IBM Integration Bus

Automated Build and Deploy

A Basic Example

Featuring:

Git Source Code Management
Jenkins Continuous Integration
Ant scripts
Testing Integration Service with CURL

June 2016

Hands-on lab built at product Version 10.0.0.5

1. INTRODUCTION 3

1.1 SCENARIO OVERVIEW 3
1.2 OUTLINE OF TASKS 4
1.3 INSTALLED PRODUCTS 4
1.4 REFERENCES 4

2. THE GIT SOURCE CODE MANAGEMENT SYSTEM 5

2.1 THE GIT REPOSITORY 5
2.2 ECLIPSE GIT PLUG-IN CONFIGURATION 8
2.3 CHECK THE TESTNODE_IIBUSER NODE 13

3. INVESTIGATE AND RUN THE ANT SCRIPT 14

4. AUTOMATE THE BUILD WITH JENKINS 20

5. UPDATE THE INTEGRATION SERVICE AND REBUILD 32

5.1 IMPORT THE GIT PROJECTS INTO AN IIB WORKSPACE 32
5.2 UPDATE THE INTEGRATION SERVICE 38

6. APPENDIX 44

6.1 CONFIGURE INTEGRATION BUS NODE TO WORK WITH DB2 44

7. END OF LAB GUIDE 44

1. Introduction

This hands-on lab provides a basic introduction to automated building, deployment and simple testing of Integration Bus applications.

There are many examples of this type of procedure documented on the internet, and this example is not intended to demonstrate any sophisticated scenarios, or replace any such articles. However, if automated builds with IIB are new to you, then this guide should provide some basic methods and pointers to develop more sophisticated scenarios.

This lab does not have any specific IIB function related to a specific version or release of IIB. It was designed using IIB 10.0.0.5, but the tools and techniques will work with all earlier versions (although screen captures may differ).

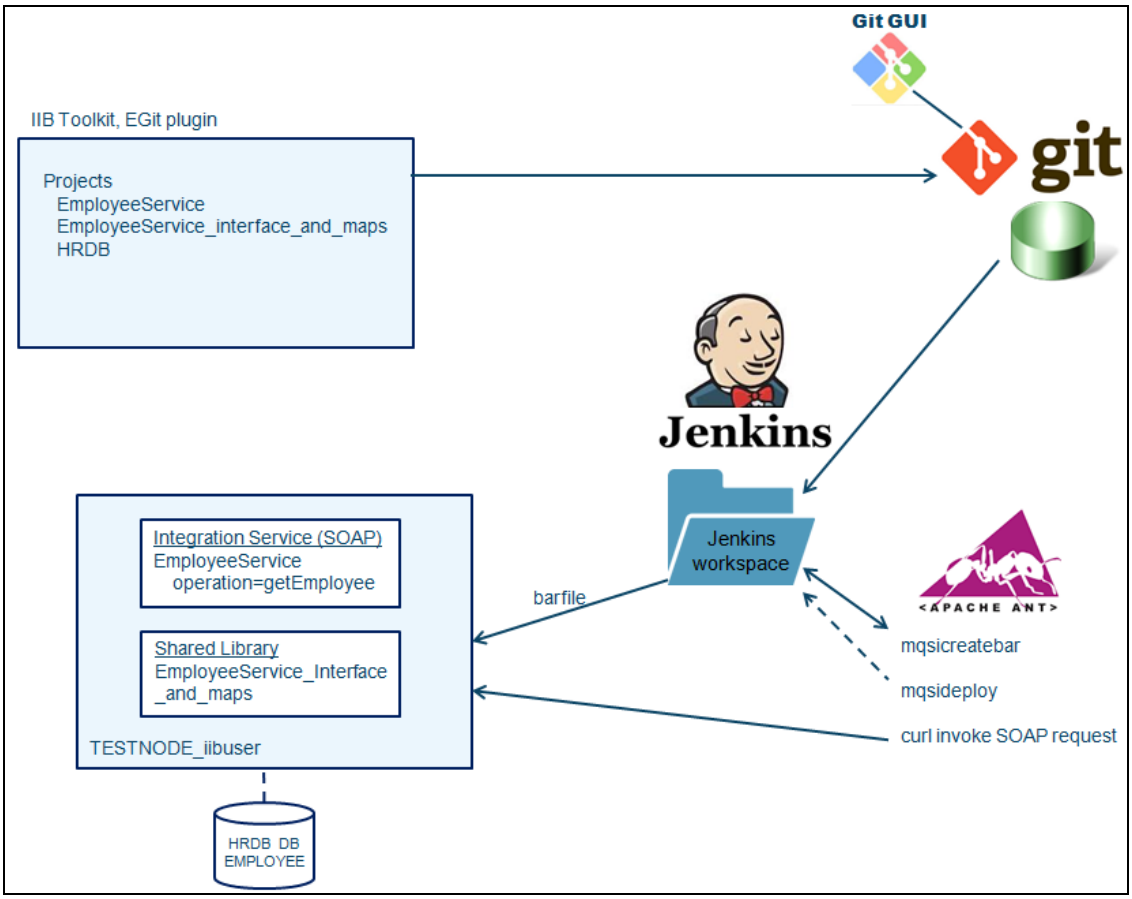
1.1 Scenario Overview

In this lab, you will use a Git repository that has been loaded with the EmployeeService application and the referenced library. You will use an ANT script to build, deploy and run a simple test. You will then combine these tools with the Jenkins build system. Finally, you will make a change to the EmployeeService, and see how this change can be automatically built and deployed.

The integration service is provided for you, and is the solution of the integration service that was developed in Lab 1 (Create Integration Service) in this series of labs.

The following schematic shows the primary components and tools that are used in this lab.

Note that the Integration Toolkit does not directly deploy IIB components to the Integration Node. All integration artefacts (projects) are stored in the Git repository. The Jenkins tool then retrieves the projects from Git, loads them into its own workspace, and invokes the supplied Ant script to build and deploy the IIB barfile. Finally, Jenkins invokes a simple "curl" command to send a SOAP request message to the deployed service.



1.2 Outline of tasks

The tasks to complete in this lab are the following:

- Investigate the Git installation and eclipse Git configuration
- Investigate and execute the supplied Ant scripts
- Investigate and configure the Jenkins installation, and execute in conjunction with Ant
- Make a change to the supplied IIB application and rebuild and retest with Jenkins and Ant

1.3 Installed products

To facilitate integration with the existing workshop lab scenarios, we have provided all the required components on a Windows environment. There are some differences in the way that these tools work in Windows, and this lab is not presented as a comprehensive guide to using these tools in this environment.

The following products and tools have been installed.

- Git
- Jenkins
- Ant
- Curl

Only basic configuration was performed, and mostly the installation was done by clicking "next" and accepting the default options. Where changes were made, we have described them in this guide.

1.4 References

Several articles have provided useful guidance in the preparation of this scenario:

- Using the EGit Eclipse plugin
  http://www.vogella.com/tutorials/EclipseGit/article.html

- Building IIB applications with Git, ANT and Jenkins on Linux
  https://developer.ibm.com/integration/blog/2015/10/02/continuous-build-and-deploy-automation-with-ibm-integration-bus-v10-using-ant-git-and-jenkins/

- Using ANT for Integration Bus applications
  http://blogs.perficient.com/ibm/2013/08/27/automated-build-and-deploy-in-websphere-message-broker-using-ant/

- Use of CURL for SOAP requests
  http://superuser.com/questions/149329/what-is-the-curl-command-line-syntax-to-do-a-post-request

  http://stackoverflow.com/questions/84007/curl-command-line-for-consuming-webservices

This lab is intended to be used in the classroom environment with the iibuser login.

Make sure that you are logged in to Windows as iibuser (password = passw0rd).

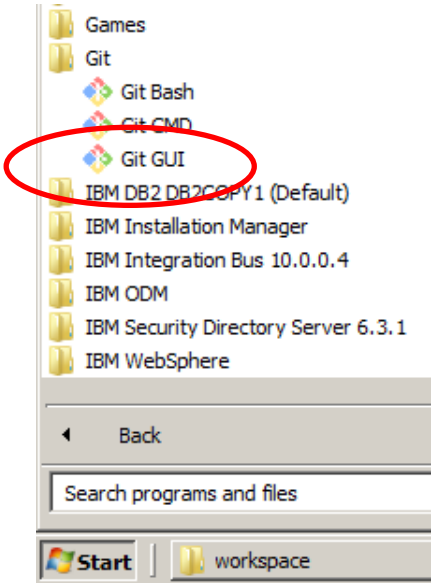
2. The Git Source Code Management System

2.1 The Git Repository

In the pre-built system, the Windows system has already been configured with a Git repository. The repository is located at c:\student10\build\GitRepo. To make the workshop scenario easy to manage, the repository was created by iibuser, and no specific security has been defined for this repository. This is unlikely to be good enough for a production system, and is not recommended as a general approach.

The Git repository has been primed with solution copies of the EmployeeService integration service and the EmployeeService_interface_and_maps library. This library also references the HRDB project, which is also in the Git repository. Take a quick look at the Git repository now.

1. From the Start menu, open Git, and select "Git GUI".

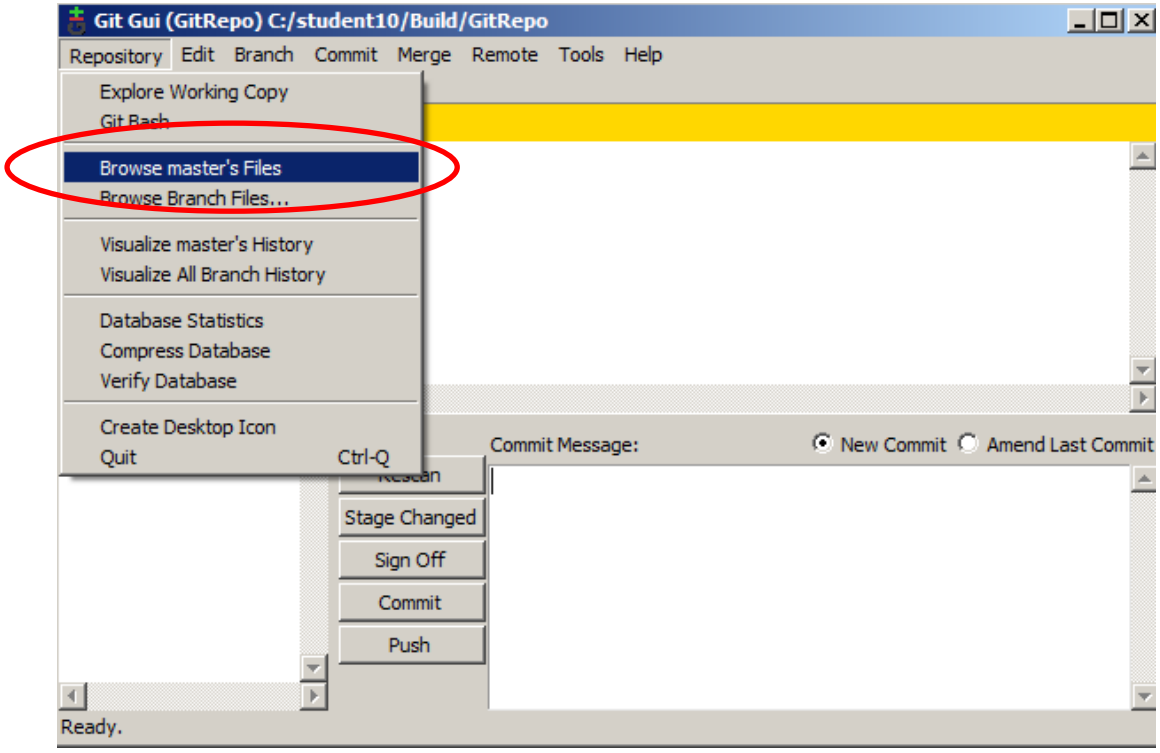


2. In Git GUI, click the recently opened repository **c:/student10/Build/GitRepo**, or use Open Existing Repository, if not shown in the recent items list.

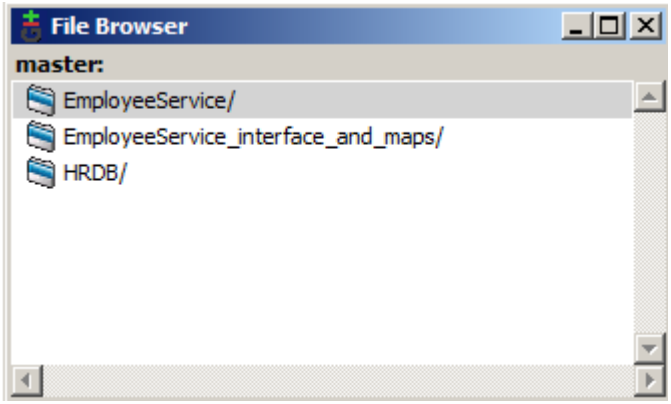
   (Note, Git Gui can also be used to create a new Git repository).

