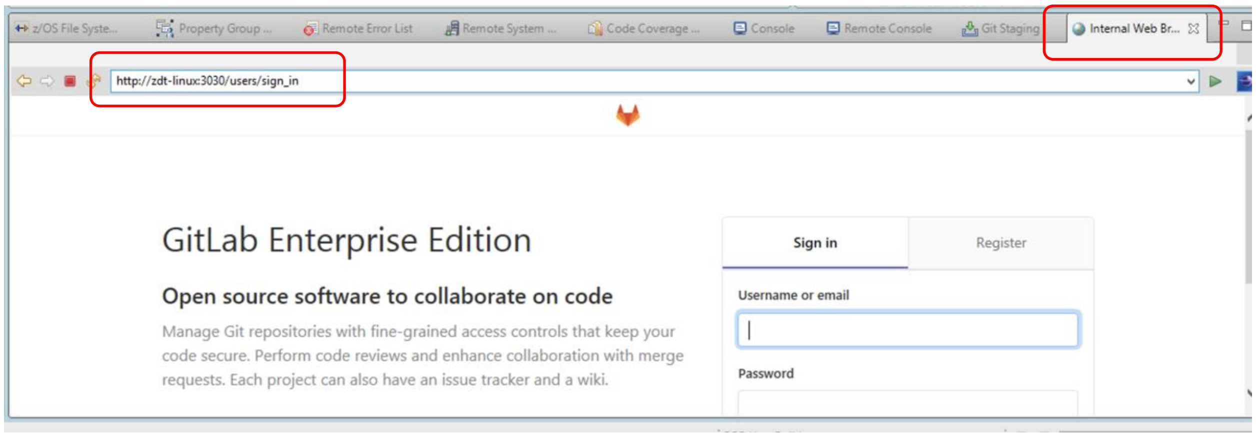
Logging into Gitlab.

1) Type the url http://zdt-linux:3030 in the url within the internal browser at the bottom of IDz.



GitLab Enterprise Edition

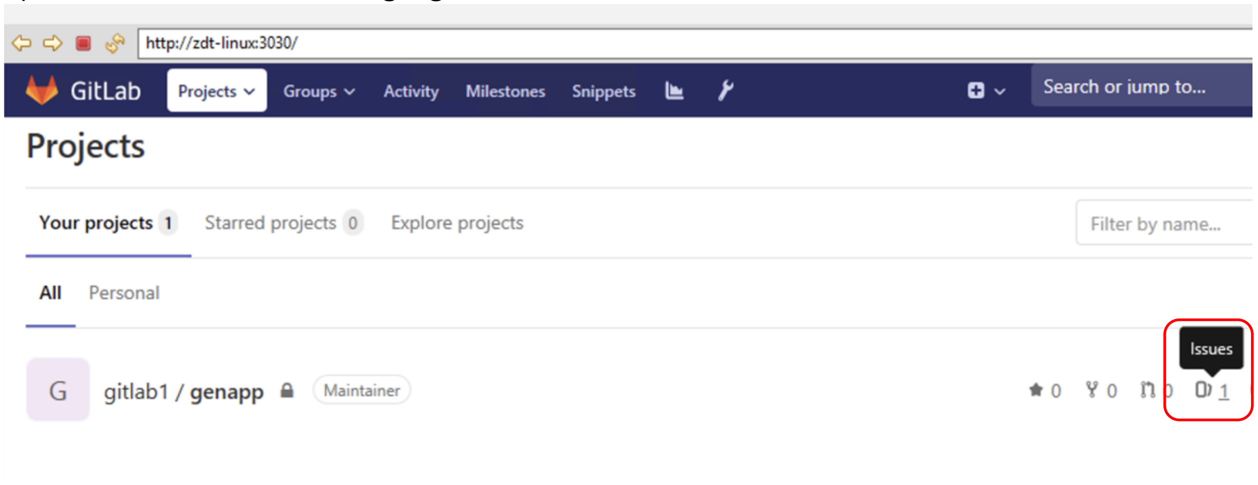
Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

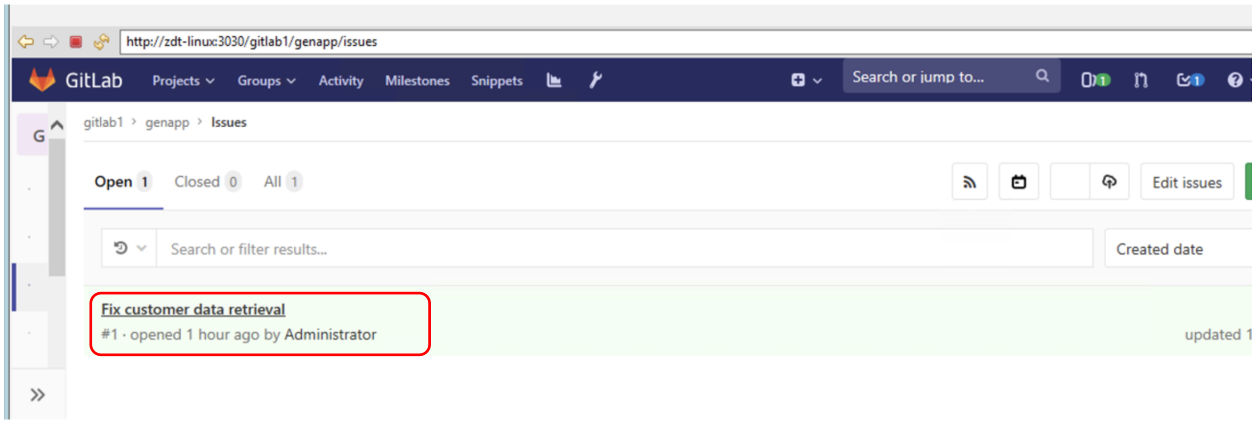
2) Login to gitlab by typing the following username and password

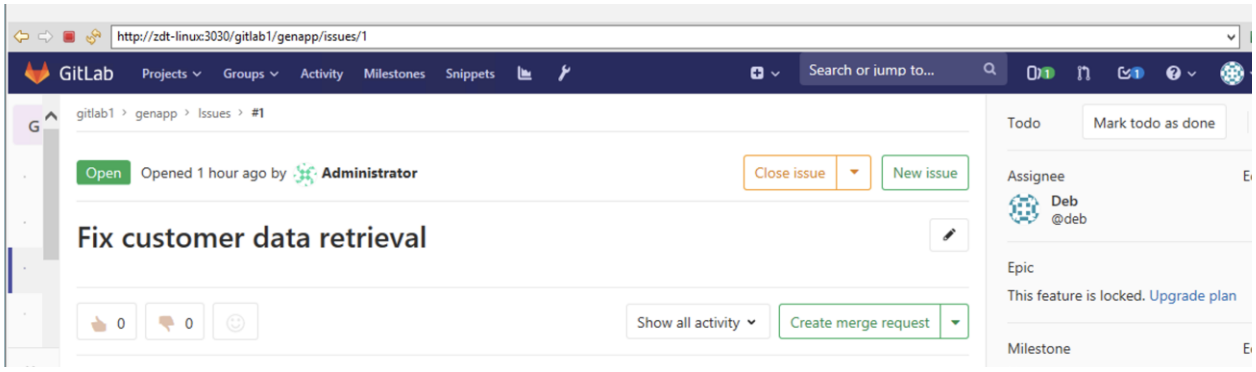
Gitlab user	: deb
pwd	: deb@zdtlinux

3) Click on the Issues icon highlighted below



4) This shows the list of assigned issues. Click on the Issue to see more details.
(There aren't any details added at this moment. Can be added in the description if required)

