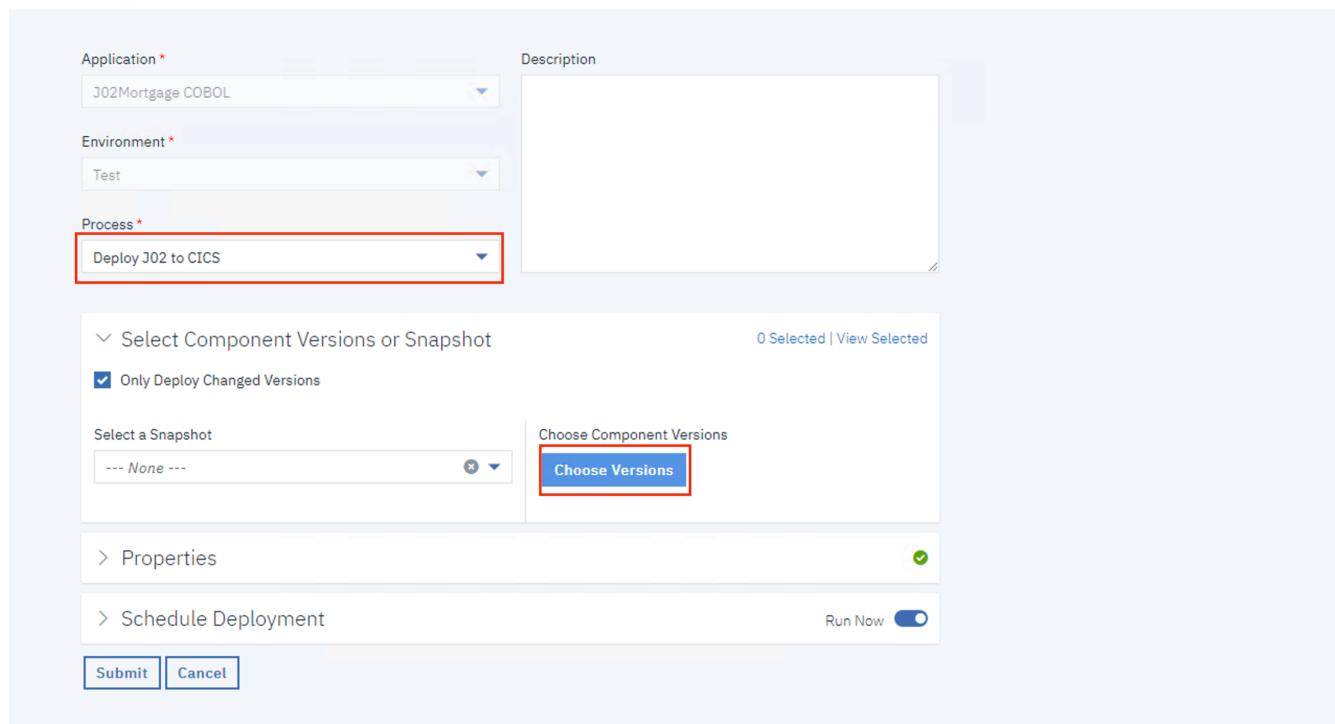
Those values in our lab were not yet defined and you will see errors when trying to deploy the application for **Test** environment..

- 10.1 ➡ Go to **Application > J02 Mortgage COBOL** and click the icon  on **Test** environment



The screenshot shows the 'Environments' tab selected in the navigation bar. Below it, a list of environments is shown with a single entry: 'Test'. To the right of the environment name, there is a 'Request Process' button, which is highlighted with a red box. Other buttons visible include 'Compare Environments' and 'Create Environment'.

- 10.2 ➡ Select the Process **Deploy J02 to CICS**, click **Choose Versions**, Run Application Process



The screenshot shows the 'Run Application Process' dialog. In the 'Process' dropdown, 'Deploy J02 to CICS' is selected and highlighted with a red box. Below the process selection, there is a section titled 'Select Component Versions or Snapshot' with a 'Choose Component Versions' button, which is also highlighted with a red box. Other buttons in this section include 'Select a Snapshot' and 'Only Deploy Changed Versions'. At the bottom of the dialog, there are 'Submit' and 'Cancel' buttons.

click **Select For All..** and select **Latest Available**. Close window.

Select Component Versions

Only Show Changed Components

Allow Invalid Versions

Component: J02Mortgage Current Environment Inventory Versions to Deploy

Filter:

Items per page: 10 | 1-1 of 1 item

- 1 Latest Available
- Latest with Status
- Versions With Name
- Current Environment Inventory
- Latest Available at Execution Time
- Latest with Status at Execution Time
- Current Environment Inventory at Execution Time
- None (Clear All)

You should see: and close window.

Select Component Versions

Only Show Changed Components

Allow Invalid Versions

Component: J02Mortgage Current Environment Inventory Versions to Deploy

Filter:

J02Mortgage	None	20240424 x	Add...
-------------	------	------------	--------

Items per page: 10 | 1-1 of 1 item

< Previous 1 Next >

10.3 ► When the dialog below is shown click **Submit**

Applications

Run Application Process

Application * J02Mortgage COBOL

Description

Environment * Test

Process * Deploy J02 to CICS

▼ Select Component Versions or Snapshot 1 Selected | View Selected

Only Deploy Changed Versions

Select a Snapshot --- None --- Choose Component Versions

Choose Versions

> Properties

> Schedule Deployment Run Now

Submit Cancel



The Deploy process start, **but it will fail as you see soon..**

10.4 ► Click **Expand all** on the far right

Applications / J02Mortgage COBOL / Process Request

Application Process Request: J02Mortgage COBOL

Process [Deploy J02 to CICS \(Version 2\)](#)
Environment [Test](#)
Only Deploy Changed true
Versions
Date Requested 4/24/2024, 9:15 AM
Requested By empot02
Scheduled For 4/24/2024, 9:15 AM

[View Deployment Request](#) [Process Request](#)

Log Properties Manifest Configuration Changes Inventory Changes

Execution Log ▾

[Expand All](#) [Collapse All](#) [Pause](#) [Cancel](#) [Force Cancel](#) [Download All Logs](#)

Step	Progress	Start Time	Duration	Status
» 1. ⚙️ Install: "J02Mortgage"	0 / 1	9:15:49 AM	0:00:23	Running
Total Execution	0 / 1	9:15:49 AM	0:00:23	Running

10.5 ► Click icon to expand .. Be patient.. this is slow on your environment.. Three steps were successful but the last one failed.

► Click on Error Log icon

Step ▾	Progress	Start Time	Duration	Status
v 1. ⚙ Install: "J02Mortgage"	1 / 1	1:11:20 PM	0:03:12	🔴 Failed
└ J02Mortgage	1 / 1	1:11:20 PM	0:03:12	🔴 Failed
└ Deploy JKE to CICS (J02Mortgage_20180709)	1	1:11:20 PM	0:03:12	🔴 Failed
1. ⚡ Download Artifacts for zOS	1	1:11:20 PM	0:01:15	🟢 Success
2. ⚙ Generate Program List	1	1:12:35 PM	0:00:35	🟢 Success
3. ⚡ Deploy Data Sets	1	1:13:11 PM	0:01:21	🟢 Success
4. ⚙ New copy resources	1 / 1	1:14:31 PM	0:00:00	🔴 Failed
Total Exec (CICS TS v. 41.20181207-0633)		1:11:20 PM	0:03:12	🔴 Failed
Error Log				

10.6 ► See the properties missing and close the dialog

The screenshot shows a tooltip box titled 'Error' with the message: 'The following properties were left unresolved: \${p:cics.cmciport}, \${p:cics.host}. Please ensure their values have been given and that the property names are spelled correctly'. A red arrow points from the top of the page to the 'Error Log' button in the table above. Another red arrow points from the bottom right of the tooltip to the close button 'X'.

10.7 We could add properties at various levels.. In our example let's make the variables at the Environment level. The missing properties are:

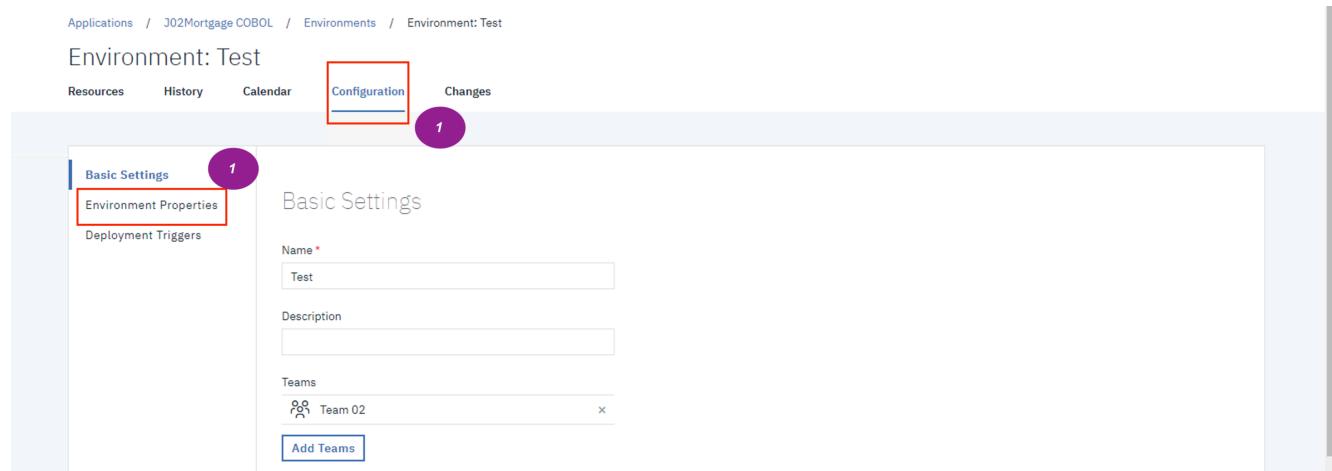
cics.cmciport and **cics.host**.

10.8 To add the properties..

► Click Applications > J02 Mortgage COBOL > Test

The screenshot shows the 'Application: J02Mortgage COBOL' details page. It includes sections for 'Created By' (empot02), 'Created On' (4/24/2024, 8:49 AM), and tabs for 'Environments', 'History', 'Configuration', 'Components', 'Blueprints', 'Snapshots', 'Processes', 'Calendar', and 'Changes'. Below the tabs, it says 'Drag environments by their names to re-order them.' and shows '1 Environment'. A search bar for 'Search by Name' or 'Search by Blueprint' is present. At the bottom, there are buttons for 'Compare Environments', 'Create Environment', 'Collapse All', 'Request Process', and other icons. A red box highlights the 'Test' environment entry in the list.

10.9  Select Configuration > Environment properties and click Add Property



The screenshot shows the 'Configuration' tab selected in the top navigation bar. A purple circle with the number '1' is positioned above the 'Changes' tab. On the left sidebar, the 'Environment Properties' tab is highlighted with a red box and a purple circle with the number '1'. The main area displays the 'Basic Settings' section with fields for Name (Test), Description, and Teams (Team 02). An 'Add Teams' button is visible.

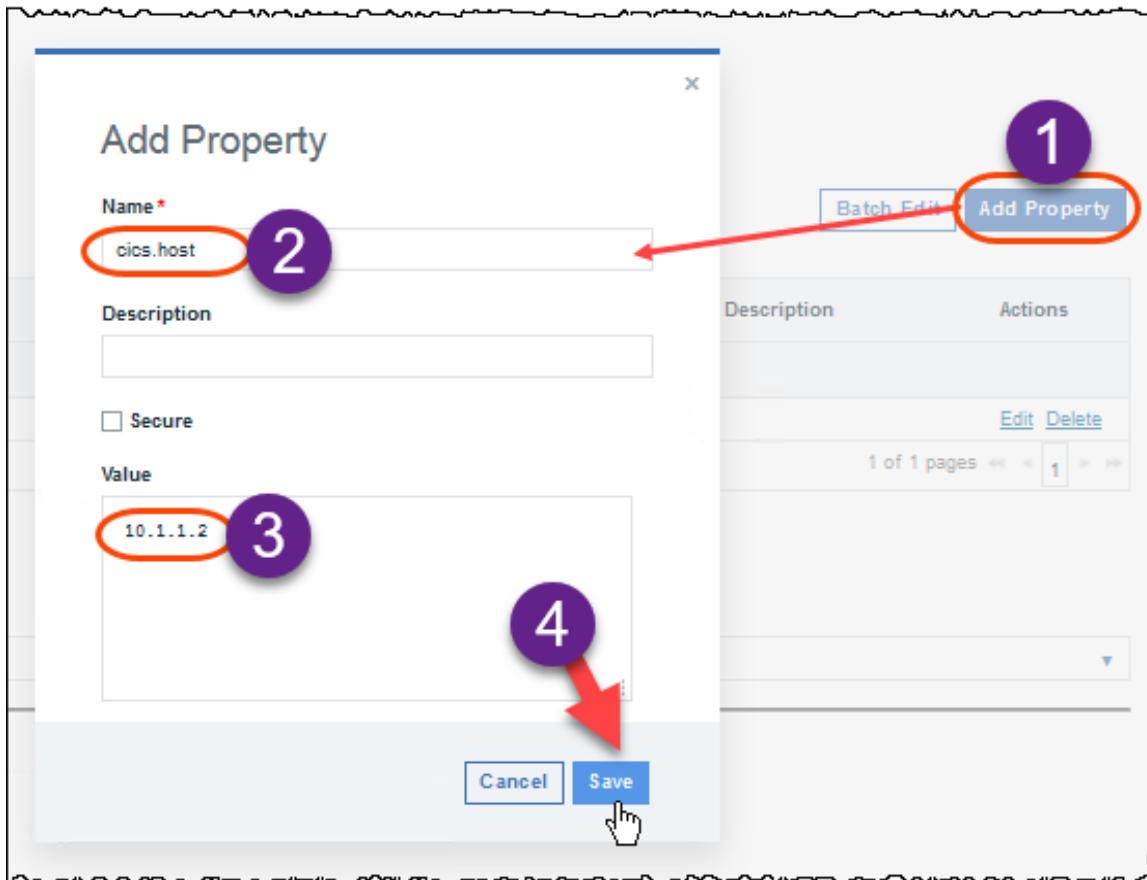


The screenshot shows the 'Environment Properties' tab selected in the top navigation bar. A purple circle with the number '1' is positioned above the 'Changes' tab. On the left sidebar, the 'Environment Properties' tab is highlighted with a red box and a purple circle with the number '1'. The main area displays the 'Environment Properties' section with a message 'Version 6 of 6'. At the bottom right, there are 'Add Property' and 'Batch Edit' buttons, with the 'Add Property' button highlighted by a red box.

- 10.10 ► Add the property **cics.cmciport** with value **1490** and click **Save**

The screenshot shows a 'Add Property' dialog box. At the top, there is a 'Name*' label followed by a text input field containing 'cics.cmciport', which is highlighted with a red oval. Below it is a 'Description' label with an empty input field. Underneath is a 'Secure' checkbox followed by a 'Value' label with a text input field containing '1490'. At the bottom right, there are two buttons: 'Cancel' and 'Save', with a large red arrow pointing towards the 'Save' button.

- 10.12 ► Add the property **cics.host** with value **10.1.1.2** and click **Save**



10.13 ► Using the Add Property button

add the property **cics.userid** with the value **empot02** (your userid)

10.14 ► Also add the property **cics.password** with the value **empot02**.

In the real world you will not add the password here and will use other secure way to authenticate., also

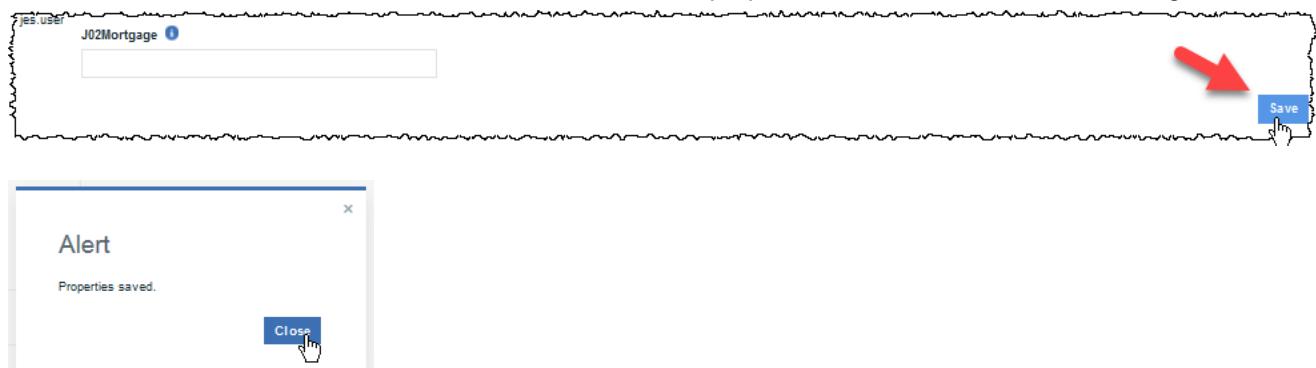
In the Edit Property dialog, you can check the secure box and this will hide (secure) the password..

When complete:

The screenshot shows a list of properties with four new entries highlighted by a red box.

Name	Description	Actions
cics.cmciport	1490	Edit Delete
cics.host	10.1.1.2	Edit Delete
cics.password	empot02	Edit Delete
empot02	empot02	Edit Delete

- 10.15 ► Scroll down and click **Save** to save the defined properties and **Close** to close the dialog.



PART 3 – Deploy the application to z/OS CICS

On this part you will deploy the Application to the z/OS CICS.

To deploy the components in the application, you must run the application process on the specified environment. We created only one environment named **Test**, you will deploy this environment.

Task 11 – Run an application process

You are now ready to test our deployment process.

- 11.1 ► Click the **Applications** tab and then click **J02 Mortgage COBOL**

Name	Tags	Template	Description	Created	By	⋮
CRSA	🕒			3/6/2023, 3:54 AM	admin	⋮
Genapp	🕒			3/3/2023, 4:26 AM	admin	⋮
J02Mortgage.COBOL	🕒			4/24/2024, 8:49 AM	empot02	⋮

- 11.2 ► Under **Test** environment click the **Request Process** icon

Applications / J02Mortgage COBOL

Application: J02Mortgage COBOL

Created By: empot02
Created On: 4/24/2024, 8:49 AM

Environments History Configuration Components Blueprints Snapshots Processes Calendar Changes

Drag environments by their names to re-order them. 1 Environment

Search by Name or Search by Blueprint

> Test Snapshot: None ● Inventory: 1 / 1 Request Process

- 11.3 ► In the Run Process window, un-select Only Changed Versions, in the Process, select the Deploy J02 to CICS and Under Component Versions, click Choose Versions

Applications

Run Application Process

Application * J02Mortgage COBOL

Environment * Test

Process * Deploy J02 to CICS

1 Select Component Versions or Snapshot

Only Deploy Changed Versions

Select a Snapshot --- None ---

2 Choose Component Versions

3 Choose Versions

Properties

Schedule Deployment

Run Now

- 11.4 ► Click Select For All > Latest available click on X

The screenshot shows the UrbanCode Deploy application interface. On the left, the 'Run Application Process' screen is displayed with fields for Application (J02Mortgage COBOL), Environment (Test), and Process (Deploy J02 to CICS). A dropdown menu under 'Select Component Versions or Snapshot' is open, showing options like 'Only Deploy Changed Versions' and 'Choose Component Versions'. On the right, a modal window titled 'Select Component Versions' is open, listing component versions for 'J02Mortgage'. The 'Latest Available' option is highlighted with a purple circle and labeled '1'. The 'Select For All' button is also highlighted with a purple circle and labeled '2'. A red arrow points from the number '3' to the close button in the top right corner of the modal.

Make sure that the version that you created on PART 1 of the lab is selected for **J02Mortgage**
If you do not select a version, that component is not deployed.

11.5 ► Click **OK** and **Submit**

Applications

Run Application Process

Application *

J02Mortgage COBOL

Environment *

Test

Process *

Deploy J02 to CICS

Description

▼ Select Component Versions or Snapshot 1 Selected | View Selected

Only Deploy Changed Versions

Select a Snapshot

--- None ---

Choose Component Versions

Choose Versions

> Properties ✓

> Schedule Deployment Run Now

Submit **Cancel**



The web page shows you the progress of the application process request. From this page, you can watch as the processes run. The following figure shows the application process partially completed. The application component process is finished and the other two component processes are running.

11.6 ►| Click **Expand All** and expand **Deploy JKE to CICS** to see all steps and WAIT the deploy...

If the process runs successfully, the request shows that each component process is finished, as in the following figure:

Execution Log				
Step	Progress	Start Time	Duration	Status
▼ 1. ⚙️ Install: "J02Mortgage"	1 / 1	12:39:01 PM	0:00:27	● Success
▼ J02Mortgage	1 / 1	12:39:01 PM	0:00:27	● Success
▼ Deploy JKE to CICS (J02Mortgage_20240424)		12:39:01 PM	0:00:27	● Success
1. 📥 Download Artifacts for zOS		12:39:01 PM	0:00:06	● Success
2. ⚡ Deploy Data Sets		12:39:07 PM	0:00:10	● Success
3. ⚙️ Generate Program List		12:39:17 PM	0:00:04	● Success
4. ⚙️ New copy resources		12:39:22 PM	0:00:06	● Success
Total Execution	1 / 1	12:39:01 PM	0:00:27	● Success

11.7 ►| Click on **Output Log** to see the CICS Programs NEWCOPY executed.

2. ⚡ Deploy Data Sets		12:39:07 PM	0:00:10	● Success	⋮
3. ⚙️ Generate Program List		12:39:17 PM	0:00:04	● Success	⋮
4. ⚙️ New copy resources		12:39:22 PM	0:00:06	● Success	⋮
Total Execution	1 / 1	12:39:01 PM	0:00:27	● Success	View Output Log View Input/Output Properties

```
WE_ACTIVITY_ID=18f1656d-86ed-d633-7037-a486c7bd9cb0
=====
2024/04/25 17:39:52.851 GMT BUZCP0006I Connected to "10.1.1.2:1490". CICS TS version: 050600.
2024/04/25 17:39:53.322 GMT BUZCP0037I Perform NEWCOPY Operation.
2024/04/25 17:39:53.544 GMT BUZCP0024I NEWCOPY PROGRAM "J02CMORT" succeeded.
2024/04/25 17:39:53.568 GMT BUZCP0024I NEWCOPY PROGRAM "J02MPMT" succeeded.
2024/04/25 17:39:53.577 GMT BUZCP0029I Summary: 2 NEWCOPY request(s) succeeded, 0 NEWCOPY request(s) failed.
```

11.8 ►| Close the web browser

Task 12 – Verifying the Deploy results at z/OS CICS

The z/OS components are unique for each student, so you can see the code that you deployed to CICS.

- 12.1 ➔ Use the 3270 emulator

Go to the windows desktop and **double click the 3270 emulation icon**



I cicsempotj02

- 12.2 Verify the CICS application. L cicsts56

➔ Type **I cicsts53** (I is L lower case)

```

z/OS V2R5 LVLI PUT2112/RSU2112                               IP Address = 10.1.1.1
                                                               VTAM Terminal = TCP00006

                           Application Developer System

                           // 0000000  SSSSS
                           // 00      00 SS
zzzzzz // 00      00 SS
zz   // 00      00 SSSS
zz   // 00      00      SS
zz   // 00      00      SS
zzzzzz // 0000000  SSSS

System Customization - ADCCD.Z25A.*


====> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
====> Enter L followed by the APPLID
====> Examples: "L TSO", "L CICSTS56", "L CICSTS55", "L IMS15"

l cicsts56_
MA A

```

24/011

12.3 ► Authenticate using your id **empot02** and password **empot02**

Signon to CICS

APPLID CICSTS53

WELCOME TO CICS TS 5.3

Type your userid and password, then press ENTER:

Userid . . . empot02 Groupid . . . _____
Password . . . -
Language . . . -

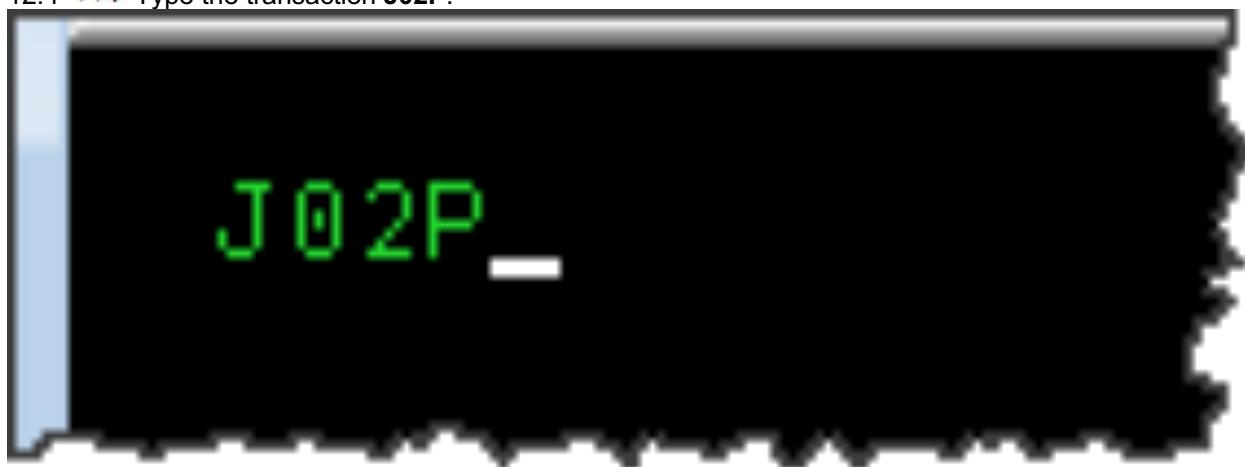
New Password . . .

DFHCE3520 Please type your userid.
F3=Exit

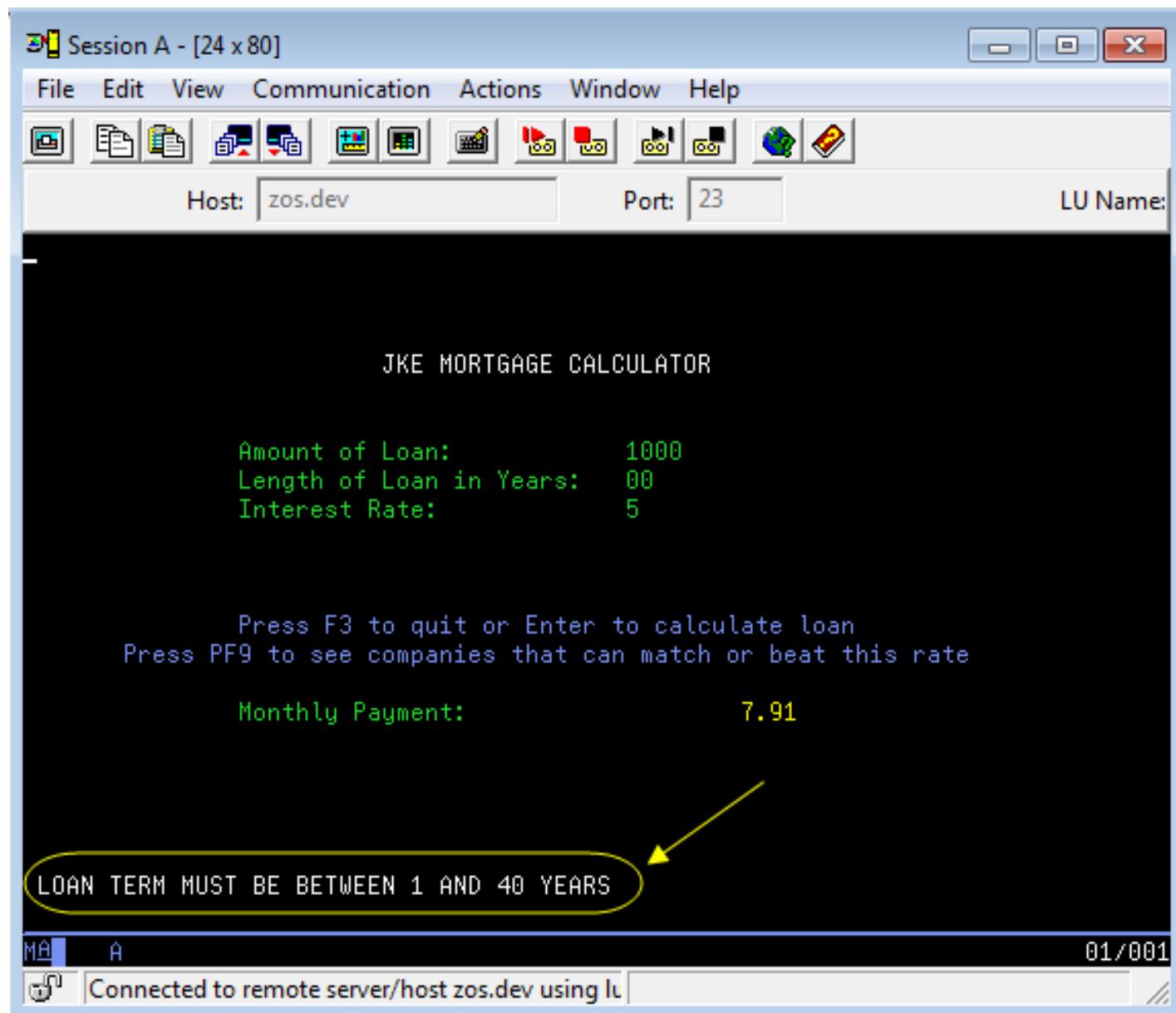
MA A

11/033

12.4 ► Type the transaction **J02P**.



12.5 ► Type **00** on Length of Loan in Years. Note that the error message.



Congratulations! You completed the lab 2

LAB 3 – Using IBM Dependency Based Build with Git, Jenkins and UCD on z/OS

Updated June,2024 (revised by Carlos Hirata/Wilbert Kho, Reviewed by Joe Huang and Frank Hernandez)

This lab will take you through the steps of using [IBM Dependency Based Build](#) (DBB) along with [Git](#), [Jenkins](#) and [UrbanCode Deploy](#) (UCD) on z/OS.

On this lab you will modify an existing COBOL/CICS application stored in Git.

You will use **IDz** (which is part of the **ADFz** "package") to change the code and perform a personal test for later delivery and commit to *Git* and then use *Jenkins* 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.

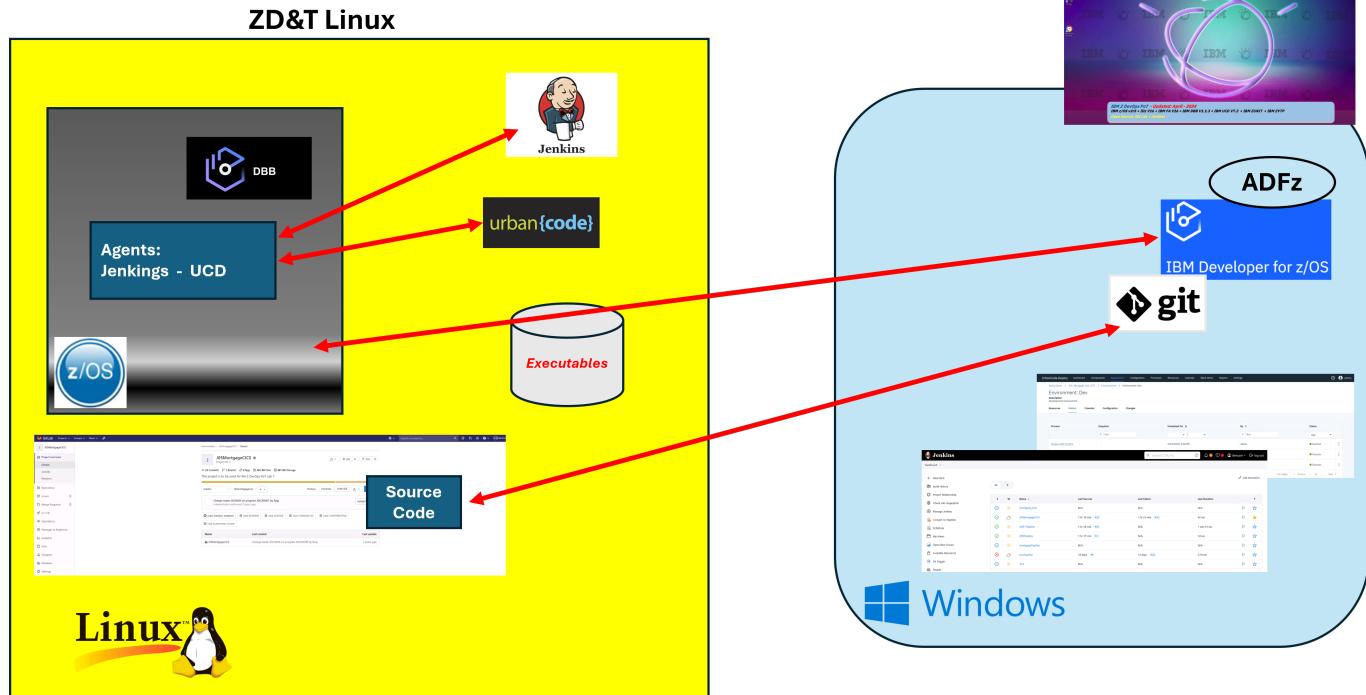
More about DBB https://www.ibm.com/support/knowledgecenter/SS6T76_1.1.0/welcome.html

The environment used on this Lab is pictured below.