3. Select **Repository** and then **Browse master's files.**

   This will show the projects and files that have been stored in the Gut repository.

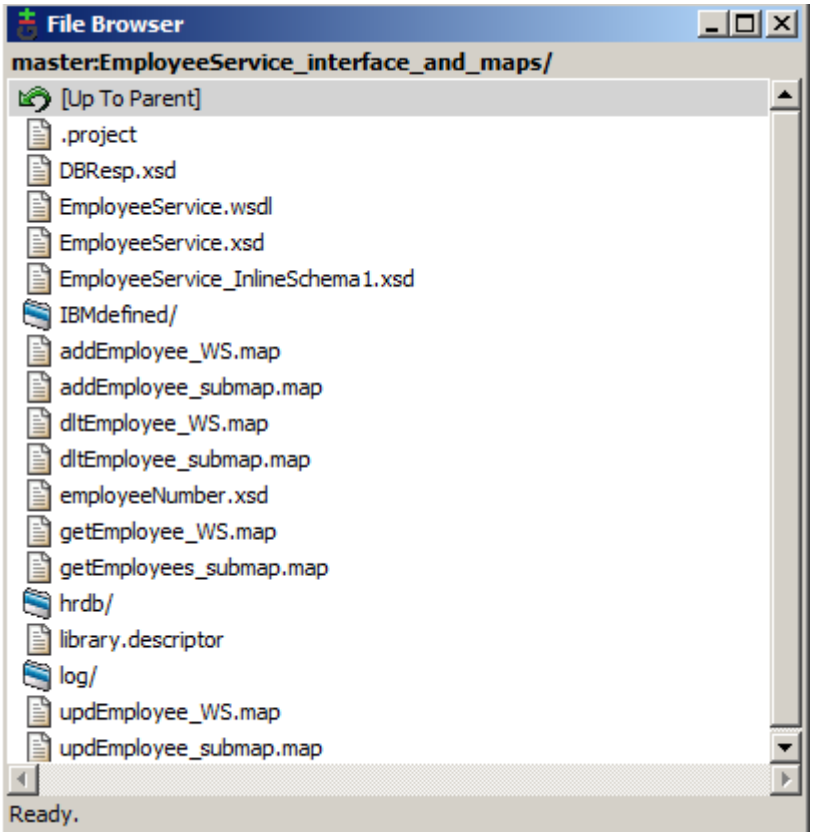


4. You will see that three folder have been stored in the Git repository. These correspond to the Eclipse projects that contain the IIB applications.



5. Double-click one of them. You will see all the files that constitute that particular project.

   Close this window when complete.



2.2 Eclipse Git Plug-in configuration

The provided system has installed the EGit Eclipse plugin into the IIB Eclipse Toolkit. This was installed in the usual way from the Eclipse Marketplace. Most of the EGit installation has been left with the default configuration. However, to make the use of EGit easier in a classroom environment, some minor configuration has been done.

Note - if the IIB product is uninstalled, or upgraded, the EGit plugin will have to be reinstalled.

For reference, the website http://www.vogella.com/tutorials/EclipseGit/article.html has provided some useful information and techniques for this combination of tools. (Note - IBM does not endorse any one provider of this type of education or material).

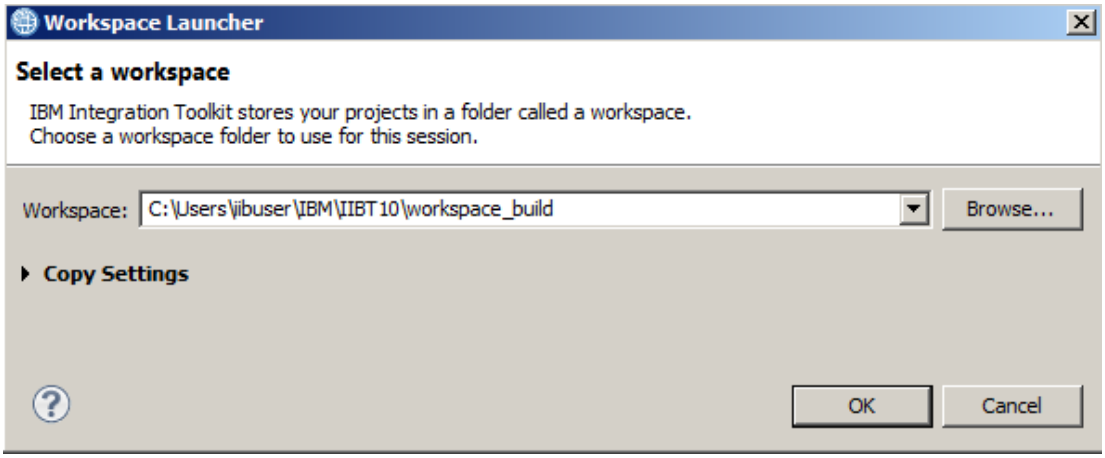
The configuration that has been done is reviewed here:

1. To make sure there is no confusion with existing projects, create a new IIB workspace.

   In the IIB Toolkit, use File, Switch workspace, and provide the name

   `c:\Users\iibuser\IBM\IIBT10\workspace_build`

   Click OK.



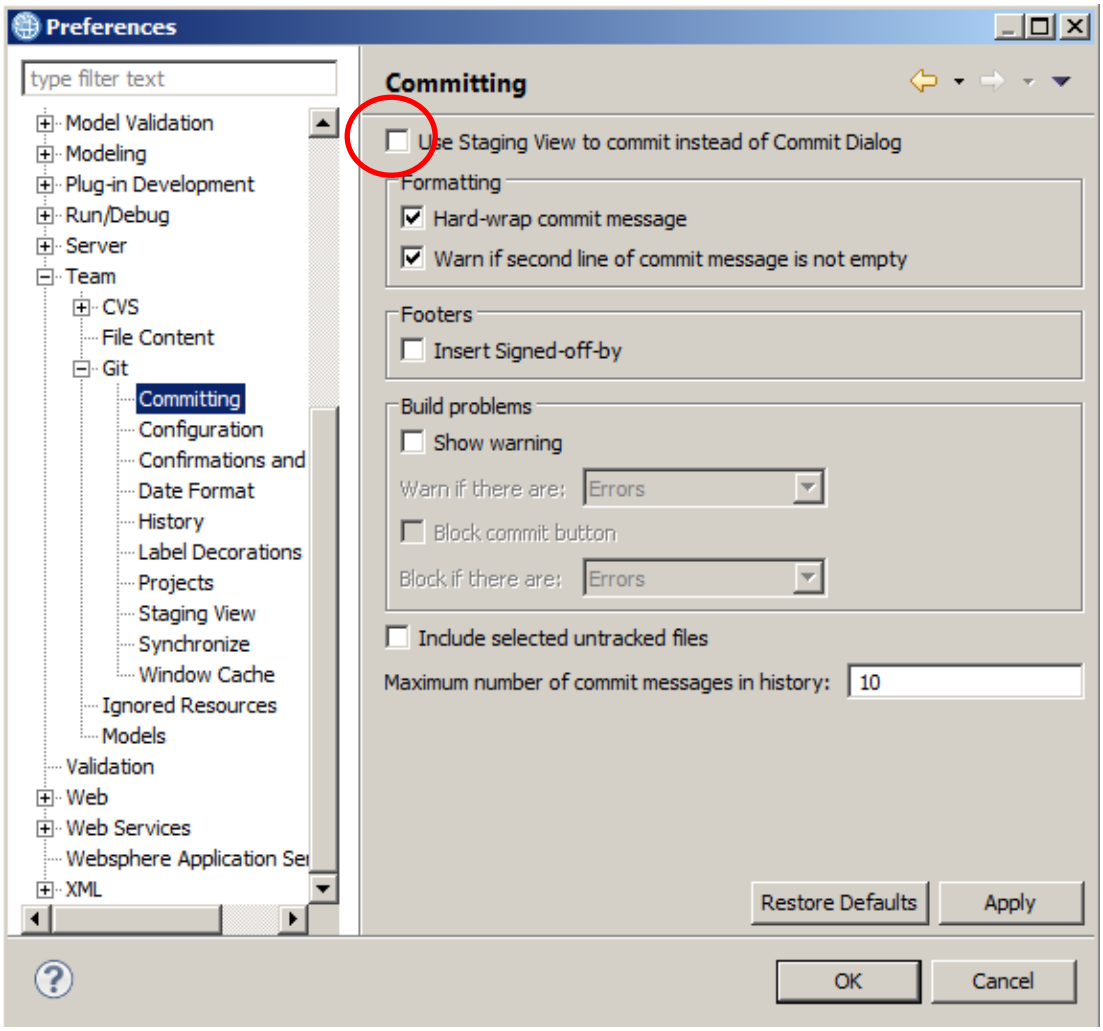
2. In the IIB Toolkit, click Window, Preferences.

   Expand Team, Git.

   Select **Committing**.

   Deselect the item "Use Staging View to commit instead of Commit Dialog". This means that a Commit operation will immediately update the Git repository, rather than requiring a second manual stage to perform the commit.

   Note: this may not be good practice in a production scenario, and is not the default option when the plugin is installed.

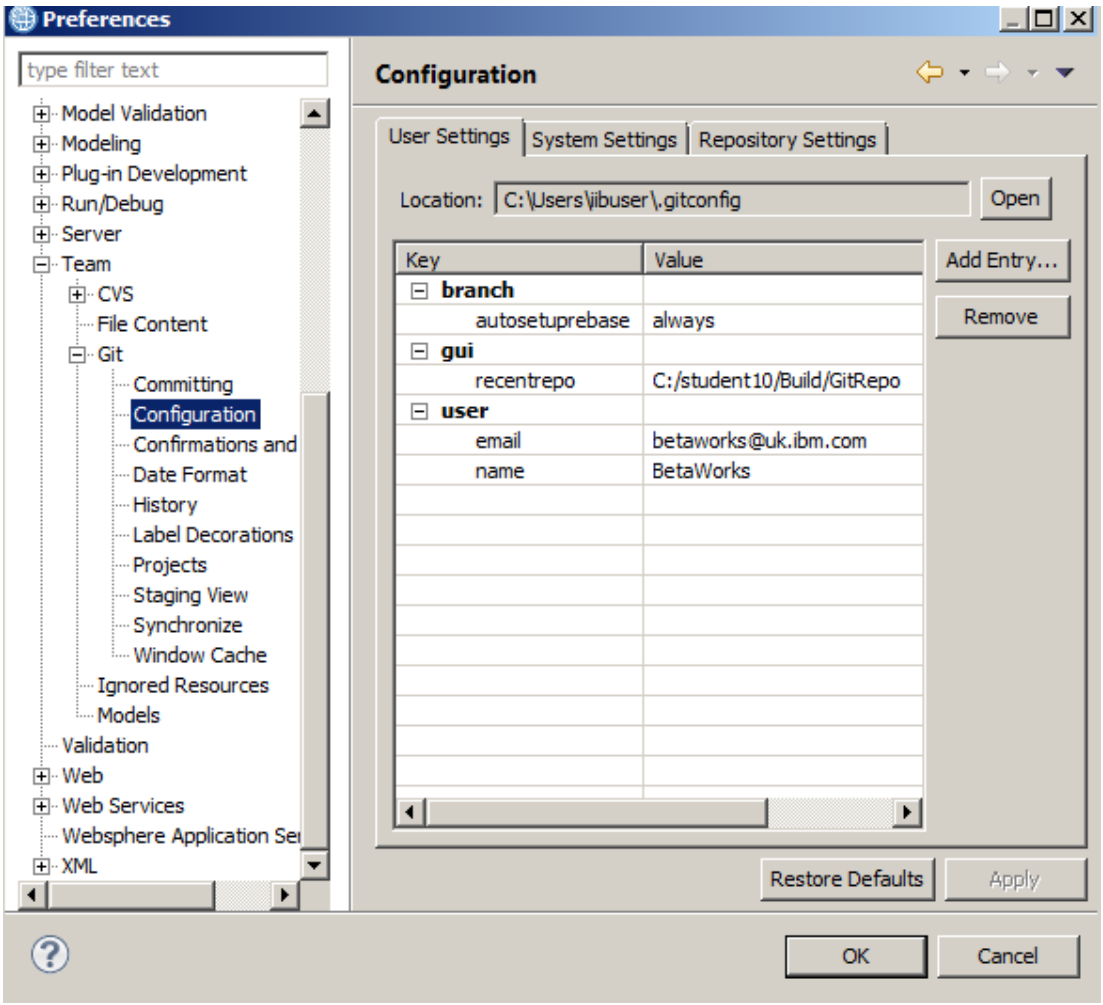


3. Select **Configuration**.

   Although not required for the lab scenarios, the keys **branch.autosetuprebase, user.email and user.name** have been defined in User Settings. The user settings are used by the EGit plugin to record information in Git about which user has made updates to items in the Git repository.