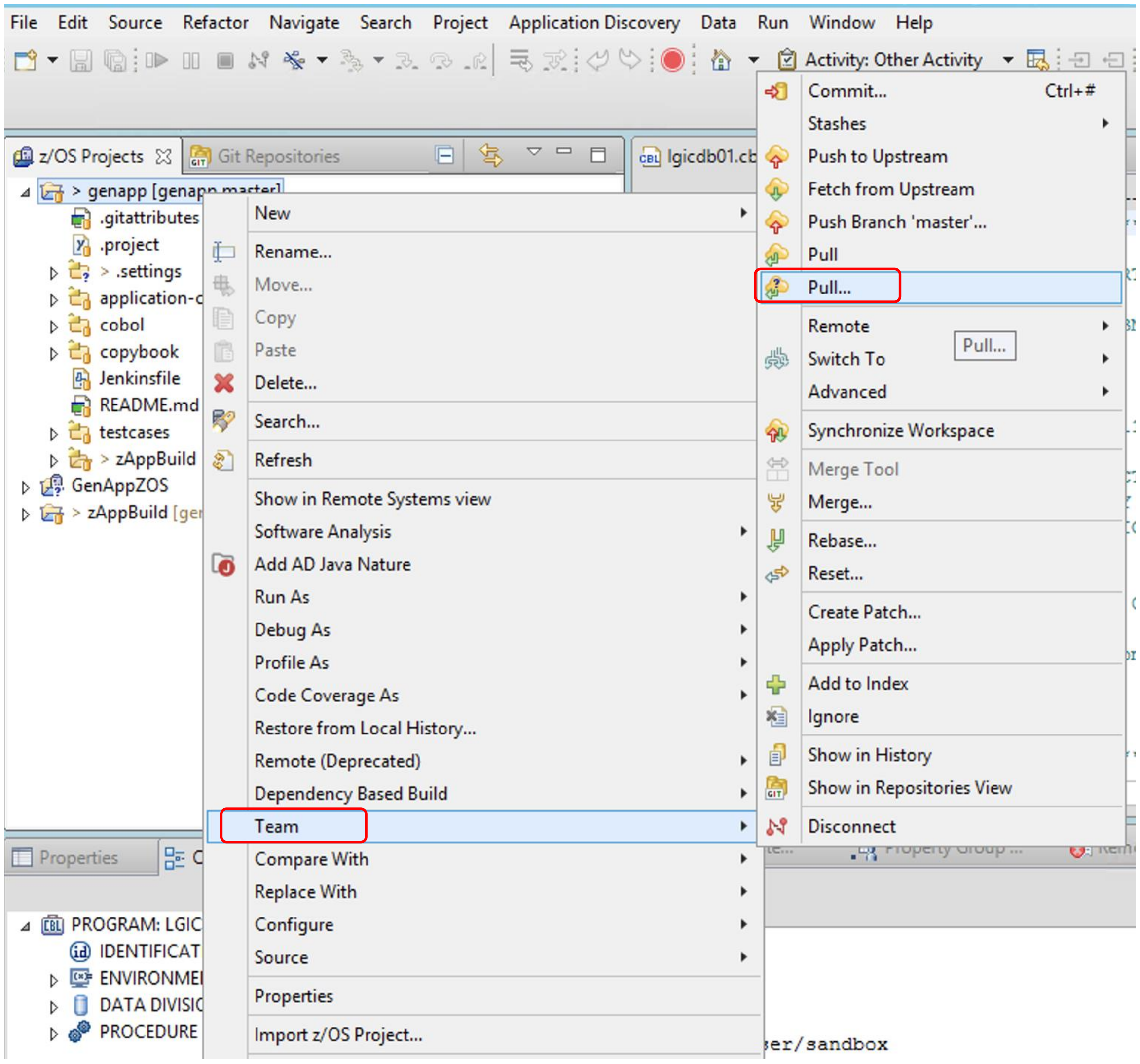

http://zdt-linux:3030/gitlab1/genapp/issues
GitLab
Projects
Groups
Activity
Milestones
Snippets
Search or jump to...
gitlab1 › genapp › Issues
Open 1
Closed 0
All 1
Edit issues
Search or filter results...
Created date
Fix customer data retrieval
#1 · opened 1 hour ago by Administrator
updated 1



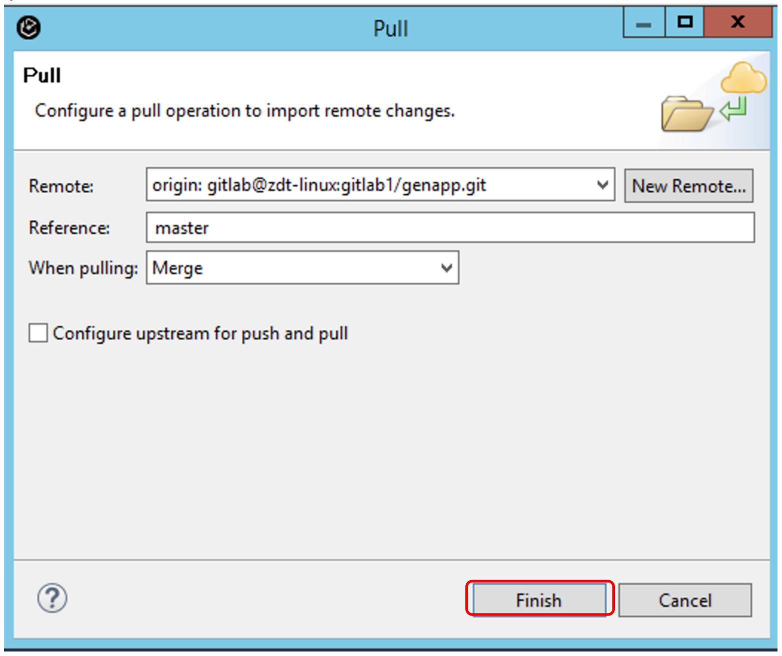
Getting started with making a source code change.

The source code of genapp from gitlab has already been cloned into IDz. This process need not be repeated. However, it is a good practice to perform a "Pull" against this code to make sure that the developer gets the latest code.

1) Right click on the project and navigate to Team -> Pull..

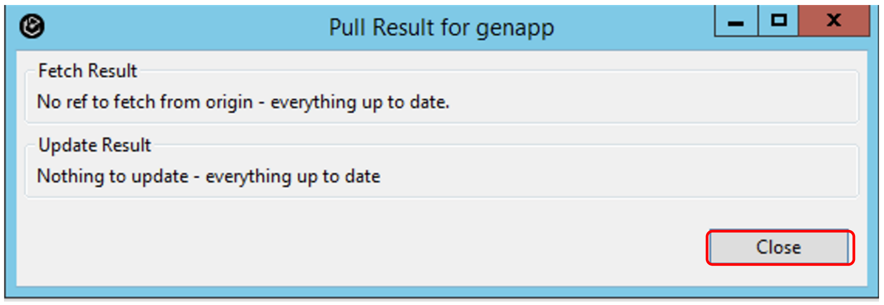


2) Click Finish on the next screen to retrieve the latest code from the master branch.



Remote: origin: gitlab@zdt-linux:gitlab1/genapp.git
Reference: master
When pulling: Merge
☐ Configure upstream for push and pull

3) The below screen shot appears if there are no new changes. However, if there are new changes, it will list the changes that have been pulled into the workspace

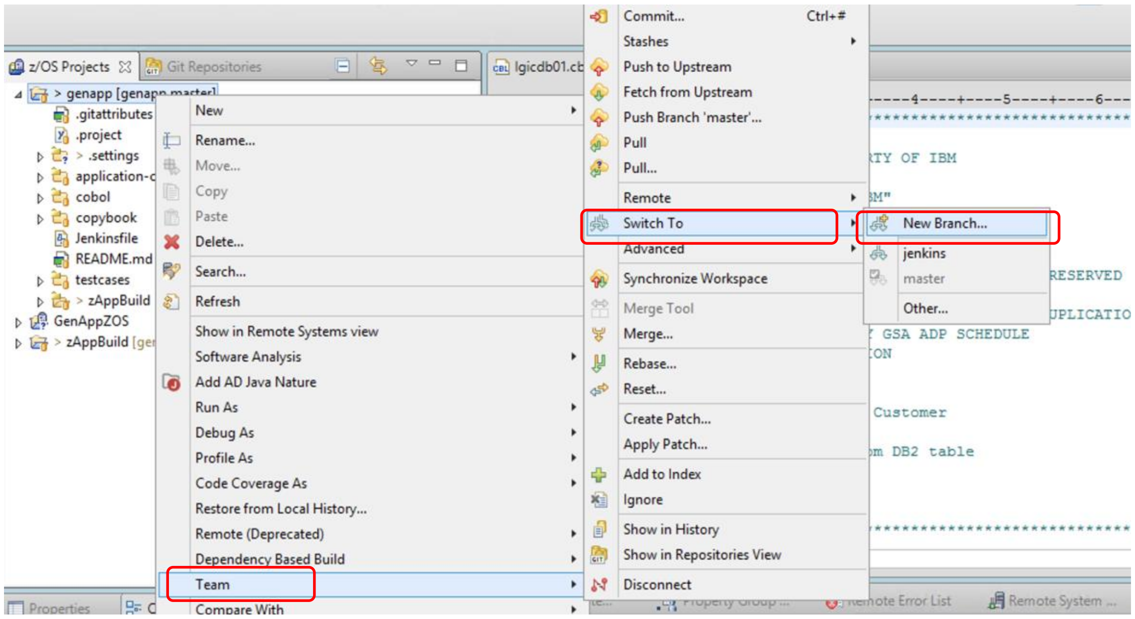


Fetch Result
No ref to fetch from origin - everything up to date.
Update Result
Nothing to update - everything up to date