Note that we have few "servers" running on Linux like **UCD (UrbanCode Deploy)** server, **Jenkins** and also the **Git Repository (Gitlab)** is on Linux.

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

Topology used on the labs



Overview of development tasks

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

1. **Get familiar with the application using the 3270 terminal**
→ You will start a 3270 emulation and execute a transaction named **J05P** to become familiar with the Application that you intend to modify.
2. **Load the source code from Git to the local IDz workspace**
→ You will load the COBOL code that is stored on Linux to your windows client to be modified.
3. **Modify the COBOL code using IDz.**
→ Using *IDz* you will modify the COBOL code to have a different message in a *C/ICS* dialog.
4. **Use IDz DBB User Build to compile/bind and perform personal tests.**
→ You will compile and link the modified code using the *DBB User Build* Function. When complete you will run the code using *C/ICS* for a personal test and verify that the change is correctly implemented.
5. **Push and Commit the changed code to Git .**
→ You will commit the changes to *Git*.
6. **Use Jenkins with Git plugin to build all the modified code committed to Git.**
→ You will use Jenkins pipeline to build the new changed code and push the executables to be deployed using *UCD*.
Note that this step makes a build of all code committed to Git, while on step 4 only the selected COBOL program was built.
7. **Use Jenkins and UCD plugin to deploy results and test the CICS transaction again**
→ You will verify the results after the final deploy to *C/ICS* using *UCD*
8. **Understanding DBB Build Reports**
→ You will understand the reports generated by *DBB* during the build.

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. For that reason we are using **GitLab** in this lab, but the steps would be exactly the same if using other tools.

GitHub offers free public repositories; **Bitbucket** also offers free private repositories; **GitLab** offers a Community Edition which is entirely free

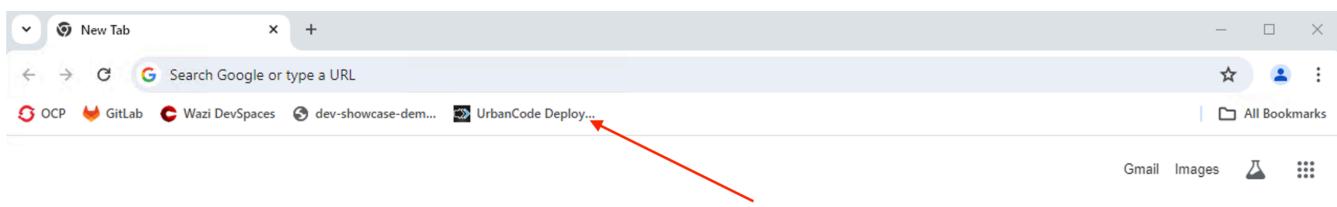
Section 0. Checking UCD agent is ONLINE.

Before start UCD lab, let's verify UCD agent (zOS) is connect to UCD server (Linux Server).

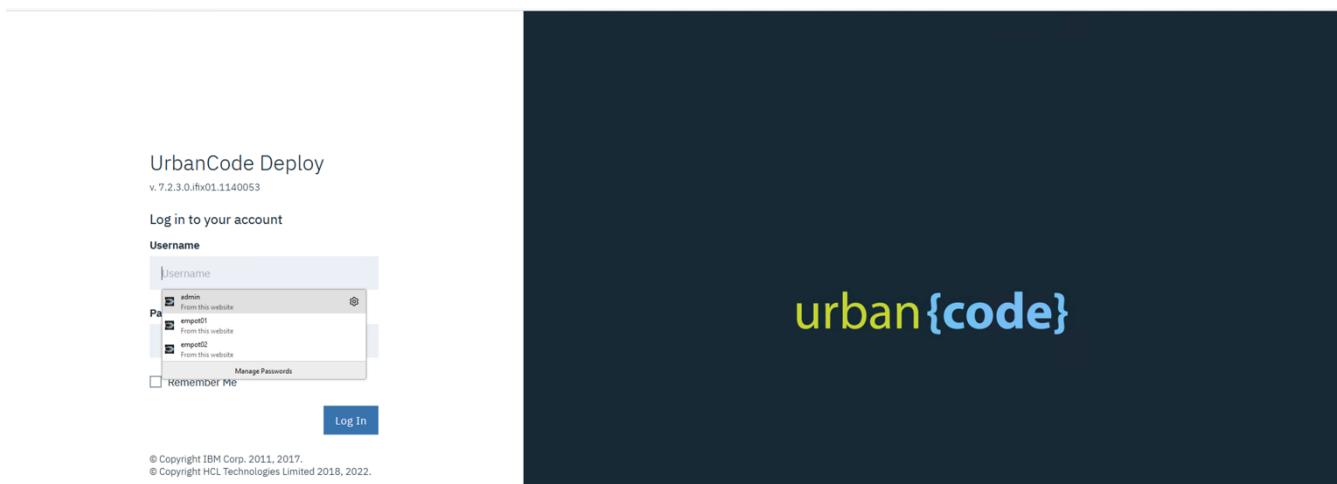
0.1.1 Open firefox browser.



0.1.2 ➡ Look at Bookmark toolbar and select **UrbanCode Deploy**



0.1.3. ➡ Login with **admin** and password **admin**



0.1.4 On Urbancode page, locate **Resources** tab. Look at the agent (zos.dev2) and check the status.

If the status is **Online**, it is good and skip to Task 1 in this lab. If the status is **Offline**. You must start the agent on zOS.

UrbanCode Deploy Dashboard Components Applications Configuration Processes **Resources** Calendar Work Items Reports Settings ? admin

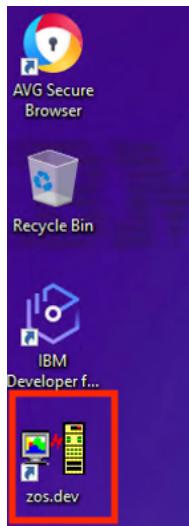
Resources

Resource Tree Resource Templates Agents Agent Configuration Templates Agent Relays Agent Pools Cloud Connections

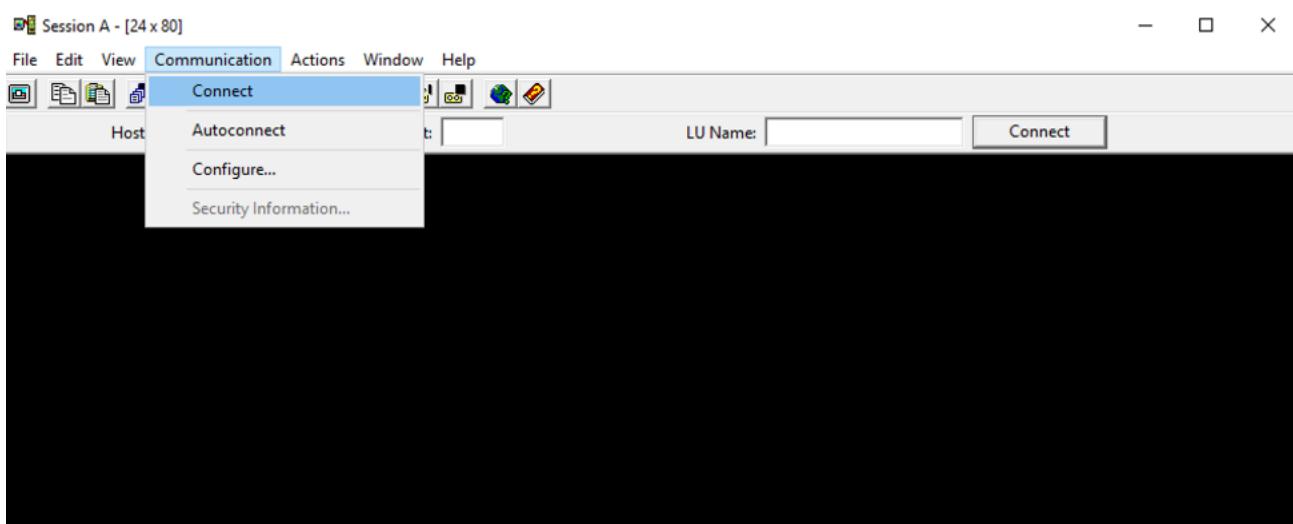
Show Bulk Actions Select All... Create Top-Level Group Customize Display

Name	Tags	Inventory	Status	Description
zos.dev2 (View Agent)		version-375 (+ 27 more)	Offline	
CBSA-backend (View Component)		version-458 (+ 15 more)		
Genapp-backend (View Component)		20240617-185833 (+ 1 more)		
J05MortgagePOT (View Component)				

0.1.5 On Windows Desktop, click ➡️ zos.dev icon and open PCMM application.



0.1.6 Click ➡️ on Communication → Connect



0.1.7 The z/OS main screen will show up and enter logon ibmuser (it can be lower case).

```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect
z/OS V2R5 LVLI PUT2112/RSU2112 IP Address = 10.1.1.1
VTAM Terminal = TCP00004

Application Developer System

        // 0000000      SSSSS
        // 00      00 SS
zzzzzz // 00      00 SS
zz // 00      00 SSSS
zz // 00      00      SS
zz // 00      00      SS
zzzzzz // 0000000  SSSS

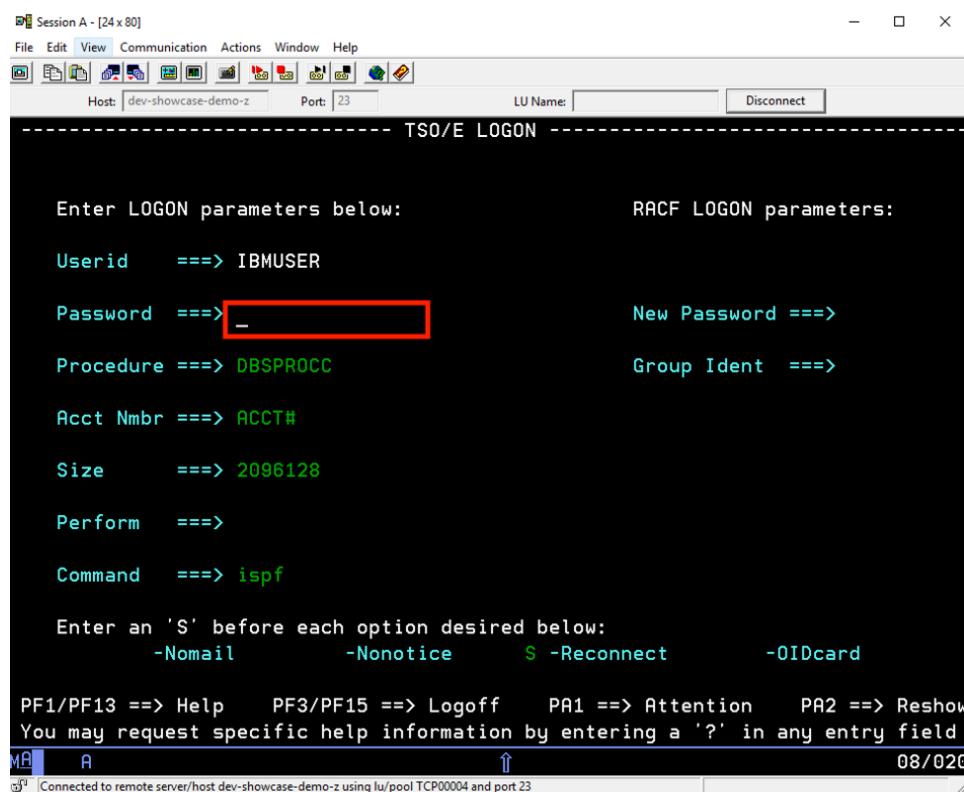
System Customization - ADCD.Z25A.*


==> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
==> Enter L followed by the APPLID
==> Examples: "L TSO", "L CICSTS56", "L CICSTS55", "L IMS15"

logon ibmuser_
MA A 24/014
Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00004 and port 23

```

0.1.8 The userid is IBMUSER and the password is sys1. Press **ENTER** and **ENTER**.

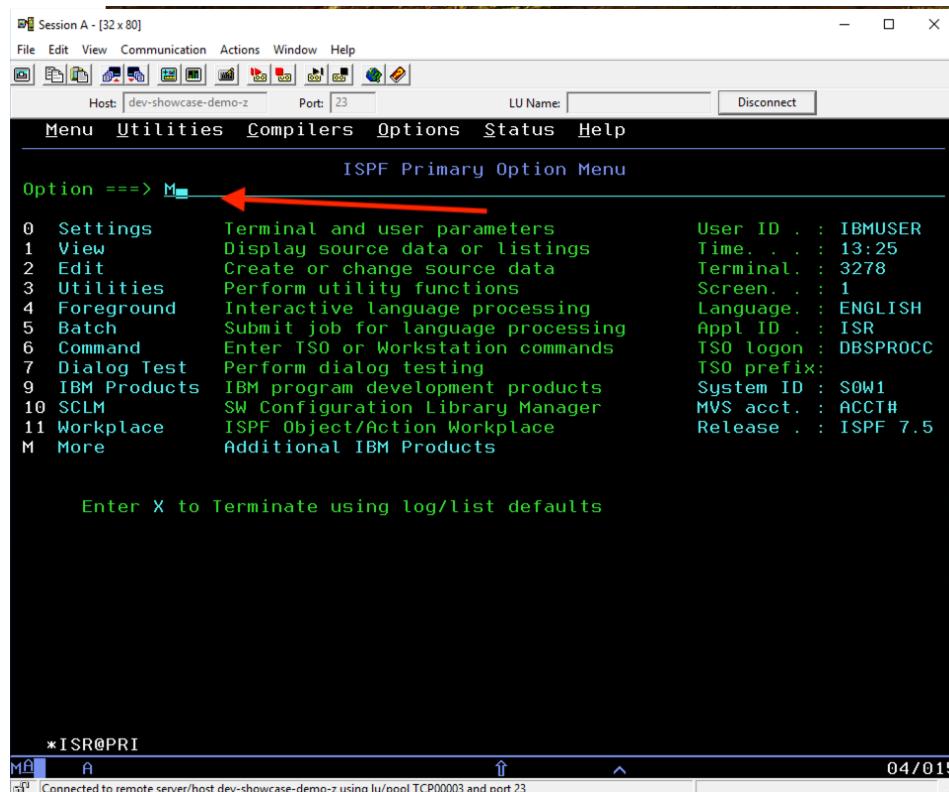


```

Session A - [24 x 80]
File Edit View Communication Actions Window Help
Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect
----- TSO/E LOGON -----
Enter LOGON parameters below:          RACF LOGON parameters:
Userid    ==> IBMUSER
Password  ==> [REDACTED] New Password ==>
Procedure ==> DBSPROCC Group Ident ==>
Acct Nmbr ==> ACCT#
Size      ==> 2096128
Perform   ==>
Command   ==> ispf
Enter an 'S' before each option desired below:
-Nomail     -Nonotice    S -Reconnect    -OIDcard
PF1/PF13 ==> Help    PF3/PF15 ==> Logoff   PA1 ==> Attention   PA2 ==> Reshow
You may request specific help information by entering a '?' in any entry field
MA A                                     08/020
Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00004 and port 23

```

0.1.9 On ISPF Primary Menu. Enter the letter M on **Option** field.Press **ENTER**.



```

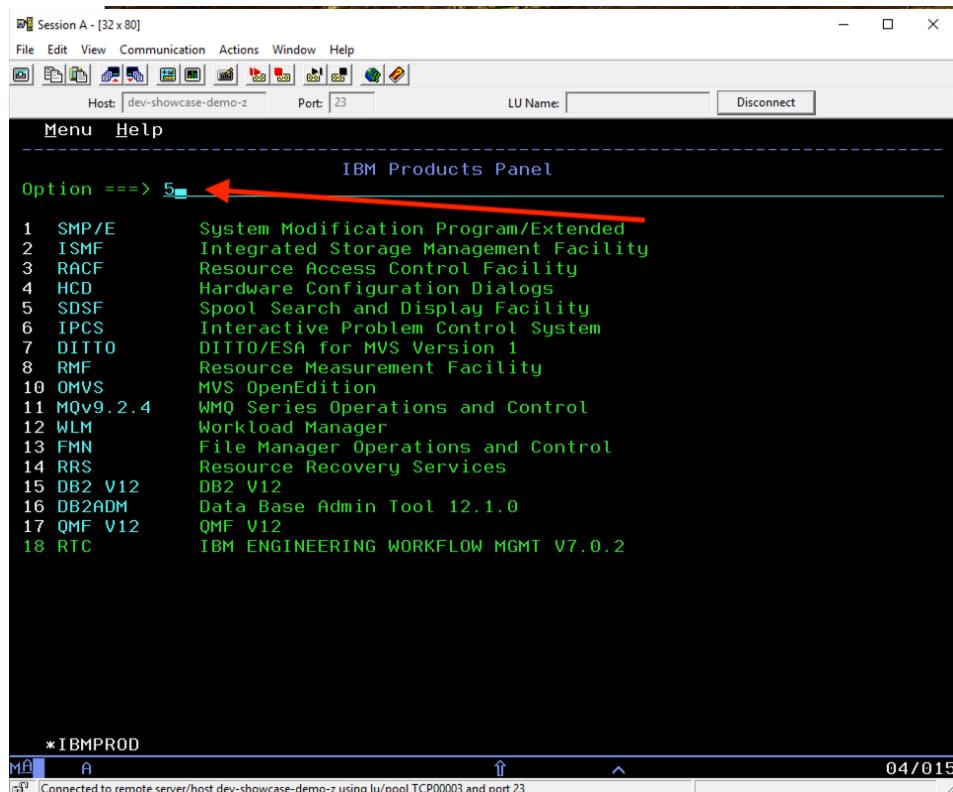
Session A - [32 x 80]
File Edit View Communication Actions Window Help
Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect
----- ISPF Primary Option Menu -----
Menu Utilities Compilers Options Status Help
Option ==> M [Red arrow pointing to the 'M' in 'Option']
0 Settings Terminal and user parameters User ID . : IBMUSER
1 View Display source data or listings Time. . . : 13:25
2 Edit Create or change source data Terminal. . : 3278
3 Utilities Perform utility functions Screen. . : 1
4 Foreground Interactive language processing Language. . : ENGLISH
5 Batch Submit job for language processing Appl ID . : ISR
6 Command Enter TSO or Workstation commands TSO logon : DBSPROCC
7 Dialog Test Perform dialog testing TSO prefix:
9 IBM Products IBM program development products System ID : SOW1
10 SCLM SW Configuration Library Manager MVS acct. . : ACCT#
11 Workplace ISPF Object/Action Workplace Release . : ISPF 7.5
M More Additional IBM Products

Enter X to Terminate using log/list defaults

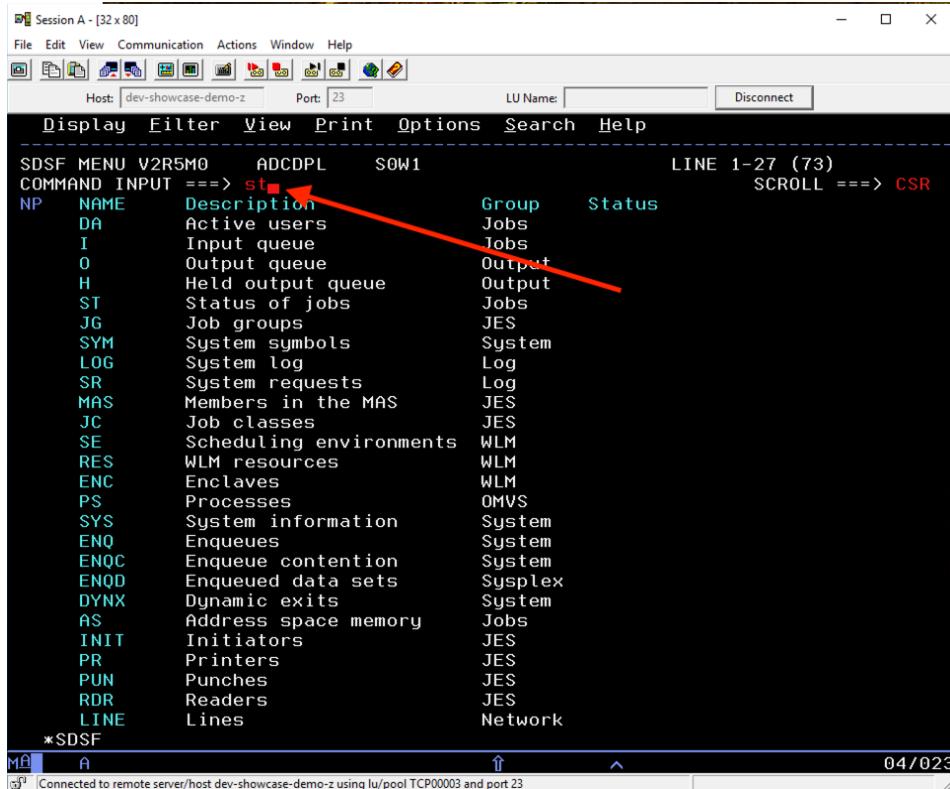
*ISR@PRI
MA A                                     04/015
Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00003 and port 23

```

0.1.10 On IBM Products Panel. Enter the number **5** to go to **Spool Search and Display Facility**.



0.1.11 On **SDSF Menu**. Type **st** command on **COMMAND INPUT** and press **ENTER**.



```

SDSF MENU V2R5M0      ADCDPL    S0W1          LINE 1-27 (73)
COMMAND INPUT ==> st  SCROLL ==> CSR

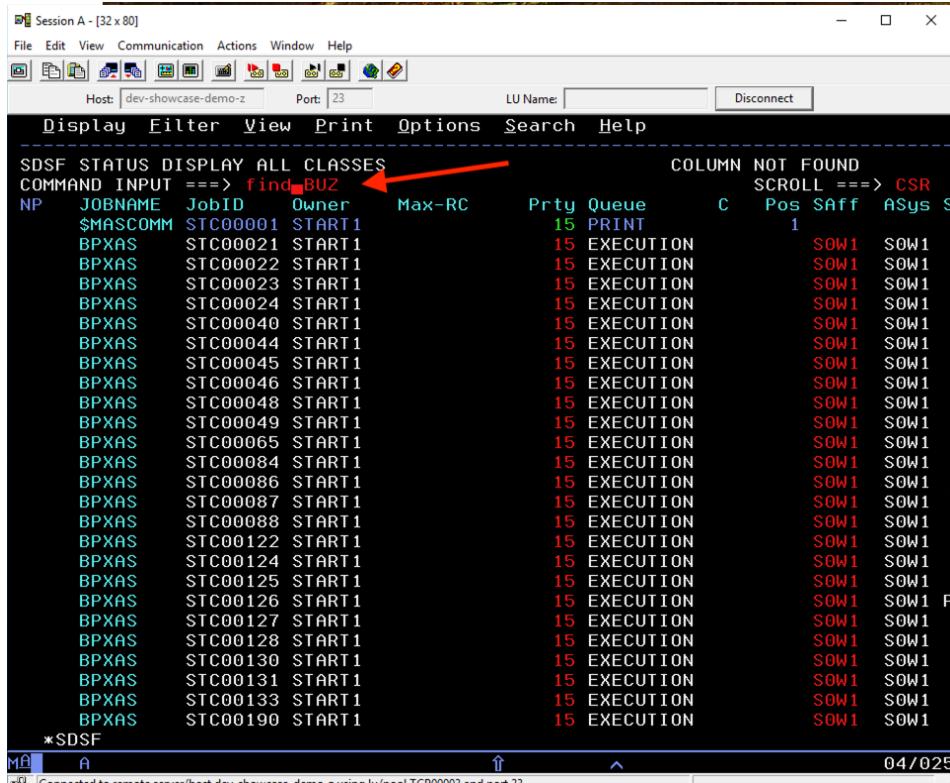
NP  NAME        Description      Group     Status
DA  Active users   Jobs
I   Input queue    Jobs
O   Output queue   Output
H   Held output queue  Output
ST  Status of jobs  Jobs
JG  Job groups    JES
SYM System symbols  System
LOG System log     Log
SR  System requests Log
MAS Members in the MAS JES
JC  Job classes    JES
SE  Scheduling environments WLM
RES WLM resources   WLM
ENC Enclaves       WLM
PS  Processes      OMVS
SYS System information System
ENQ Enqueues       System
ENQC Enqueue contention System
ENQD Enqueued data sets Sysplex
DYNX Dynamic exits   System
AS  Address space memory Jobs
INIT Initiators    JES
PR  Printers       JES
PUN Punches        JES
RDR Readers        JES
LINE Lines          Network

*SDSF

```

Connected to remote server/host dev-showcase-demo-z using lpu/pool TCP00003 and port 23

0.1.12 On SDSF STATUS DISPLAY ALL CLASSES, type find buz (3 letters is enough) and press ENTER.



```

SDSF STATUS DISPLAY ALL CLASSES          COLUMN NOT FOUND
COMMAND INPUT ==> find BUZ  SCROLL ==> CSR

NP  JOBNAM JobID  Owner  Max-RC  Prty Queue  C  Pos  Staff  Asys  S
SMASCOMM STC00001 START1  15 PRINT   1
BPXAS    STC00021 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00022 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00023 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00024 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00040 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00044 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00045 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00046 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00048 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00049 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00065 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00084 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00086 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00087 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00088 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00122 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00124 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00125 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00126 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00127 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00128 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00130 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00131 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00133 START1  15 EXECUTION  S0W1  S0W1
BPXAS    STC00190 START1  15 EXECUTION  S0W1  S0W1

*SDSF

```

Connected to remote server/host dev-showcase-demo-z using lpu/pool TCP00003 and port 23

0.1.13 BUZAGNT agent might be started, but it is not connected to the UCD server. It is necessary to cancel this JOB and start it again. On the column NP, type letter c the front of BUZAGNT job and press ENTER.

```

Session A - [32 x 80]
File Edit View Communication Actions Window Help
Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect
Display Filter View Print Options Search Help
SDSF STATUS DISPLAY ALL CLASSES LINE 38-63 (118)
COMMAND INPUT ==> SCROLL ==> CSR
NP JOBNAME JobID Owner Max-RC Prty Queue C Pos SAff ASys S
BUZAGNT STC00464 START1 CANCELED 15 EXECUTION SOW1 SOW1
CFZCIM STC00027 CFZSRV 15 EXECUTION SOW1 SOW1
CICSTS56 STC00175 CICSUSER 15 EXECUTION SOW1 SOW1
CICSTS56 STC00028 CICSUSER CC 0001 1 PRINT 11
CNMPROC STC00068 START1 15 EXECUTION SOW1 SOW1
CSF STC00016 START1 15 EXECUTION SOW1 SOW1
DBB STC00074 IBMUSER 15 EXECUTION SOW1 SOW1
DBBS STC00075 IBMUSER 15 EXECUTION SOW1 SOW1
DBCADM1 STC00055 START1 15 EXECUTION SOW1 SOW1
DBCDBM1 STC00050 START1 15 EXECUTION SOW1 SOW1
DBC DIST STC00054 START1 15 EXECUTION SOW1 SOW1
DBC GIRLM STC00041 START1 15 EXECUTION SOW1 SOW1
DBC GMSTR STC00029 START1 15 EXECUTION SOW1 SOW1
DBG MGR STC00026 START1 15 EXECUTION SOW1 SOW1
EQAPROF STC00077 STCEQA 15 EXECUTION SOW1 SOW1
EQARMTD STC00078 START1 15 EXECUTION SOW1 SOW1
GITLABR STC00076 IBMUSER CC 3840 1 PRINT 9
HTTPD1 STC00085 WEBSRV 15 EXECUTION SOW1 SOW1
IBMUSER TSU00463 IBMUSER 15 EXECUTION SOW1 SOW1
IMSGNJCL JOB00064 START1 CC 0000 1 PRINT A 5
IMS1CTL STC00033 START1 15 EXECUTION SOW1 SOW1
IMS1DLI STC00051 START1 15 EXECUTION SOW1 SOW1
IMS1DRC STC00052 START1 15 EXECUTION SOW1 SOW1
IMS1ERE JOB00061 START1 CC 0000 1 PRINT A 2
IMS1ERE JOB00062 START1 CC 0008 1 PRINT A 3
IMS1ERE JOB00063 START1 CC 0008 1 PRINT A 4
*SDSF
M A ↑ ^ 06/003
Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00003 and port 23

```

0.1.14 BUZAGNT is canceled and now start the BUZAGNT. On **COMMAND INPUT**, type **/s BUZAGNT**. Press **ENTER**.

```

Session A - [32 x 80]
File Edit View Communication Actions Window Help
Host: dev-showcase-demo-z Port: 23 LU Name: Disconnect
Display Filter View Print Options Search Help
SDSF STATUS DISPLAY ALL CLASSES LINE 38-63 (118)
COMMAND INPUT ==> /s buzagnt 1
NP JOBNAME JobID Owner Max-RC Prty Queue C Pos SAff ASys S
BUZAGNT STC00464 START1 CANCELED 1 PRINT 37
CFZCIM STC00027 CFZSRV 15 EXECUTION SOW1 SOW1
CICSTS56 STC00175 CICSUSER 15 EXECUTION SOW1 SOW1
CICSTS56 STC00028 CICSUSER CC 0001 1 PRINT 11
CNMPROC STC00068 START1 15 EXECUTION SOW1 SOW1
CSF STC00016 START1 15 EXECUTION SOW1 SOW1
DBB STC00074 IBMUSER 15 EXECUTION SOW1 SOW1
DBBS STC00075 IBMUSER 15 EXECUTION SOW1 SOW1
DBCADM1 STC00055 START1 15 EXECUTION SOW1 SOW1
DBCDBM1 STC00050 START1 15 EXECUTION SOW1 SOW1
DBC DIST STC00054 START1 15 EXECUTION SOW1 SOW1
DBC GIRLM STC00041 START1 15 EXECUTION SOW1 SOW1
DBC GMSTR STC00029 START1 15 EXECUTION SOW1 SOW1
DBG MGR STC00026 START1 15 EXECUTION SOW1 SOW1
EQAPROF STC00077 STCEQA 15 EXECUTION SOW1 SOW1
EQARMTD STC00078 START1 15 EXECUTION SOW1 SOW1
GITLABR STC00076 IBMUSER CC 3840 1 PRINT 9
HTTPD1 STC00085 WEBSRV 15 EXECUTION SOW1 SOW1
IBMUSER TSU00463 IBMUSER 15 EXECUTION SOW1 SOW1
IMSGNJCL JOB00064 START1 CC 0000 1 PRINT A 5
IMS1CTL STC00033 START1 15 EXECUTION SOW1 SOW1
IMS1DLI STC00051 START1 15 EXECUTION SOW1 SOW1
IMS1DRC STC00052 START1 15 EXECUTION SOW1 SOW1
IMS1ERE JOB00061 START1 CC 0000 1 PRINT A 2
IMS1ERE JOB00062 START1 CC 0008 1 PRINT A 3
IMS1ERE JOB00063 START1 CC 0008 1 PRINT A 4
*SDSF
M A ↑ ^ 04/031
Connected to remote server/host dev-showcase-demo-z using lu/pool TCP00003 and port 23