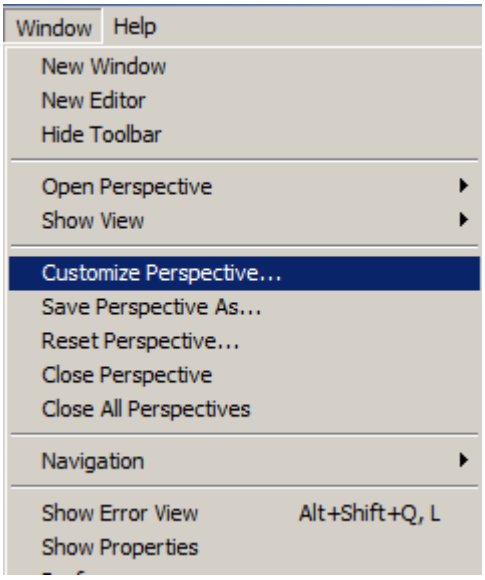
   Refer to the webpage http://www.vogella.com/tutorials/EclipseGit/article.html for further discussion on this topic.

   Click OK when complete.



4. Now add the Git Command Groups to the Toolkit perspective.

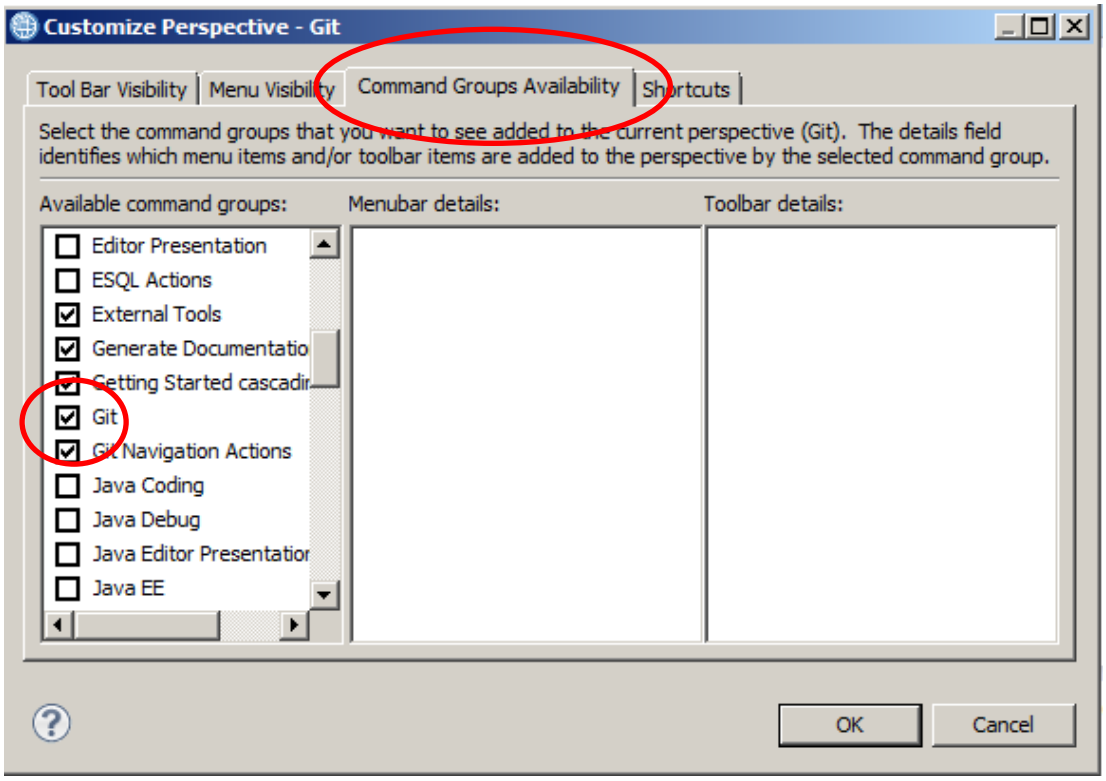
   Click Window, Customize Perspective.



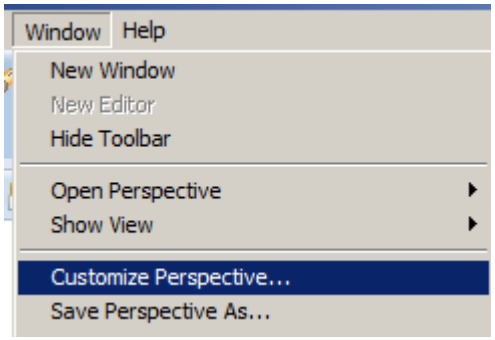
5. In the Customize Perspective window, click the "Command Groups Availability" tab.

   Select the "Git" command group.

   Menu items on other tabs are not populated until you click OK, so click OK now.

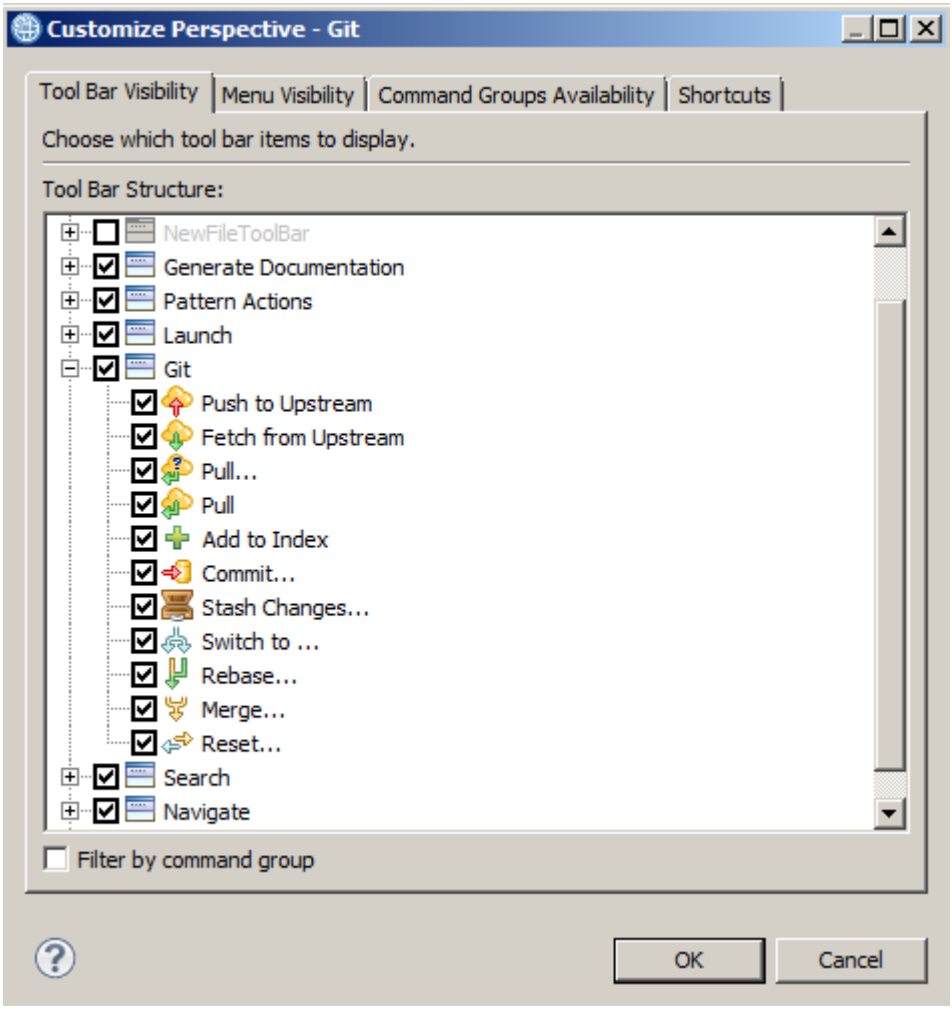


6. In the IIB Toolkit, again click Window, Customize Perspective.



7. In the Tool Bar Visibility tab, ensure "Git" is selected. If you wish, you can select/deselect any of the individual Git items.

   Click OK.

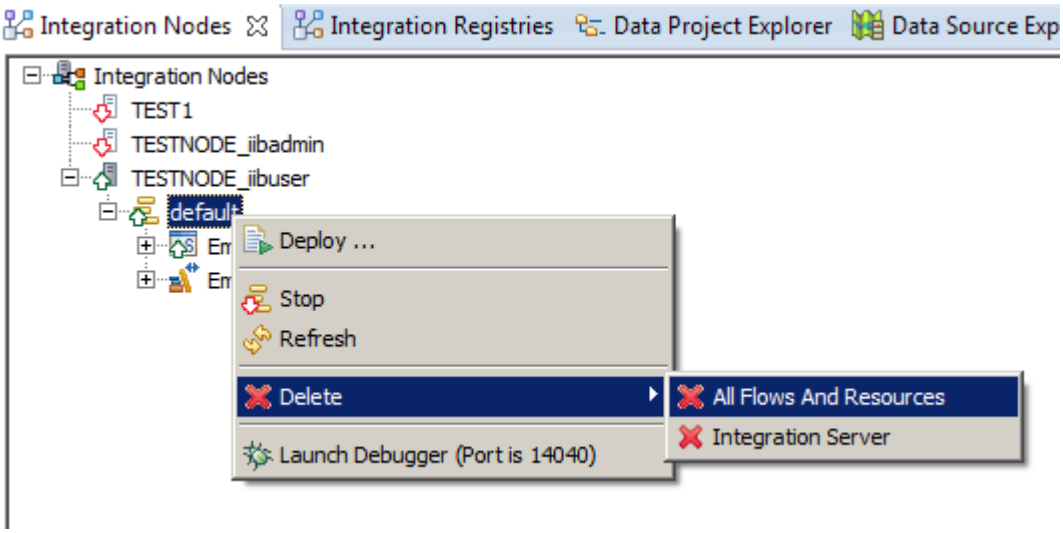


2.3 Check the TESTNODE_iibuser node

1. In the IIB Toolkit, ensure that the TESTNODE_iibuser node is running, and that no services or other artefacts are deployed to the default server.

   If necessary, Delete All Flows and Resources from the node.

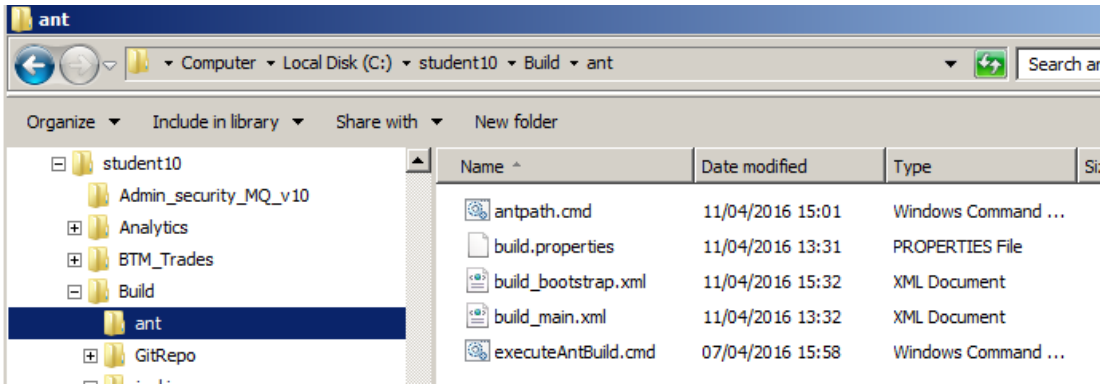
   Also, ensure that no other nodes are running (to ensure that the required node is listening on port 7800).



3. Investigate and run the Ant script

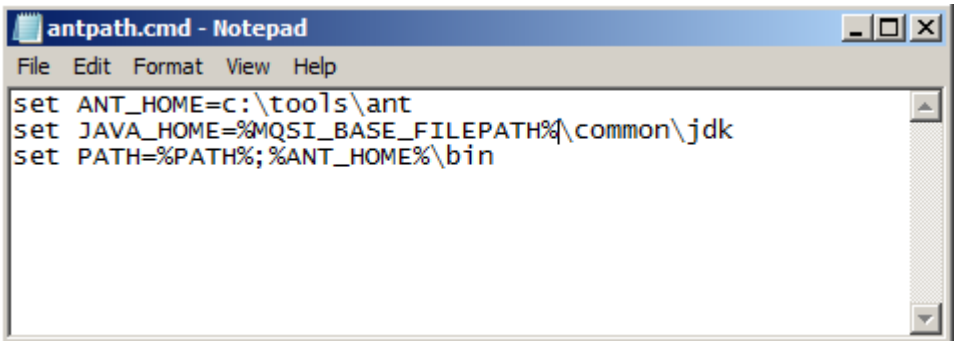
In this implementation of the automated IIB build system, we have used ANT scripts to control and invoke the various mqsi commands that are required to perform this automation. These scripts can be invoked directly. Later in this lab, these scripts will be invoked by the Jenkins build control system.