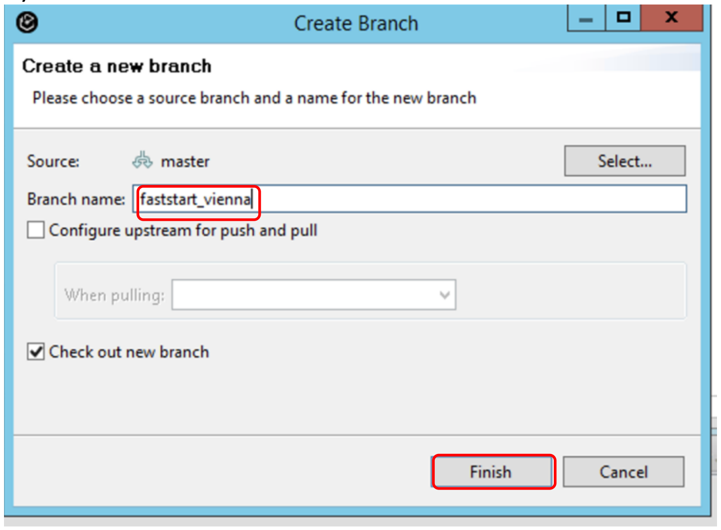
Creating a branch

One of the reasons to choose modern SCMs is to utilize their parallel development capability i.e. allowing multiple developers to work on the same applications at the same time. This could be for different releases. Hence, the first task for a developer before he/she starts coding is to create a new branch.

1) Right click on the project > Team > Switch To > New Branch



2) Provide the name of the branch and click on Finish.


Create Branch

Create a new branch

Please choose a source branch and a name for the new branch

Source: master

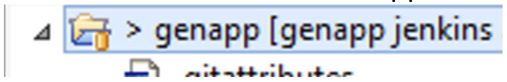
Branch name: faststart_vienna

☐ Configure upstream for push and pull

When pulling:

☑ Check out new branch

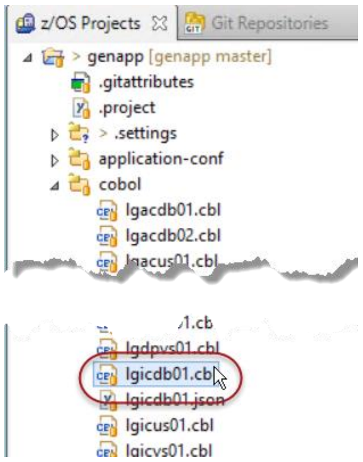
3) The name of the branch will appear next to the git project name ->



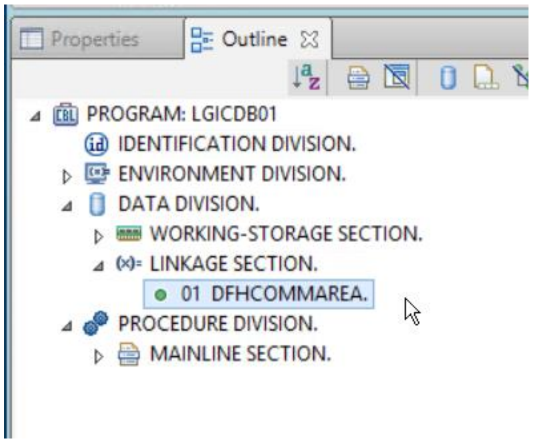
Editing a program

IDz provides unparalleled editing and analysis capabilities

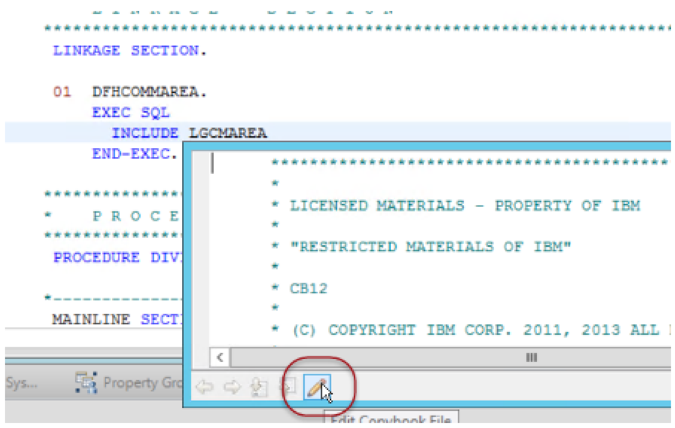
1) Double click on the program lgicdb01.dbl



2) Highlight outlive view, navigate by clicking on DFHCOMMAREA:



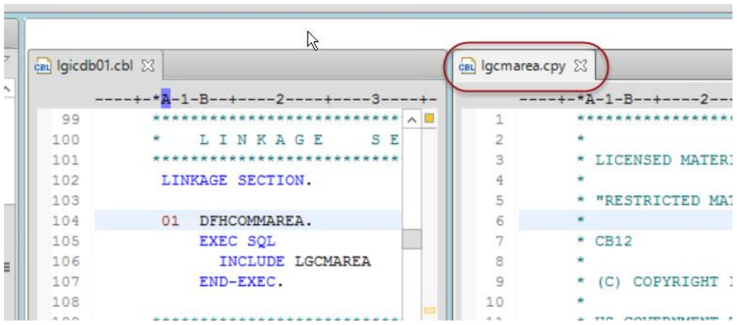
3) In editor area, hover over included SQLCA LGCMAREA member, press F2 and click edit icon to open:



```
      *****************************************************************
       LINKAGE SECTION.

       01  DFHCOMMAREA.
           EXEC SQL
             INCLUDE LGCMAREA
           END-EXEC.

      **************
      *    P R O C E
      **************
       PROCEDURE DIV

      *-------------
       MAINLINE SECT
```

```
      ******************************************
      *
      * LICENSED MATERIALS - PROPERTY OF IBM
      *
      * "RESTRICTED MATERIALS OF IBM"
      *
      * CB12
      *
      * (C) COPYRIGHT IBM CORP. 2011, 2013 ALL
```

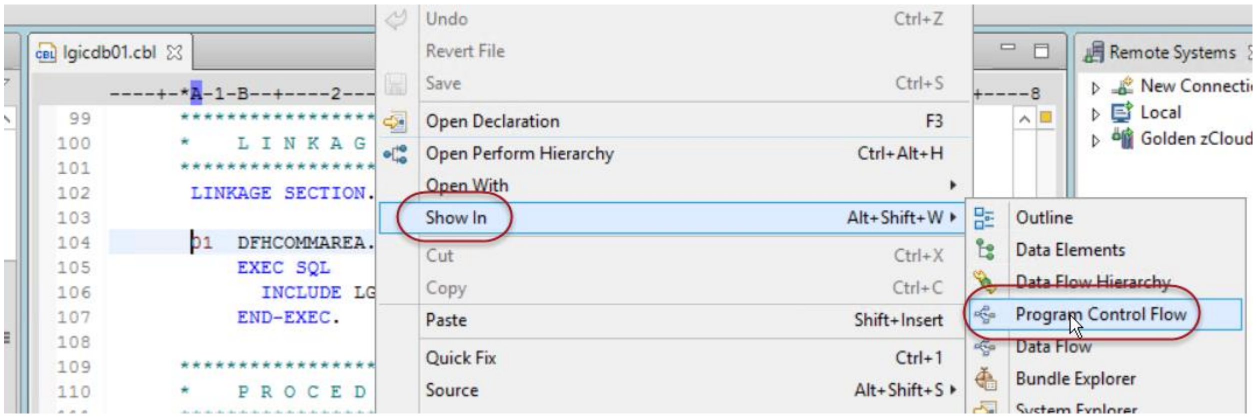
4) Drag lgcmarea.copy tab to the right to create two windows:



```
      ----+-*A-1-B--+----2----+----3----+-
      ***************************
      *    L I N K A G E    S E
      ***************************
       LINKAGE SECTION.

       01  DFHCOMMAREA.
           EXEC SQL
             INCLUDE LGCMAREA
           END-EXEC.
```