```

0.1.15 BUZAGNT agent is connect.

SDSF STATUS DISPLAY ALL CLASSES

NP	JOBNAME	JobID	Owner	Max-RC	Prtv	Queue	C	Pos	Saff	ASys	S
	BUZAGNT	STC00465	START1		15	EXECUTION			SOW1	SOW1	
	BUZAGNT	STC00464	START1	CANCELED	1	PRINT		37			
	CFZCIM	STC00027	CFZSRV		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00175	CICSUSER		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00028	CICSUSER	CC 0001	1	PRINT		11			
	CNMPROC	STC00068	START1		15	EXECUTION			SOW1	SOW1	
	CSF	STC00016	START1		15	EXECUTION			SOW1	SOW1	
	DBB	STC00074	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBBS	STC00075	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBCADM	STC00055	START1		15	EXECUTION			SOW1	SOW1	
	DBCDBM1	STC00050	START1		15	EXECUTION			SOW1	SOW1	
	DBCGDIST	STC00054	START1		15	EXECUTION			SOW1	SOW1	
	DBCGLRML	STC00041	START1		15	EXECUTION			SOW1	SOW1	
	DBCGMSTR	STC00029	START1		15	EXECUTION			SOW1	SOW1	
	DBGMGR	STC00026	START1		15	EXECUTION			SOW1	SOW1	
	EQAPROF	STC00077	STCEQA		15	EXECUTION			SOW1	SOW1	
	EQARMTD	STC00078	START1		15	EXECUTION			SOW1	SOW1	
	GITLADR	STC00076	IBMUSER	CC 3840	1	PRINT		9			
	HTTPD1	STC00085	WEBSRV		15	EXECUTION			SOW1	SOW1	
	IBMUSER	TSU00463	IBMUSER		15	EXECUTION			SOW1	SOW1	
	IMSGNJCL	JOB00064	START1	CC 0000	1	PRINT	A	5			
	IMS1CTL	STC00033	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DLI	STC00051	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DRC	STC00052	START1		15	EXECUTION			SOW1	SOW1	
	IMS1ERE	JOB00061	START1	CC 0000	1	PRINT	A	2			
	IMS1ERE	JOB00062	START1	CC 0008	1	PRINT	A	3			

*SDSF

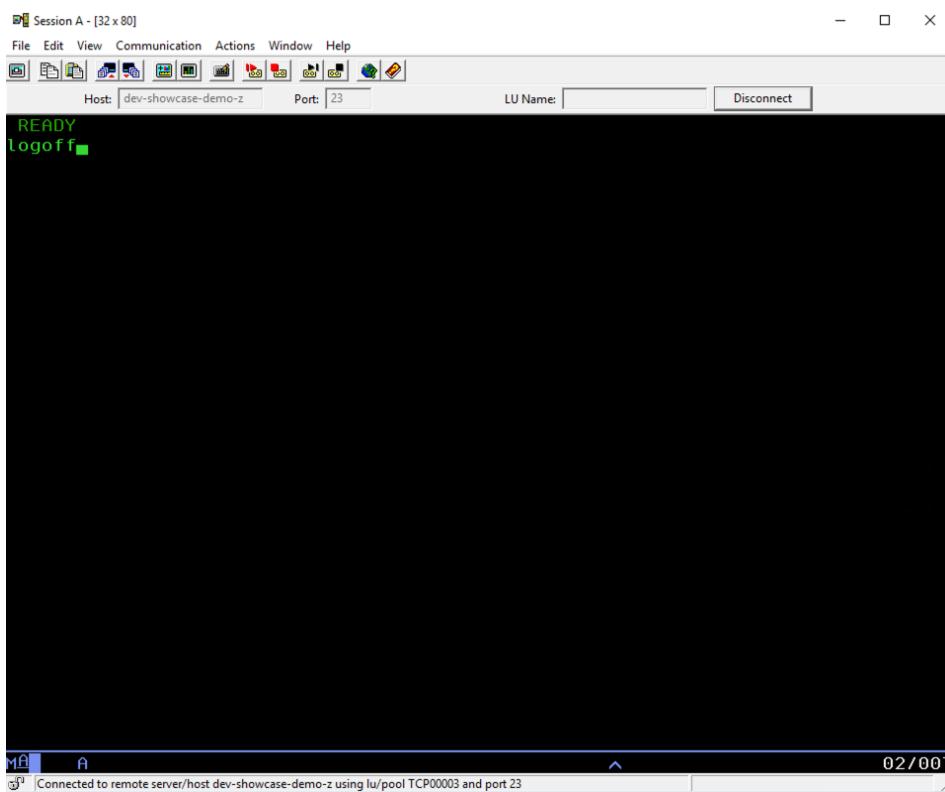
SDSF STATUS DISPLAY ALL CLASSES

CHARS 'BUZ' FOUND

COMMAND INPUT ==> =x_

NP	JOBNAME	JobID	Owner	Max-RC	Prtv	Queue	C	Pos	Saff	ASys	S
	BUZAGNT	STC00465	START1		15	EXECUTION			SOW1	SOW1	
	BUZAGNT	STC00464	START1	CANCELED	1	PRINT		37			
	CFZCIM	STC00027	CFZSRV		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00175	CICSUSER		15	EXECUTION			SOW1	SOW1	
	CICSTS56	STC00028	CICSUSER	CC 0001	1	PRINT		11			
	CNMPROC	STC00068	START1		15	EXECUTION			SOW1	SOW1	
	CSF	STC00016	START1		15	EXECUTION			SOW1	SOW1	
	DBB	STC00074	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBBS	STC00075	IBMUSER		15	EXECUTION			SOW1	SOW1	
	DBCADM	STC00055	START1		15	EXECUTION			SOW1	SOW1	
	DBCDBM1	STC00050	START1		15	EXECUTION			SOW1	SOW1	
	DBCGDIST	STC00054	START1		15	EXECUTION			SOW1	SOW1	
	DBCGLRML	STC00041	START1		15	EXECUTION			SOW1	SOW1	
	DBCGMSTR	STC00029	START1		15	EXECUTION			SOW1	SOW1	
	DBGMGR	STC00026	START1		15	EXECUTION			SOW1	SOW1	
	EQAPROF	STC00077	STCEQA		15	EXECUTION			SOW1	SOW1	
	EQARMTD	STC00078	START1		15	EXECUTION			SOW1	SOW1	
	GITLADR	STC00076	IBMUSER	CC 3840	1	PRINT		9			
	HTTPD1	STC00085	WEBSRV		15	EXECUTION			SOW1	SOW1	
	IBMUSER	TSU00468	IBMUSER		15	EXECUTION			SOW1	SOW1	
	IMSGNJCL	JOB00064	START1	CC 0000	1	PRINT	A	5			
	IMS1CTL	STC00033	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DLI	STC00051	START1		15	EXECUTION			SOW1	SOW1	
	IMS1DRC	STC00052	START1		15	EXECUTION			SOW1	SOW1	
	IMS1ERE	JOB00061	START1	CC 0000	1	PRINT	A	2			
	IMS1ERE	JOB00062	START1	CC 0008	1	PRINT	A	3			

*SDSF



0.1.17 Back to Firefox browser on UCD page, locate **Resource** and check **zos.dev2** is **Online**.

Name	Tags	Inventory	Status	Description
zos.dev2 (View Agent)		version-375 (+ 27 more)	Online	
CBSA-backend (View Component)		version-458 (+ 15 more)		
Genapp-backend (View Component)		20240617-185833 (+ 1 more)		
J05MortgagePOT (View Component)				

Section 1. Get familiar with the application using the 3270 terminal

You will start a 3270 emulation and execute a transaction named J05P to become familiar with the Application that you intend to update.

1.1 Connect to z/OS and emulating a CICS 3270 terminal

1.1.1 Start *IBM Developer for z Systems version 16* if it is not already started

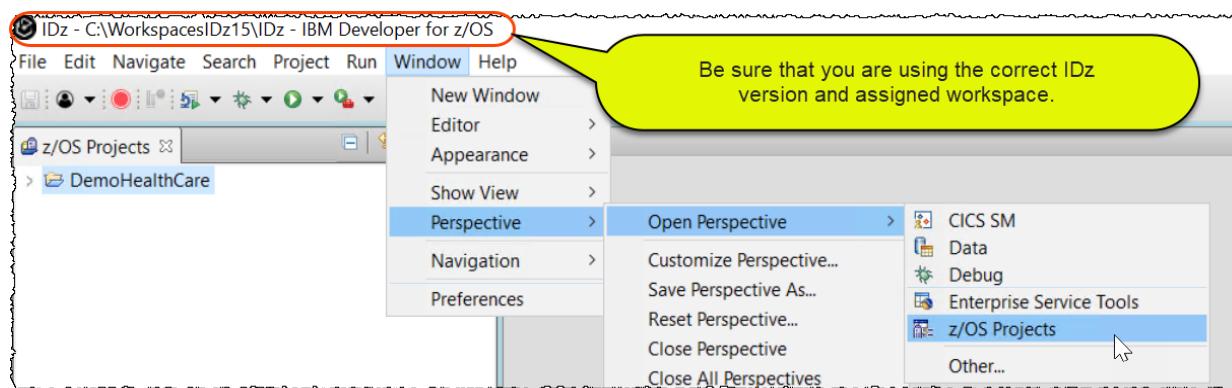
►► Using the desktop double click on **IDz V16** icon.

►► Verify that the message indicates that it is Version **16.0.3**

IMPORTANT -> This icon will start an eclipse 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**

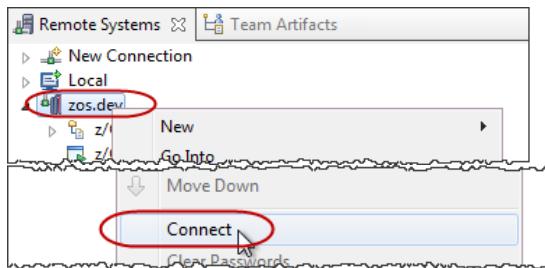


1.1.3 On other labs you used userid **empot01**.

Your new userid now will be **ibmuser**.

You need to disconnect and reconnect to the other userid.

►► If you are already connected, right click on **zos.dev** and select **Disconnect**. 1.1.4 ►► Right click on **zos.dev** and select **Connect**.

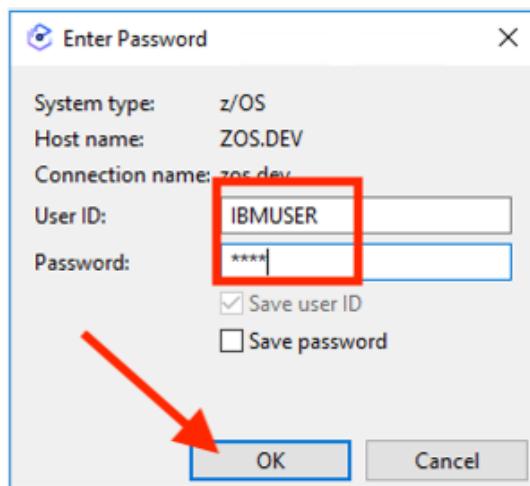


1.1.5 ►► Type **ibmuser** as userid and **sys1** as password.

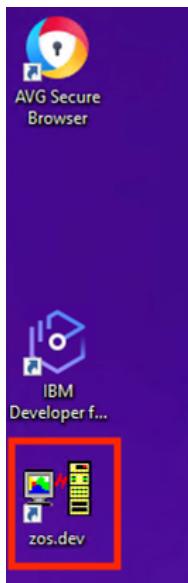
The userid and password can be any case; don't worry about having it in UPPER case.

Click **OK** to connect to z/OS.

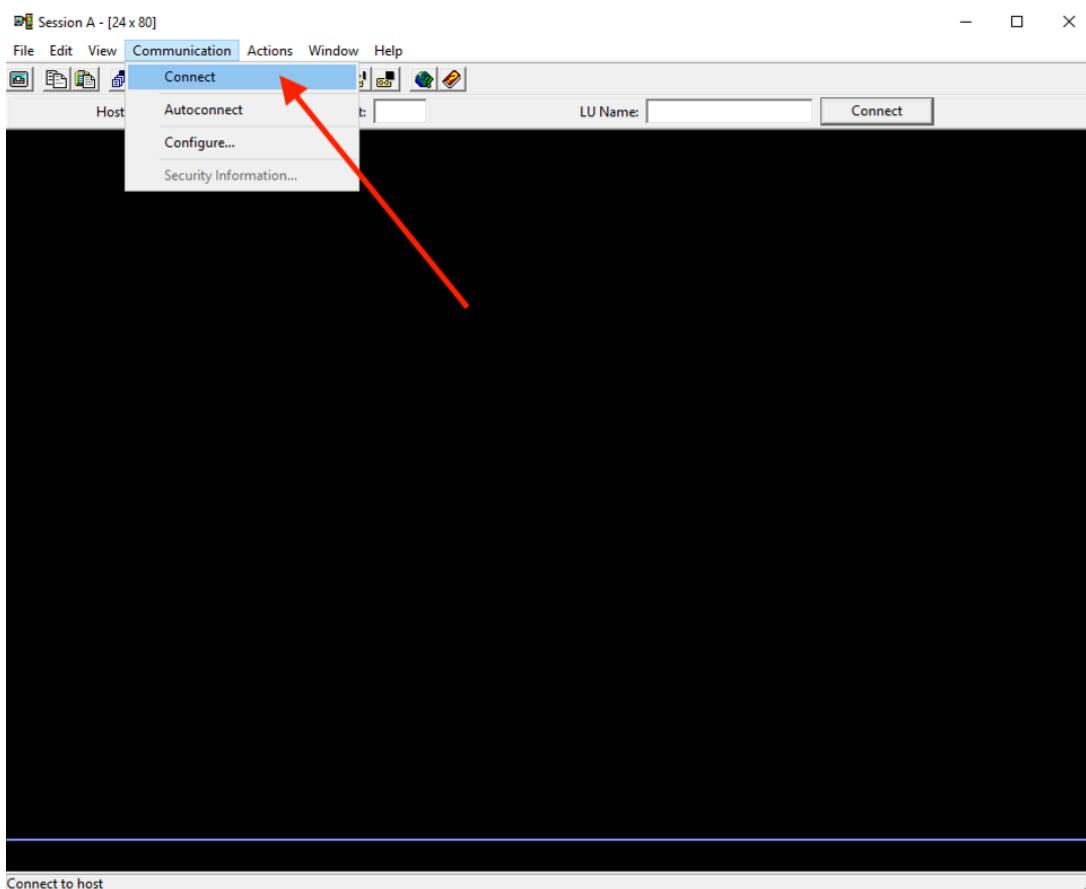
Notice → On this lab your userid will be **ibmuser** (not empot01 used in previous labs)



1.1.6 ►► On Windows Desktop click on **zos.dev** to access z/OS using PCOMM



1.1.7 ► Since you cannot connect. On the top of PCOM window, select Connect and you see VTAM welcome screen.



Session A - [24 x 80]

File Edit View Communication Actions Window Help

Host: zos.dev Port: 23 LU Name: Disconnect

z/OS V2R5 LVL1 PUT2112/RSU2112 IP Address = 10.1.1.1
VTAM Terminal = TCP00024

Application Developer System

```
      // 0000000  SSSSS
      // 00      00 SS
zzzzzz // 00      00 SS
      zz // 00      00 SSSS
      zz // 00      00      SS
      zz // 00      00      SS
zzzzzz // 0000000  SSSS
```

System Customization - ADCD.Z25A.*

==> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
==> Enter L followed by the APPLID
==> Examples: "L TSO", "L CICSTS56", "L CICSTS55", "L IMS15"

MA A 24/002

Connected to remote server/host zos.dev using lu/pool TCP00024 and port 23

1.1.8 ► Type **I cicsts56.** (where "I" is the lower case of letter "L") and press **Enter key**.

The screenshot shows a z/OS terminal window titled "Session A - [24 x 80]". The window includes a menu bar with File, Edit, View, Communication, Actions, Window, and Help. Below the menu is a toolbar with various icons. The main display area shows the following text:

```
z/OS V2R5 LVL1 PUT2112/RSU2112          IP Address = 10.1.1.1
                                         VTAM Terminal = TCP00022

Application Developer System

        // 0000000  SSSSS
        // 00      00 SS
zzzzzz // 00      00 SS
zz  // 00      00 SSSS
zz  // 00      00   SS
zz  // 00      00       SS
zzzzzz // 0000000  SSSS

System Customization - ADCD.Z25A./*

==> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
==> Enter L followed by the APPLID
==> Examples: "L TSO", "L CICSTS56", "L CICSTS55", "L IMS15"

l cicsts56
```

The status bar at the bottom of the window shows "Connected to remote server/host zos.dev using lu/pool TCP00022 and port 23" and the date "24/011".

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

The screenshot shows a z/OS terminal window titled "Session A - [24 x 80]". The window has a menu bar with "File", "Edit", "View", "Communication", "Actions", "Window", and "Help". Below the menu is a toolbar with various icons. The main area displays the following text:

```

File Edit View Communication Actions Window Help
Host: zos.dev Port: 23 LU Name: Disconnect
Signon to CICS APPLID CICSTS56
WELCOME TO CICS TS 5.6

Type your userid and password, then press ENTER:

Userid . . . . ibmuser      Groupid . . .
Password . . . . -
Language . . . . -
New Password . . .

DFHCE3520 Please type your userid.
F3=Exit
MA A 11/030
Connected to remote server/host zos.dev using lu/pool TCP00022 and port 23

```

The message "DFHCE3520 Please type your userid." is displayed in red at the bottom of the screen.

- 1.1.10 The sign-on message is displayed

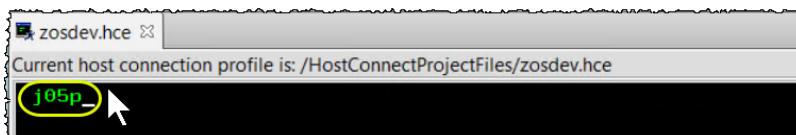
DFHCE3549 Sign-on is complete (Language ENU).

MA a

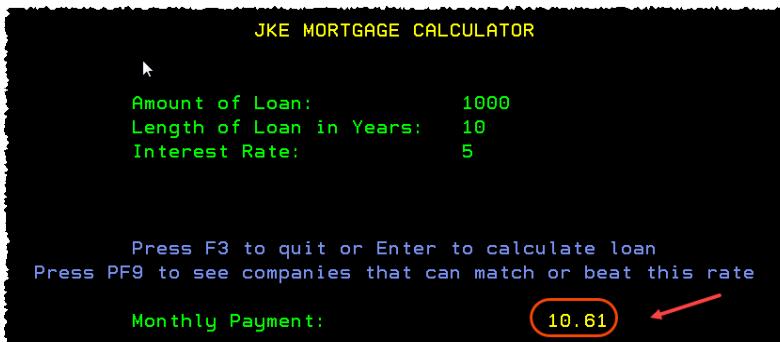
1.2 Run CICS transaction J05P

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

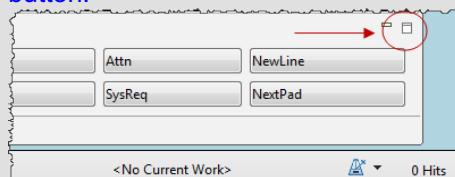
- 1.2.1 ► Type the CICS transaction **j05p** and press the **Enter** key.



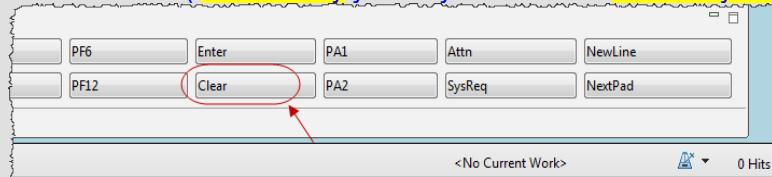
1.2.2 ➡ Press **enter** to see the monthly payment for the default data .



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 ➡ Press **F1** key and verify the message displayed “**INVALID KEY PRESSED**” . Your mission will be modifying the COBOL program that send this message. Your new message must be “**YYYYMMDD - INVALID KEY PRESSED**”.

Mac Users: Press **fn + F1**

Mac Users:

Where YYYYMMDD will be the today's date.

```
JKE MORTGAGE CALCULATOR

Amount of Loan:      1000
Length of Loan in Years:  10
Interest Rate:        5

Press F3 to quit or Enter to calculate loan
Press PF9 to see companies that can match or beat this rate

Monthly Payment:      10.61

INVALID KEY PRESSED.
```

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

```
zosdev.hce X
Current host connection profile is: /HostConnectProjectFiles/zosdev.hce
END OF TRANSACTION - THANK YOU
```

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 **J05P** and verified a simple interaction with the Mortgage application. The objective here was to show the code that you will update.

Section 2. Load the source code from Git to the local IDz workspace

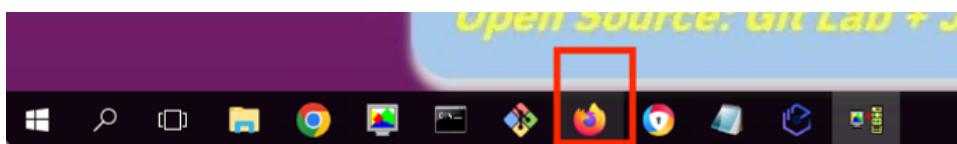
You will load the COBOL code that is stored on Linux to your Windows client to be modified.

2.1 Cloning the Git Repository

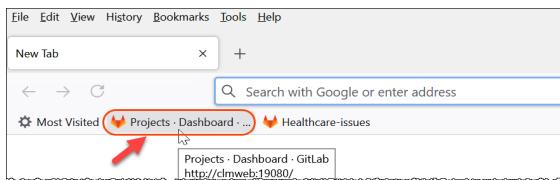
The Mortgage Application used in this lab has its source code in a *Git Repository* that is on Linux system. You must connect to the Git Repository and clone the Git project into the IDz. This brings the source code into your IDz workspace for edits and build. *GitLab* is open source and for that reason we are using *GitLab* in this lab. This lab would work also if using GitHub or Bitbucket.

2.1.1 Before cloning the Git repository, you should visualize the code stored at GitLab.

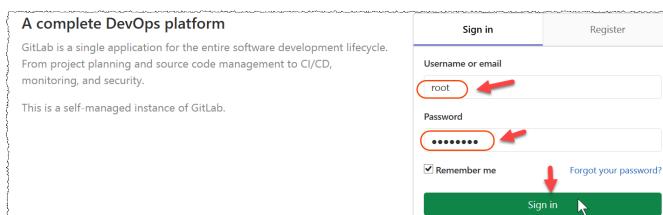
Start a browser clicking in the icon in the bottom of your screen



2.1.2 ► Click on this bookmark below to start **GitLab**. The URL used is http://dev-showcase-demo-x86:8282/users/sign_in



► If ask for credentials use **root** and password: **passw0rd**



2.1.3 ► Scroll down and click on the hyperlink below to see the **j05mortgageCICS** repository.

Owner	Name	Stars	Issues	Merge Requests	Last Commit
C	IBMZSoftware / CBSA_OpenBanking	0	0	0	Updated 1 year ago
C	IBMZSoftware / CICS Bank Sample Application	0	0	0	Updated 10 months ago
D	IBMZSoftware / db2z-node.js-app	0	0	1	Updated 1 year ago
D	IBMZSoftware / db2z-node.js-app-nopackages	0	0	0	Updated 1 year ago
D	IBMZSoftware / dbb	0	0	0	Updated 3 weeks ago
D	IBMZSoftware / dbb-pipeline	0	0	0	Updated 2 weeks ago
D	IBMZSoftware / dbb-zappbuild	0	0	0	Updated 1 week ago
J	root/j05mortgagecics	0	0	0	Updated 5 minutes ago

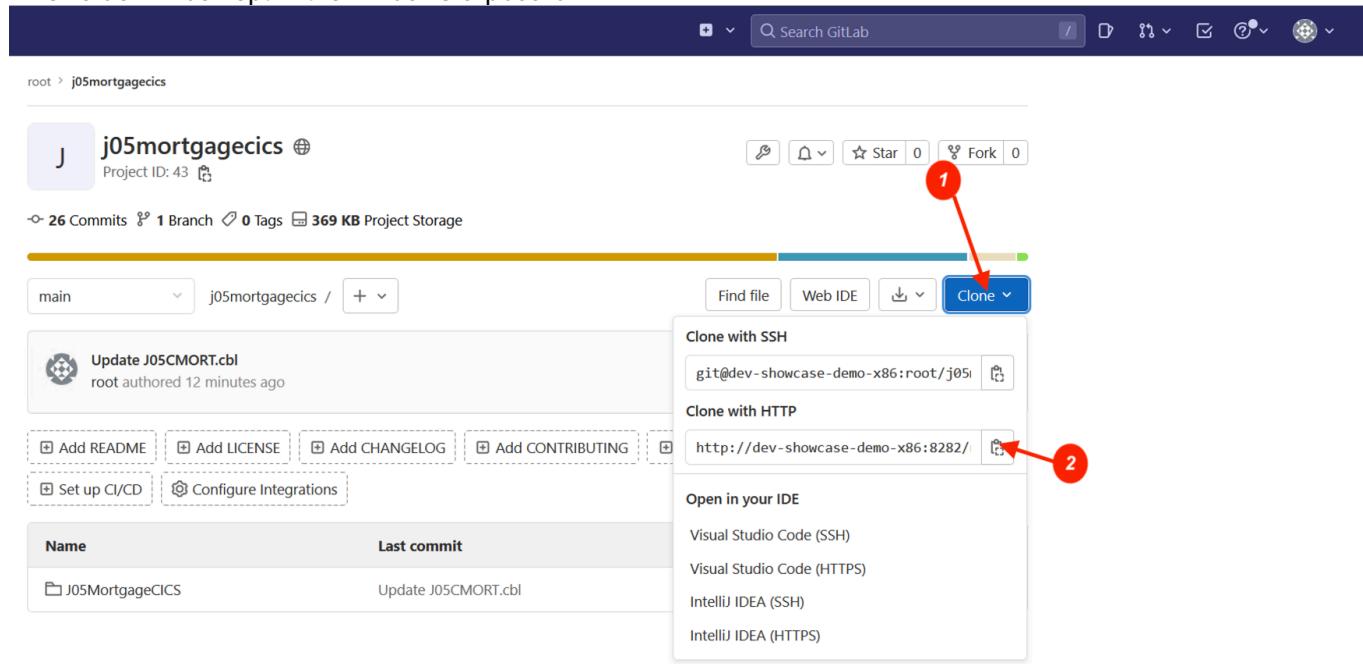
2.1.4 ► You can see the **J05MortgageCICS** repository.

You may browse the content if you want. The source code to be used on this lab is there

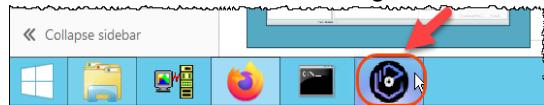
2.1.5 In order to clone it at your window desktop you could use **HTTP** or **SSH**.

Since **SSH** is blocked in our environment we need to use **HTTP**.

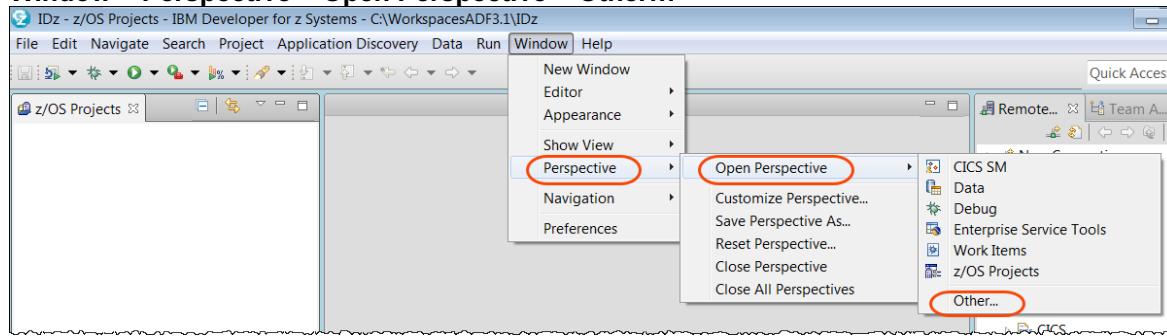
- ▶ 1 Click on **Clone** (the blue button) and 2 in the icon to copy the URL
This value will be kept in the windows clipboard.



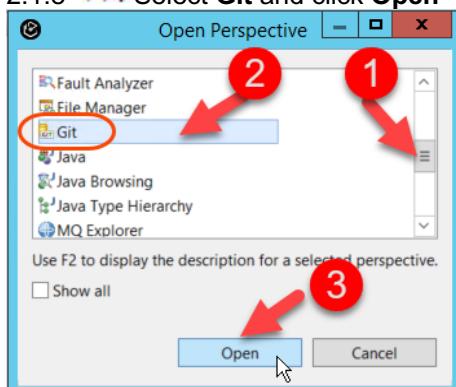
2.1.6 ▶ Go back to IDz using the icon that is on the base of your screen:



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



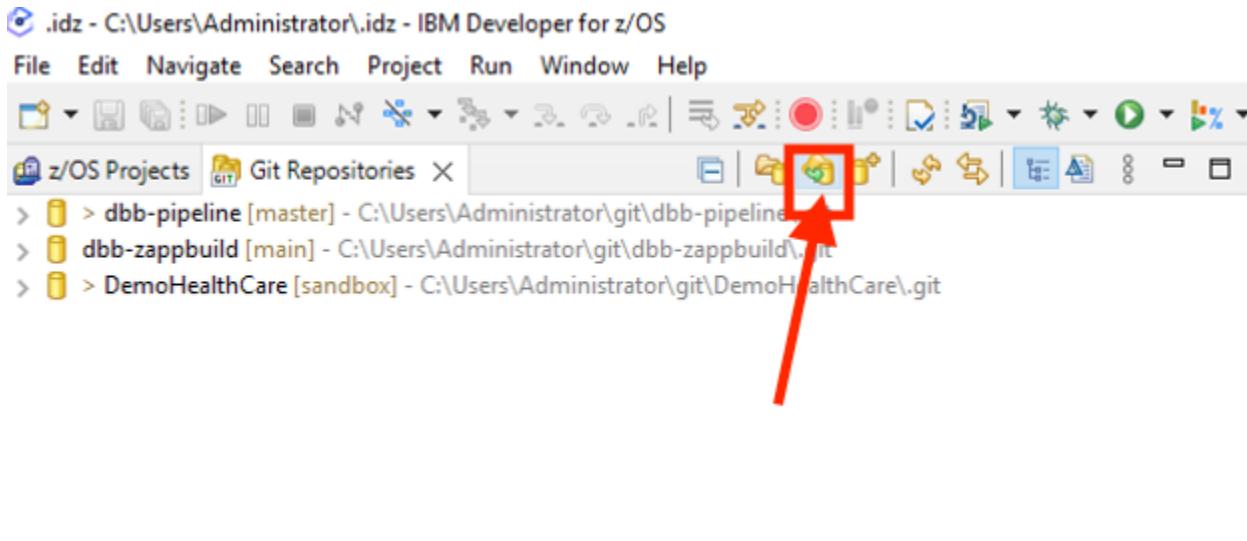
2.1.8 ►| Select **Git** and click **Open**



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

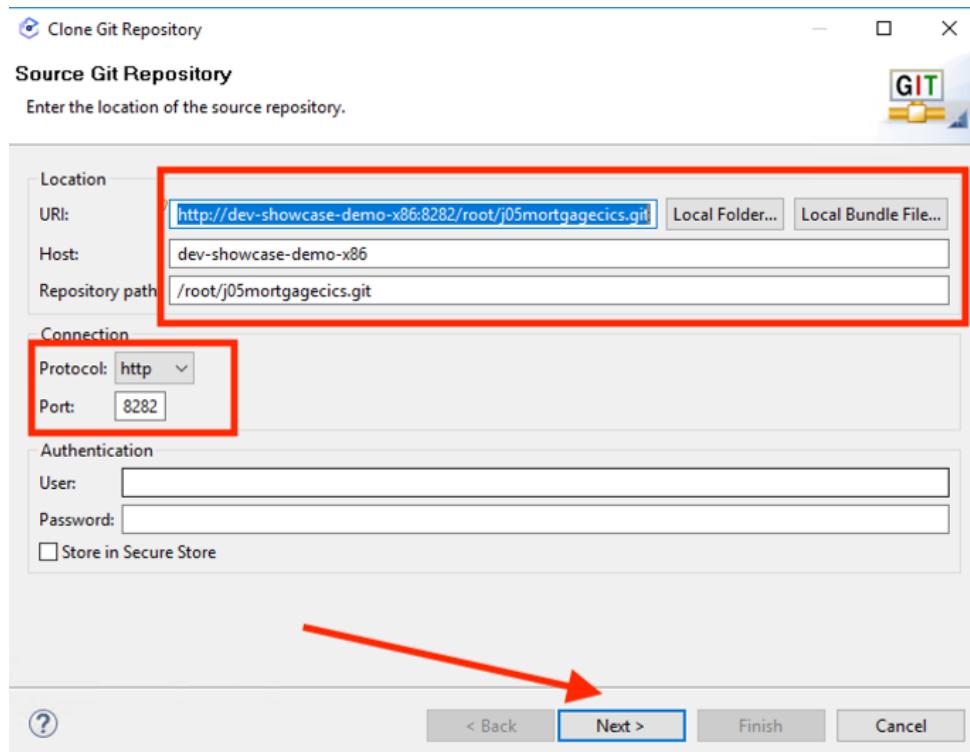


You might see Git Repositories with a list of projects. ►| Click on the Icon **Clone a Git Repository ...**



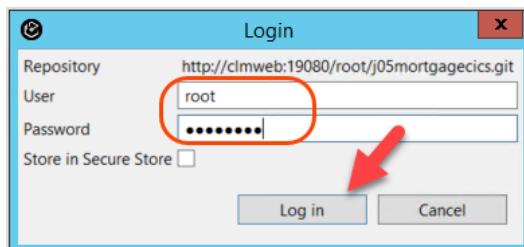
2.1.10 ►| The values copied from the web page (copy URL) will be shown in the Clone Git Repository.
Tip: In case you don't see it, got back to the page and copy it again (steps 2.1.1-2.1.5 above).