1. In Windows Explorer, navigate to c:\student10\Build\ant.

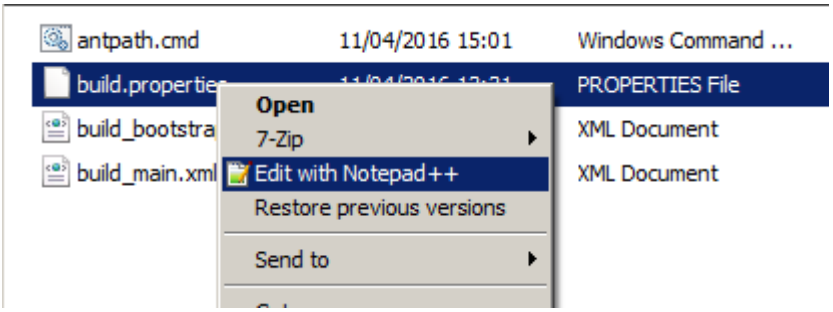


2. The first file to note is **antpath.cmd**. Open this with the Notepad editor.

   This cmd file is provided for info only. The cmd file can be used to run ANT in a DOS window, if the environment variables have not been previously set. However, in the workshop system, these values have already been set using Windows System Environment variables. Use the SET command in a DOS window to take a look if you wish.



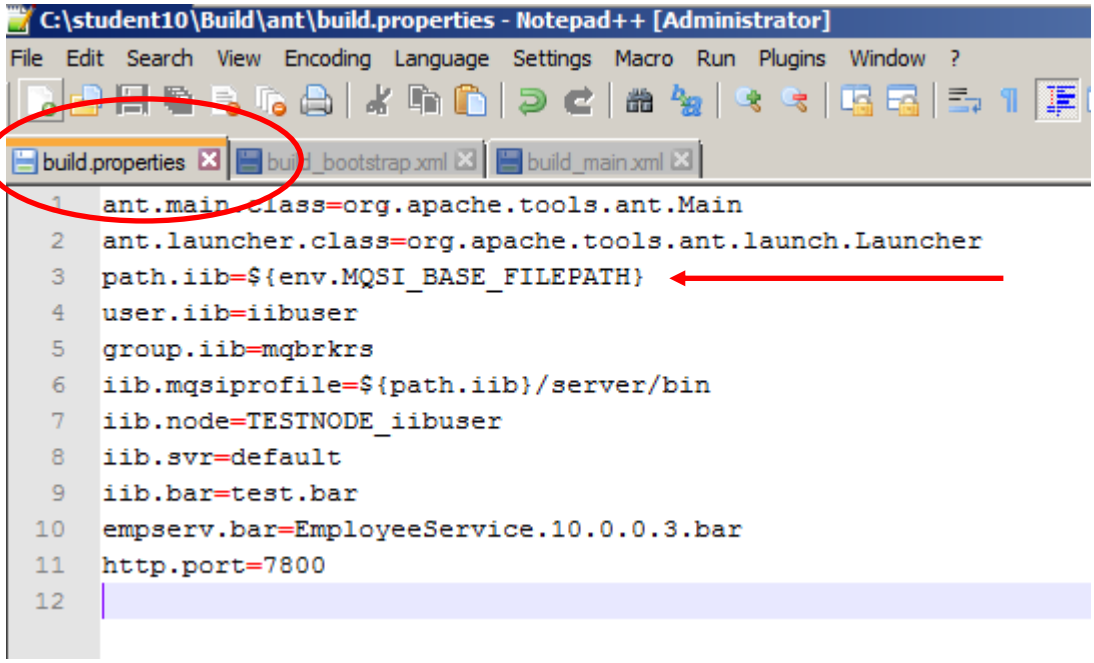
3. Open the three files prefixed "build_" in Notepad++.



4. In Notepad++, edit the **build.properties** file. This file is used to provide a convenient way of changing the value of properties that may need to be adjusted for each build. The file is referenced by the main Ant script files.

   Although we have provided a pre-built instance of this file, in a specific folder, it would be possible to store this file in the Git repository itself, associated with the IIB projects that are to be built.

   Note the use of the ANT facilities to determine the value of system environment variables, in this example to set the value of the IIB installation folder with the environment variable MQSI_BASE_FILEPATH.

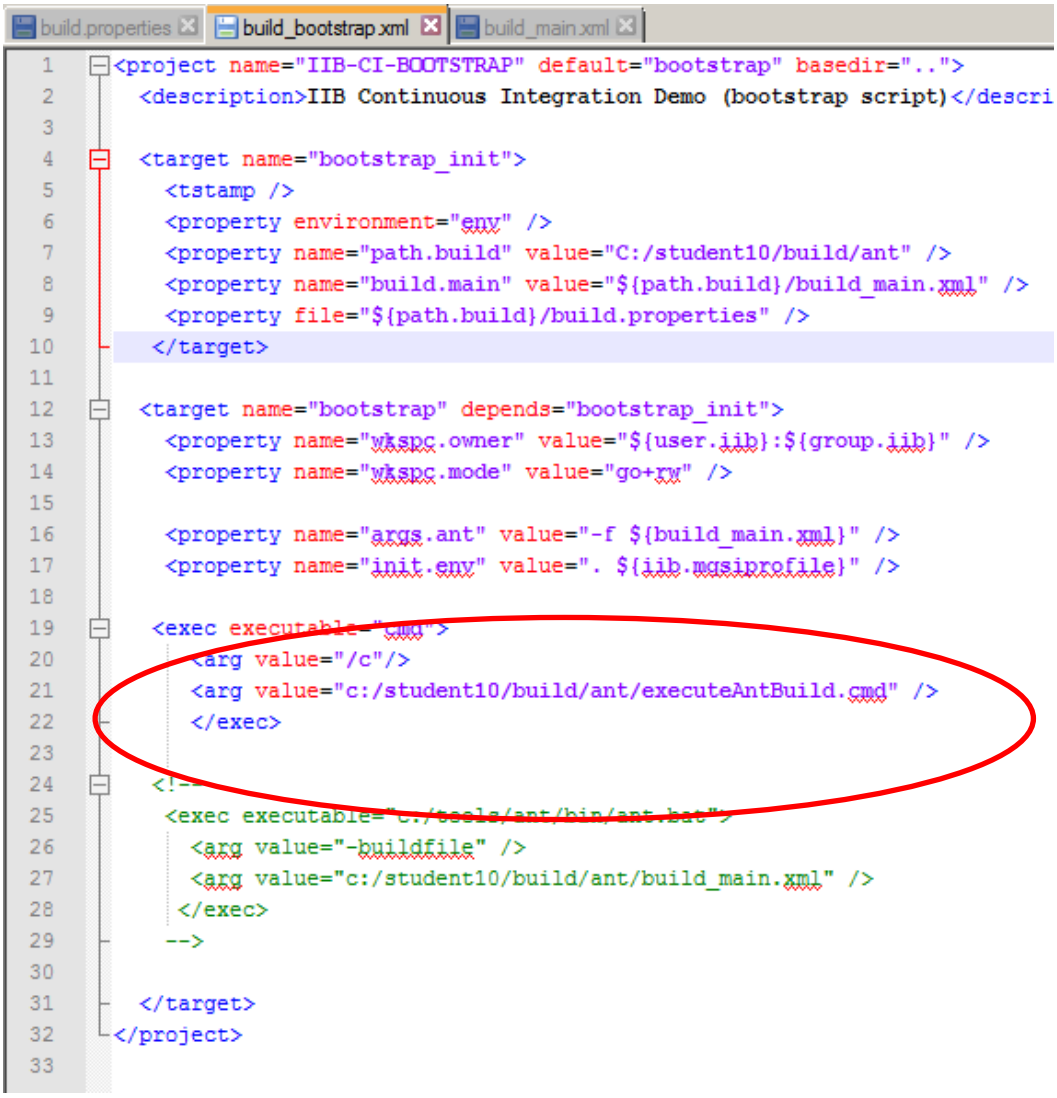


5. Now switch to the **build_bootstrap.xml** file.

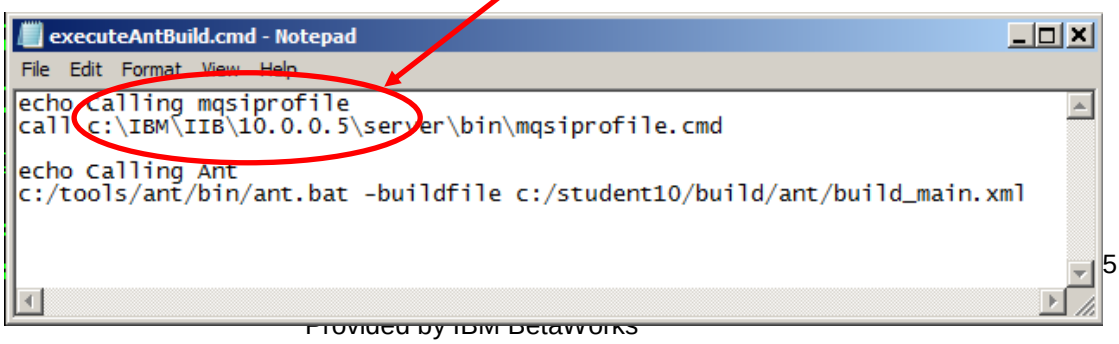
   Note that the main part of this Ant script file is the stanza that contains the executable "**cmd**". This invokes a Windows CMD file, and executes the CMD file **c:\student10\build\ant\executeAntBuild.cmd** within the DOS window.

   This is required because some of the IIB commands that are required must be run after executing the mqsiprofile.cmd. The executeAntBuild cmd file performs this, and then executes the main Ant build script.

   On Unix systems, it is easier to directly invoke the mqsiprofile directly from the Ant script, followed by the primary Ant build script.



Important: The executeAntBuild command file references the installation folder of the IIB system. If this is changed, or a new release of IIB is supplied, this command file must be changed to reflect that.



6. Now switch to the build.xml file.

   Things to note:

   - The basedir variable is set to c:\student10\build\jenkins\workspace. This is because Jenkins has the concept of a workspace. When Jenkins extracts files or projects from Git, it places them in its own workspace. The name of this workspace can be defaulted by Jenkins, but in this case, for clarity, we have provided a specific folder for this purpose. However, if you are going to run the Ant script stand-alone, this version of the Ant script requires the Jenkins workspace to be manually populated with the Eclipse projects. In the IIB workshop, this has been prepared for you in advance.

   - The barfile build is performed with the mqsicreatebar command. This command requires an installation of IIB, since it creates a headless version of the Eclipse Toolkit. The mqsipackagebar command may also be used. This does not require a full installation of IIB, although there are some limitations to this command.

   - The mqsideployscript command is used to deploy the newly created barfile to the IIB node TESTNODE_iibuser.