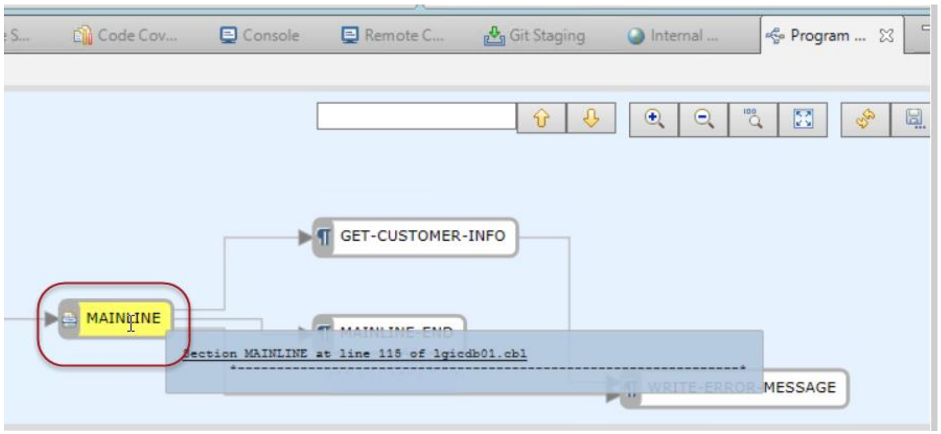
```
      ----+-*A-1-B--+----2--
      ****************
      *
      * LICENSED MATERI
      *
      * "RESTRICTED MAT
      *
      * CB12
      *
      * (C) COPYRIGHT I
      *
```

5) Right-click in lgcdb01.cbl editor area and choose -> Show In -> Program Control Flow:



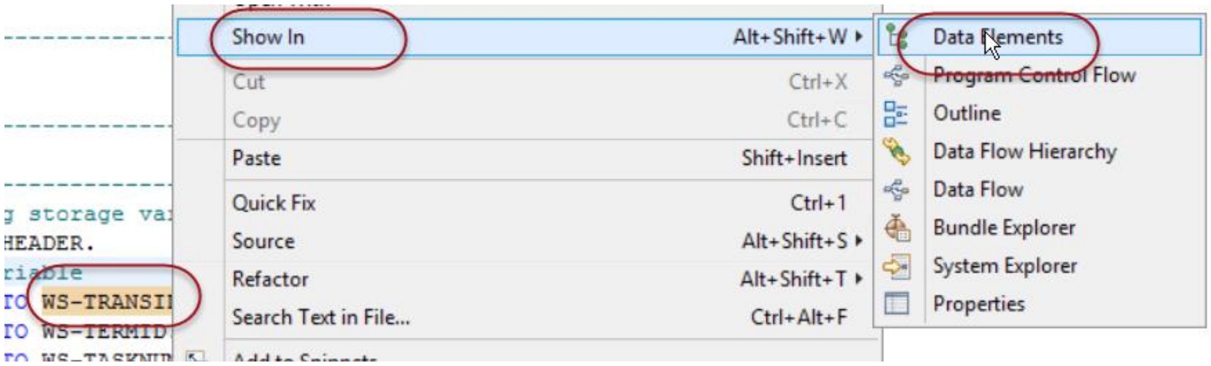
```
      ----+-*A-1-B--+----2--
      *****************
      *    L I N K A G
      *****************
       LINKAGE SECTION.

       01  DFHCOMMAREA.
           EXEC SQL
             INCLUDE LG
           END-EXEC.

      *****************
      *    P R O C E D
```

6) Click Mainline Section:



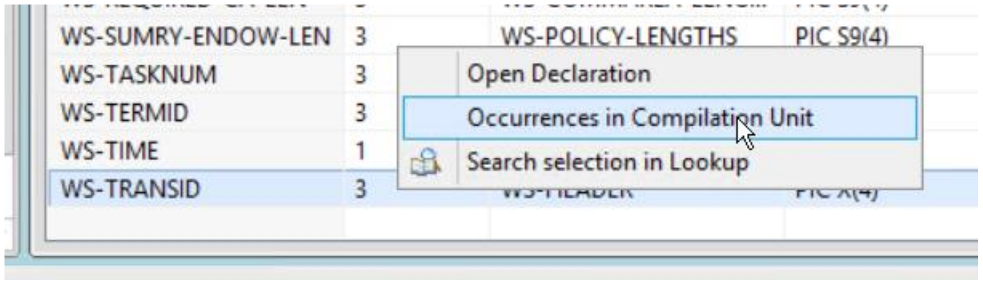
Section MAINLINE at line 115 of lgicdb01.cbl

7) Double-click WS-TRANSID to select, then right click and choose Show In -> Data Elements:



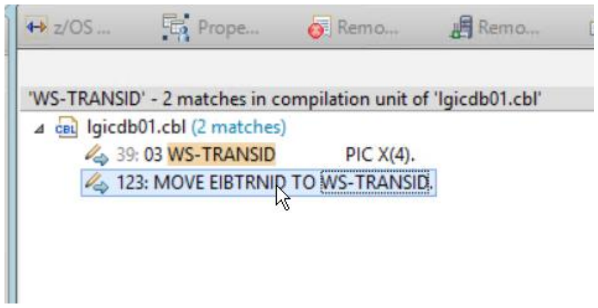
```
g storage va
HEADER.
riable
TO WS-TRANSI
TO WS-TERMID
```

8) Right-click on WS-TRANSID in the Data Elements view, and choose occurrences in compilation unit:



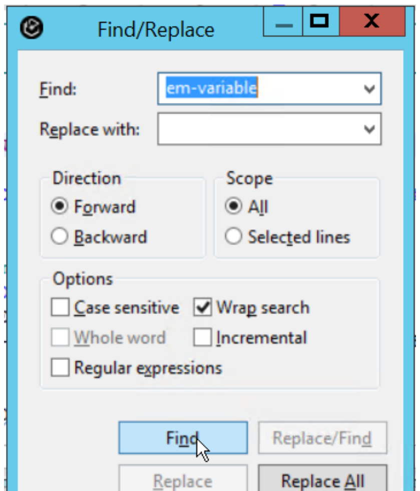
WS-SUMRY-ENDOW-LEN	3	WS-POLICY-LENGTHS	PIC S9(4)
WS-TASKNUM	3		
WS-TERMID	3		
WS-TIME	1		
WS-TRANSID	3		

9) You can navigate to any instance where WS-TRANSID is used by double-clicking:



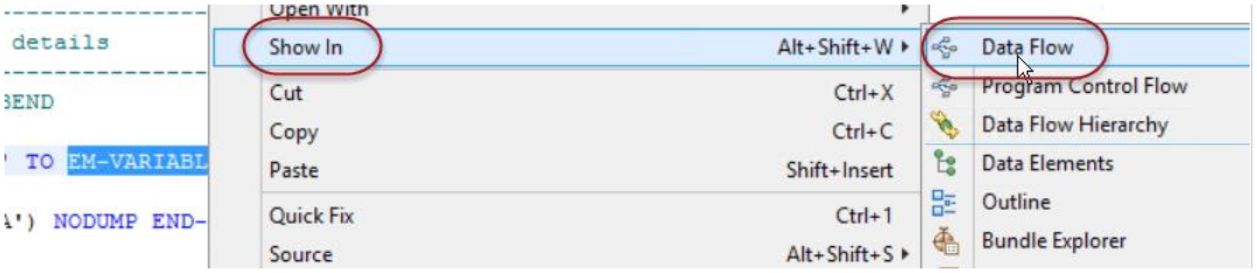
```
'WS-TRANSID' - 2 matches in compilation unit of 'lgicdb01.cbl'
lgicdb01.cbl (2 matches)
    39: 03 WS-TRANSID          PIC X(4).
    123: MOVE EIBTRNID TO WS-TRANSID.
```

10) Press CTRL-F to find string EM-VARIABLE:

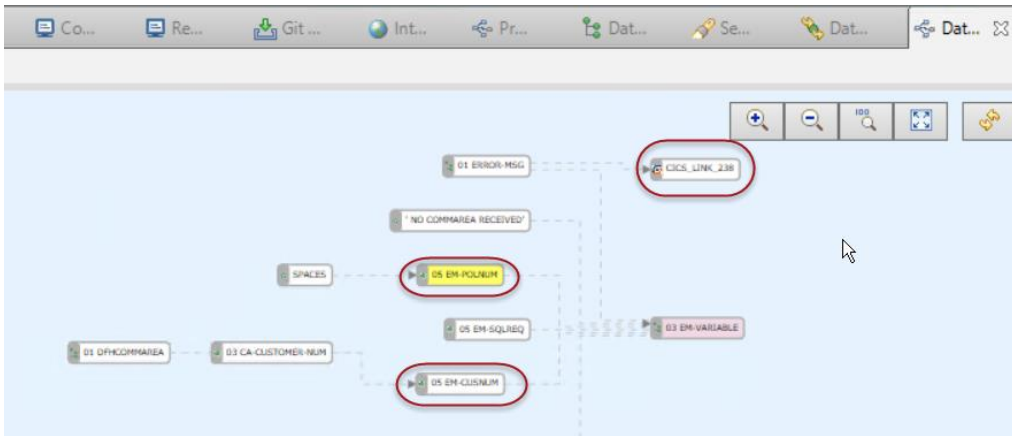


Find: em-variable
Replace with:
Direction: [✓ Forward] Backward
Scope: [✓ All] Selected lines
Options: Case sensitive, [✓ Wrap search], Whole word, Incremental, Regular expressions