►| Click **Next**



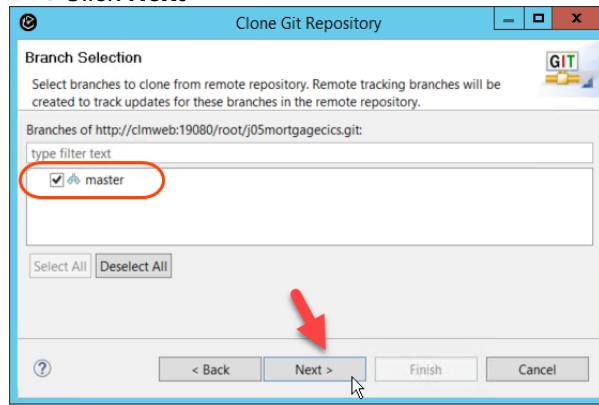
2.1.11 You might need to login using the credentials are a use user: **root** and password **passw0rd (0 is zero)** . Otherwise skip to the next step.

▶| Click Log in



2.1.12 We have only one branch named master.

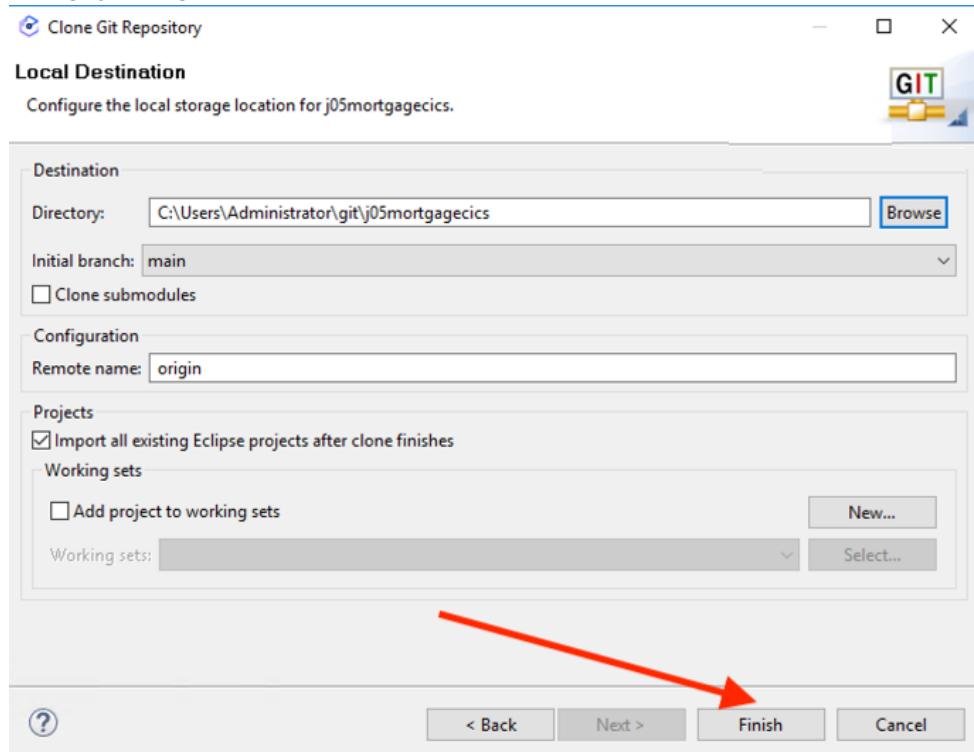
▶▶ Click Next



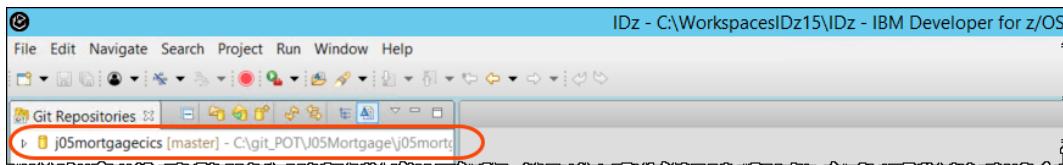
2.1.13 The *Directory* now should point to C:\User\Administrator\.git\j05mortgagecics

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

▶▶ Click Finish



2.1.14 The Mortgage 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:

The screenshot shows the expanded Git Repositories view for the 'j05mortgagecics' repository. The expanded tree structure includes:

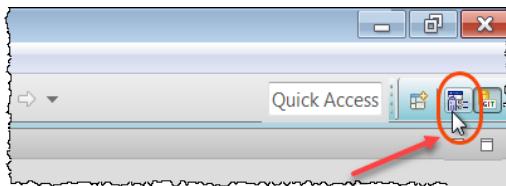
- Branches** (Local):
 - main
 - origin
- Tags**
- References** (HEAD [refs/heads/main] 265ac47 Update J05CMORT.cbl)
- Remotes** (origin)
 - http://dev-showcase-demo-x06:8282/root/j05mortgagecics.git
 - http://dev-showcase-demo-x06:8282/root/j05mortgagecics.git
- Working Tree** - C:\Users\Administrator\git\j05mortgagecics
 - .git
 - J05MortgageCICS
 - settings
 - application-conf
 - bms
 - build
 - cobol
 - cobol_cics
 - copybook
 - link
 - .gitattributes
 - .project

2.2 Verify the code cloned using z/OS projects perspective

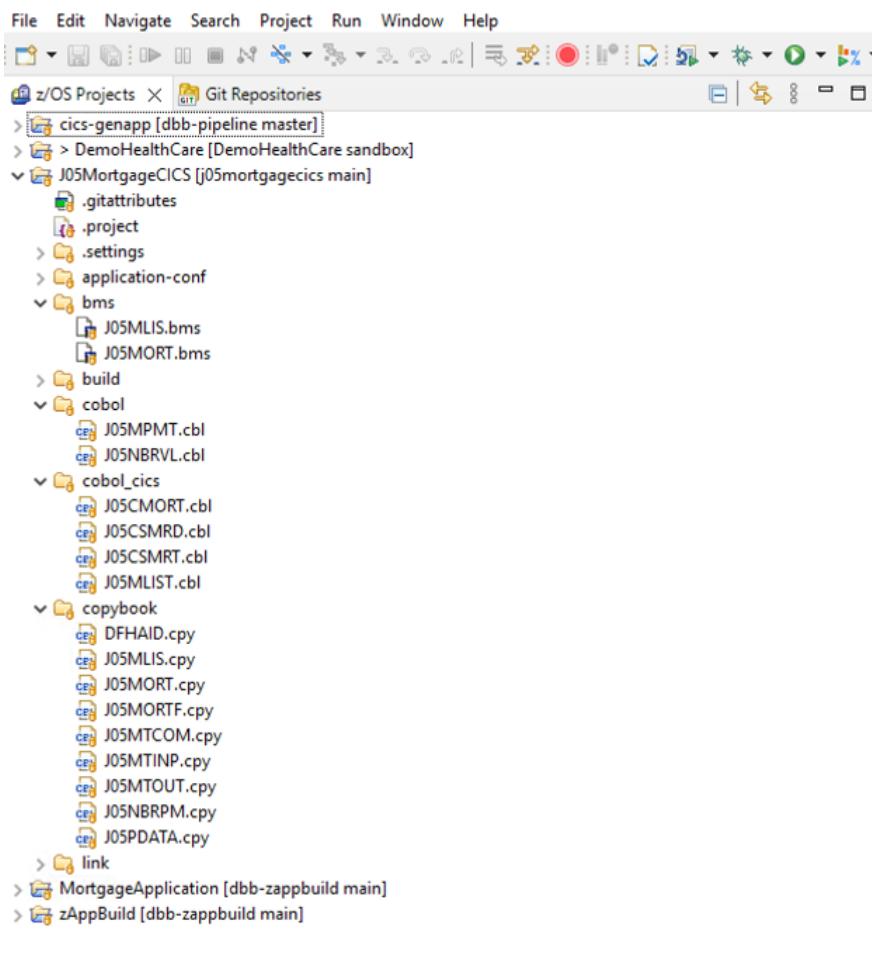
The z/OS projects perspective is the IDz perspective that developers use to work with the source code.

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 **J05MortgageCICS** 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.



Section 3. Modify the COBOL code using IDz.

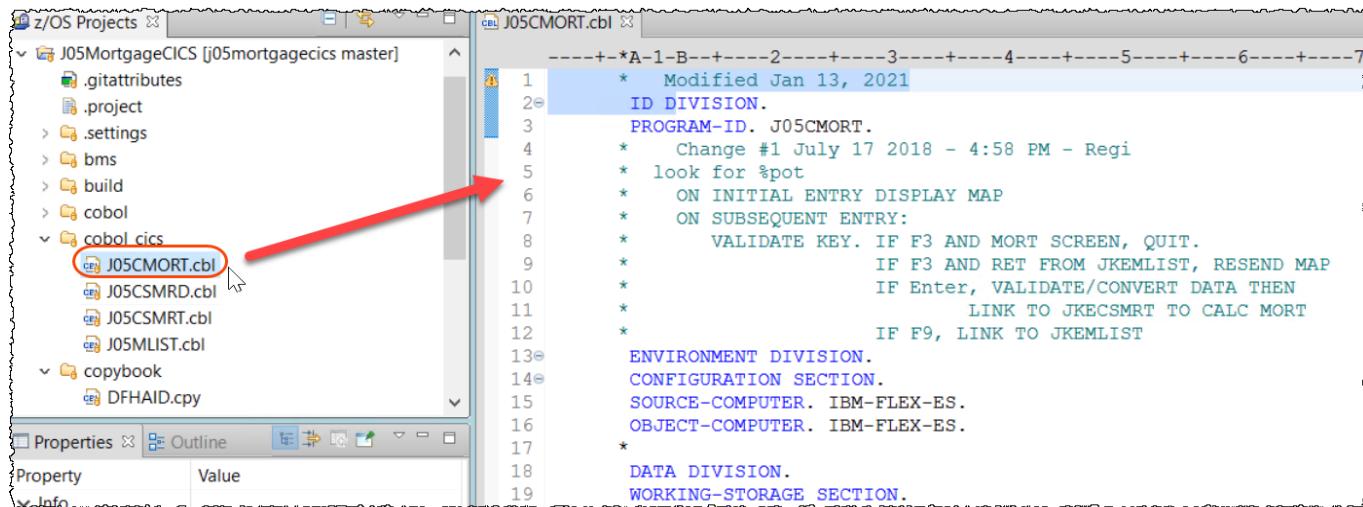
Using IDz you will modify the COBOL to have a different message in a CICS dialog.
You will replace the message "**INVALID KEY PRESSED**" when pressing F1 by another message:
"YYYYMMDD - INVALID KEY PRESSED".

3.1 Edit and modify the code that send the message

The COBOL code to be modified is the program **J05CMORT** that is under the folder **cobol_cics**.

3.1.1 ► Using z/OS Projects view double click **J05CMORT** under **cobol_cics**

This is the program that you will update



The screenshot shows the z/OS Projects interface. On the left, the project structure is displayed under 'J05MortgageCICS [j05mortgagecics master]'. The 'cobol_cics' folder contains several files: .gitattributes, .project, settings, bms, build, cobol, and J05CMORT.cbl. A red arrow points from the text above to the 'J05CMORT.cbl' file, which is highlighted with a red oval. On the right, the content of the J05CMORT.cbl file is shown in a code editor window. The code includes comments about modifications and various sections like ID DIVISION, PROGRAM-ID, and ENVIRONMENT DIVISION.

```

-----+--A-1-B---+--2---+--3---+--4---+--5---+--6---+--7
      * Modified Jan 13, 2021
      ID DIVISION.
      PROGRAM-ID. J05CMORT.
      * Change #1 July 17 2018 - 4:58 PM - Regi
      * look for %pot
      * ON INITIAL ENTRY DISPLAY MAP
      * ON SUBSEQUENT ENTRY:
      *   VALIDATE KEY. IF F3 AND MORT SCREEN, QUIT.
      *           IF F3 AND RET FROM JKEMLIST, RESEND MAP
      *           IF Enter, VALIDATE/CONVERT DATA THEN
      *               LINK TO JKECMSRT TO CALC MORT
      *           IF F9, LINK TO JKEMLIST
      ENVIRONMENT DIVISION.
      CONFIGURATION SECTION.
      SOURCE-COMPUTER. IBM-FLEX-ES.
      OBJECT-COMPUTER. IBM-FLEX-ES.
      *
      DATA DIVISION.
      WORKING-STORAGE SECTION.
  
```

3.1.2 ► Use Ctrl + f to find the "%pot" on line 137 (second Find hit). You will modify the line 141..

```

-----+*A-1-B-----2-----3-----4-----5-----6-----7-----|+-
131 * Process Enter Key to calculate the loan amount
132     PERFORM A100-PROCESS-MAP
133     END-IF
134     WHEN OTHER
135 * Invalid key
136     MOVE LOW-VALUES TO JKEMENU
137 * %pot - below is the message to be changed -
138 * Replace YYYYMMDD Example: 20180712      *
139 *
140     MOVE 'INVALID KEY PRESSED.' TO MSGERRO
141     MOVE 'INVALID KEY PRESSED.' TO MSGERRO
142     SET SEND-DATAONLY TO TRUE
143     PERFORM A300-SEND-MAP
144     END-EVALUATE
145     EXEC CICS
146         RETURN TRANSID(EIBTRNID)
147         COMMAREA(W-COMMUNICATION-AREA)
148         LENGTH(W-COMAREA-LENGTH)
149     END-EXEC.
150

```

3.1.3 ► Close the find dialog and modify the line 141

From

MOVE 'INVALID KEY PRESSED.' TO MSGERRO

To

MOVE 'YYYYMMDD - INVALID KEY PRESSED.' TO MSGERRO

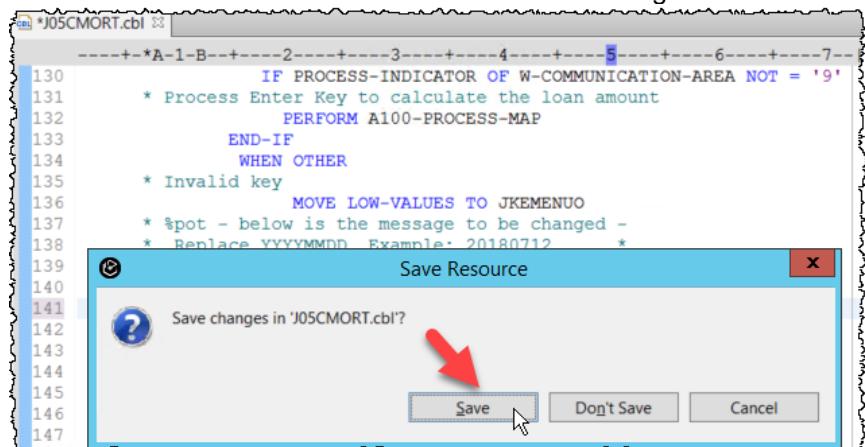
Where YYYYMMDD is the today's date.. Example, I changed to 20190529..

```

-----+*A-1-B-----2-----3-----4-----5-----6-----7-----|+-
129
130     WHEN EIBAID = DFHENTER
131         IF PROCESS-INDICATOR OF W-COMMUNICATION-AREA NOT = '9'
132             * Process Enter Key to calculate the loan amount
133             PERFORM A100-PROCESS-MAP
134             END-IF
135             WHEN OTHER
136             * Invalid key
137             MOVE LOW-VALUES TO JKEMENU
138             * %pot - below is the message to be changed -
139             * Replace YYYYMMDD Example: 20180712      *
140             *
141             MOVE '20190529 - INVALID KEY PRESSED.' TO MSGERRO
142             MOVE '20190529 - INVALID KEY PRESSED.' TO MSGERRO
143             SET SEND-DATAONLY TO TRUE
144             PERFORM A300-SEND-MAP
145             END-EVALUATE
146             EXEC CICS
147                 RETURN TRANSID(EIBTRNID)
148                 COMMAREA(W-COMMUNICATION-AREA)
149             END-EXEC.
150

```

- 3.1.4 ► Use **Ctrl + Shift + F4** to close all code being edited. Click **Save** to save the modified code.

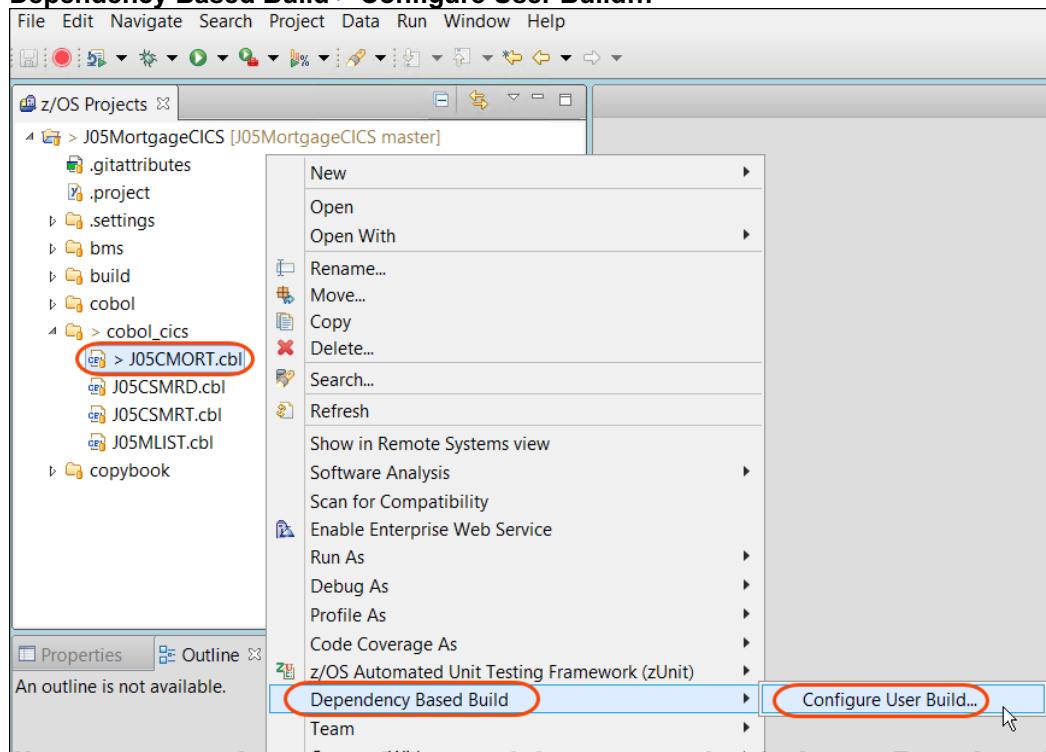


Section 4. Use IDz DBB User Build to compile/bind and perform personal tests.

You will compile and link the modified code using the DBB User Build function. When complete you will run the code using CICS for a personal test and verify that the change is correctly implemented.

4.1 Using Dependency Based Building option

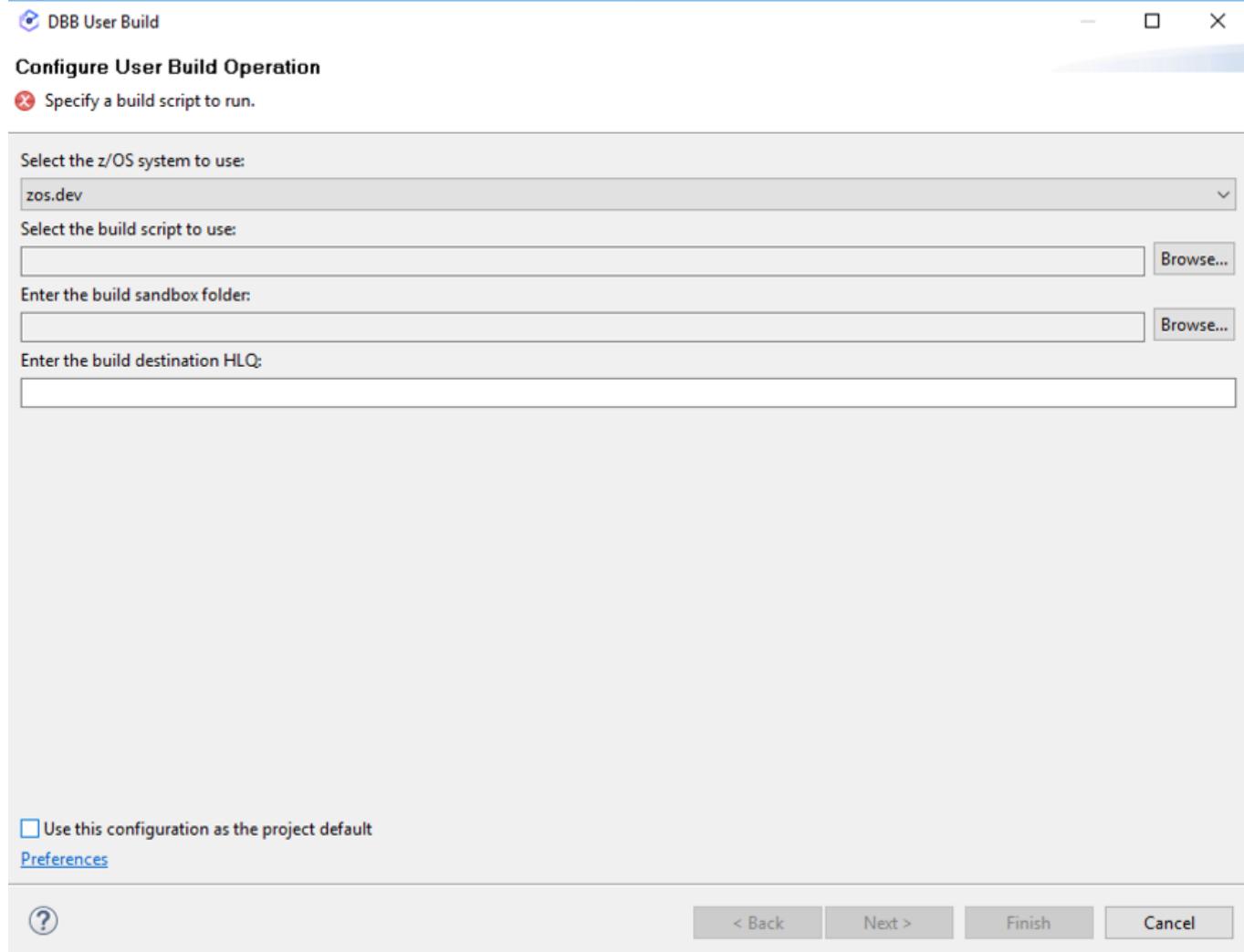
- 4.1.1 ► Under **z/OS Projects** view right click **J05CMORT** and select **Dependency Based Build > Configure User Build...**



4.1.2  Be sure that those values are already assigned for the build, otherwise use the **Browse** button and select the values as below

z/OS system: **zos.dev**

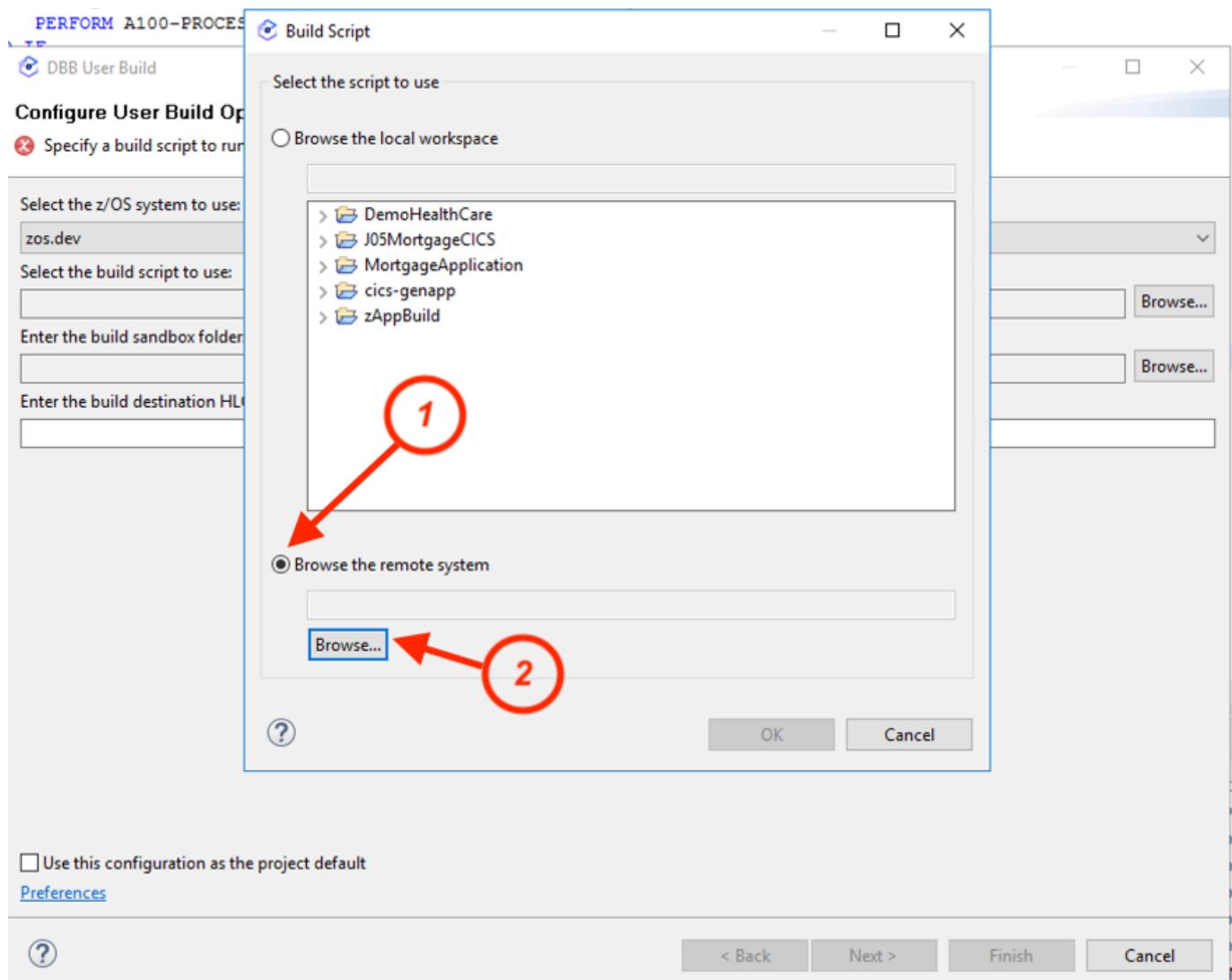
```
END-EXEC
IEN EIBAID = DFHENTER
IF PROCESS-INDICATOR OF W-COMMUNICATION-AREA NOT = '9'
  Inter Key to calculate the loan amount
    PERFORM A100-PROCESS-MAP
  
```



Build script : We will use the latest version of build.groovy (use **Browse the local workspace**)

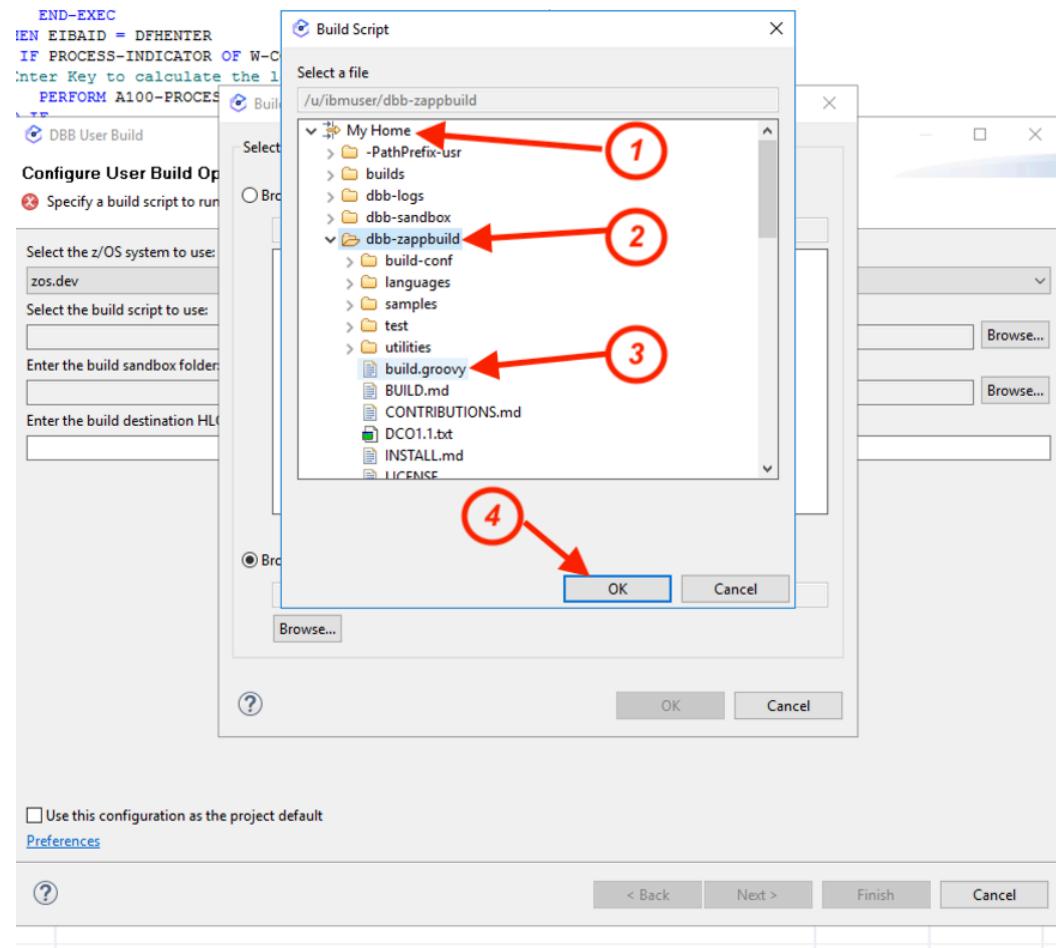
Click on Browse button on DBB User Build Window