   - The CURL command is used to send a simple SOAP Request message to the deployed service. The full CURL command is (formatted for readability):

     ```
     curl -H SOAPAction: http://EmployeeService/getEmployee
       -H Content-Type:text/xml;charset=UTF-8
       -X POST
       -d @c:\student10\build\test\EmployeeService_run.txt
       http://betaworks-esb10:7800/EmployeeService
     ```

     The file EmployeeService_run.txt contains the SOAP request message to retrieve a record from the EMPLOYEE table.

build.properties | build_bootstrap.xml | build_main.xml

```
<project name="IIB-CI-MAIN" default="dev" basedir="c:\student10\build\jenkins\workspace">
  <description>IIB Continuous Integration Demo (main script)</description>

  <target name="init">
    <tstamp />
    <property environment="env" />
    <property name="path.build" value="C:/student10/build/ant" />
    <property file="${path.build}/build.properties" />
  </target>

  <target name="dev" depends="init">

    <exec executable="${path.iib}\server\bin\mqsiprofile.cmd">
    </exec>

    <exec executable="${path.iib}\server\bin\mqsiservice">
      <arg value="-v" />
    </exec>
    <exec executable="${path.iib}\server\bin\mqsilist" />
    <exec executable="${path.iib}\server\bin\mqsilist">
      <arg value="${iib.node}" />
    </exec>

    <exec executable="${path.iib}\tools\mqsicreatebar">
      <arg value="-data" />
      <arg value="${basedir}" />
      <arg value="-b" />
      <arg value="${empserv.bar}" />
      <arg value="-a" />
      <arg value="EmployeeService" />
      <arg value="-l" />
      <arg value="EmployeeService_interface_and_maps" />
      <arg value="-deployAsSource" />
    </exec>

    <exec executable="${path.iib}\server\bin\mqsideployscript.bat">
      <arg value="${iib.node}" />
      <arg value="-e" />
      <arg value="${iib.svr}" />
      <arg value="-a" />
      <arg value="${empserv.bar}" />
      <arg value="-w" />
    </exec>

    <exec executable="c:/tools/curl/curl.exe">
      <arg line="-H SOAPAction: http://EmployeeService/getEmployee -H Content-Type:text/xml;
    </exec>


  </target>
</project>
```

7. Now execute the Ant script manually.

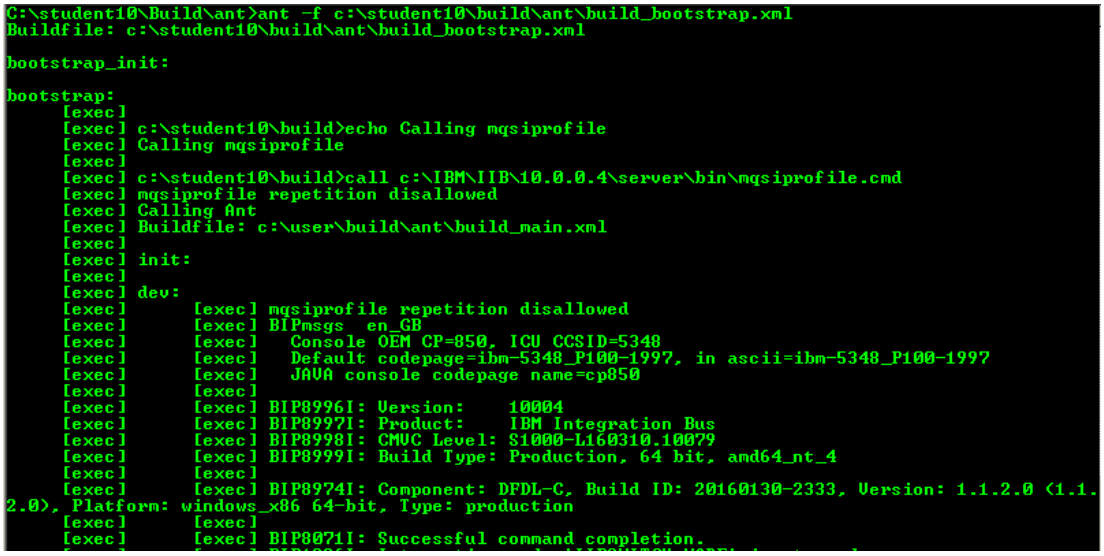
   Open an IIB Command Console, and navigate to c:\student10\build\ant.

   Execute the command

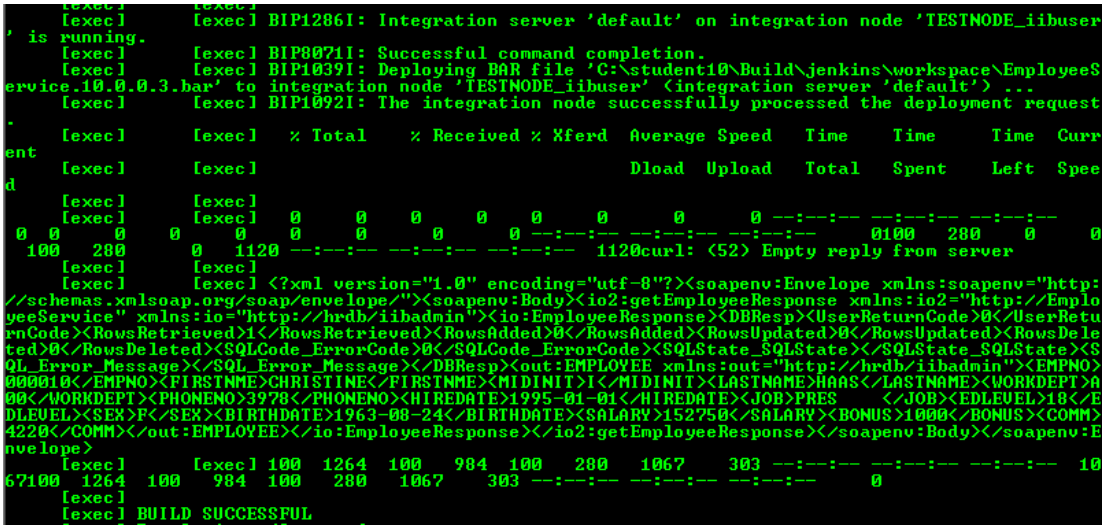
   `ant -f c:\student10\build\ant\build_bootstrap.xml`

   The following output will be seen (some output is omitted). Note that the barfile EmployeeService.10.0.0.3.bar is created in the folder c:\student10\Build**jenkins\workspace.**. Any folder could be used for this, but we have configured this for use with Jenkins. You will investigate Jenkins further in the next chapter.

   Note that the CURL tool has been used to send a single request to EmployeeService to invoke the getEmployee operation (port 7800), with the employee ID "000010". You will see the employee details returned in the SOAP response message.



...

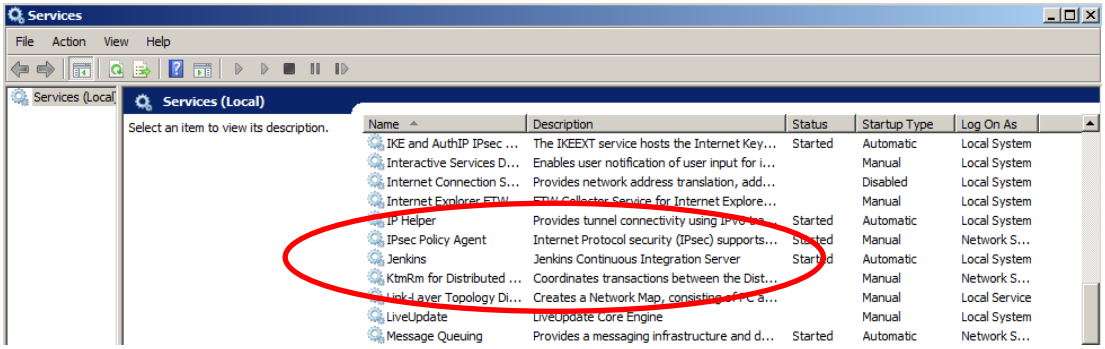


4. Automate the Build with Jenkins

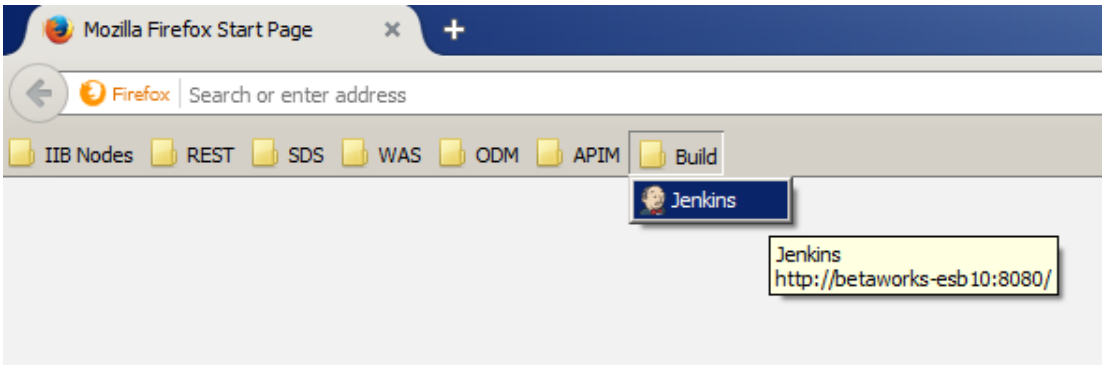
The next step of this lab is to automate the build and deploy process. We have selected the Jenkins tools to do this, but many other similar tools are available.

1. First, check that Jenkins is running.

   Open Windows Services. The Jenkins Continuous Integration Server is set to start automatically, so should be running. If not, then start it now.



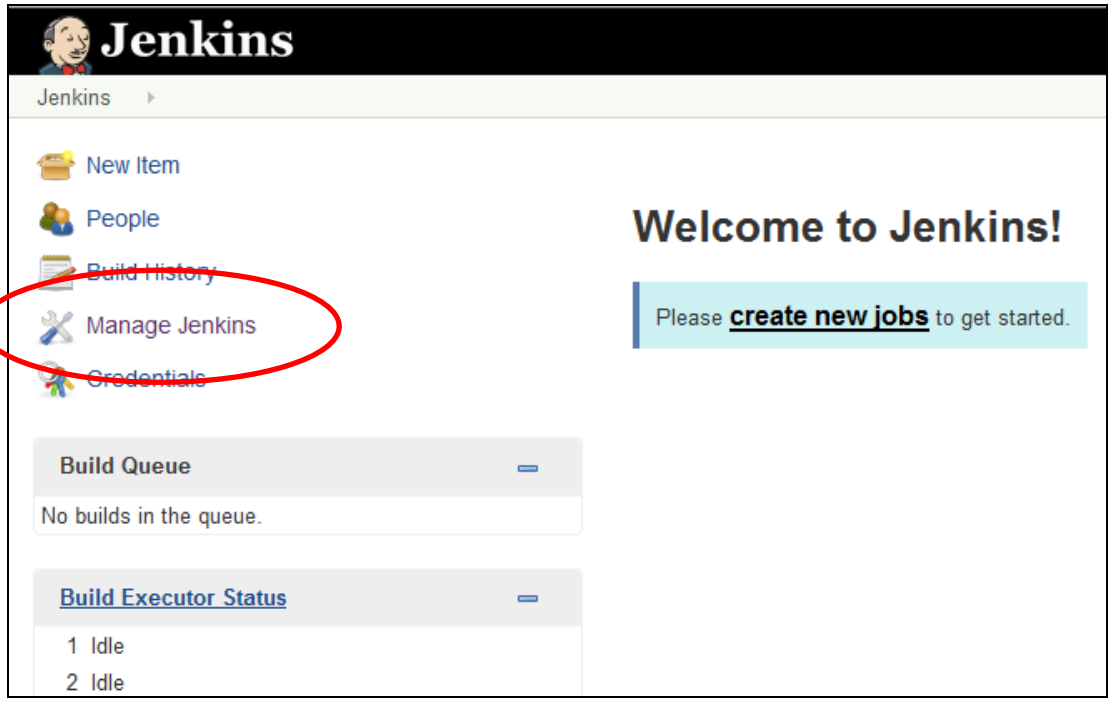
2. In a Firefox browser, open the URL **http://betaworks-esb10:8080** (or use the provided short-cut in the Build folder).



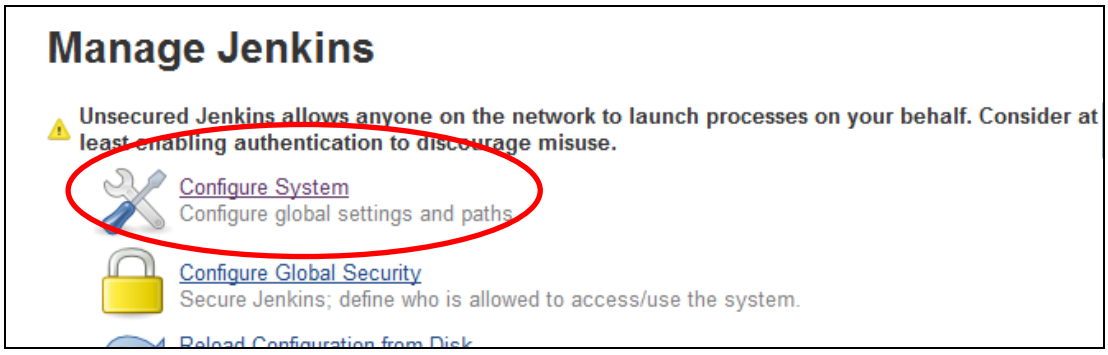
3. The Jenkins Welcome page will be displayed.

   Now take a look at the configuration changes that have already been made.

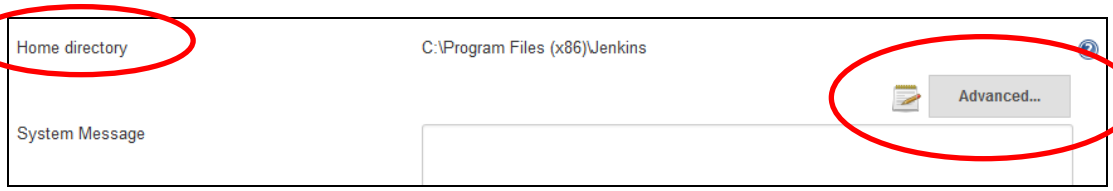
   Click Manage Jenkins.