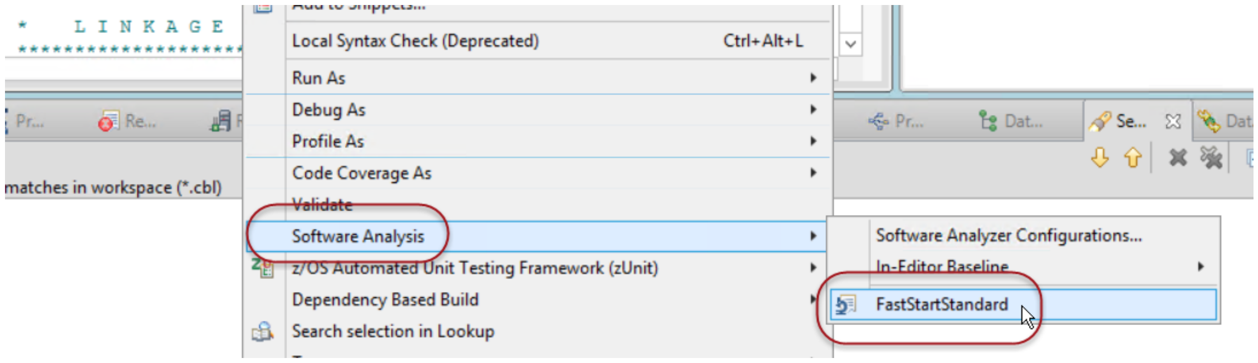
11) Double-click to select, then right-click and choose Data Flow:


```
----------------
 details
----------------
3END
' TO EM-VARIABL
A') NODUMP END-
```

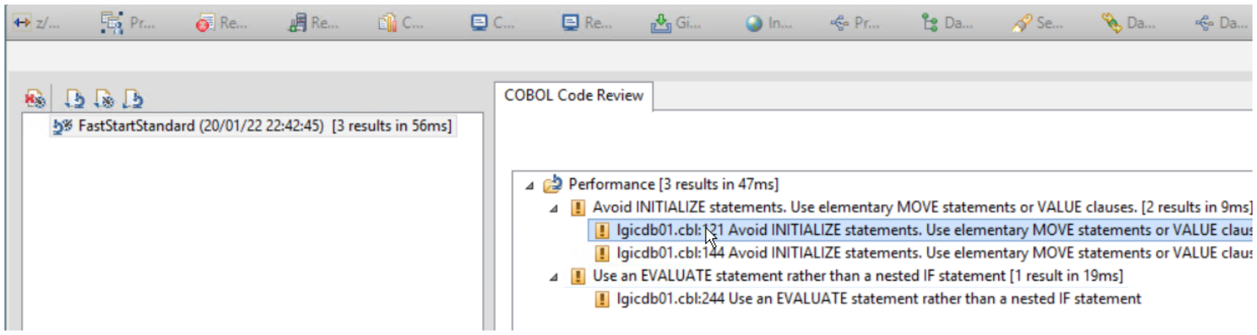
12) Navigating the objects in the diagram will take you to code and statements that modify EM-VARIABLE:



13) Right-click within lgicdb01.cbl, choose Software Analysis, and choose FastStart Standard:



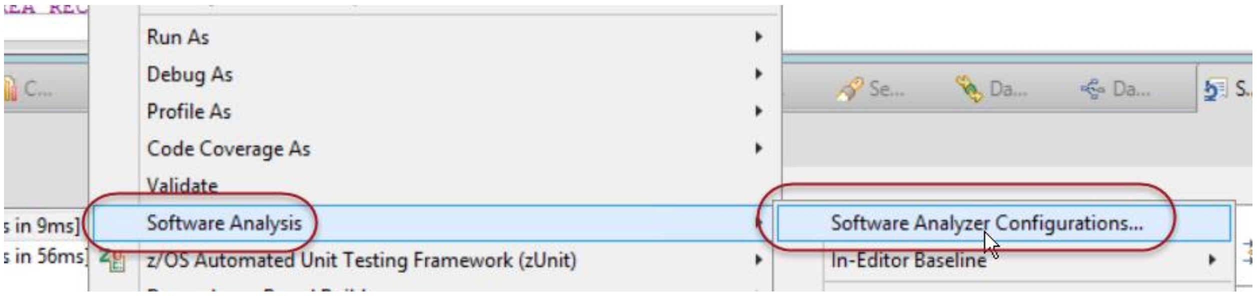
14) Code Review results will be displayed:



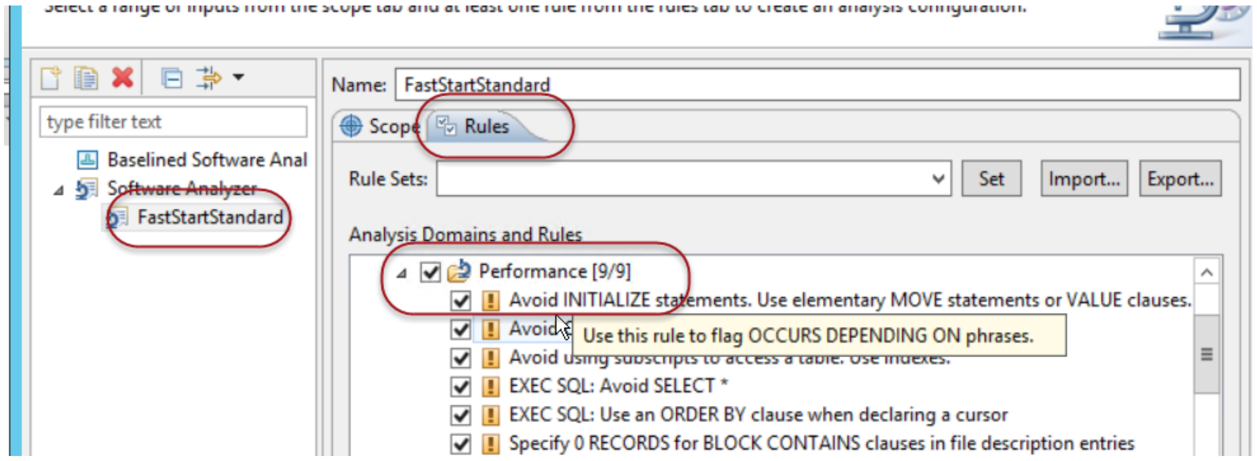
FastStartStandard (20/01/22 22:42:45) [3 results in 56ms]

```
Performance [3 results in 47ms]
  Avoid INITIALIZE statements. Use elementary MOVE statements or VALUE clauses. [2 results in 9ms]
    lgicdb01.cbl:121 Avoid INITIALIZE statements. Use elementary MOVE statements or VALUE claus
    lgicdb01.cbl:144 Avoid INITIALIZE statements. Use elementary MOVE statements or VALUE claus
  Use an EVALUATE statement rather than a nested IF statement [1 result in 19ms]
    lgicdb01.cbl:244 Use an EVALUATE statement rather than a nested IF statement
```

15) Right-click on Software Analysis again, and choose Software Analyzer Configurations:



16) Click FastStartStandard, Rules, and expand Performance. For this configuration we chose only the Performance Rules (for a Fast Start … sorry, it's late …):



Select a range of inputs from the scope tab and at least one rule from the rules tab to create an analysis configuration.