1. On Build Script Window. Check **Browse the remote system**
2. Click **Browse** Button

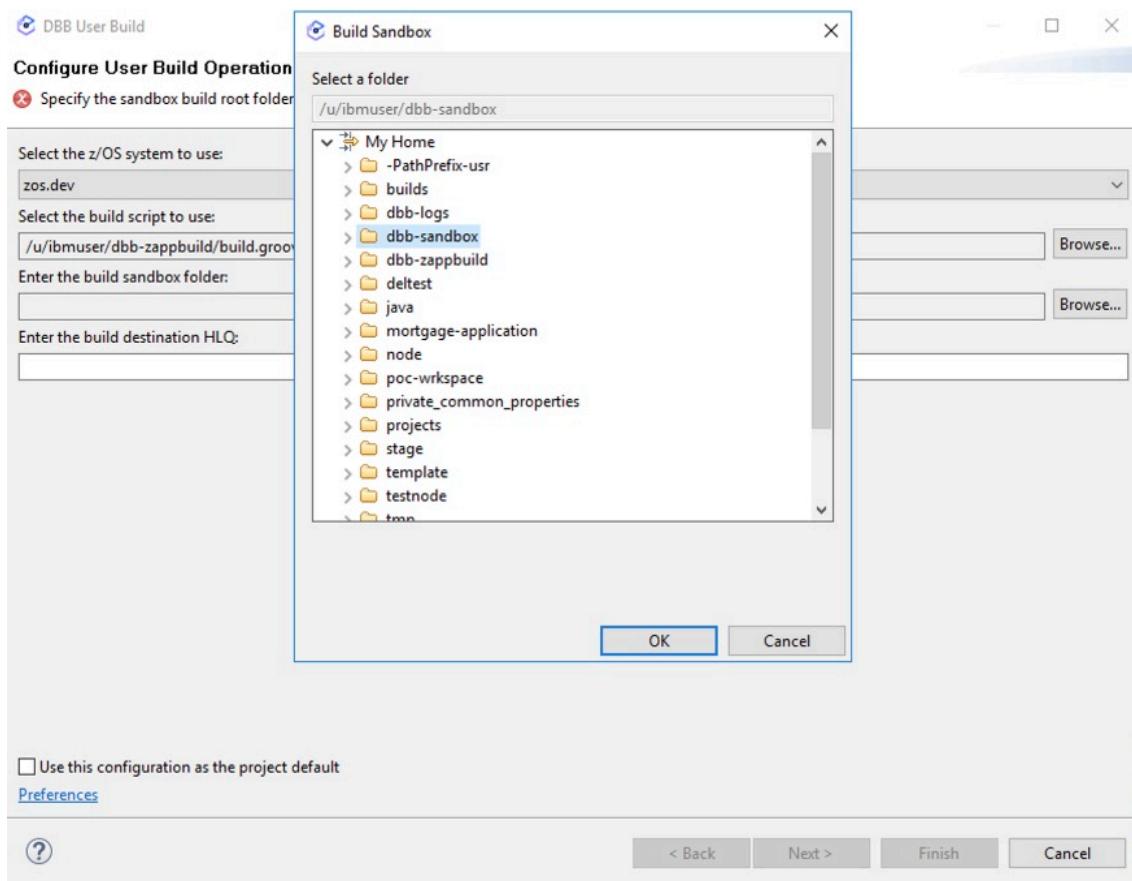


On Build Script Window:

1. Select **My Home** folder
2. Select **dbb-zappbuild** folder
3. Click **build.groovy**
4. Click **Ok**

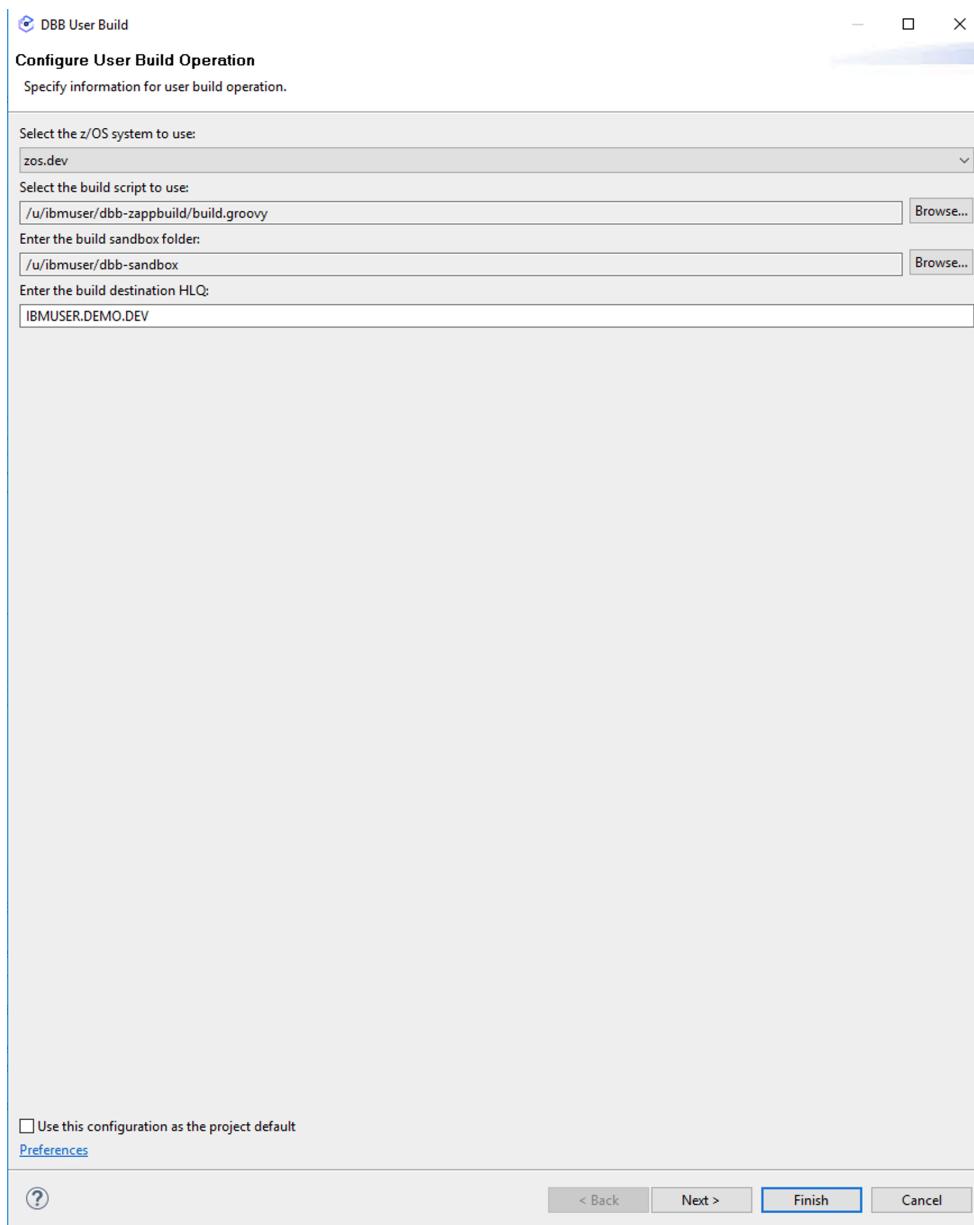


Build sandbox : Select the sandbox folder on remote server (Use **Browse**, expand **MyHome**, click **dbb-sandbox**)

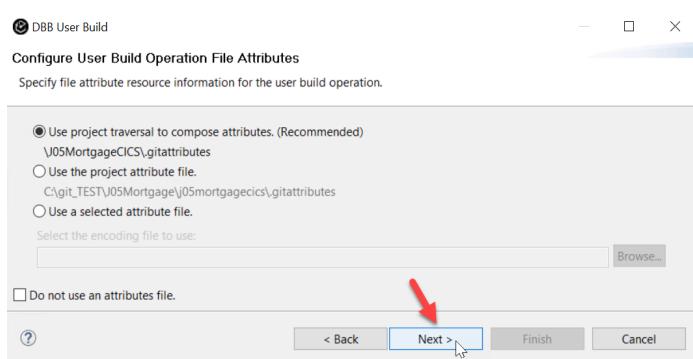


Build destination HLO: IBMUSER.DEMO.DEV (must type)

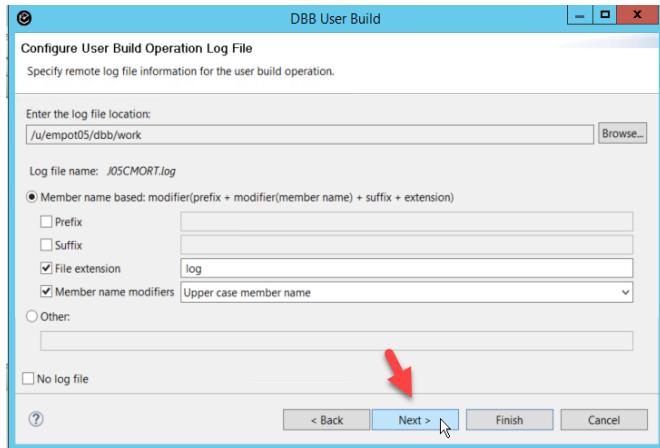
▶| Click **Next**



4.1.3 ► 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.

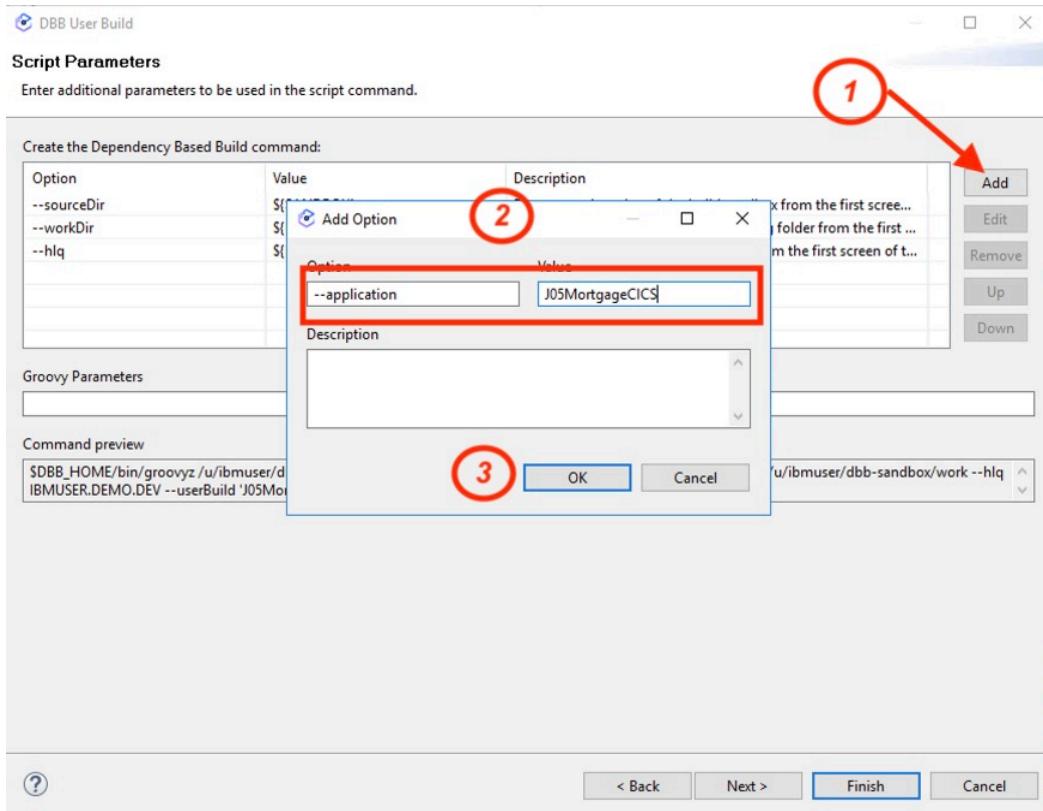


4.1.4 ► On *Configure User Build Operation Log File* click **Next**
 You will use the default values.



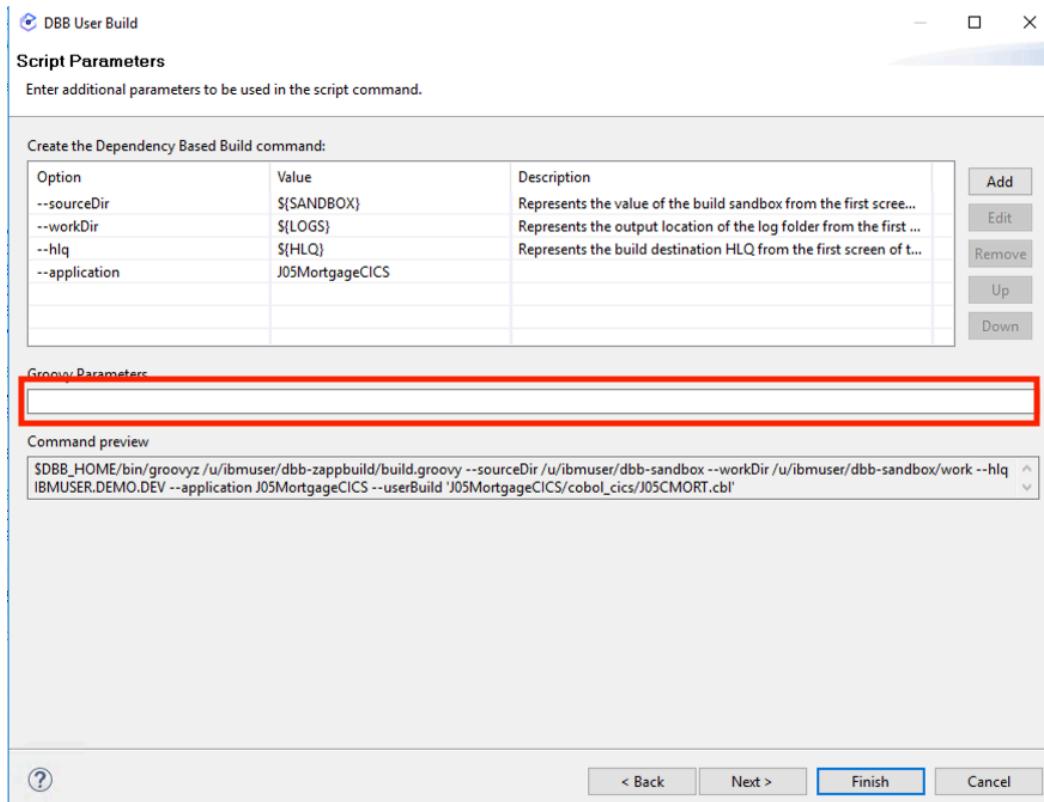
4.1.5 ► On Script parameters:

1. Click Add Button
2. Enter on Option : **--application**
3. Enter Value: **J05MortgageCICS**



Be sure that the field **Groovy Parameters** is empty and click **Next**

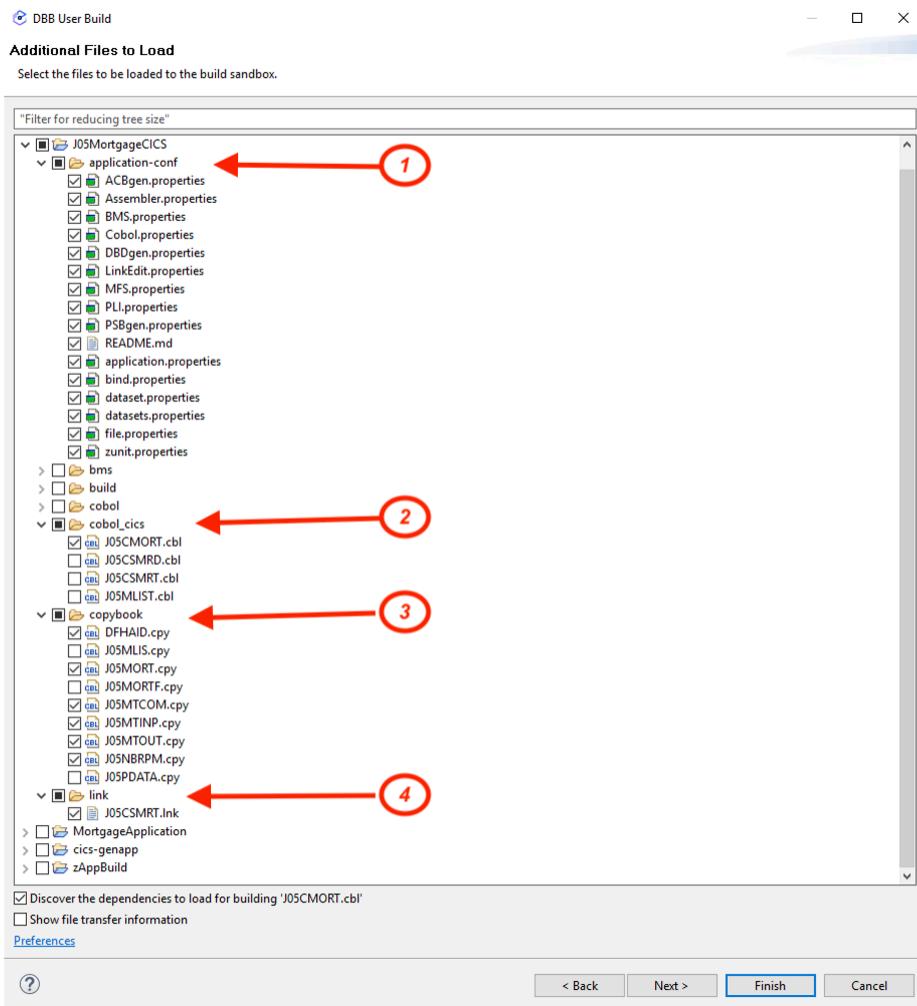
Notice the preview of the commands that will be execute at the z/OS by the DBB toolkit.



4.1.6 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 folders **J05Mortgage CICS** and the **cobol_cics** and **copybook** directories

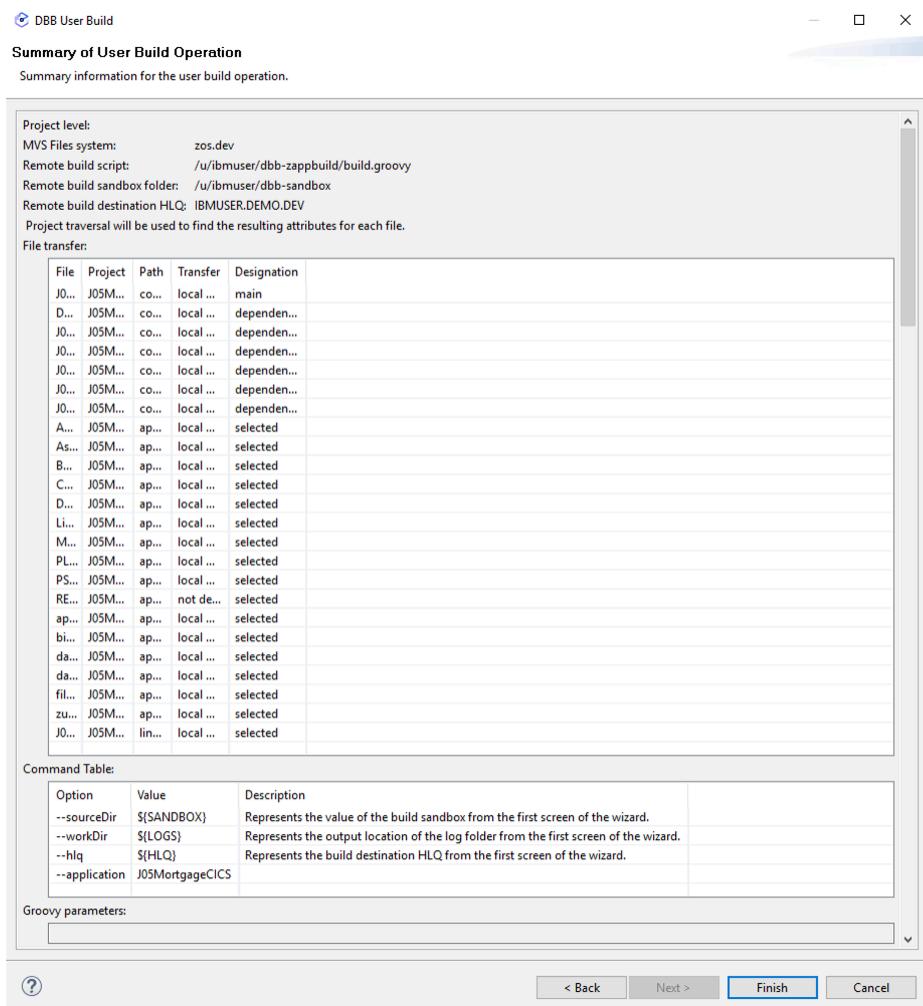
1. Check **application-conf** on J05MortgageCICS folder
2. Check J05CMORT.cbl on **cobol_cics** folder
3. Verify **copybook** folder (DBB checks all copybooks necessities)
4. Check J05CSMRT.lnk on **link** folder



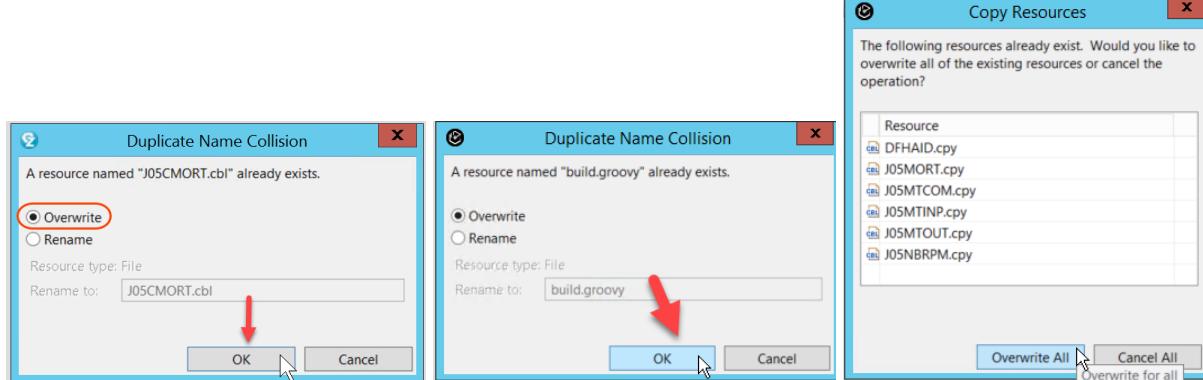
▶ Since you will also need the related copybooks for a clean compile be sure that “**Discover the dependencies to load for building J05CMORT.cbl**“ is selected (automatically checked).

▶ Click **Next**

4.1.7 ▶ Verify the summary of the User Build Operation and click **Finish**



4.1.8 ► If the dialogs below pops up select Overwrite and click OK
Also select Overwrite All for the copybooks



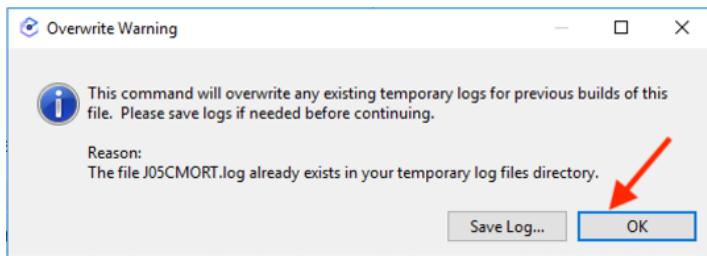
The DBB User build will be running and the messages will be shown at the Console view..

► Click on **Console** tab on lower right corner The execution will start, and this will take a while .

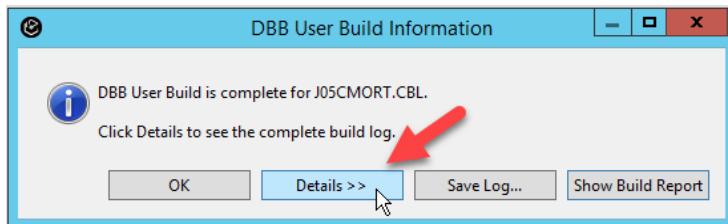
4.1.9 When complete you will have the message below:

On Overwriting Warning.

Click **OK**



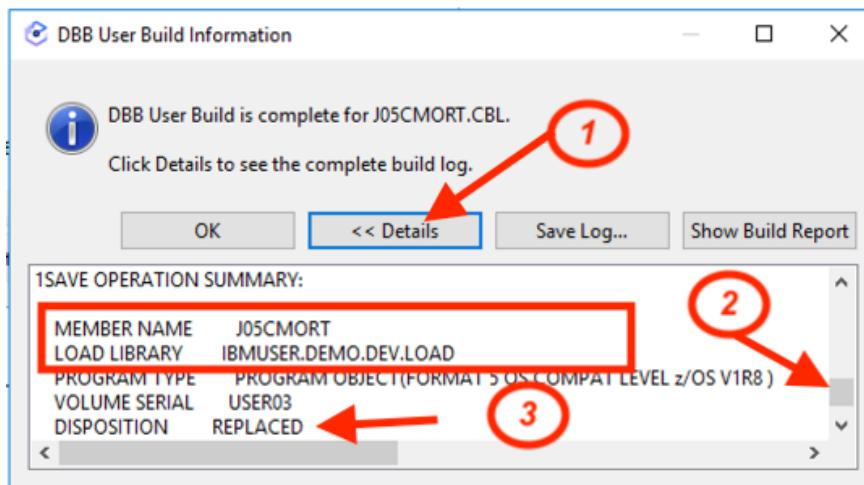
▶| Click Details



__ 4.2 Verify the DBB User Build results

4.2.1 ▶| After clicking **Details >>** scroll down and verify that a new load module name **J05CMORT** was replaced on the dataset **IBMUSER.DEMO.DEV.LOAD**

▶| Click **OK** to close this dialog.



4.2.2 ► Verify at the Condole view that the Build State should show a **CLEAN** build.

```
** Writing build report to /u/ibmuser/dbb-sandbox/work/BuildReport.html
* Build ended at Tue Jun 18 08:41:42 CDT 2024
* Build State : CLEAN
** total files processed : 1
** Total build time : 6.890 seconds

** Build finished

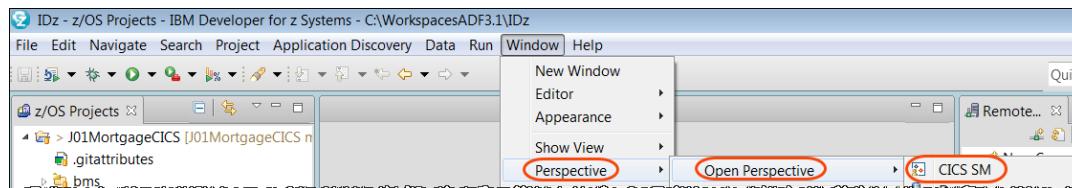
/u/ibmuser/dbb-sandbox>
```

4.3 Issuing a CICS New copy

4.3.1 At this point you may test your new load module. But since this is a CICS system you will need to make a **CICS NEWCOPY**. Let's use IDz and CICS Explorer to do that.

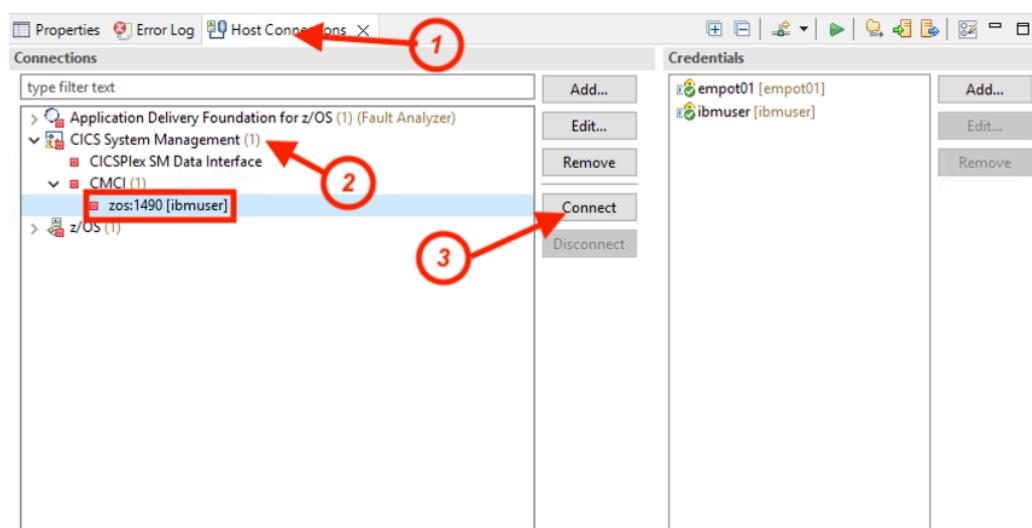
Notice that the new copy could be done by the Groovy scripts, but I want to show CICS Explorer nice capabilities here..

► Open the CICS SM perspective selecting **Window > Perspective > Open Perspective > CICS SM**

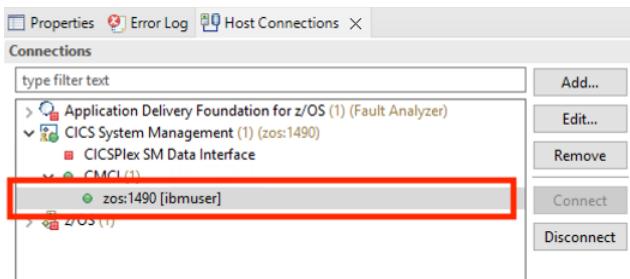


4.3.2 ► Click on tab **Host Connections** on the bottom, expand the **CICS System Management, CMCI**, select **CICSTS56** and click on **Connect**

Note: If it ask you for userid and password then use **ibmuser** and **sys1** for username and password



4.3.3 The red dot will turn green once is connected to CICS



4.3.4 ► On top left expand **CICSTS56** and click on **CICSTS56 (CICSTS56)**.

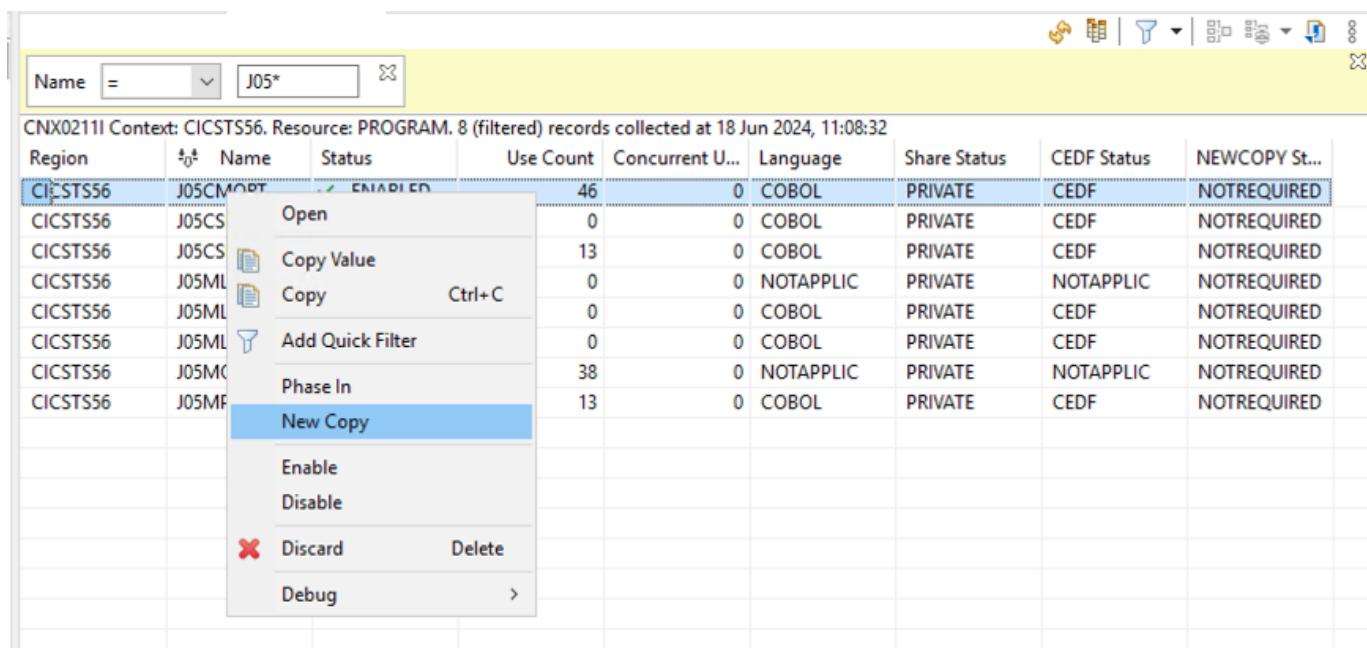
Region	Name	Status	Use Count	Concurrent U...	Language	Share Status	CEDF Status	NEWCOPY St...
CICSTS56	AZUCMGT	ENABLED	0	0	C	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	AZUCREST	ENABLED	0	0	C	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF410	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF420	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF510	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF520	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF530	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF540	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF550	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF560	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCF610	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCICSA	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCICSE	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCICSN	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCICSS	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCICST	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCICSW	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	BZUCICSX	ENABLED	0	0	NOTDEFINED	PRIVATE	CEDF	NOTREQUIRED

► Click on the **Programs** tab.