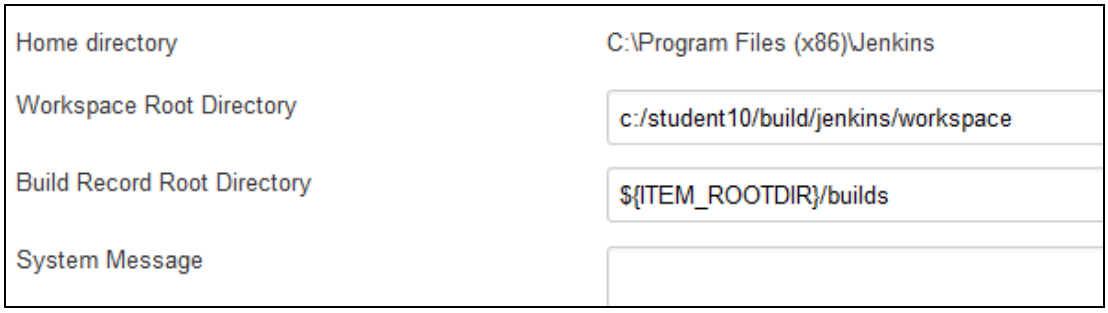
4. Click Configure System.



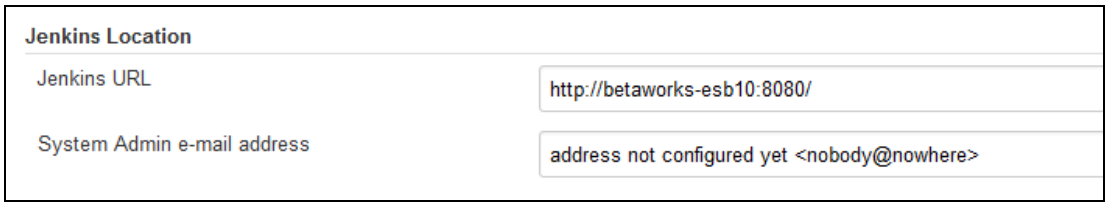
5. In the **Home directory** section, click Advanced.



6. Note that the Workspace Root Directory is set to **c:/student10/build/jenkins/workspace**. (forward slashes are used and recommended, to ease movement to unix systems).



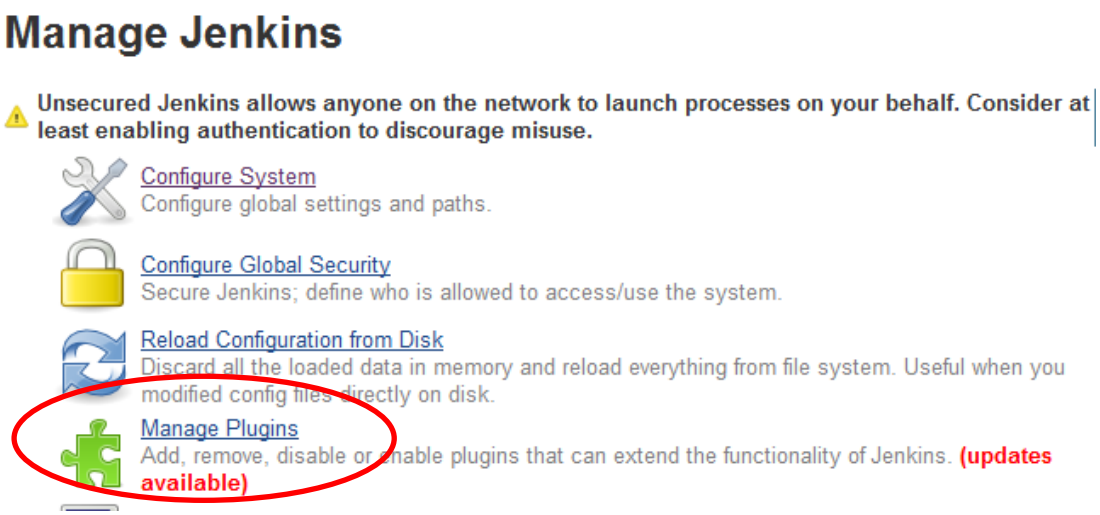
7. Scroll down the page to the **Jenkins Location**. The Jenkins URL is set to the hostname of the local system (this is the default when Jenkins is installed).



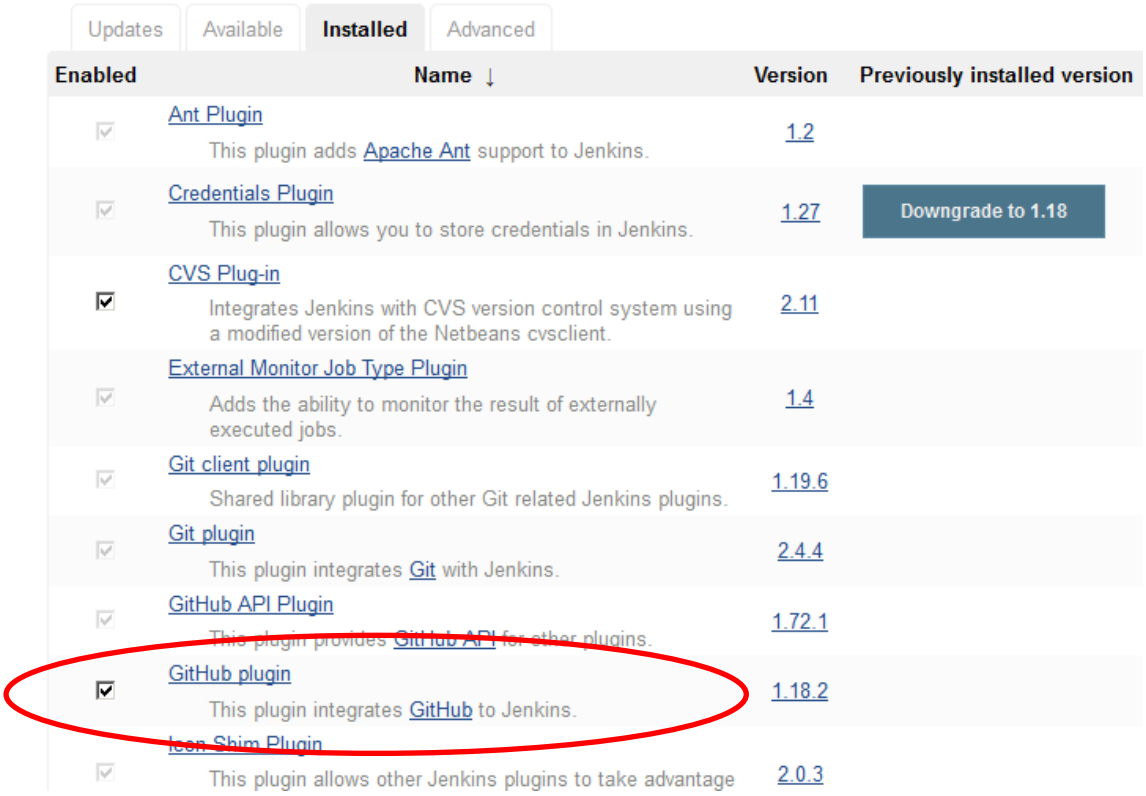
These are the only changes that have been made to Jenkins on this page.

8. Back on the main navigator, click "**Manage Jenkins**" again.

This time, click **Manage Plugins**.



9. Click the **Installed** tab. Note that the GitHub plugin has been installed.

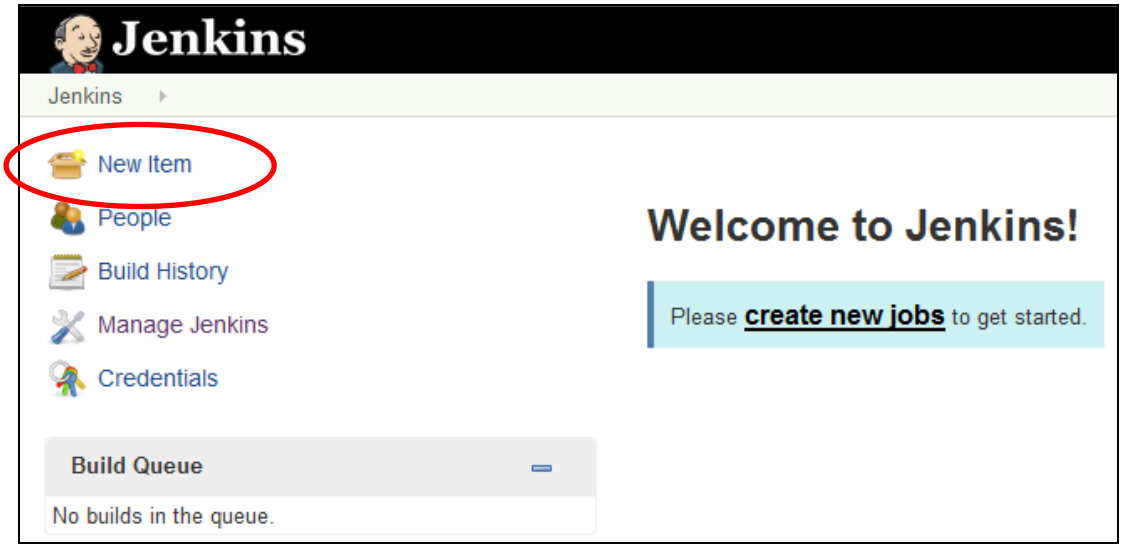


10. This completes the Jenkins configuration, although there are many more levels of sophistication that are available.

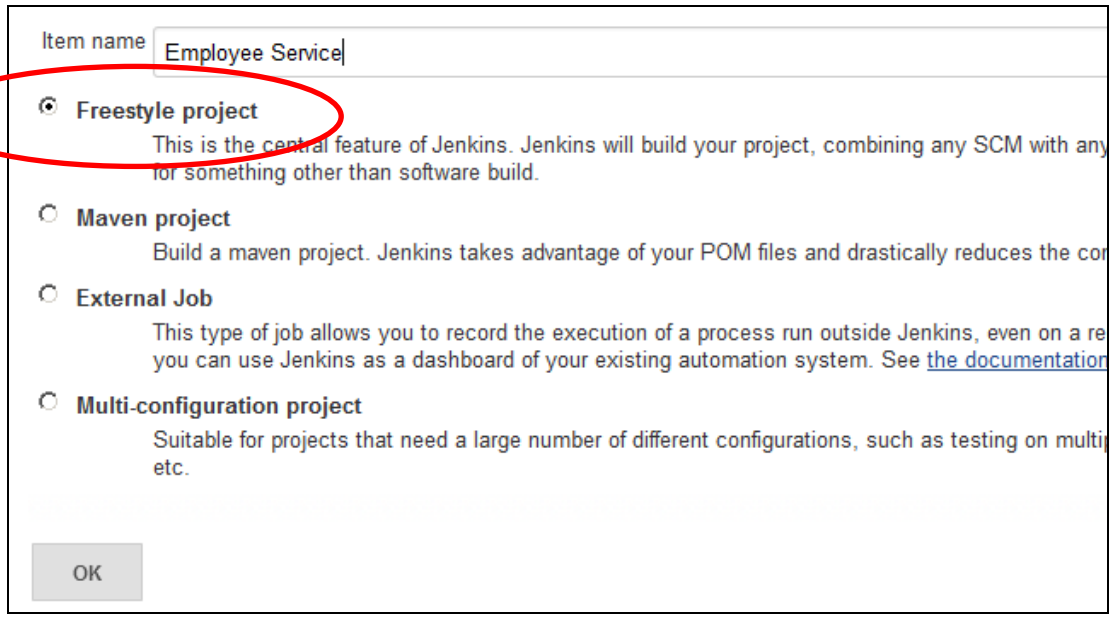
    Click "Back to Dashboard".

11. You will now create a new Jenkins item to control the IIB Build and Deploy ANT script.

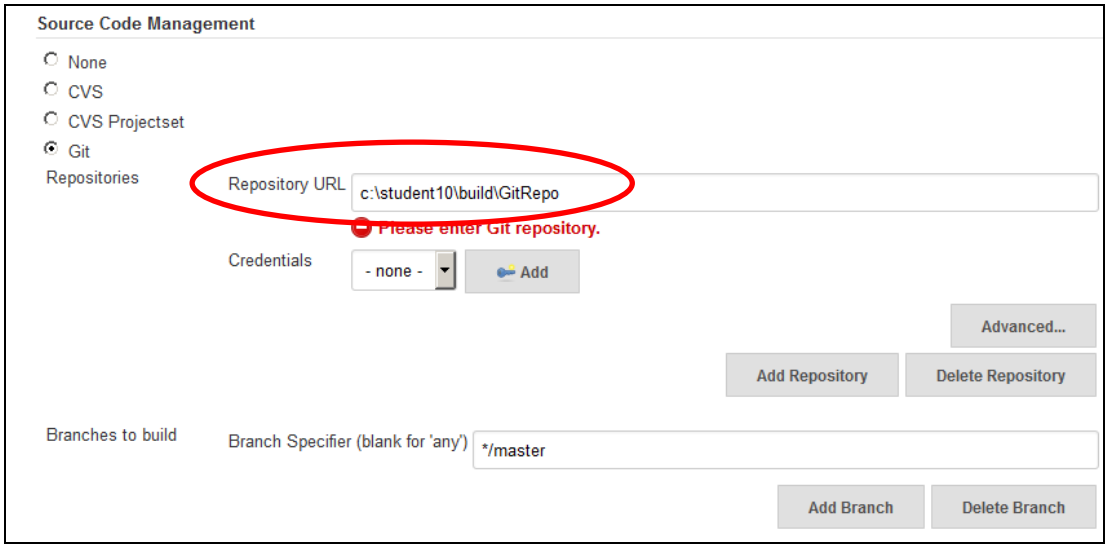
    Click New Item.