Name: FastStartStandard

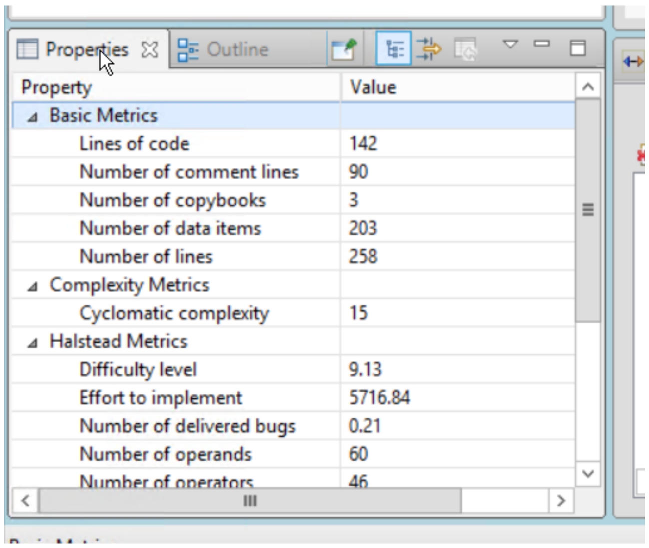
Rule Sets:

Analysis Domains and Rules

```
[✓] Performance [9/9]
    [✓] Avoid INITIALIZE statements. Use elementary MOVE statements or VALUE clauses.
    [✓] Avoid
    [✓] Avoid using subscripts to access a table. Use indexes.
    [✓] EXEC SQL: Avoid SELECT *
    [✓] EXEC SQL: Use an ORDER BY clause when declaring a cursor
    [✓] Specify 0 RECORDS for BLOCK CONTAINS clauses in file description entries
```

Use this rule to flag OCCURS DEPENDING ON phrases.

17) Related to our Code Review, the Properties View in the lower left hand corner of the workbench will display our program's metrics:

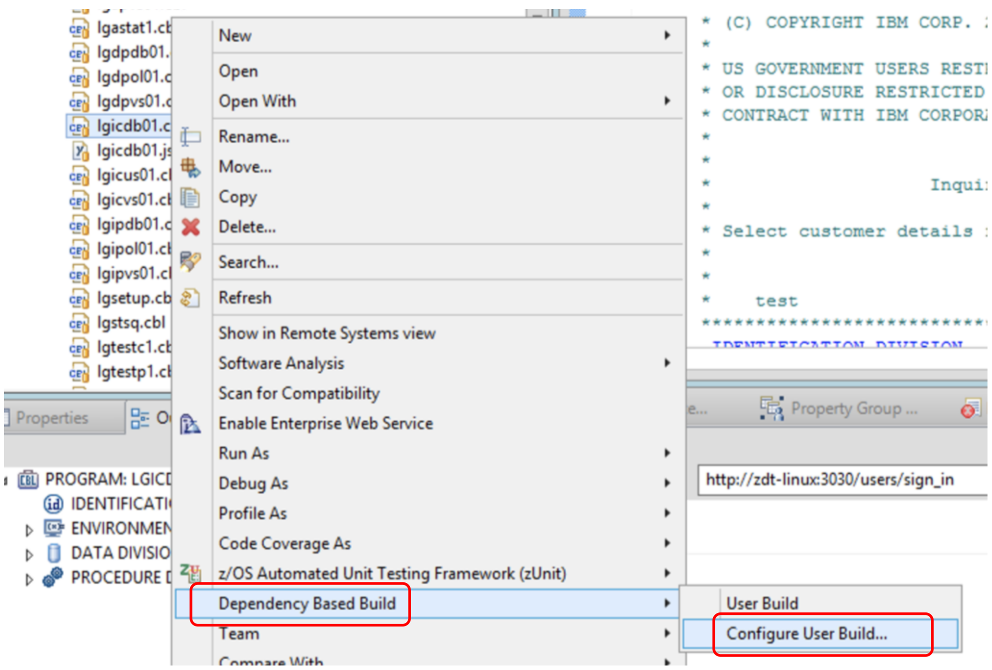


Property	Value
Basic Metrics	
Lines of code	142
Number of comment lines	90
Number of copybooks	3
Number of data items	203
Number of lines	258
Complexity Metrics	
Cyclomatic complexity	15
Halstead Metrics	
Difficulty level	9.13
Effort to implement	5716.84
Number of delivered bugs	0.21
Number of operands	60
Number of operators	46

18) **Close the program**

Performing a user build

1) Right click on the program and select Dependency Based Build > **Configure User Build..**

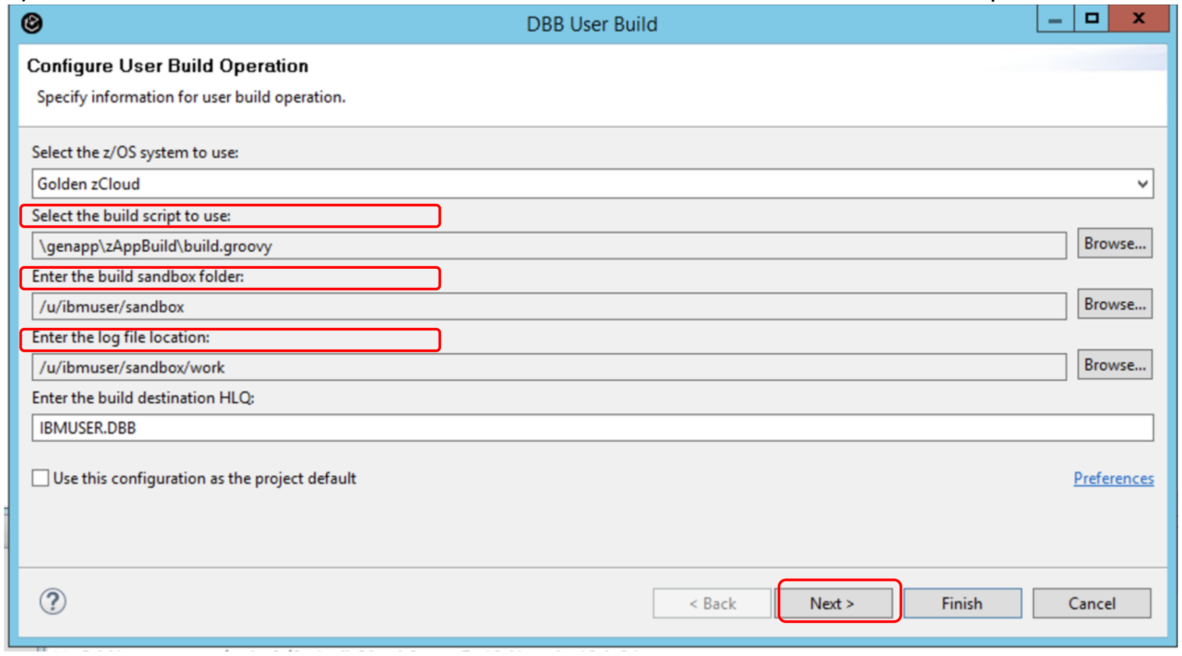


```
* (C) COPYRIGHT IBM CORP.
*
* US GOVERNMENT USERS RESTI
* OR DISCLOSURE RESTRICTED
* CONTRACT WITH IBM CORPOR
*
*
*                 Inqui
*
* Select customer details
*
*
*     test
***************************
```

http://zdt-linux:3030/users/sign_in

The above setup is to be done when invoking the user build for the first time. Once setup, it is sufficient to just click on **User Build** from the menu and not **Configure User Build..** But it is good to show this so everyone can understand how the user build works.

2) Click Next on the below screen. The different fields on this screen are explained below


DBB User Build

Configure User Build Operation

Specify information for user build operation.