Since the filter is already made for “**Name = J05***” and press **ENTER** you should see the list below:

Region	Name	Status	Use Count	Concurrent U...	Language	Share Status	CEDF Stat...
CICSTS56	J05CMORT	ENABLED	46	0	COBOL	PRIVATE	CEDF
CICSTS56	J05CSMRD	ENABLED	0	0	COBOL	PRIVATE	CEDF
CICSTS56	J05CSMRT	ENABLED	13	0	COBOL	PRIVATE	CEDF
CICSTS56	J05MLIS	ENABLED	0	0	NOTAPPLIC	PRIVATE	NOTAPPLIC
CICSTS56	J05MLISD	ENABLED	0	0	COBOL	PRIVATE	CEDF
CICSTS56	J05MLIST	ENABLED	0	0	COBOL	PRIVATE	CEDF
CICSTS56	J05MORT	ENABLED	38	0	NOTAPPLIC	PRIVATE	NOTAPPLIC
CICSTS56	J05MPMT	ENABLED	13	0	COBOL	PRIVATE	CEDF

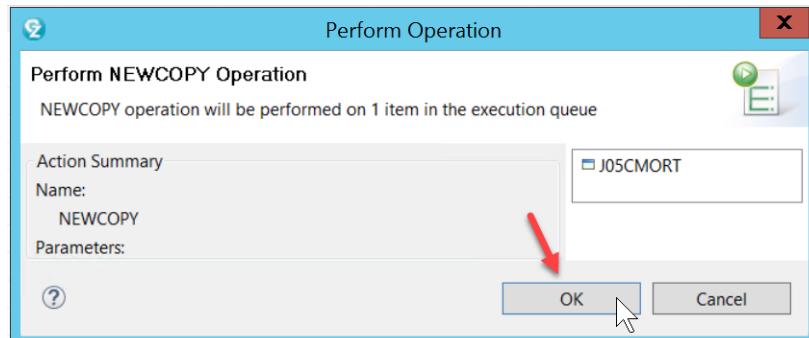
4.3.5 ► Right click on **J05CMORT** and select **New Copy** as the example below



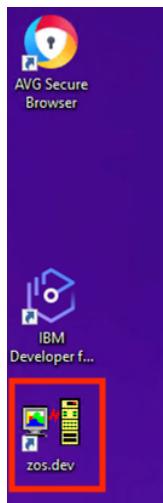
The screenshot shows a table of resources. A context menu is open over the row for item **J05CMORT**. The menu items are: Open, Copy Value, Copy (with a keyboard shortcut Ctrl+C), Add Quick Filter, Phase In, New Copy (which is highlighted with a blue selection bar), Enable, Disable, Discard, Delete, and Debug.

Region	Name	Status	Use Count	Concurrent U...	Language	Share Status	CEDF Status	NEWCOPY St...
CICSTS56	J05CMORT	ENABLED	46	0	COBOL	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	J05CS	Open	0	0	COBOL	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	J05CS	Copy Value	13	0	COBOL	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	J05ML	Copy	0	0	NOTAPPLIC	PRIVATE	NOTAPPLIC	NOTREQUIRED
CICSTS56	J05ML	Ctrl+C	0	0	COBOL	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	J05ML	Add Quick Filter	0	0	COBOL	PRIVATE	CEDF	NOTREQUIRED
CICSTS56	J05MC	Phase In	38	0	NOTAPPLIC	PRIVATE	NOTAPPLIC	NOTREQUIRED
CICSTS56	J05MF	New Copy	13	0	COBOL	PRIVATE	CEDF	NOTREQUIRED
		Enable						
		Disable						
		Discard						
		Delete						
		Debug		>				

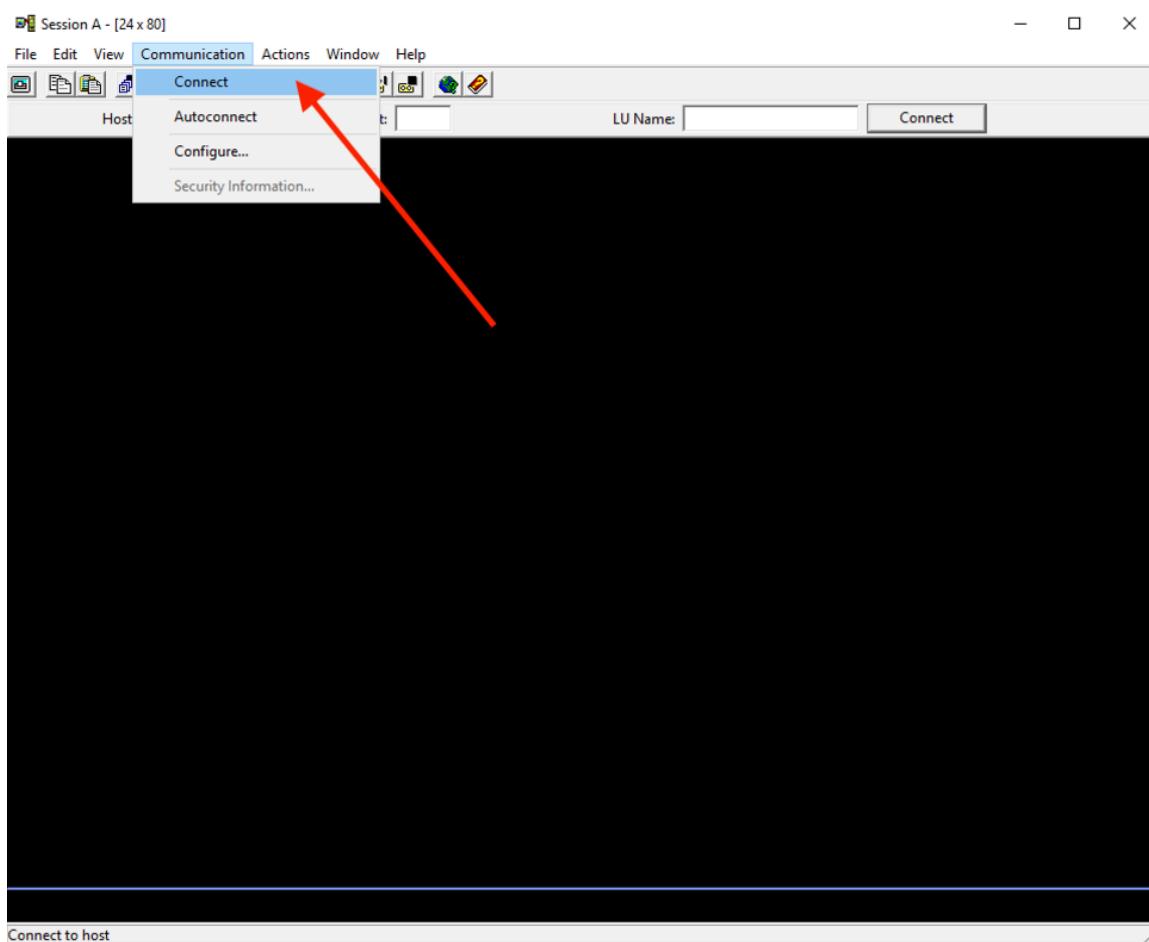
4.3.6 ► Select J05CMORT and click **OK**



4.3.7 Let's start the 3270 emulation again ..



4.3.8 ► If a black screen shows up . Select Communications → Connect



4.3.9 ➤ Type **I cicsts56.** and press **Enter key.**

The screenshot shows a VTAM terminal window titled "Session A - [24 x 80]". The window has a toolbar with various icons. The connection details are shown at the top: Host: zos.dev, Port: 23, LU Name: (empty), and Disconnect button. The main screen displays the following text:

```
z/OS V2R5 LVLI PUT2112/RSU2112          IP Address = 10.1.1.1
                                         VTAM Terminal = TCP00025

Application Developer System

      // 0000000  SSSSS
      // 00      00 SS
zzzzzz // 00      00 SS
      zz // 00      00 SSSS
      zz // 00      00      SS
zzzzzz // 0000000  SSSS

System Customization - ADCD.Z25A.*
```

At the bottom of the screen, there is a message:

```
==> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
==> Enter L followed by the APPLID
==> Examples: "L TSO", "L CICSTS56", "L CICSTS55", "L IMS15"
```

The command **I cicsts56** is typed in the input field at the bottom left. The status bar at the bottom right shows "Connected to remote server/host zos.dev using lu/pool TCP00025 and port 23" and the date "24/011".

- 4.3.11 ► Logon using **ibmuser** and password **sys1** and press **Enter**.
j05p

zos.dev 23 LU Name: Disconnect

Signon to CICS APPLID CICSTS56

WELCOME TO CICS TS 5.6

Type your userid and password, then press ENTER:

```
Userid . . . . ibmuser      Groupid . . .
Password . . . . -
Language . . . . -
New Password . . .
```

DFHCE3520 Please type your userid.

F3=Exit

MA A 11/030

Connected to remote server/host zos.dev using lu/pool TCP00022 and port 23

- 4.3.12 The sign-on message is displayed



4.4 Running **j05p** CICS transaction

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

- 4.4.1 ► Type the CICS transaction **j05p** and press the **Enter** key.

4.4.2 ➡ Press **enter** to see the monthly payment for the default data .

```
JKE MORTGAGE CALCULATOR

Amount of Loan: 1000
Length of Loan in Years: 10
Interest Rate: 5

Press F3 to quit or Enter to calculate loan
Press PF9 to see companies that can match or beat this rate

Monthly Payment: 10.61
```

4.4.3 ➡ Press **F1** key and verify the message displayed
This was your mission. As you see the change has been implemented.

```
Session A - [24x80]
File Edit View Communication Actions Window Help
Host: zos.dev Port: 23 LUName: Disconnect

JKE MORTGAGE CALCULATOR - July 12-, 2018

Amount of Loan: 1000
Length of Loan in Years: 10
Interest Rate: 5

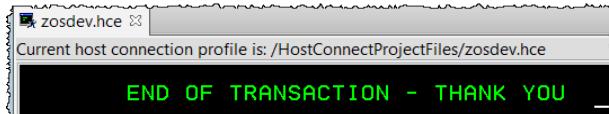
Press F3 to quit or Enter to calculate loan
Press PF9 to see companies that can match or beat this rate

Monthly Payment: 10.61

This is your
YYYYMMDD

YYYYMMDD - INVALID KEY PRESSED.
```

4.4.4 ➡ Press F3 to end the CICS transaction.



Section 5. Push and Commit the changed code to Git.

Using IDz you will commit the changes you made to Git and later perform the final building and deploy using Jenkins and UCD.

5.1 Using IDz with the Git plugin to compare the change versus original

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



5.1.2 ➡ ① Expand the nodes and click on the program **J05CMORT.cbl** that you have modified.
Notice that the changed program is listed in the *Git Staging* view..

➡ ② Double click on the **J05CMORT.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.

The screenshot shows the Eclipse IDE interface with several open editors:

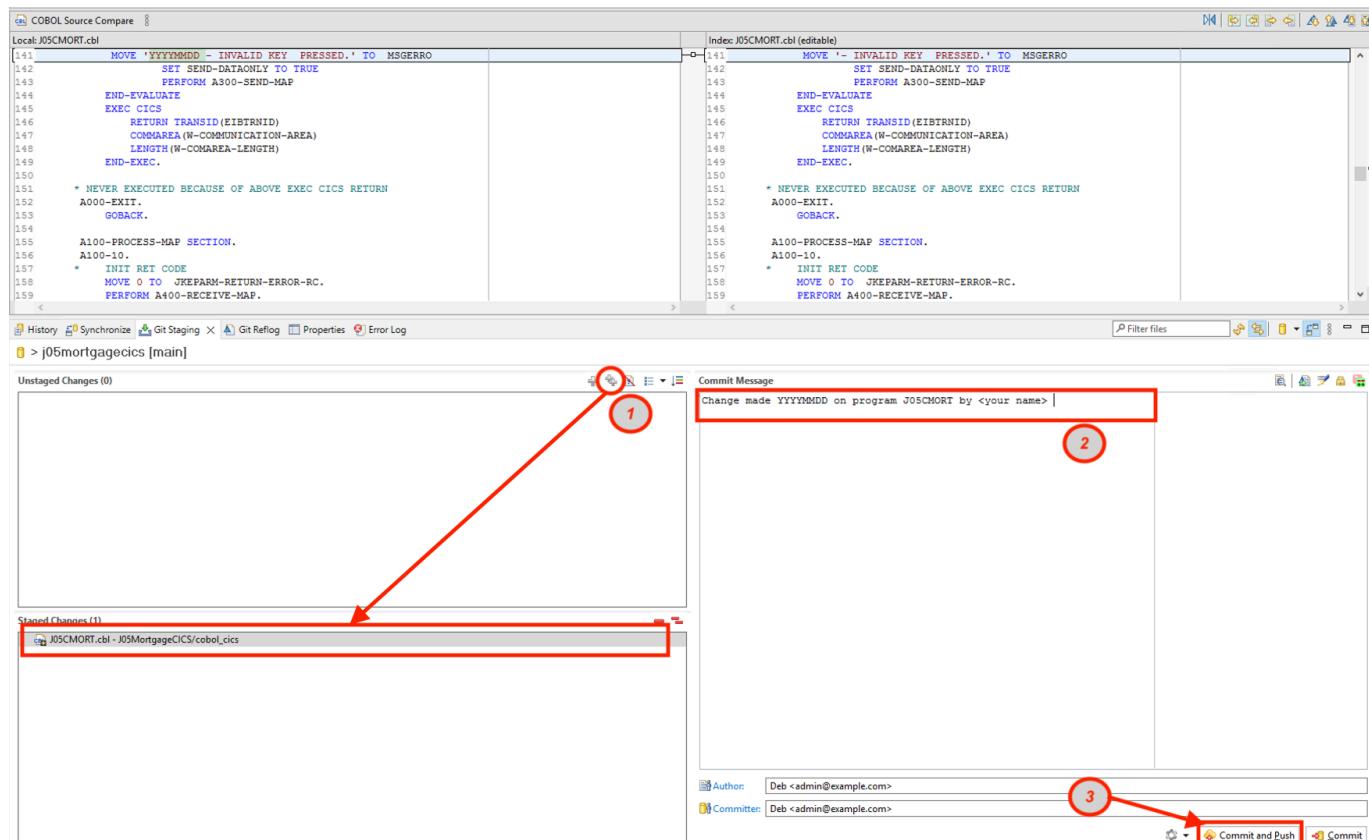
- Git Repositories:** Shows local branches (main, origin), tags, and remotes (origin). A red circle labeled '1' points to the 'J05CMORT.cbl' file in the 'Working Tree' section.
- COBOL Structure Compare:** Compares the local version of J05CMORT.cbl with the version in the index. A yellow callout bubble says 'The date you type will be here' pointing to the date field in the diff view. A red circle labeled '2' points to the 'J05CMORT.cbl' entry in the 'Unstaged Changes' list.
- Unstaged Changes:** Lists the modified file 'J05CMORT.cbl'.

5.1.3 ➡ Use **Ctrl + Shift + F4** to close the editors.

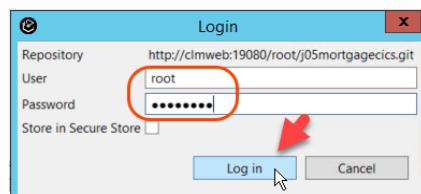
5.2 Push and Commit the changes to Git

5.2.1 ► Using Git Staging view , ,

- 1 Click on the icon 
- 2 Write a commit message like: **Change made yyyymmdd on program J05CMORT by your name**
- 3 Click **Commit and Push ...**

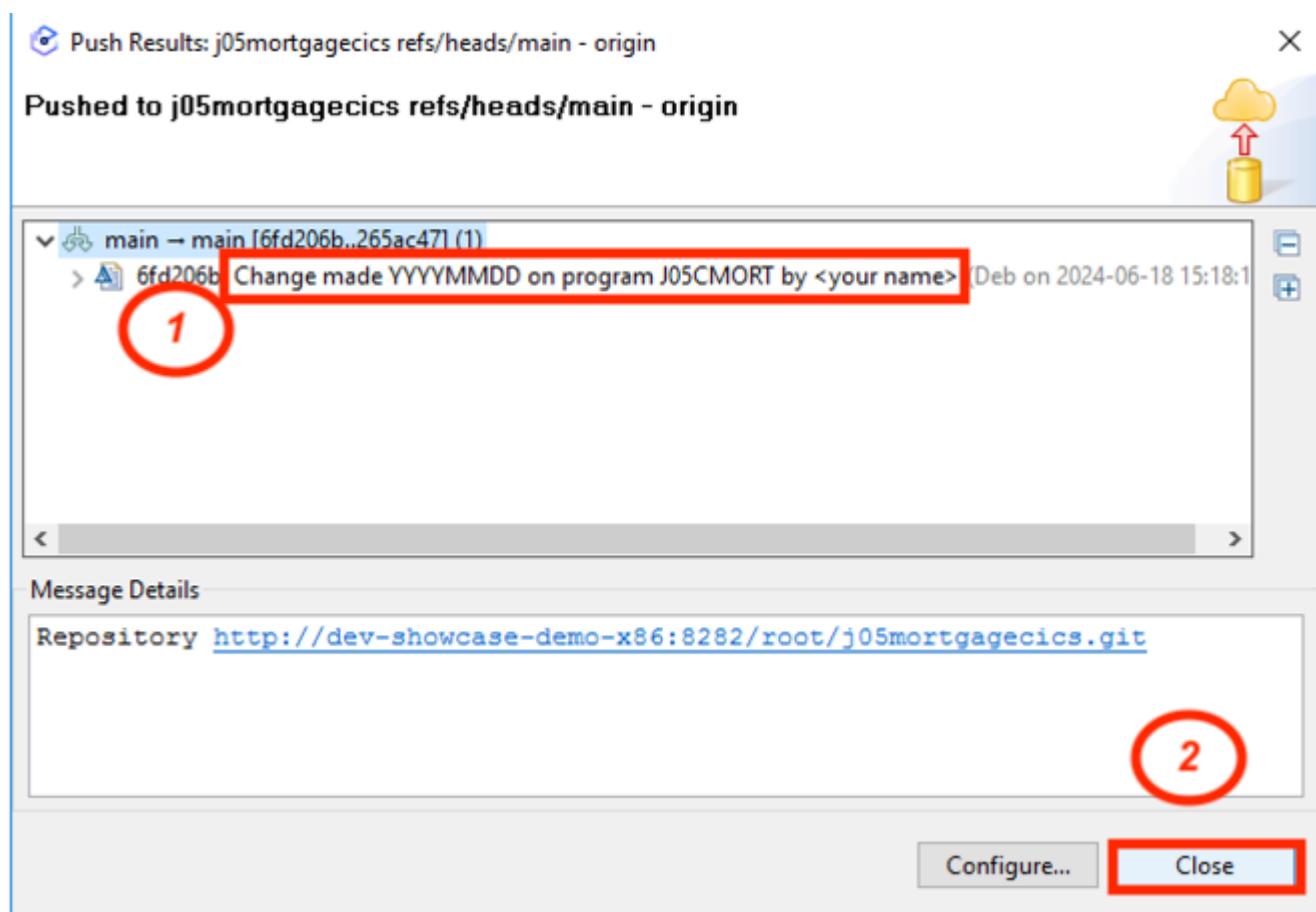


5.2.2 ► If it prompts for login credentials use **root** and password **passw0rd (0 is number zero)** and click **Log in**



5.2.3 ► Verify that the commit was successful and click **Close**

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



5.2.4 ► 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).

Name	Last commit	Last update
J05MortgageCICS	Change made YYYYMMDD on program J05CMORT by <your name>	4 minutes ago

5.2.5 ► You also can click on **Change made yyyyymmdd on program J05CMORT by xxxx** to see the changes

Change made YYYYMMDD on program J05CMORT by <your name>

```

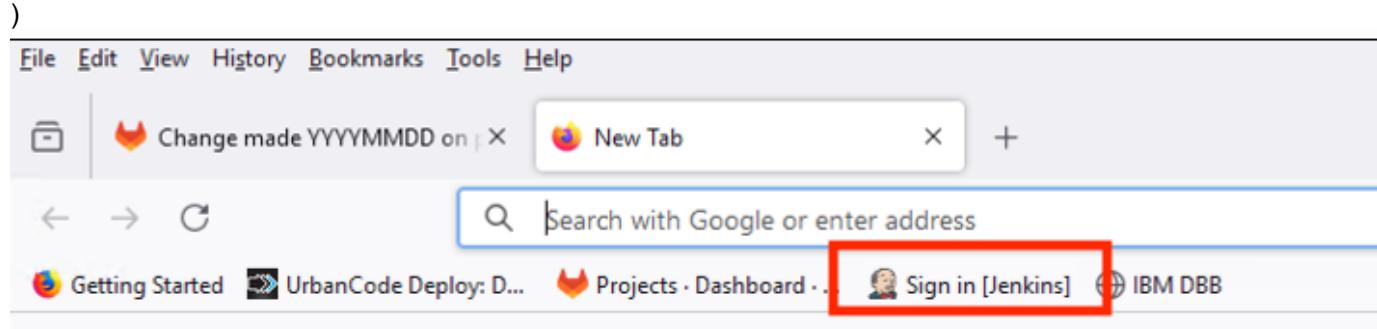
@@ -138,7 +138,7 @@
      * Replace YYYYMMDD Example: 20180712      *
      *           MOVE 'INVALID KEY PRESSED.' TO MSGERRO
      *
-     MOVE '- INVALID KEY PRESSED.' TO MSGERRO
+     MOVE 'YYYYMMDD - INVALID KEY PRESSED.' TO MSGERRO
      SET SEND-DATAONLY TO TRUE
      PERFORM A300-SEND-MAP
      END-EVALUATE
  
```

Section 6. Use Jenkins with Git plugin to build all the modified code committed to Git.

At this point the changed code is delivered into the Git repository that is on Linux. You will use Jenkins pipeline to build the new changed code and push the executables to be deployed using UCD. Note that this step makes a build of all code committed to Git, while on Section 4 only the selected COBOL program was built.

6.1 Logon to Jenkins using a Web Browser and start the z/OS agent

6.1.1 ► Using the **Bookmarks Tool Bar** click on **Sign in (Jenkins)**



6.1.2 ► If required type the user **ibmuser** and password **passw0rd (0 is number zero)** and click **log in**.



Sign in to Jenkins

<input type="text" value="ibmuser"/> <input type="password" value="••••"/>	<p><input type="checkbox"/> Keep me signed in</p> <p>Sign in</p>
---	---

6.1.3 ► Scroll down and click on **zos-Agent**

The screenshot shows the Jenkins dashboard with a sidebar on the left containing links like Project Relationship, Check File Fingerprint, Manage Jenkins, Convert To Pipeline, SCM2Job, My Views, Open Blue Ocean, Lockable Resources, P4 Trigger, and People. Below the sidebar are two expandable sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (listing 'Built-In Node' with 1 Idle, 2 Idle, and 'zos-Agent' which is highlighted with a red box). The 'zos-Agent' entry is also listed under 'Idle'.

6.1.4 ► If the agent zos-Agent is online go to the next step . If not On the right corner click on the **Bring this node back online** “blue button”.

This will bring the Jenkins agent online again.

Agent zos-Agent

Mark this node temporarily offline ?

Monitoring Data ▾ Add description

Projects tied to zos-Agent

S	W	Name	Last Success	Last Failure	Last Duration	F
		J05MortgageCICS	1 day 1 hr #51	1 day 1 hr #50	42 sec	> ★
		J05PDeploy	1 day 1 hr #3	N/A	34 sec	> ★

The agent is online

6.1.5 Click on Jenkins on top left:

Dashboard > Nodes > zos-Agent

█ Status

Agent zos-Agent

6.2 Starting the Jenkins Pipeline

6.2.1 The homepage lists all of the Jenkins jobs and pipelines.

Click on the hyperlink for the project **J05P_Pipeline** that represents your CICS application.

All +

S	W	Name	Last Success	Last Failure	Last Duration	F
		J05Deploy_test	N/A	N/A	N/A	> ★
		J05MortgageCICS	1 day 1 hr #51	1 day 2 hr #50	42 sec	> ★
		J05P_Pipeline	1 day 1 hr #21	N/A	1 min 31 sec	> ★
		J05PDeploy	1 day 1 hr #3	N/A	34 sec	> ★

6.2.2 This opens the **J05P_Pipeline** view. This pipeline has been designed with two stages:

1 – **Stage 1** builds ONLY the changed code from Git. This is an example of a **Build pipeline**.

2 – **Stage 2** deploys the changes into CICS Development Region (Dev) using UrbanCode Deploy (UCD).
This is an example of a **Delivery pipeline**.

This pipeline is a simplistic form of an actual delivery pipeline. Any other activity such as, test cases, code reviews, batch job runs, rexx scripts, table queries and other manual tasks that are currently being carried out in a shop before code is moved to QA (or any other region) can be built into a pipeline.

▶ Click on **Build Now** on the left.

IMPORTANT : This Jenkins Build will take some **Minutes** since the cloud instances are slow compared to a real environment.

The screenshot shows the Jenkins Pipeline interface for the 'J05P_Pipeline'. At the top, there's a breadcrumb navigation: Dashboard > J05P_Pipeline >. Below it, a navigation bar includes 'Status' (highlighted with a grey background), 'Changes', 'Build Now' (which has a red border around it), 'Configure', 'Delete Pipeline', and 'Full Stage View'. The main title 'J05P_Pipeline' is displayed with a green checkmark icon. A subtitle below it reads 'Pipeline to build and Deploy J05P - CICS Transaction'. On the right, there's a section titled 'Stage View' with a progress bar, which is currently empty.

- 6.2.3 ① Notice that the new build has appeared under **Build History** indicating the currently started Pipeline.
② The **Stage View** also shows a new line with an in-progress bar.

J05P_Pipeline

Pipeline to build and Deploy J05P - CICS Transaction

Stage View

Average stage times:
(Average full run time: ~22min)

#22	Jun 18 16:17	No Changes	36s
		41s	21min 50s
		32s	

#21	Jun 17 13:57	No Changes	50s	41s	
#19		Jun 17 09:24	No Changes	41s	15s

Build History

trend ▾

Filter... /

#22 Jun. 18, 2024, 5:17 p.m.

6.3 Checking the results

6.3.1 Since this z/OS is running in a small processor in the cloud, this activity may take some minutes, but you don't need to wait to be completed to check the results.
You can follow what is going on as described below ..

- ▶ Move the cursor and click on the **green area** 1 and then click on **Logs** 2.

Stage View

- trash Delete Pipeline
- search Full Stage View
- star Favorite
- blue-ocean Open Blue Ocean
- stages Stages
- edit Rename
- comment Build Review
- question Pipeline Syntax

Build History trend

#	Date	Time	Changes
#22	Jun 18	16:17	No Changes
#21	Jun 17	13:57	No Changes
#19	Jun 17	09:24	No Changes
#17	Jun 17	08:18	No Changes

Average stage times:
(Average full run time: ~17min)

Stage	Run #	Time	Duration
Starting J05MortgageCICS Build and Push to UCD	#22	Jun 18 16:17	51s
Deploy J05 into DEV with UCD	#22	Jun 18 16:17	39s
Starting J05MortgageCICS Build and Push to UCD	#21	Jun 17 13:57	50s
Deploy J05 into DEV with UCD	#21	Jun 17 13:57	41s
Starting J05MortgageCICS Build and Push to UCD	#19	Jun 17 09:24	41s
Deploy J05 into DEV with UCD	#19	Jun 17 09:24	15s
Starting J05MortgageCICS Build and Push to UCD	#17	Jun 17 08:18	42s
Deploy J05 into DEV with UCD	#17	Jun 17 08:18	1h 4min

6.3.2 The dialog *Stage Logs* will open.

Right click on **J05MortgageCICS #nn** and select **Open Link in New Tab**

Stage Logs (Starting J05MortgageCICS Build and Push to UCD)

Building J05MortgageCICS -- J05MortgageCICS (self time 51s)

```
Scheduling project: J05MortgageCICS
Starting building: J05MortgageCICS #52
Build J05MortgageCICS #52
```

- trash Delete Pipeline
- search Full Stage View
- star Favorite
- blue-ocean Open Blue Ocean
- stages Stages
- edit Rename
- comment Build Review
- question Pipeline Syntax

Build History trend

#22	Jun 18	16:17	No Changes
#21	Jun 17	13:57	No Changes
#19	Jun 17	09:24	No Changes
#17	Jun 17	08:18	No Changes

Average stage times:
(Average full run time: ~17min)

Stage	Run #	Time	Duration
Starting J05MortgageCICS Build and Push to UCD	#22	Jun 18 16:17	51s
Deploy J05 into DEV with UCD	#22	Jun 18 16:17	39s
Starting J05MortgageCICS Build and Push to UCD	#21	Jun 17 13:57	50s
Deploy J05 into DEV with UCD	#21	Jun 17 13:57	41s
Starting J05MortgageCICS Build and Push to UCD	#19	Jun 17 09:24	41s
Deploy J05 into DEV with UCD	#19	Jun 17 09:24	15s
Starting J05MortgageCICS Build and Push to UCD	#17	Jun 17 08:18	42s
Deploy J05 into DEV with UCD	#17	Jun 17 08:18	1h 4min

Permalinks

[Atom feed for all](#) [Atom feed for failures](#)

6.3.3 This opens the current running task for the first step of the J05Mortgage Application Build.

▶ Click on the **Console Output** to view the log.

The screenshot shows the Jenkins interface for build #51 of the J05MortgageCICS project. The top navigation bar includes the Jenkins logo, the word "Jenkins", and links for "Dashboard", "J05MortgageCICS", and "#51". The main content area displays the build status as "Success" (#51, Jun. 17, 2024, 2:57:28 p.m.). On the left, a sidebar lists various build actions: "Changes", "Console Output" (which is highlighted with a red box), "Edit Build Information", "Delete build '#51'", "Timings", "Git Build Data", "Open Blue Ocean", "Previous Build", and "Next Build". To the right, detailed build information is shown, including the changes made (1. Add new file), the start cause (Started by upstream project J05P_Pipeline build number 21), the user who started it (ibmuser), the time spent (7.5 sec waiting, 42 sec build duration, 50 sec total from scheduled to completion), and the git details (Revision: 694f2ab63473f6a7c9b01403842cf7c2fdf6692c, Repository: http://dev-showcase-demo-x86:8282/root/j05mortgagecics.git, origin/main).

6.3.4 The Console Output log of first pipeline shows the various Groovy scripts being executed. You need to understand what is going on !!.

Dashboard > J05MortgageCICS > #52 > Console Output

</> Changes

Console Output

View as plain text

Edit Build Information

Delete build '#52'

Timings

Git Build Data

Open Blue Ocean

Previous Build

```

Started by upstream project "J05P_Pipeline" build number 22
originally caused by:
Started by user ibmuser
Running as SYSTEM
Building remotely on zos-Agent in workspace /opt/jenkins/workspace/J05MortgageCICS
The recommended git tool is: NONE
using credential 9d4ae3f-d8d3-48da-bd9c-c0d31c3ab0ff
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh rev-parse --resolve-git-dir /opt/jenkins/workspace/J05MortgageCICS/.git # timeout=10
Fetching changes from the remote Git repository
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh config remote.origin.url http://dev-showcase-demo-x86:8282/root/j05mortgagetects.git # timeout=10
Fetching upstream changes from http://dev-showcase-demo-x86:8282/root/j05mortgagetects.git
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh --version # timeout=10
> git --version # 'git' version 2.26.2-78'
using GIT_ASKPASS to set credentials Gitlab connection
Using name charset 'IBM1047'
Using password charset 'IBM1047'
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh fetch --tags --force --progress -- http://dev-showcase-demo-x86:8282/root/j05mortgagetects.git +refs/heads/*:refs/remotes/origin/* # timeout=10
Seen branch in repository origin/main
Seen 1 remote branch
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh show-ref --tags -d # timeout=10
Checking out Revision 6fd206b3b6e4f61d0b6ea9fc77fa1ea85594e21e (origin/main)
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh config core.sparsecheckout # timeout=10
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh checkout -f 6fd206b3b6e4f61d0b6ea9fc77fa1ea85594e21e # timeout=10
Commit message: "Change made YYYYMMDD on program J05CMORT by <your name>"
```

> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh rev-list --no-walk 694f2ab63473f6a7c9b01403842cf7c2fdf6692c # timeout=10

[J05MortgageCICS] \$ /usr/lpp/IBM/dbb/groovy/bin/groovy -cp /usr/lpp/IBM/dbb/lib/* -c IBM-1047 /opt/jenkins/workspace/J05MortgageCICS/J05MortgageCICS/build/impacts.groovy --buildHash 6fd206b3b6e4f61d0b6ea9fc77fa1ea85594e21e --sourceDir /opt/jenkins/workspace/J05MortgageCICS --workDir /opt/jenkins/workspace/J05MortgageCICS/BUILD-52

** Build properties at startup:

6.4 Understand the results

This task performs five sub-steps.