12. Set the Item name = "EmployeeService" and select FreeStyle project. Click OK.



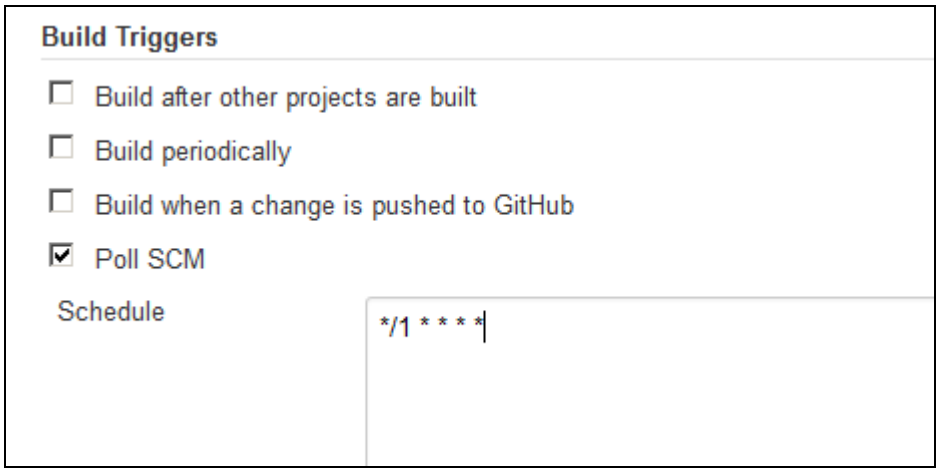
13. In the **Source Code Management** section, select Git, and set the Repository URL to c:\student10\build\GitRepo.



14. In the **Build Triggers** section, select "Poll SCM", and set the Schedule to

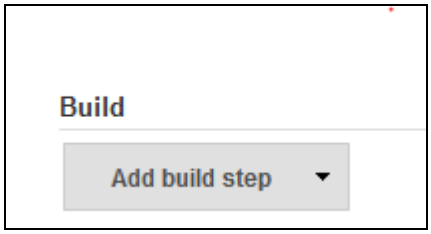
```
*/1 * * * *
```

Formatting note: the first three characters, */1, are consecutive.
The following four * characters are all separated with a space character.

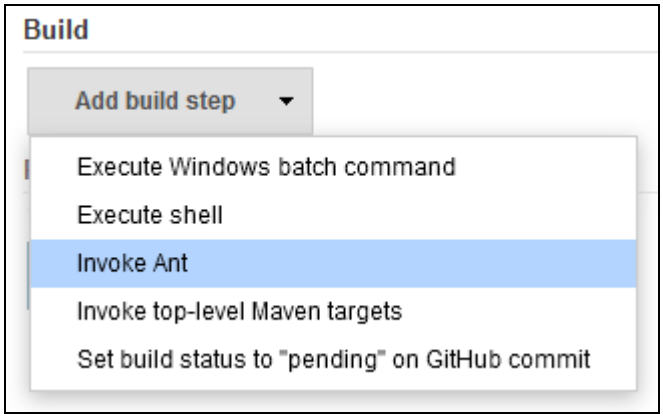


Important note: This schedule means that Jenkins will poll the Git repository every minute. This is clearly not optimal for a production scenario, but is configured like this for a classroom environment. Although it is possible to configure Git to automatically trigger a Jenkins build, this requires the addition of the GitHub server component, and adds complexity in a classroom environment.

15. In the Build step, click Add build step.

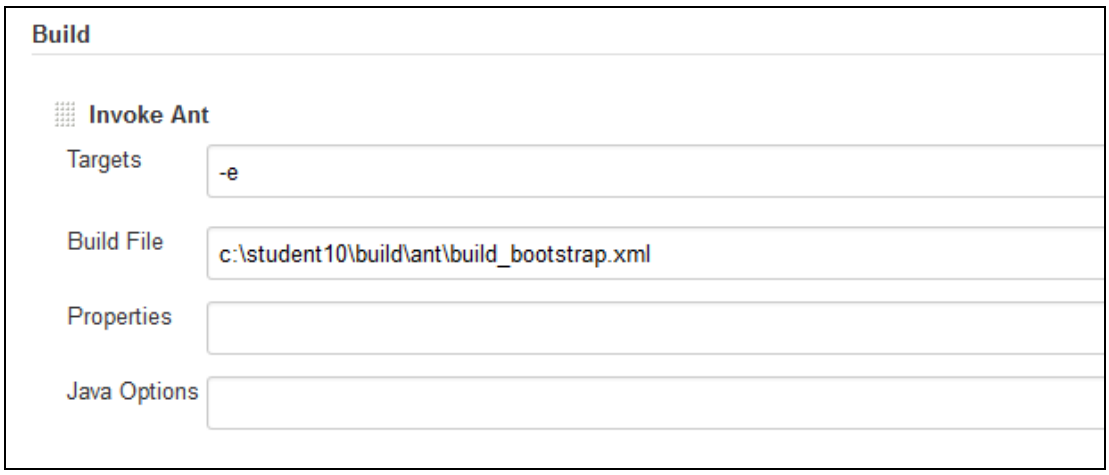


16. Select **Invoke Ant** from the drop-down menu.,



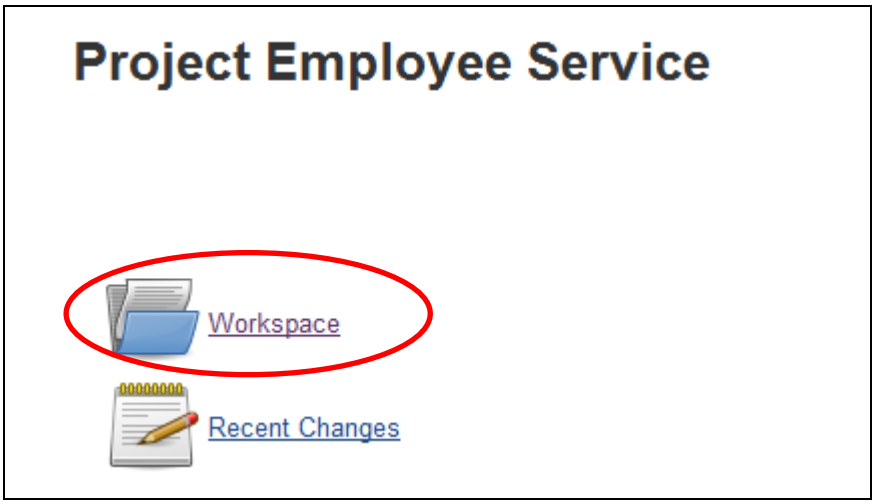
17. Click the Advanced button.

    - Set **Targets** to **-e** (to prevent Ant from adorning each line in the log excessively with prefixes)
    - Set **Build File** to c:\student10\build\ant\build_bootstrap.xml

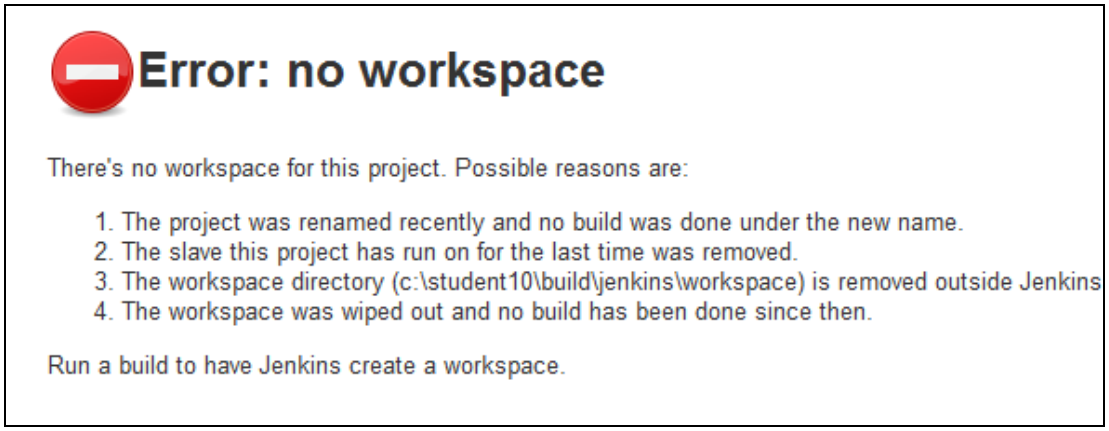


18. At the bottom of the page, click Save.

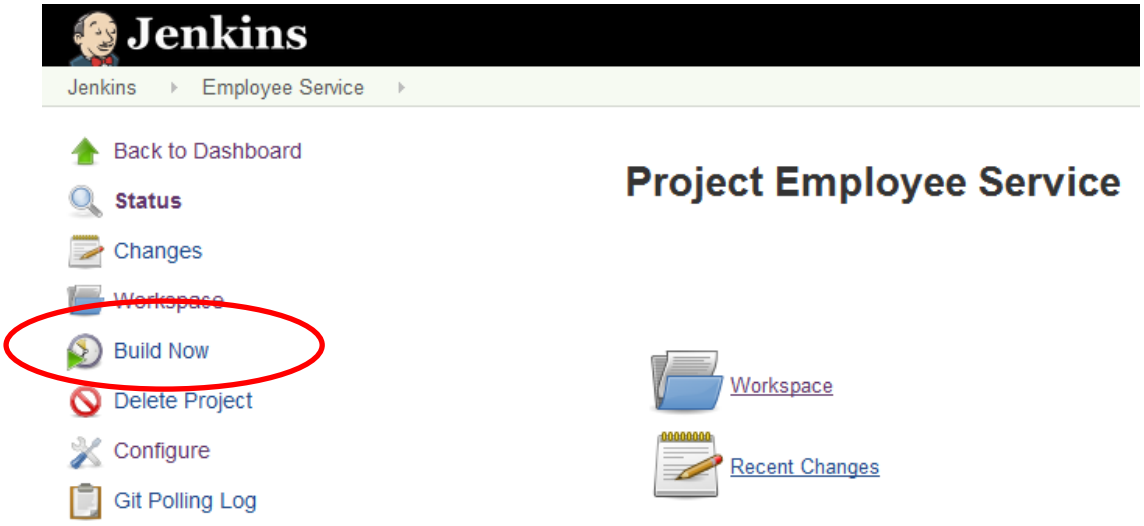
19. The project is now ready to use. Before you run the Jenkins project, take a look at the contents of the Employee Service Jenkins workspace. Click workspace.



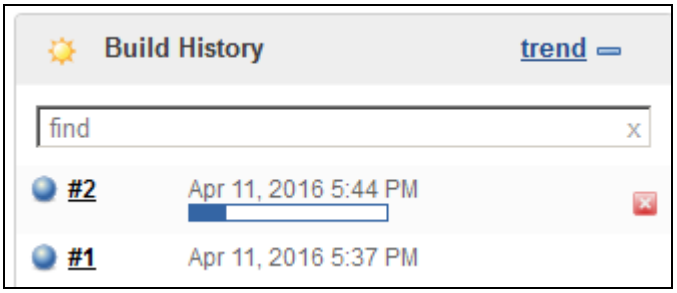
20. Since you have not run the Jenkins build yet, the workspace is empty.