Select the z/OS system to use:
Golden zCloud

Select the build script to use:
\genapp\zAppBuild\build.groovy

Enter the build sandbox folder:
/u/ibmuser/sandbox

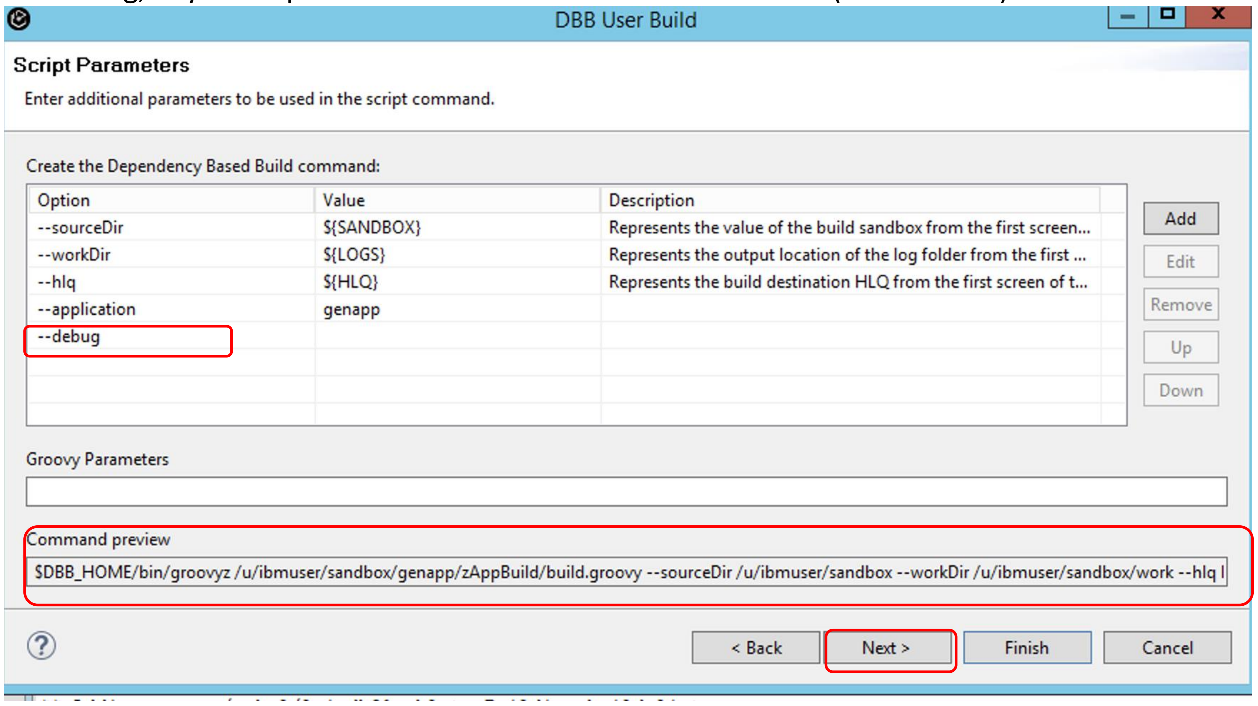
Enter the log file location:
/u/ibmuser/sandbox/work

Enter the build destination HLQ:
IBMUSER.DBB

☐ Use this configuration as the project default

a) Build script – points to the build groovy script that will run the build. In this project the build groovy script is in the folder zAppBuild. This will be explained in detail in the DBB session
b) The sandbox is a folder on the USS where the programs to be build, the dependencies, the groovy scripts will be copied over so that the build can be executed
c) The log file contains the build logs etc.
d) During the groovy build, the programs will be copied over to the MVS along with the copybooks and dependencies and compiled and linked into a LOAD PDS also in MVS. The destination HLQ refers to the HLQ for these PDSes. These will be created if it doesn't exist.

3) The following screen shows a preview of the build script that will be run. Additional options like debug, any other parameters can be added into this screen (new feature). Click on Next.



Script Parameters

Enter additional parameters to be used in the script command.

Create the Dependency Based Build command:

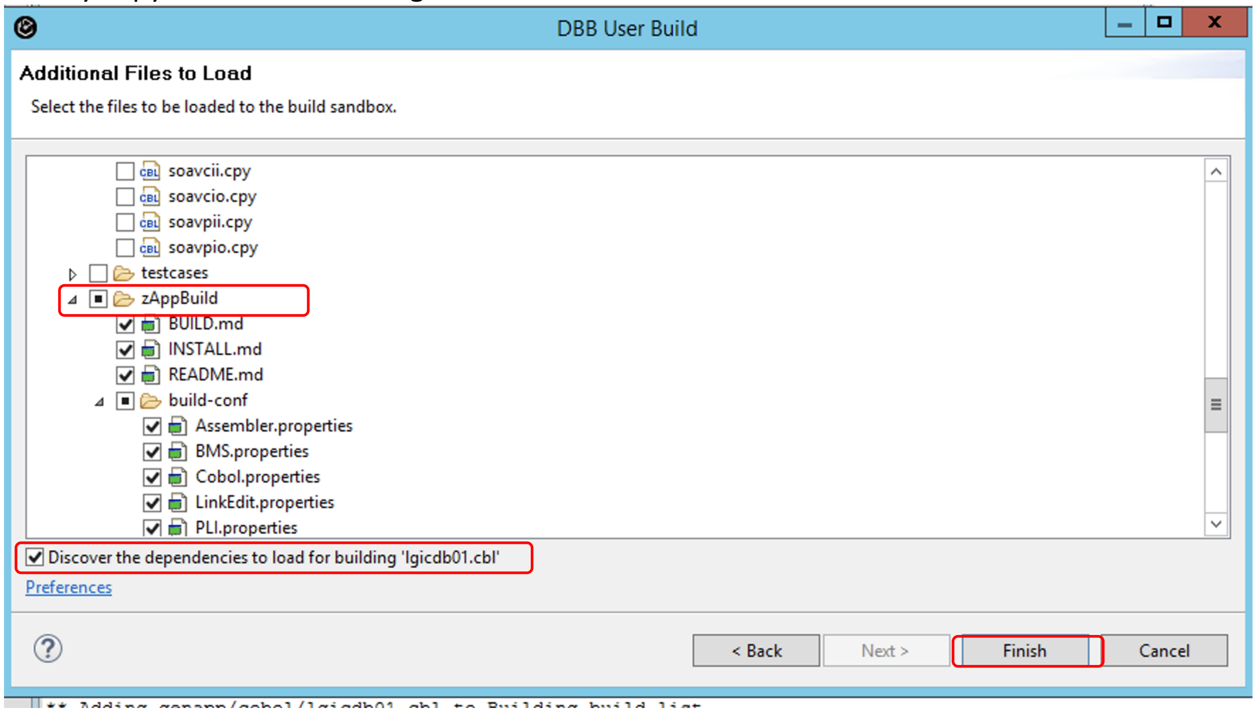
Option	Value	Description
--sourceDir	${SANDBOX}	Represents the value of the build sandbox from the first screen...
--workDir	${LOGS}	Represents the output location of the log folder from the first ...
--hlq	${HLQ}	Represents the build destination HLQ from the first screen of t...
--application	genapp	
--debug		

Groovy Parameters

Command preview

```
$DBB_HOME/bin/groovyz /u/ibmuser/sandbox/genapp/zAppBuild/build.groovy --sourceDir /u/ibmuser/sandbox --workDir /u/ibmuser/sandbox/work --hlq I
```

4) The below screen gives an option to select the files to be copied over to USS from the local workspace in the laptop. By default, the dependencies check box is selected. This allows dependent copybooks to be copied over to USS. The folders containing the build script are also checked. All these need to be checked during the first build. After which it is required to only copy over the files changed. Click on Finish.


DBB User Build

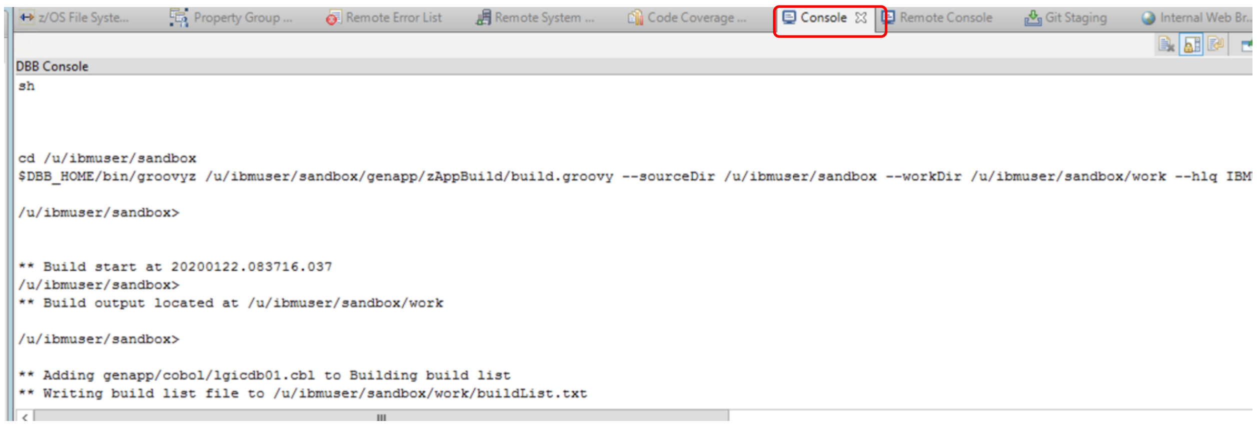
Additional Files to Load

Select the files to be loaded to the build sandbox.