► Scroll down and locate the highlighted portions shown below from the build console output to know more about the build

6.4.1 Check out of code from Git Repositories based on the latest commit

Dashboard > J05MortgageCICS > #52 > Console Output

</> Changes

Console Output

View as plain text

Edit Build Information

Delete build '#52'

Timings

Git Build Data

Open Blue Ocean

Previous Build

```

Started by upstream project "J05P_Pipeline" build number 22
originally caused by:
Started by user ibmuser
Running as SYSTEM
Building remotely on zos-Agent in workspace /opt/jenkins/workspace/J05MortgageCICS
The recommended git tool is: NONE
using credential 9d4ae3f-d8d3-48da-bd9c-c0d31c3ab0ff
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh rev-parse --resolve-git-dir /opt/jenkins/workspace/J05MortgageCICS/.git # timeout=10
Fetching changes from the remote Git repository
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh config remote.origin.url http://dev-showcase-demo-x86:8282/root/j05mortgagetects.git # timeout=10
Fetching upstream changes from http://dev-showcase-demo-x86:8282/root/j05mortgagetects.git
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh --version # timeout=10
> git --version # 'git' version 2.26.2-78'
using GIT_ASKPASS to set credentials Gitlab connection
Using name charset 'IBM1047'
Using password charset 'IBM1047'
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh fetch --tags --force --progress -- http://dev-showcase-demo-x86:8282/root/j05mortgagetects.git +refs/heads/*:refs/remotes/origin/* # timeout=10
Seen branch in repository origin/main
Seen 1 remote branch
> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh show-ref --tags -d # timeout=10
Checking out Revision 6fd206b3b6e4f61d0b6ea9fc77fa1ea85594e21e (origin/main)
```

> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh config core.sparsecheckout # timeout=10

> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh checkout -f 6fd206b3b6e4f61d0b6ea9fc77fa1ea85594e21e # timeout=10

Commit message: "Change made YYYYMMDD on program J05CMORT by <your name>"

> /usr/lpp/IBM/dbb/bin/git-jenkins2.sh rev-list --no-walk 694f2ab63473f6a7c9b01403842cf7c2fdf6692c # timeout=10

[J05MortgageCICS] \$ /usr/lpp/IBM/dbb/groovy/bin/groovy -cp /usr/lpp/IBM/dbb/lib/* -c IBM-1047 /opt/jenkins/workspace/J05MortgageCICS/J05MortgageCICS/build/impacts.groovy --buildHash 6fd206b3b6e4f61d0b6ea9fc77fa1ea85594e21e --sourceDir /opt/jenkins/workspace/J05MortgageCICS --workDir /opt/jenkins/workspace/J05MortgageCICS/BUILD-52

** Build properties at startup:

BUILD_MFS=false

6.4.2 An Impact Analysis to identify if any programs have been changed since the last build using DBB Impact scripts. The DBB server that is installed on USS on z/OS keeps track of all builds.

```

Dashboard > J05MortgageCICS > #52 > Console Output
10-AUADMIN
pwfile=J05MortgageCICS/build/ADMIN.pw
repo=https://dev-showcase-demo-x86:9443/dbb/
scriptMapping=BMSPprocessing::J05MortgageCICS/bms/*.bms
scriptMapping=CobolCompile::J05MortgageCICS/cobol/J05MPPMT.cbl,J05MortgageCICS/cobol_cics/*.cbl
scriptMapping=Compile::J05MortgageCICS/cobol/J05CMORT.cbl,J05MortgageCICS/cobol/J05MBRVL.cbl
scriptMapping=LinkEdit::J05MortgageCICS/link/*.ink
sourceDir=/opt/jenkins/workspace/J05MortgageCICS
jenkinsUrl=https://jenkins:8080/jenkins/workspace/J05MortgageCICS/BUILD-52
** Impact analysis start at 20240618.091753.017
** Searching for last successful build commit hash for build group J05MortgageCICS
Last successful build commit hash located. label : build.20240617.065817.058 , buildHash : 694f2ab63473f6a7c9b01403842cf7c2fdf6692c
** Executing Git command: git diff --name-only 694f2ab63473f6a7c9b01403842cf7c2fdf6692c 6fd206b3b6e4f61d0b6ea9fc77fa1e85594e21e
Number of changed files detected since build build.20240617.065817.058 : 1
J05MortgageCICS/cobol_cics/J05CMORT.cbl

** Scan the changed file list to collect the latest dependency data
Scanning changed file J05MortgageCICS/cobol_cics/J05CMORT.cbl
** Store the dependency data in repository collection 'J05MortgageCICS'
HTTP/1.1 200 OK
** Creating build list by resolving impacted programs/files for changed files
Found build script mapping for J05MortgageCICS/cobol_cics/J05CMORT.cbl. Adding to build list.
** Writing buildlist to /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/buildlist.txt
** Impact analysis finished at Tue Jun 18 21:17:55 GMT 2024
** Total # build files calculated = 1
** Total analysis time : 1.458 seconds
[J05MortgageCICS] $ /usr/lpp/IBM/dbb/groovy -cp /usr/lpp/IBM/dbb/lib/* -c IBM-1047 /opt/jenkins/workspace/J05MortgageCICS/J05MortgageCICS/build/build.groovy --buildHash
6fd206b3b6e4f61d0b6ea9fc77fa1e85594e21e --sourceDir /opt/jenkins/workspace/J05MortgageCICS --workDir /opt/jenkins/workspace/J05MortgageCICS/BUILD-52 --hlq IBMUSER.DEMO.DEV /opt/jenkins/
workspace/J05MortgageCICS/BUILD-52/buildlist.txt
** Build properties at startup:
BUILD_MFS=false
COLLID=
COLLDIR=
MACLIB=SYS1.MACLIB
OWNER=
QUAL=
REFERAL=
RUN_DB2_BIND=False
SASMMOD1=HLA.SASMMOD1
SCEELKED=CEE.SCEELKED

```

6.4.3 A dependency scan to pick out all the dependencies of the changed programs, The DBB server that is installed on USS on z/OS keeps track of all dependencies.

```

Dashboard > J05MortgageCICS > #52 > Console Output
workDir=/opt/jenkins/workspace/J05MortgageCICS/BUILD-52
** Impact analysis start at 20240618.091753.017
** Searching for last successful build commit hash for build group J05MortgageCICS
Last successful build commit hash located. label : build.20240617.065817.058 , buildHash : 694f2ab63473f6a7c9b01403842cf7c2fdf6692c
** Executing Git command: git diff --name-only 694f2ab63473f6a7c9b01403842cf7c2fdf6692c 6fd206b3b6e4f61d0b6ea9fc77fa1e85594e21e
Number of changed files detected since build build.20240617.065817.058 : 1
J05MortgageCICS/cobol_cics/J05CMORT.cbl

** Scan the changed file list to collect the latest dependency data
Scanning changed file J05MortgageCICS/cobol_cics/J05CMORT.cbl
** Store the dependency data in repository collection 'J05MortgageCICS'
HTTP/1.1 200 OK
** Creating build list by resolving impacted programs/files for changed files
Found build script mapping for J05MortgageCICS/cobol_cics/J05CMORT.cbl. Adding to build list.
** Writing buildlist to /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/buildlist.txt
** Impact analysis finished at Tue Jun 18 21:17:55 GMT 2024
** Total # build files calculated = 1
** Total analysis time : 1.458 seconds
[J05MortgageCICS] $ /usr/lpp/IBM/dbb/groovy -cp /usr/lpp/IBM/dbb/lib/* -c IBM-1047 /opt/jenkins/workspace/J05MortgageCICS/J05MortgageCICS/build/build.groovy --buildHash
6fd206b3b6e4f61d0b6ea9fc77fa1e85594e21e --sourceDir /opt/jenkins/workspace/J05MortgageCICS --workDir /opt/jenkins/workspace/J05MortgageCICS/BUILD-52 --hlq IBMUSER.DEMO.DEV /opt/jenkins/
workspace/J05MortgageCICS/BUILD-52/buildlist.txt
** Build properties at startup:
BUILD_MFS=false
COLLID=
COLLDIR=
MACLIB=SYS1.MACLIB
OWNER=
QUAL=
REFERAL=
RUN_DB2_BIND=False
SASMMOD1=HLA.SASMMOD1
SCEELKED=CEE.SCEELKED

```



6.4.4 A compile/link-edit of the changed programs

Dashboard > J05MortgageCICS > #52 > Console Output

```

m+-----+-----+-----+-----+-----+-----+-----+-----+
report=https://dev-showcase-demo-x86:9443/dbb/
scriptMapping=BMSProcessing::J05MortgageCICS/bms/*.bms
scriptMapping=CobolCompile::J05MortgageCICS/cobol/J05MMRT.cbl,J05MortgageCICS/cobol_cics/*.cbl
scriptMapping=Compile::J05MortgageCICS/cobol/J05CMORT.cbl,J05MortgageCICS/cobol/J05NBRVL.cbl
scriptMapping=LinkEdit::J05MortgageCICS/link/*.lnk
sourceDir=/opt/jenkins/workspace/J05MortgageCICS
workDir=/opt/jenkins/workspace/J05MortgageCICS/BUILD-52
** Build start at 20240618.091803.018
** Build result created at https://dev-showcase-demo-x86:9443/dbb/rest/buildResult/10595
** Build output will be in /opt/jenkins/workspace/J05MortgageCICS/BUILD-52
** Building files listed in /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/buildlist.txt
** Scan the build list to collect dependency data
Scanning J05MortgageCICS/cobol_cics/J05CMORT.cbl
** Storing remaining 1 logical files in repository collection 'J05MortgageCICS'
HTTP/1.1 200 OK
** Invoking build scripts according to build order: Compile, LinkEdit, CobolCompile
* Building J05MortgageCICS/cobol_cics/J05CMORT.cbl using cobolCompile.groovy script
Copying /opt/jenkins/workspace/J05MortgageCICS/J05MortgageCICS/cobol_cics/J05CMORT.cbl to IBMUSER.DEMO.DEV.COBL(J05CMORT)
Resolving dependencies for file J05MortgageCICS/cobol_cics/J05CMORT.cbl and copying to IBMUSER.DEMO.DEV.COPYBOOK
Compiling and link editing program J05MortgageCICS/cobol_cics/J05CMORT.cbl
** Build Finished at Tue Jun 18 21:18:08 GMT 2024
** Build State = CLEAN
** Total files processed : 1
** Total build time : 4.963 seconds
[J05MortgageCICS] $ /usr/lpp/IBM/dbb/groovy/bin/groovy -cp /usr/lpp/IBM/dbb/lib/* -c IBM-1047 /opt/jenkins/workspace/J05MortgageCICS/J05MortgageCICS/build/deploy.groovy --buztool /var/ucd/agent72/bin/buztool.sh -wkdir /opt/jenkins/workspace/J05MortgageCICS/BUILD-52 --component J05MortgagePOT
** Create version start at 20240618.091814.018
** Properties at startup:
  component -> J05MortgagePOT
  buztoolPath -> /var/ucd/agent72/bin/buztool.sh
  startTime -> 20240618.091814.018

```

6.4.5 Creation of a deployable binary version

of code that can be used by *UrbanCode Deploy* to deploy to the necessary environments. Here Jenkins is using the UCD plugin to store an executable version at UCD server for later deploy.

Dashboard > J05MortgageCICS > #52 > Console Output

```

m+-----+-----+-----+-----+-----+-----+-----+-----+
** Create version start at 20240618.091814.018
** properties at startup:
  component -> J05MortgagePOT
  buztoolPath -> /var/ucd/agent72/bin/buztool.sh
  startTime -> 20240618.091814.018
  workDir -> /opt/jenkins/workspace/J05MortgageCICS/BUILD-52
** Read build report data from /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/BuildReport.json
** Find deployable outputs in the build report
  IBMUSER.DEMO.DEV.LOAD(J05CMORT), LOAD
** Generate UCD ship list file
** Create ship list file to /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/shiplist.xml
** Create version by running UCD buztool
** Ignore > Error adding properties to version in UrbanCode Deploy server
/var/ucd/agent72/bin/buztool.sh createzosversion -c J05MortgagePOT -s /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/shiplist.xml -o /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/
buztool.output
zos toolkit config : /var/ucd/agent72/ (7.2.0.0,20210527-0014)
zos toolkit binary : /var/ucd/agent72/ (7.2.0.0,20210527-0014)
zos toolkit data set : BUZ720 (7.2.0.0,20210615-0431)
Reading parameters:
....Command : createzosversion
....Component : J05MortgagePOT
....Generate version name : 20240618-211818
....Shiplist file : /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/shiplist.xml
....Output file:/opt/jenkins/workspace/J05MortgageCICS/BUILD-52/buztool.output
Verifying version
....Repository location : /var/ucd/agent72/var/repository/J05MortgagePOT/20240618-211818
Pre-processing shiplist:
....Shiplist after processing :/var/ucd/agent72/var/repository/J05MortgagePOT/20240618-211818/shiplist.xml
Packaging data sets:
....Location to store zip : /var/ucd/agent72/var/repository/J05MortgagePOT/20240618-211818
....Zip name : package.zip

```

Ignore the error message below. This does not impact the UCD version creation..

```
Dashboard > J05MortgageCICS > #52 > Console Output
** Create version by running UCD buztool
** Ignore > Error adding properties to version in Urbancode Deploy server
/var/ucd/agent72/bin/buztool.sh createzosversion -c J05MortgagePOT -s /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/shiplist.xml -o /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/buztool.output
zos_toolkit config : /var/ucd/agent72/ (7.2.0.0,20210527-0014)
zos_toolkit binary : /var/ucd/agent72/ (7.2.0.0,20210527-0014)
zos_toolkit data set : BUZT20 (7.2.0.0,20210615-0431)
Reading parameters:
....Command : createzosversion
....Component : J05MortgagePOT
....Generate version name : 20240618-211818
....Shiplist file : /opt/jenkins/workspace/J05MortgageCICS/BUILD-52/shiplist.xml
....Output File:/opt/jenkins/workspace/J05MortgageCICS/BUILD-52/buztool.output
Verifying version
....Repository location : /var/ucd/agent72/var/repository/J05MortgagePOT/20240618-211818
Pre-processing shiplist:
....Shiplist after processing :/var/ucd/agent72/var/repository/J05MortgagePOT/20240618-211818/shiplist.xml
Packaging data sets:
....Location to store zip : /var/ucd/agent72/var/repository/J05MortgagePOT/20240618-211818
....Zip name : package.zip
....IBMUSER.DEMO.DEV.LOAD.bin
....Elapsed time for data set package or deploy operation : 0.421123
Post-processing package:
PackageManifest file post-processing completed.
Create version and store package:
....Version created in UCD server
....Version:20240618-211818 created
Elapsed time: 4.0 seconds.

Finished: SUCCESS
```

Section 7. Use Jenkins and UCD plugin to deploy results and test the CICS transaction again

You will verify the results of the second stage (deploy using UCD) .

7.1 Checking the deploy results (second stage)

- 7.1.1 ➡ Click on Jenkins to back to Dashboard.

The screenshot shows the Jenkins dashboard. At the top left is the Jenkins logo (a cartoon character) inside a red-bordered box. Below it is the word "Jenkins". Underneath is a navigation bar with the items "Dashboard" > "Nodes" > "zos-Agent".

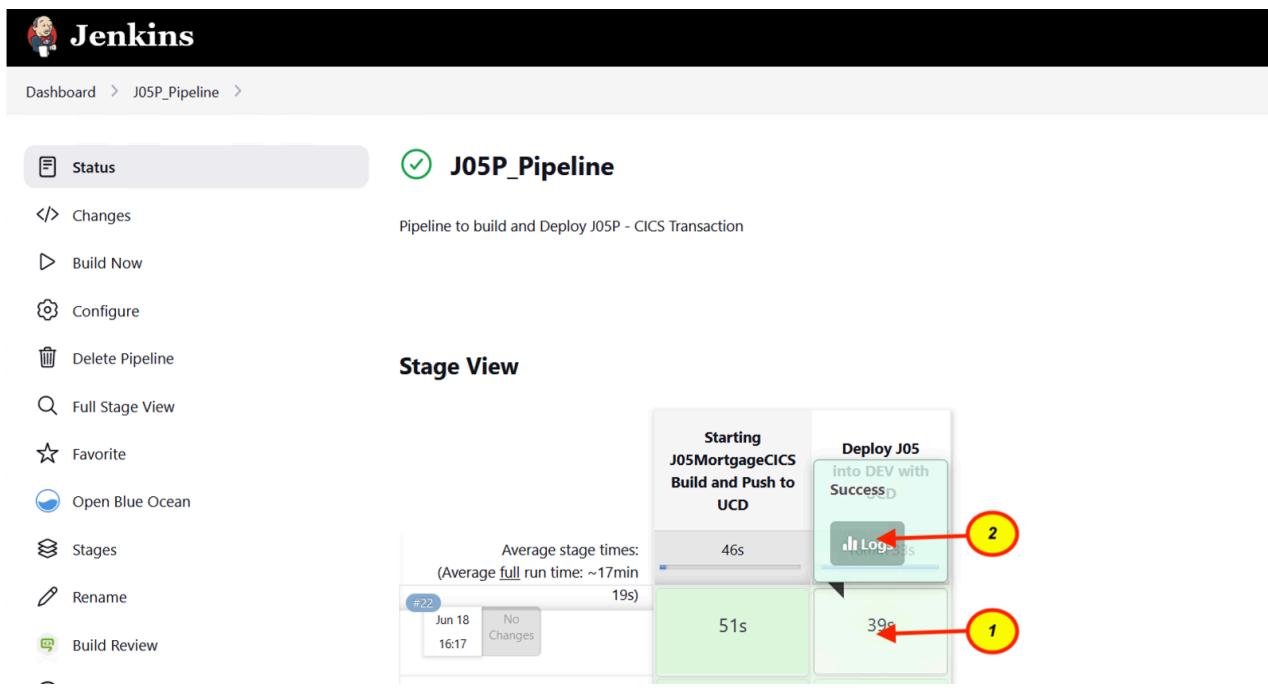
Status

Agent zos-Agent

- 7.1.2 ➡ Click on **J05P_Pipeline**

S	W	Name ↓	Last Success	Last Failure	Last Duration	F
...	☀	J05Deploy_test	N/A	N/A	N/A	▶ ☆
✓	☁	J05MortgageCICS	1 day 1 hr #51	1 day 2 hr #50	42 sec	▶ ★
✓	☀	J05P_Pipeline	1 day 1 hr #21	N/A	1 min 31 sec	▶ ☆
✓	☀	J05PDeploy	1 day 1 hr #3	N/A	34 sec	▶ ☆

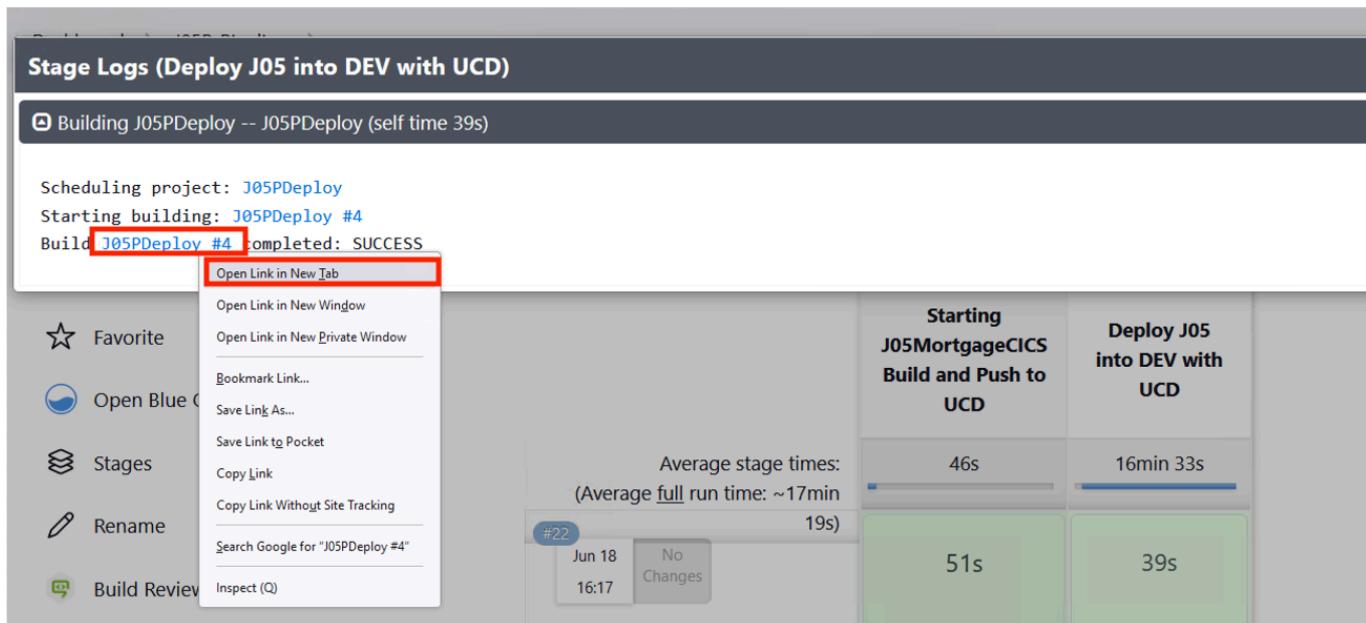
- 7.1.3 ➡ Click as show in the **blue area** of the Deploy stage ① and then click on **Logs** ② .



The screenshot shows the Jenkins Pipeline status page for the 'J05P_Pipeline'. The pipeline is labeled 'J05P_Pipeline' with a green checkmark icon. Below it, the description 'Pipeline to build and Deploy J05P - CICS Transaction' is displayed. On the left, there is a sidebar with various Jenkins management options like 'Status', 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Full Stage View', 'Favorite', 'Open Blue Ocean', 'Stages', 'Rename', and 'Build Review'. The main area is titled 'Stage View' and shows two stages: 'Starting J05MortgageCICS Build and Push to UCD' (46s) and 'Deploy J05 into DEV with SuccessD' (39s). A tooltip indicates an average stage time of 17 minutes. A red arrow labeled '1' points to the 'Deploy J05 into DEV with SuccessD' stage, and another red arrow labeled '2' points to its 'Log' button.

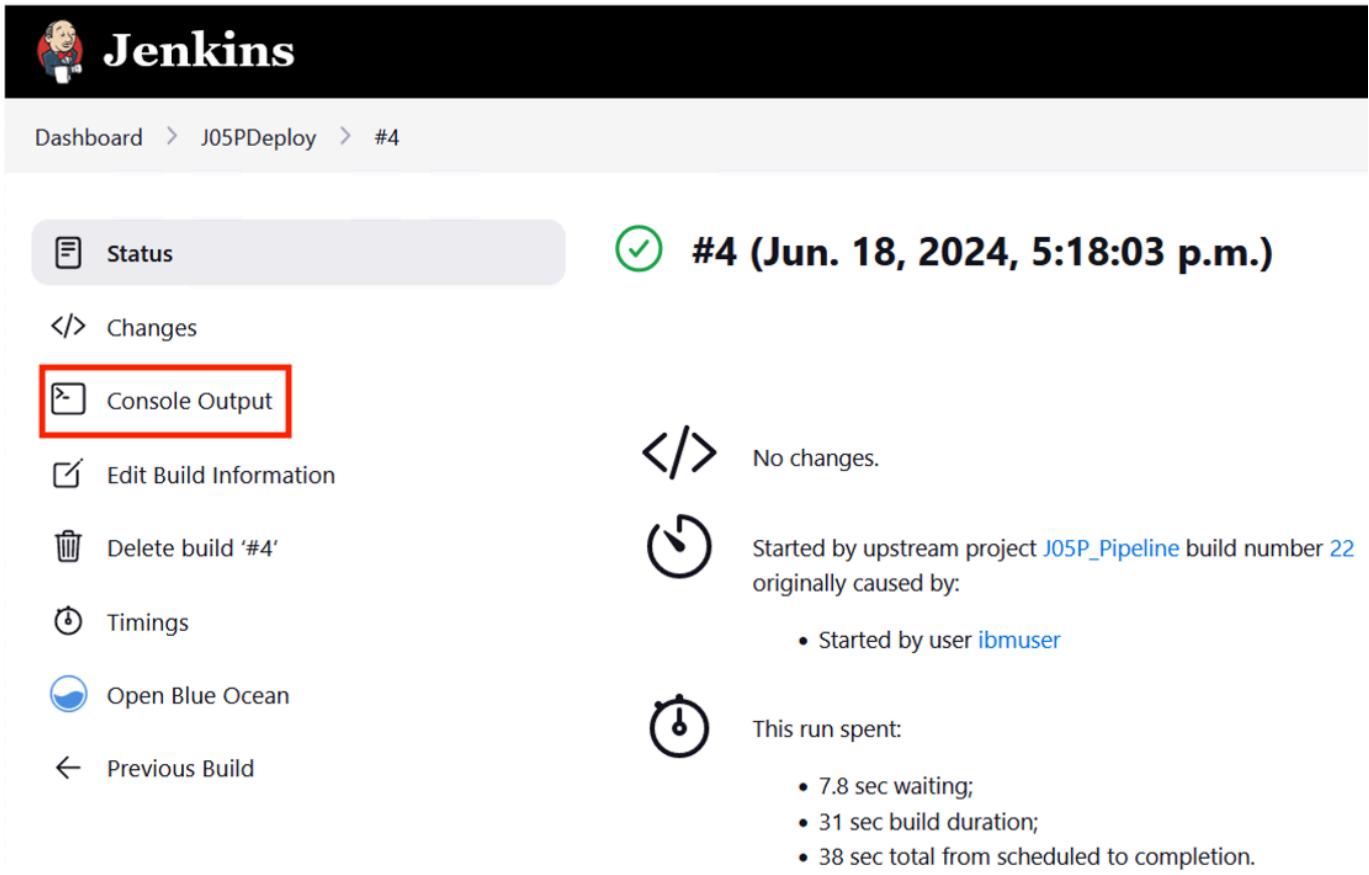
7.1.4 The dialog Stage Logs will open.

▶ Right click on **J05PDeploy #nn** and select **Open Link in New Tab**



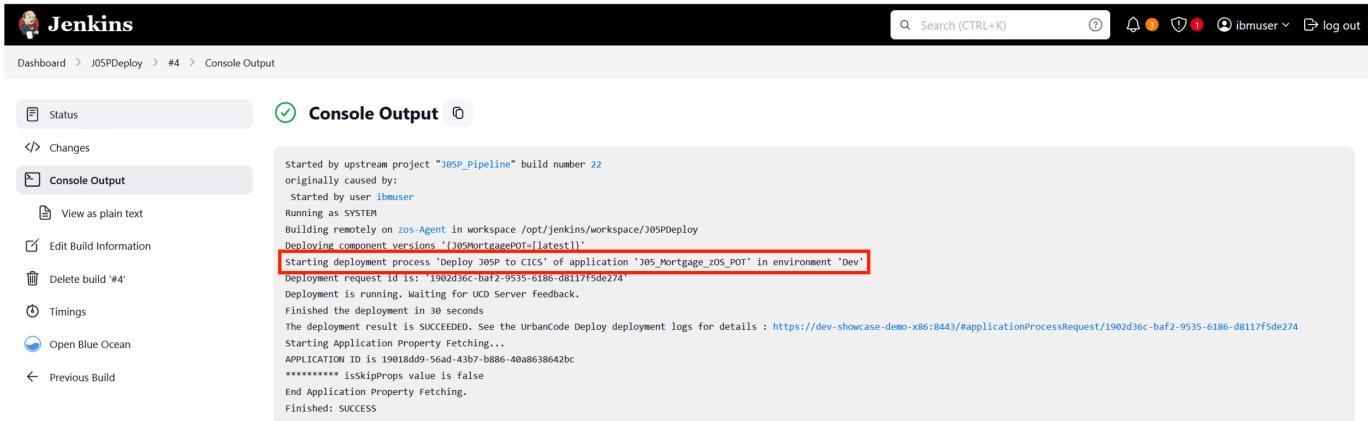
The screenshot shows the 'Stage Logs (Deploy J05 into DEV with UCD)' dialog. It displays the log output for the 'J05PDeploy' build, specifically for build #4. The log shows the project starting, building, and then completing successfully. A context menu is open over the log entry for build #4, with the option 'Open Link in New Tab' highlighted by a red box. The background shows the Jenkins Stage View interface with the same two stages and their respective times.

7.1.5  Click on the **Console Output** to view the logs.



The screenshot shows the Jenkins interface for a job named "J05PDeploy" with build number #4. The top navigation bar includes links for "Dashboard", "J05PDeploy", and "#4". On the left, there's a sidebar with various options like "Status", "Changes", "Console Output" (which is highlighted with a red box), "Edit Build Information", "Delete build '#4'", "Timings", "Open Blue Ocean", and "Previous Build". To the right, a summary section displays a green checkmark icon and the text "#4 (Jun. 18, 2024, 5:18:03 p.m.)". Below this, it says "No changes." and provides details about the build's origin: "Started by upstream project J05P_Pipeline build number 22 originally caused by: • Started by user ibmuser". It also shows the run's duration: "This run spent: • 7.8 sec waiting; • 31 sec build duration; • 38 sec total from scheduled to completion."

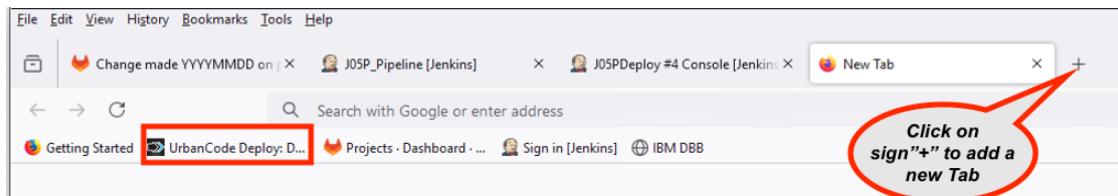
7.1.6 The Console Output log of second pipeline shows that the program that you delivered will be deployed to CICS Dev environment using UCD..



The screenshot shows the Jenkins "Console Output" page for build #4. The top navigation bar includes links for "Dashboard", "J05PDeploy", and "#4". The sidebar on the left is identical to the previous screenshot. The main content area shows the console output log, which includes deployment details such as "Started by upstream project "J05P_Pipeline" build number 22 originally caused by: Started by user ibmuser", "Running as SYSTEM", "Building remotely on zos-Agent in workspace /opt/jenkins/workspace/J05PDeploy", "Deploying component versions 'J05MortgageOT-[latest]'", "starting deployment process 'Deploy J05P to CICS' of application 'J05_Mortgage_ZOS_POT' in environment 'Dev'", "Deployment request id is: '1902d36c-baf2-9535-6186-d8117f5de274'", "Deployment is running. Waiting for UCD Server feedback.", "Finished the deployment in 38 seconds", "The deployment result is SUCEEDED. See the UrbanCode Deploy deployment logs for details : https://dev-showcase-demo-x86:8443/#applicationProcessRequest/1902d36c-baf2-9535-6186-d8117f5de274", "Starting Application Property Fetching...", "APPLICATION ID is 19018dd9-56ad-43b7-b886-40a8638642bc", "***** isSkipProps value is false", "End Application Property Fetching.", and "Finished: SUCCESS".

7.2 Checking UrbanCode Deploy logs

7.2.1 ► Open a new Tab on Firebox. And start UCD console using the Bookmarks ToolBar below:



7.2.2 ► Using **admin** and password **admin** and click **Login**

UrbanCode Deploy

v. 7.2.3.0.ifix01.1140053

Log in to your account

Username

Password

Remember Me

Log In

urban{code}

© Copyright IBM Corp. 2011, 2017.
© Copyright HCL Technologies Limited 2018, 2022.