- [ ] soavcii.cpy
- [ ] soavcio.cpy
- [ ] soavpii.cpy
- [ ] soavpio.cpy
- [ ] testcases
- [■] zAppBuild
  - [✓] BUILD.md
  - [✓] INSTALL.md
  - [✓] README.md
  - [■] build-conf
    - [✓] Assembler.properties
    - [✓] BMS.properties
    - [✓] Cobol.properties
    - [✓] LinkEdit.properties
    - [✓] PLI.properties

[✓] Discover the dependencies to load for building 'lgicdb01.cbl'

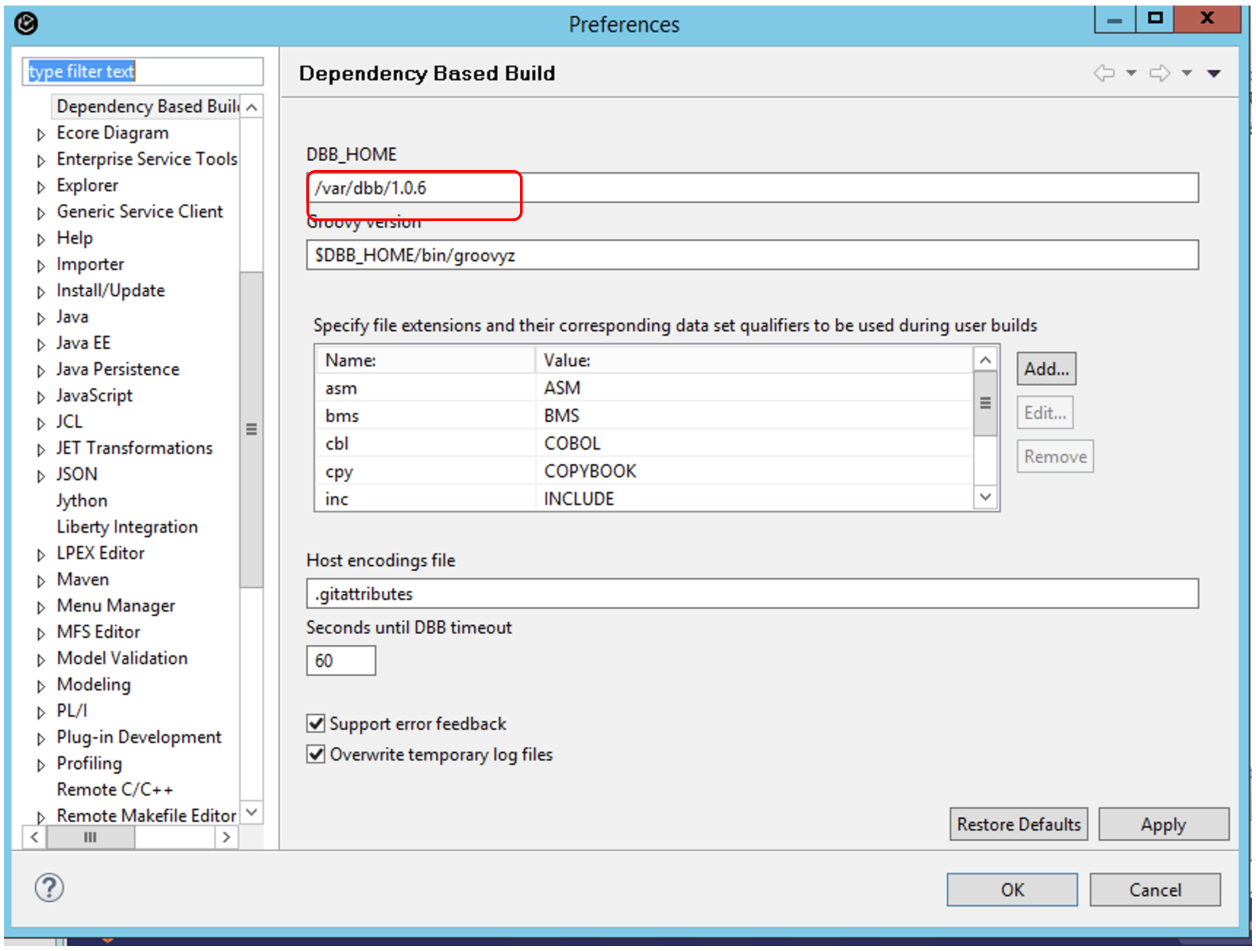
5) Finish on the above screen invokes the DBB user build. The build progress can be viewed in the console



```
sh



cd /u/ibmuser/sandbox
$DBB_HOME/bin/groovyz /u/ibmuser/sandbox/genapp/zAppBuild/build.groovy --sourceDir /u/ibmuser/sandbox --workDir /u/ibmuser/sandbox/work --hlq IBM

/u/ibmuser/sandbox>


** Build start at 20200122.083716.037
/u/ibmuser/sandbox>
** Build output located at /u/ibmuser/sandbox/work

/u/ibmuser/sandbox>

** Adding genapp/cobol/lgicdb01.cbl to Building build list
** Writing build list file to /u/ibmuser/sandbox/work/buildList.txt
```

Other preferences and settings

1) DBB preferences can be accessed from Windows > Preferences > Dependency Based Build
   For the build to work, the DBB_HOME i.e. the installation directory of the DBB toolkit has to be provided



Dependency Based Build

DBB_HOME: /var/dbb/1.0.6

Groovy version: $DBB_HOME/bin/groovyz

Specify file extensions and their corresponding data set qualifiers to be used during user builds

Name:	Value:
asm	ASM
bms	BMS
cbl	COBOL
cpy	COPYBOOK
inc	INCLUDE

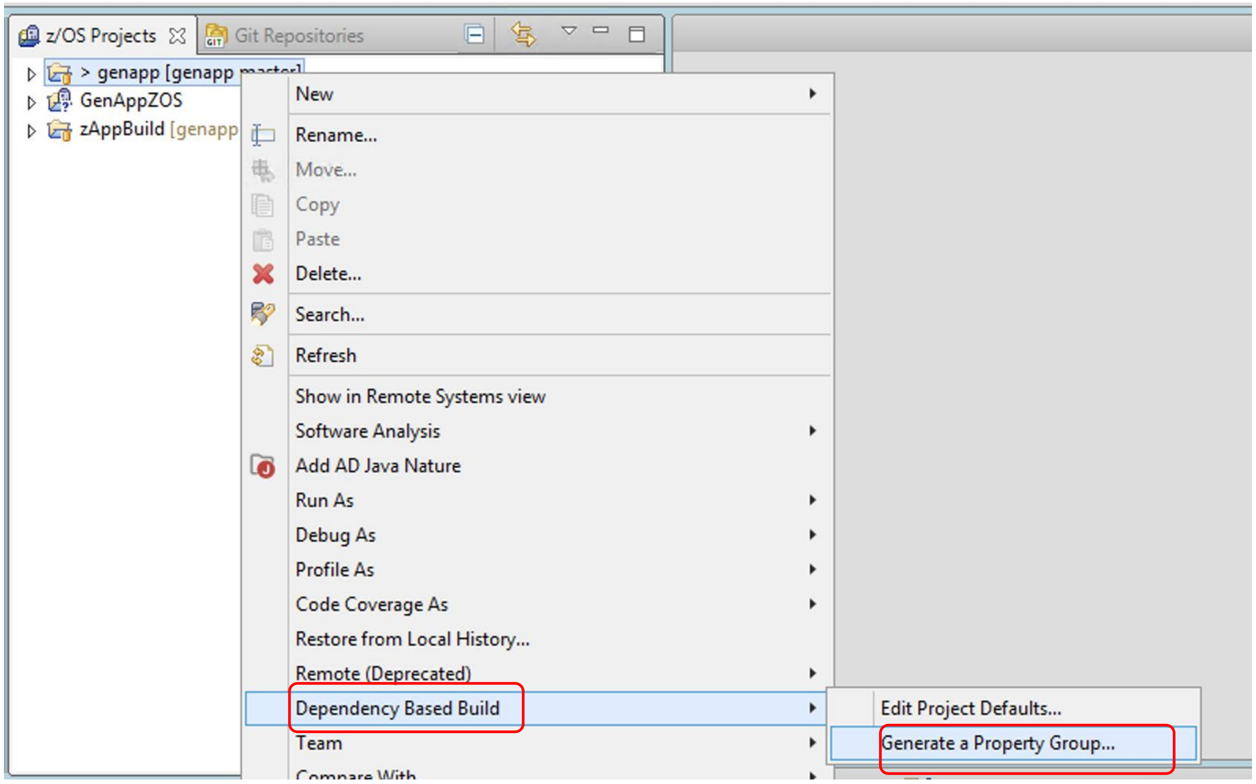
Host encodings file: .gitattributes

Seconds until DBB timeout: 60

☑ Support error feedback

☑ Overwrite temporary log files

2) When you clone a project for the first time, IDz provides an option to create a local property group for this project. This enables correct copybook detection etc. To do this, right click on the project and select **Dependency Based Build > Generate a property group ..**



3) For the zD&T Image, the following dataset needs to be deleted from TSO before the build can be run. This is a zDT issue for this instance.

```
IBMUSER.DBB.U4839255.SYSXMLSD.XML Volume USER04
```