7.2.3 ► Click on **Applications** and find your application **J05_Mortgage_zOS_POT** and click on it

Name	Template	Description
J04 Mortgage zOS and Worklight		J04 CICS + JKE Mobile
J04_Mortgage_zOS_POT		Deploy J04P for DevOps PoT
J05 Mortgage zOS and Worklight		J05 CICS + JKE Mobile
J05_Mortgage_zOS_POT		Deploy J05P for DevOps PoT
J06 Mortgage zOS and Worklight		J06 CICS + JKE Mobile

Component	Status
Build Status	Success
Deployment Status	Success
Logs	Success
Metrics	Success
Logs	Success
Metrics	Success
Logs	Success
Metrics	Success

7.2.4 ► Click on the **History** tab

The screenshot shows a web-based application management interface. At the top, it displays the application name "Application: J05_Mortgage_zOS_POT" and a "Show details" link. Below the application name is a navigation bar with several tabs: Environments, History, Configuration, Components, Blueprints, Snapshots, Processes, Calendar, and Changes. The "History" tab is highlighted with a red circle and a red arrow pointing to it from below. Underneath the tabs, there is a message: "Drag environments by their names to re-order them. 2 Environments". Below this message are two search input fields: "Search by Name" and "Search by Blueprint", separated by the word "or". To the right of these fields is a dropdown menu icon. At the bottom left of the search area is a checkbox labeled "Show Inactive Environments". On the far right, there is a blue "Create" button.

7.2.5 ➔ Click on 3 dots icon and select **View Request** entry that is related to your deploy (check the date/time)

Process	Environment	Snapshot	Scheduled For	By	Status
Deploy J05P to CICS	Dev		6/18/2024, 4:18 PM	admin	Success 1
Deploy J05P to CICS	Dev		6/17/2024, 1:58 PM	admin	Success 2

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

Step	Progress	Start Time	Duration	Status
1. Install: "J05MortgagePOT"	1 / 1	4:18:04 PM	0:00:26	Success
↳ J05MortgagePOT	1 / 1	4:18:04 PM	0:00:26	Success
↳ Deploy J05P to CICS (J05MortgagePOT 20240618-211818)		4:18:04 PM	0:00:26	Success
1. Download Artifacts for zOS		4:18:04 PM	0:00:05	Success
2. Deploy Data Sets		4:18:09 PM	0:00:10	Success
3. Generate Program List		4:18:19 PM	0:00:04	Success
4. New copy resources		4:18:23 PM	0:00:06	Success
Total Execution	1 / 1	4:18:04 PM	0:00:26	Success

7.2.7 ➔ Once the **New copy** is green, click on 3 Dots and click **View Output log** (last UCD step)

Step	Progress	Start Time	Duration	Status
1. Install: "J05MortgagePOT"	1 / 1	4:18:04 PM	0:00:26	Success
↳ J05MortgagePOT	1 / 1	4:18:04 PM	0:00:26	Success
↳ Deploy J05P to CICS (J05MortgagePOT 20240618-211818)		4:18:04 PM	0:00:26	Success
1. Download Artifacts for zOS		4:18:04 PM	0:00:05	Success
2. Deploy Data Sets		4:18:09 PM	0:00:10	Success
3. Generate Program List		4:18:19 PM	0:00:04	Success
4. New copy resources		4:18:23 PM	0:00:06	Success
Total Execution	1 / 1	4:18:04 PM	0:00:26	Success

7.2.8 Notice that **J05CMORT** program was deployed to CICS Dev environment. and a CICS NEWCOPY was succeeded.

Output Log

Working Directory /var/ucd/agent72/var/work//J05MortgagePOT

```

1 =====
2 plugin: CICS TS, id: com.ibm.ucd.plugin.cics, version: 44, release: 44.20220614-1106
3 plugin command: '/usr/lpp/java/J8.0_64/bin/java' '-cp' '/var/ucd/agent72/var/plugins/com.ibm.ucd.plugin.cics_44_bfbaf3488848d60758529719b9e044e95df18031a2be53259
4 working directory: /S0W1/var/ucd/agent72/var/work/J05MortgagePOT
5 properties:
6 PLUGIN_INPUT_PROPS=/var/ucd/agent72/var/temp/cd1edec7-0d65-4465-a0f7-e5bda7d1d4a7/input.props
7 PLUGIN_OUTPUT_PROPS=/var/ucd/agent72/var/temp/logs/cd1edec7-0d65-4465-a0f7-e5bda7d1d4a7/output.props
8 cicsplex=
9 cicsscope=
10 hostname=10.1.1.2
11 ks_location=
12 ks_password=
13 ks_type=
14 maxRetryTimes=
15 password=
16 port=1490
17 resourceNameList=J05CMORT,
18 resourceType=PROGRAM
19 retryInterval=3
20 ssl=
21 ts_location=
22 ts_password=
23 ts_type=
24 username=
25 environment:
26 AGENT_HOME=/var/ucd/agent72
27 AH_AUTH_TOKEN=*****
28 AH_WEB_URL=https://dev-showcase-demo-x86:8443
29 AUTH_TOKEN=*****
30 DS_AUTH_TOKEN=*****
31 DS_SYSTEM_ENCODING=IBM-1047
32 JAVA_OPTS=-Dfile.encoding=IBM-1047 -Dconsole.encoding=IBM-1047
33 PLUGIN_HOME=/var/ucd/agent72/var/plugins/com.ibm.ucd.plugin.cics_44_bfbaf3488848d60758529719b9e044e95df18031a2be53259a01c8723ff3ce4
34 UD_DIALOGUE_ID=cd1edec7-0d65-4465-a0f7-e5bda7d1d4a7
35 WE_ACTIVITY_ID=1902d36c-e84c-fe84-c146-b531d42bc567
36 =====
37 2024/06/18 21:18:55.045 GMT BUZCP0006I Connected to "10.1.1.2:1490" CICS TS version: 050600
38 2024/06/18 21:18:55.611 GMT BUZCP0037I Perform NEWCOPY Operation.
39 2024/06/18 21:18:55.761 GMT BUZCP0024I NEWCOPY PROGRAM "J05CMORT" succeeded.
40 2024/06/18 21:18:55.778 GMT BUZCP0029I Summary: 1 NEWCOPY request(s) succeeded, 0 NEWCOPY request(s) failed.
41

```

Close

7.2.9 You could verify the binaries generated by the build (LOADLIB) at the UCD Code station. You can have versions of executables stored under UCD.

▶ Close the Output Log and click on Components

The screenshot shows the UrbanCode Deploy interface. At the top, there is a navigation bar with tabs: Dashboard, Components (which is highlighted with a red oval), Applications, Configuration, and Processes. Below the navigation bar, the URL is shown as Home / Components. Under Components, there are two sub-options: Components (which is highlighted with a red oval) and Templates.

7.2.10 ▶ Scroll down and click on J05MortgagePOT

The screenshot shows a list of components. One component, "J05MortgagePOT", is highlighted with a red rectangle. The list includes columns for the component name, last modified date (2024-06-18 21:18:18), timestamp (20240618-211818), and last modified by (admin). There is also a delete icon (trash bin) next to each component entry.

7.2.11 ▶ Click on Versions and the version created by your Jenkins run (it is a timestamp value)

Components / J05MortgagePOT

Component: J05MortgagePOT

Created By admin
Created On 6/14/2024, 5:08 PM

Used By
[J05_Mortgage_zOS_POT](#)

[View All](#)

[Versions](#) [History](#) [Usage](#) [Configuration](#) [Calendar](#) [Processes](#) [Changes](#)

Component Versions

Version	Statuses	Type	Created By	Date	Is Importing	Description	More
20240617-185833	Any	Incremental	admin	6/17/2024, 1:58 PM	false		⋮

7.2.12 ► Expand **IBMUSER.DEMO.DEV.LOAD** and you will see the load module created by the build on Jenkins pipeline. This is the load module that will be deployed.

Artifacts
Total add: 1 members in 1 data sets

[Expand All](#) [Collapse All](#) [Show SYS properties](#) [Download All](#)

Name	Artifact Type	Deploy Type	Inputs	Last Modified	Properties
IBMUSER.DEMO.DEV.LOAD	[PDS,ADD]				
JO5CMORT		LOAD	J05MortgageCICS/cobol_cics/J05CMORT.cbl		buildcommand=IEWBLINK buildoptions=MAP,RENT,COMPAT(PMS)

7.2.13 ► Minimize the browser if you will do the last optional step (#8) or close it.

7.3 Using Jenkins Blue Ocean Plugin

This is a nice plugin. We could have used this on the Jenkins activities. Log in Jenkins

7.3.1 ► On J05P_Pipeline. Under Build History and click on the build number .

Dashboard > J05P_Pipeline >

Status
 J05P_Pipeline

- </> Changes Pipeline to build and Deploy J05P - CICS Transaction
- ▷ Build Now
- ⚙ Configure
- trash Delete Pipeline
- 🔍 Full Stage View
- ⭐ Favorite
- 🌐 Open Blue Ocean
- ⠇ Stages
- ✍ Rename
- 👀 Build Review
- ⓘ Pipeline Syntax

Build History
trend ▾

#	Date	Time	Changes
#22	Jun 18	16:17	No Changes
#21	Jun 17	13:57	No Changes
#19	Jun 17	09:24	No Changes
#17	Jun 17	08:18	No Changes

Stage View

Starting J05MortgageCICS Build and Push to UCD	Deploy J05 into DEV with UCD
46s	16min 33s
51s	39s
50s	41s
41s	15s
42s	1h 4min

7.3.2 On the Build page you select. On the left Click Open Blue Ocean

Status

Build #22 (Jun. 18, 2024, 5:17:03 p.m.)

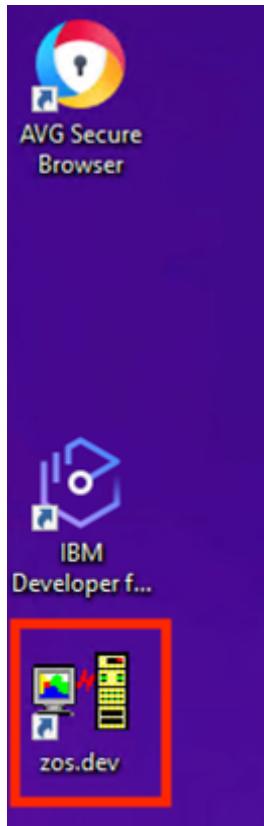
- </> Changes
- Console Output
- Edit Build Information
- Delete build '#22'
- Timings
- Open Blue Ocean**
- Build timeline
- Pipeline Overview
- Pipeline Console
- Replay
- Pipeline Steps
- Workspaces
- ← Previous Build

7.3.3 In this page shows the pipeline in another way. This plugin is handy when using “Jenkinsfile”

Triggered Builds	Duration	Timestamp
4 J05PDeploy	31s	16 hours ago

7.4 Testing the CICS code deployed to Dev environment

- 7.4.1 ➔ Double click on the **3270 terminal emulator** that is on the Windows desktop



- 7.4.2 ➔ Type **I cicsts56.** (I as logon) and press **Enter** key. (**Ctrl** in some keyboards).

Session A - [24 x 80]

File Edit View Communication Actions Window Help

Host: zos.dev Port: 23 LU Name: Disconnect

z/OS V2R5 LVL1 PUT2112/RSU2112 IP Address = 10.1.1.1
VTAM Terminal = TCP00022

Application Developer System

 // 0000000 SSSSS
 // 00 00 SS
zzzzzz // 00 00 SS
zz // 00 00 SSSS
zz // 00 00 SS
zz // 00 00 SS
zzzzzz // 0000000 SSSS

System Customization - ADCCD.Z25A.*

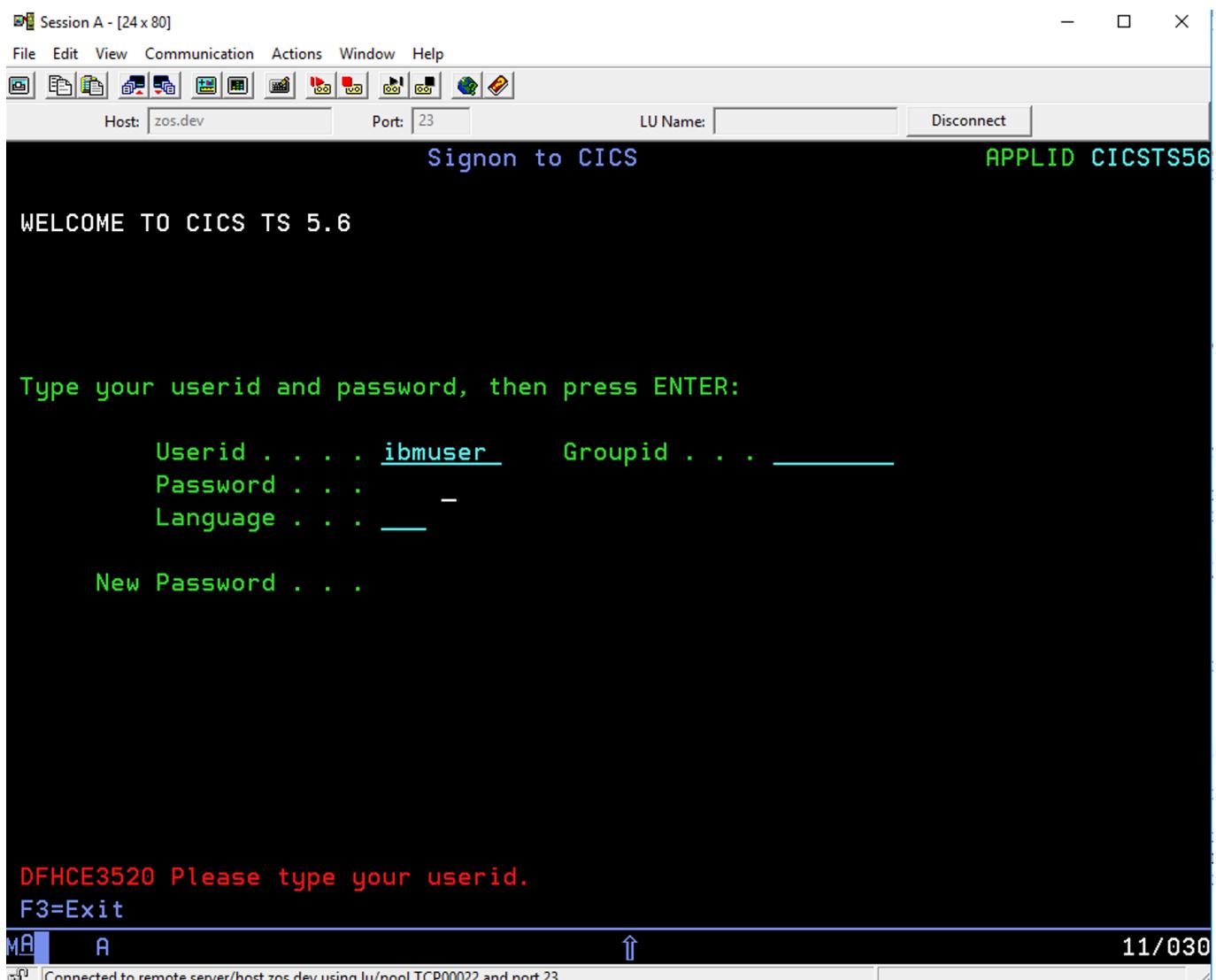
====> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
====> Enter L followed by the APPLID
====> Examples: "L TSO", "L CICSTS56", "L CICSTS55", "L IMS15"

l cicsts56

M A ↑ 24/011

Connected to remote server/host zos.dev using lu/pool TCP00022 and port 23

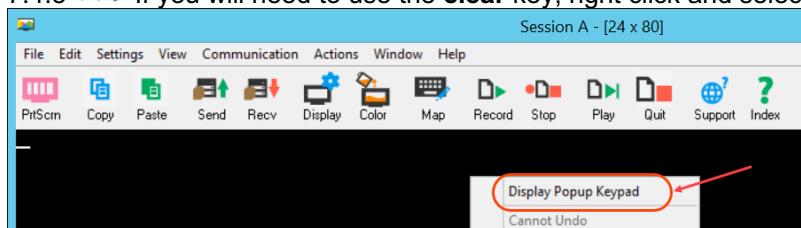
7.4.3 ► Logon using your z/OS user id and password (**ibmuser and sys1**) and press **Enter**



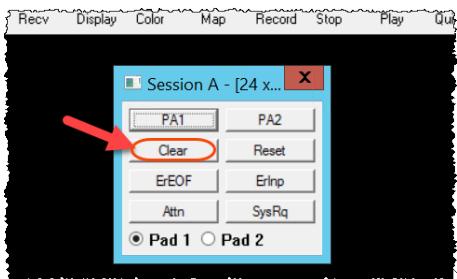
7.4.4 The sign-on message is displayed

DFHCE3549 Sign-on is complete (Language ENU).

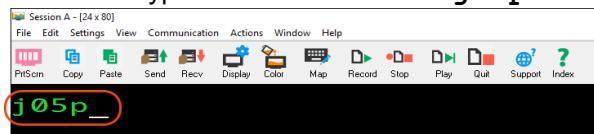
7.4.5 If you will need to use the clear key, right click and select Display Popup Keypad



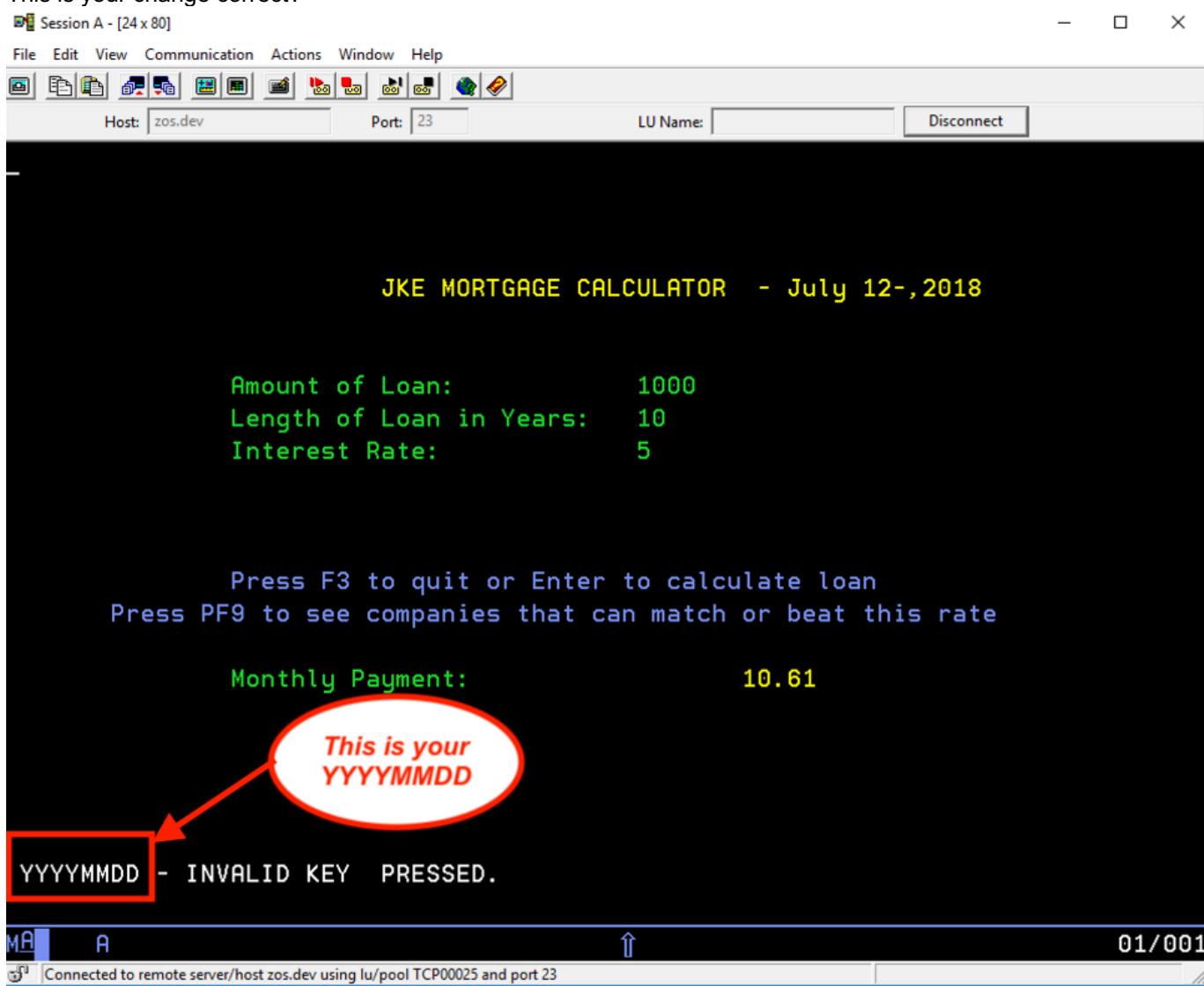
and then select Clear



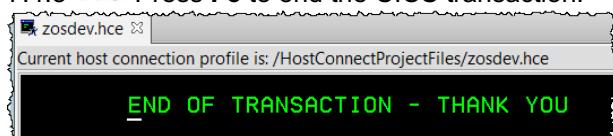
7.4.6 ➡ Type the CICS transaction **j05p** and press the **Enter** key.



7.4.7 ➡ Press **Enter** and then the **F1** key
and verify the message displayed "YYYYMMDD - INVALID KEY PRESSED".
This is your change correct?



7.4.8 ►| Press **F3** to end the CICS transaction.



zosdev.hce

Current host connection profile is: /HostConnectProjectFiles/zosdev.hce

END OF TRANSACTION - THANK YOU

7.4.9 ►| Close the terminal emulator.

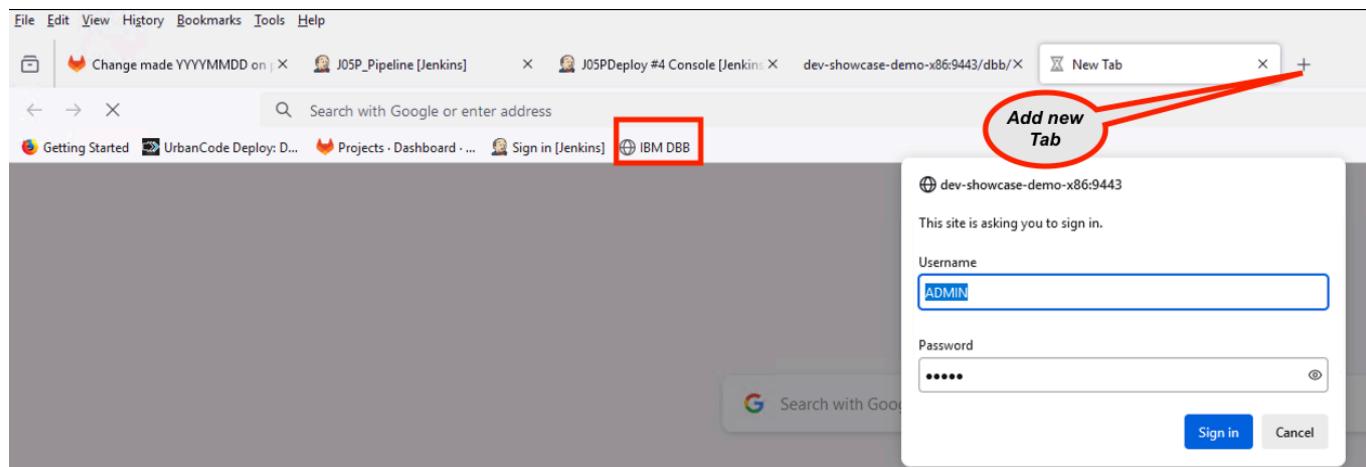
Section 8. Understanding DBB Build Reports

DBB has an application server (our is running on Linux) that capture the various build events. Here you will take a quick look of those reports.

8.1 Login to the DBB server using the browser

8.1.1 ► Using the Browser, add a new Tab (use "+"). On the new Tab click **Bookmarks Toolbar** click on **IBM DBB**

► (If necessary) Type the user **ADMIN** and password **ADMIN** (uppercase) and click **OK**.



DBB Collections

In order to use the build dependency information collected by the *DependencyScanner* for dependency resolution and impact analysis, all scanned source files (both programs and dependency files) will need their resulting logical files stored in the DBB repository database as part of a dependency **Collection**.

A collection is a repository container for logical files. The scope of a collection is determined by the user but generally a collection contains all the logical files of a Git branch. This way the logical files in a collection can use the scanned file's relative path from the *sourceDir* as a unique identifier in the collection.

Collections themselves can have any name but it is recommended to use the name of the Git branch of the source files being scanned

Collection names must be unique in the DBB repository database and an error will occur when trying to create a collection that already exists. A good practice is to first check if the collection with that name already exists before creating it

8.2 Understand the DBB Collections

8.2.1 On IBM Dependency Based Build Server. You see a list of collections Click on the arrow of the collection name **J05MortgageCICS**

Collection	Collection Owner	Latest Build	Build Owner
J05MortgageCICS	ADMIN	2024-06-18	ADMIN
▼ nazare-demo-genapp-11-force-old-policy-numbers-between-0-10-to-expire-by-2024-01-01-outputs	ADMIN	2023-06-08	ADMIN
▼ nazare-demo-genapp-11-need-to-fix-broken-ops-error	ADMIN	2022-10-27	ADMIN

8.2.2 This opens the DBB Collection details..

Click the link for the recent build

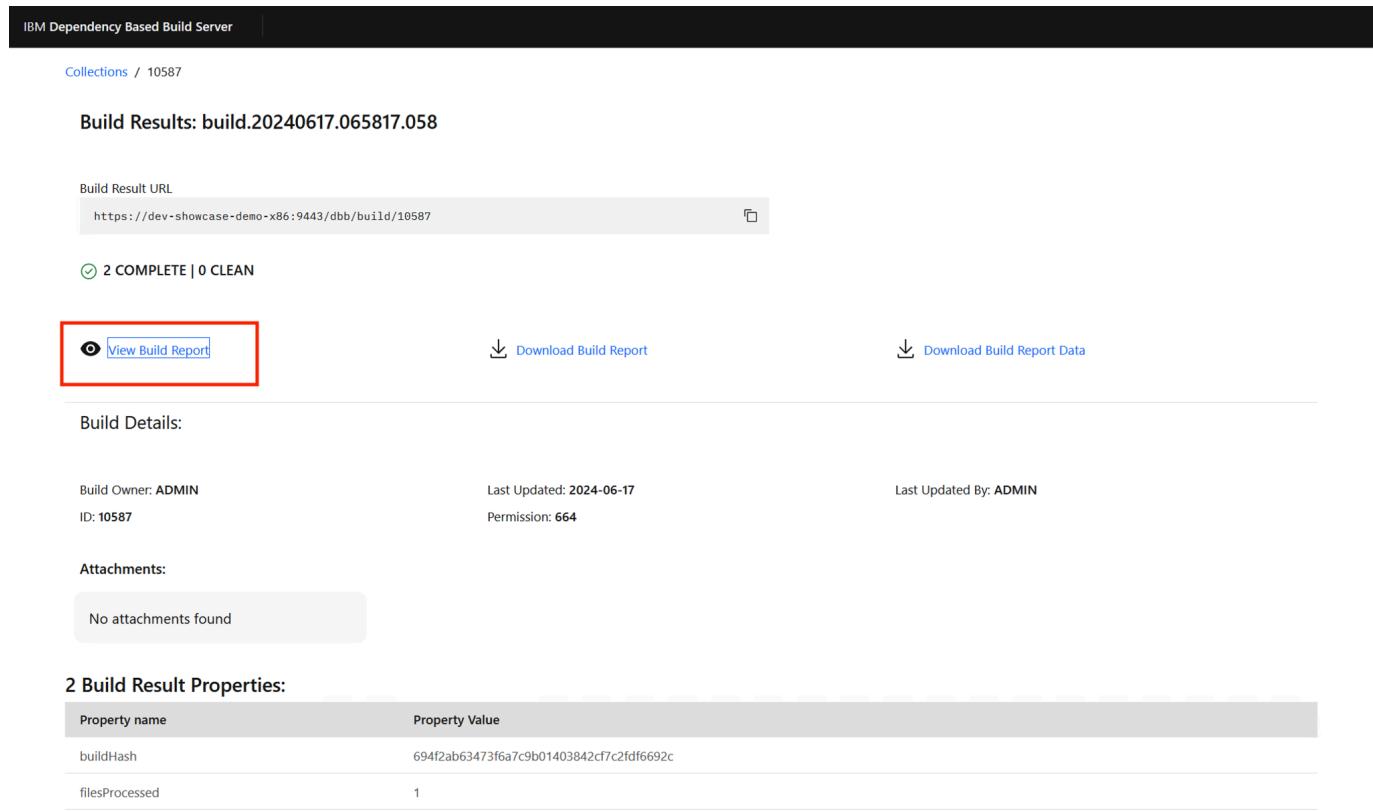
Collection	Collection Owner	Latest Build	Build Owner
^ J05MortgageCICS	ADMIN	2024-06-18	ADMIN

Build Results: 22

Build	State	Status	Build Owner	Last Update	Delete
build.20240617.011911.0	2 COMPLETE	0 CLEAN	ADMIN	2024-06-17	
build.20240617.022512.0	2 COMPLETE	0 CLEAN	ADMIN	2024-06-17	
build.20240617.065817.0	2 COMPLETE	0 CLEAN	ADMIN	2024-06-17	

8.3 Understand the Build Results

8.3.1 In this page,you see the build results. ➡ Click on **View Build Report** hyperlink,



The screenshot shows the IBM Dependency Based Build Server interface. At the top, it says "IBM Dependency Based Build Server" and "Collections / 10587". Below that, it displays "Build Results: build.20240617.065817.058". It includes a "Build Result URL" field with the value "https://dev-showcase-demo-x86:9443/dbb/build/10587" and a copy icon. Below the URL, there's a status summary: "2 COMPLETE | 0 CLEAN". There are three buttons at the top right: "View Build Report" (highlighted with a red box), "Download Build Report", and "Download Build Report Data". Under "Build Details:", it shows "Build Owner: ADMIN", "Last Updated: 2024-06-17", "Last Updated By: ADMIN", "ID: 10587", "Permission: 664", and "Attachments: No attachments found". A section titled "2 Build Result Properties:" lists two properties: "Property name" and "Property Value".

Property name	Property Value
buildHash	694f2ab63473f6a7c9b01403842cf7c2fdf6692c
filesProcessed	1

8.3.2 ➡ This opens the *Build Report*.. Notice the load module created on this build
You also may explore other hyperlinks. Click **Show dependencies**.

[Main Content](#)

Build Report

Toolkit Version:

Version: 1.1.3
Build: 151
Date: 28-Feb-2022 17:26:26

Build Summary

Number of files being built: 1

	File	Commands	RC	Data Sets	Outputs	Deploy Type	Logs
1	J05MortgageCICS/cobol_cics/J05CMORT.cbl Show Dependencies	IGYCRCTL IEWLINK	4 0	IBMUSER.DEMO.DEV.COBO(J05CMORT)		LOAD	J05CMORT.log

8.3.3 ➡ This opens the *Build Summary Report*.. Notice the files that are required for this build (dependencies)

[Main Content](#)

Build Report

Toolkit Version:

Version: 1.1.3

Build: 151

Date: 28-Feb-2022 17:26:26

Build Summary

Number of files being built: 1

	File	Commands	RC	Data Sets	Outputs	Deploy Type	Logs
1	J05MortgageCICS/cobol_cics/J05CMORT.cbl <ul style="list-style-type: none"> • J05MortgageCICS/copybook/J05NBRPM.cpy COPY • DFHAID COPY • J05MortgageCICS/copybook/J05MORT.cpy COPY • J05MortgageCICS/copybook/J05MTINP.cpy COPY • J05MortgageCICS/copybook/J05MTOUT.cpy COPY • J05MortgageCICS/copybook/J05MTCOM.cpy COPY Hide Dependencies	IGYCRCTL	4	IBMUUSER.DEMO.DEV.COBOL(J05CMORT)			J05CMORT.log
		IEWBLINK	0		IBMUUSER.DEMO.DEV.LOAD(J05CMORT)	LOAD	

8.3.4  Close the browser.

Congratulations! You have completed the Lab 3. . .



© Copyright IBM Corporation 2019.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml

Other company, product and service names may be trademarks or service marks of others.



Please Recycle