21. Return to the dashboard for the Employee Service, and click Build Now.

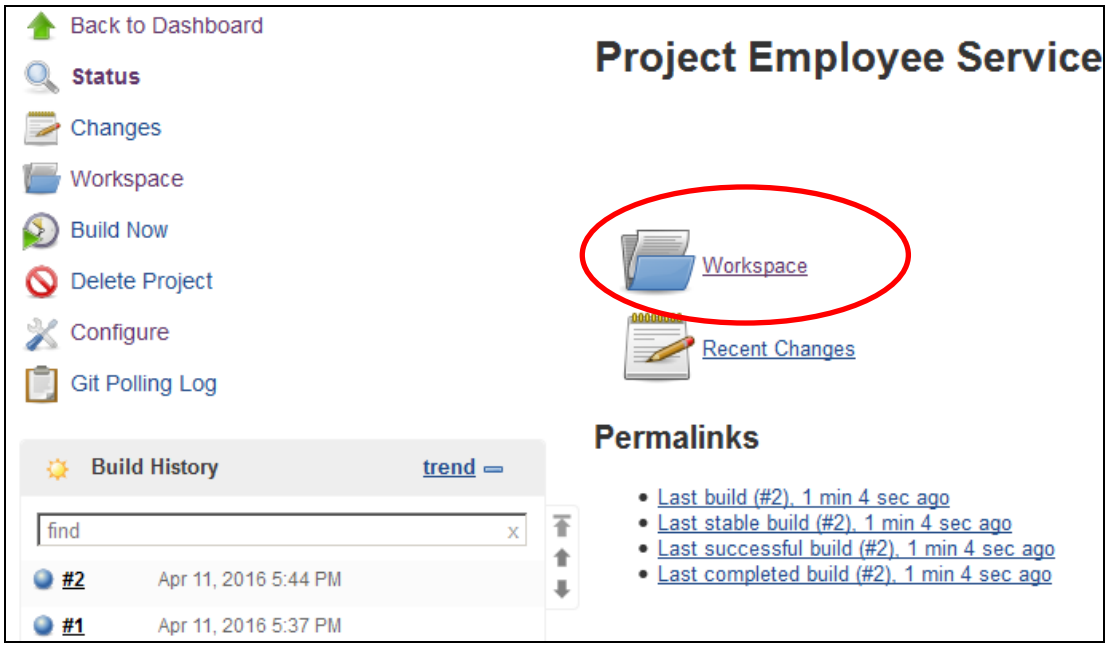


22. In the Build History pane, you will see the progress of the project build.



23. The build will take a minute or two. When complete, if successful, the icon will turn blue.

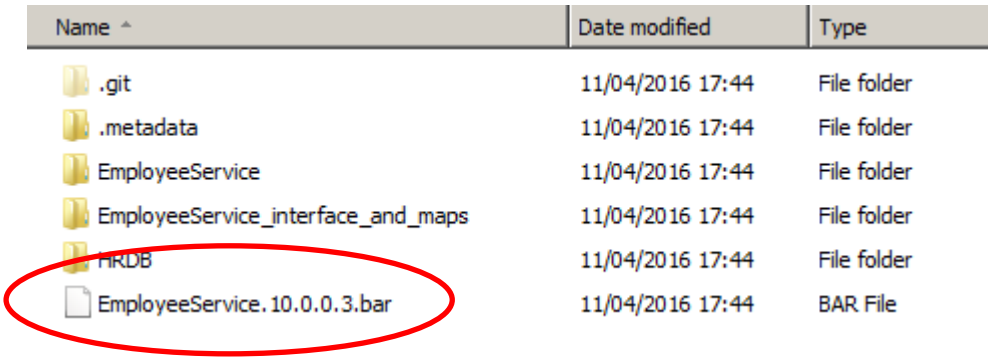
    Before reviewing the output of the build, take another look at the Workspace (click Workspace).



24. You will now see the Employee Service projects in the Jenkins workspace. These projects have been extracted from Git and placed in the local Jenkins workspace.

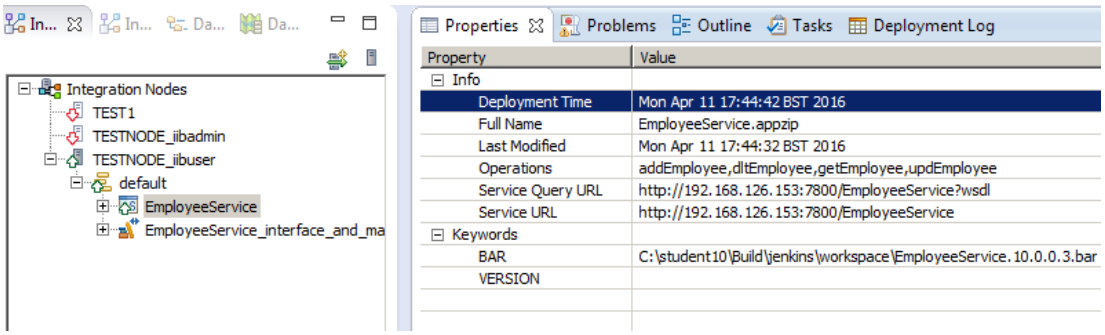
25. You can also see these files directly on the Windows file system, in c:\student10\build\jenkins\workspace, using Windows Explorer.

    Look at the "Date modified" information for EmployeeService.10.0.0.3.bar file. This should have just been created.



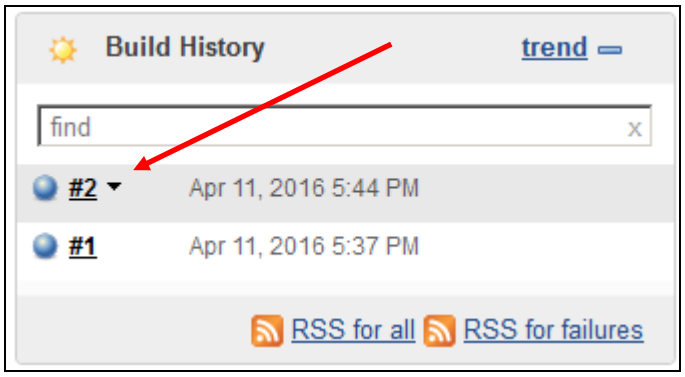
26. Now switch to the Integration Toolkit to check the details of the newly deployed integration service and library.

    In the Integration Nodes pane, select EmployeeService. Note the deployment time of the EmployeeService is also very recent, and should be consistent with the creation time of the barfile.

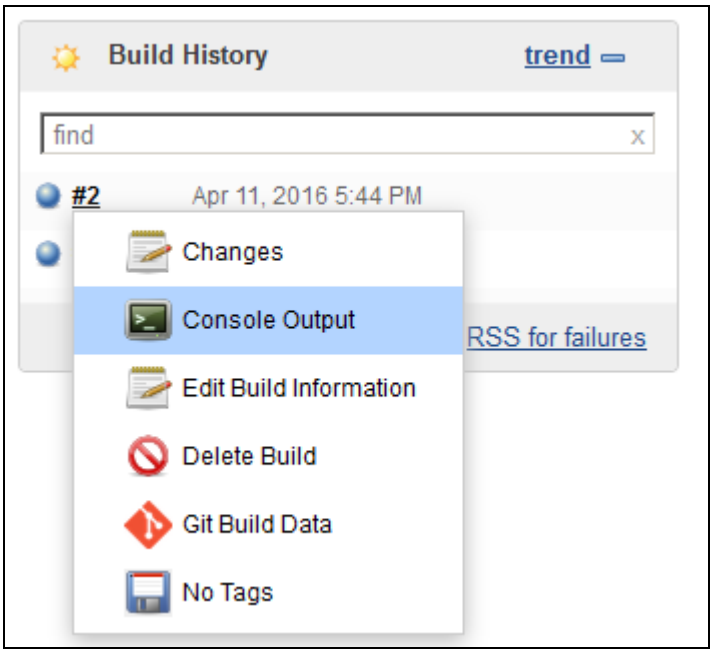


27. Switch back to the Jenkins page in the Firefox browser.

    In the Build History pane, hover the mouse just to the right of the build number.



    Click the down-arrow, and select Console Output.



28. Review the build output. In particular, at the bottom of the page, note the SOAP response message to the SOAP request that was sent by the CURL command.

```
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:B
ody><io2:getEmployeeResponse xmlns:io2="http://EmployeeService"
xmlns:io="http://hrdb/iibadmin"><io:EmployeeResponse><DBResp><UserRe
turnCode>0</UserReturnCode><RowsRetrieved>1</RowsRetrieved><RowsAdde
d>0</RowsAdded><RowsUpdated>0</RowsUpdated><RowsDeleted>0</RowsDelet
ed><SQLCode_ErrorCode>0</SQLCode_ErrorCode><SQLState_SQLState></SQLS
tate_SQLState><SQL_Error_Message></SQL_Error_Message></DBResp><out:E
MPLOYEE
xmlns:out="http://hrdb/iibadmin"><EMPNO>000010</EMPNO><FIRSTNME>CHRI
STINE</FIRSTNME><MIDINIT>I</MIDINIT><LASTNAME>HAAS</LASTNAME><WORKDE
PT>A00</WORKDEPT><PHONENO>3978</PHONENO><HIREDATE>1995-01-
01</HIREDATE><JOB>PRES
</JOB><EDLEVEL>18</EDLEVEL><SEX>F</SEX><BIRTHDATE>1963-08-
24</BIRTHDATE><SALARY>152750</SALARY><BONUS>1000</BONUS><COMM>4220</
COMM></out:EMPLOYEE></io:EmployeeResponse></io2:getEmployeeResponse>
</soapenv:Body></soapenv:Envelope>

BUILD SUCCESSFUL
Total time: 47 seconds
Finished: SUCCESS
```

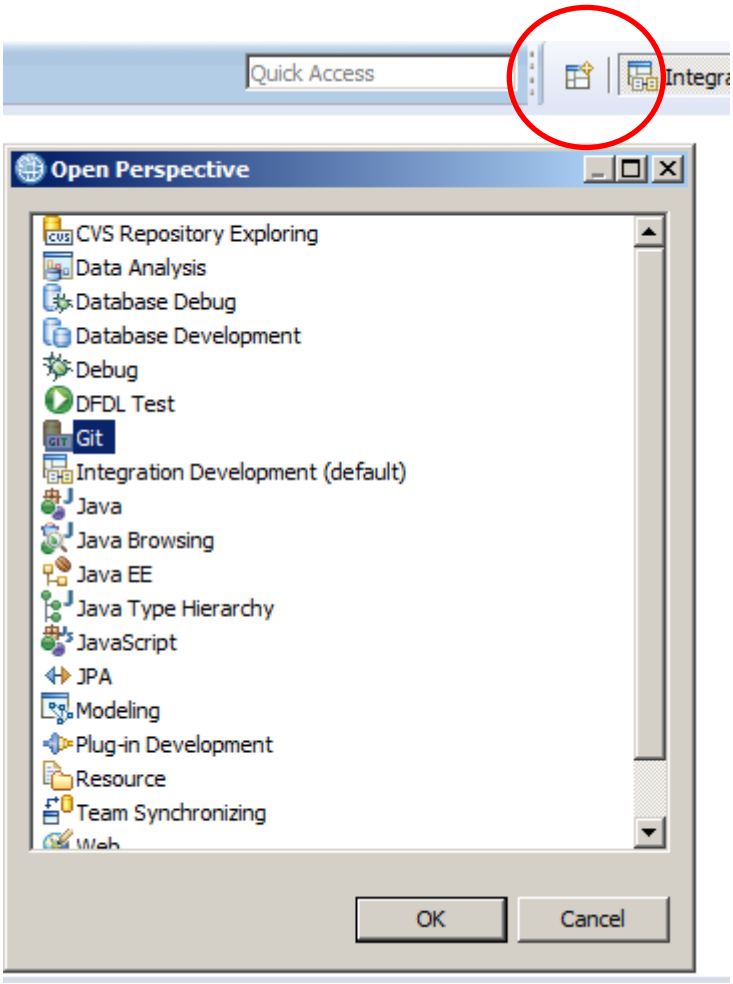
5. Update the Integration Service and Rebuild

You will now make a change to the Integration Service, and check this in to the Git repository. The Jenkins project should run automatically, rebuild and redeploy the barfile, and rerun the simple execution test.

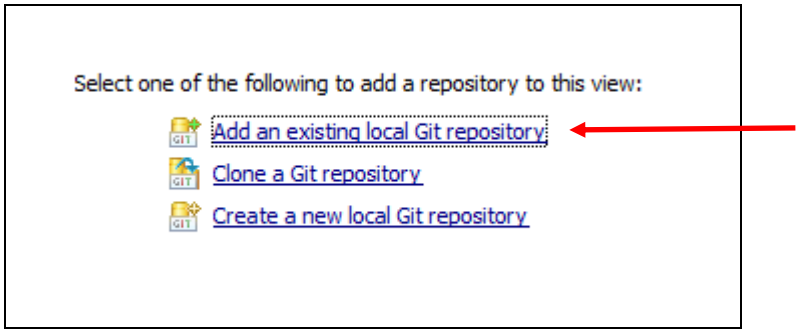
5.1 Import the Git projects into an IIB workspace

1. In the Integration Toolkit, open the Git perspective. Do this by clicking the "Open Perspective" icon, and selecting Git.

   Click OK.

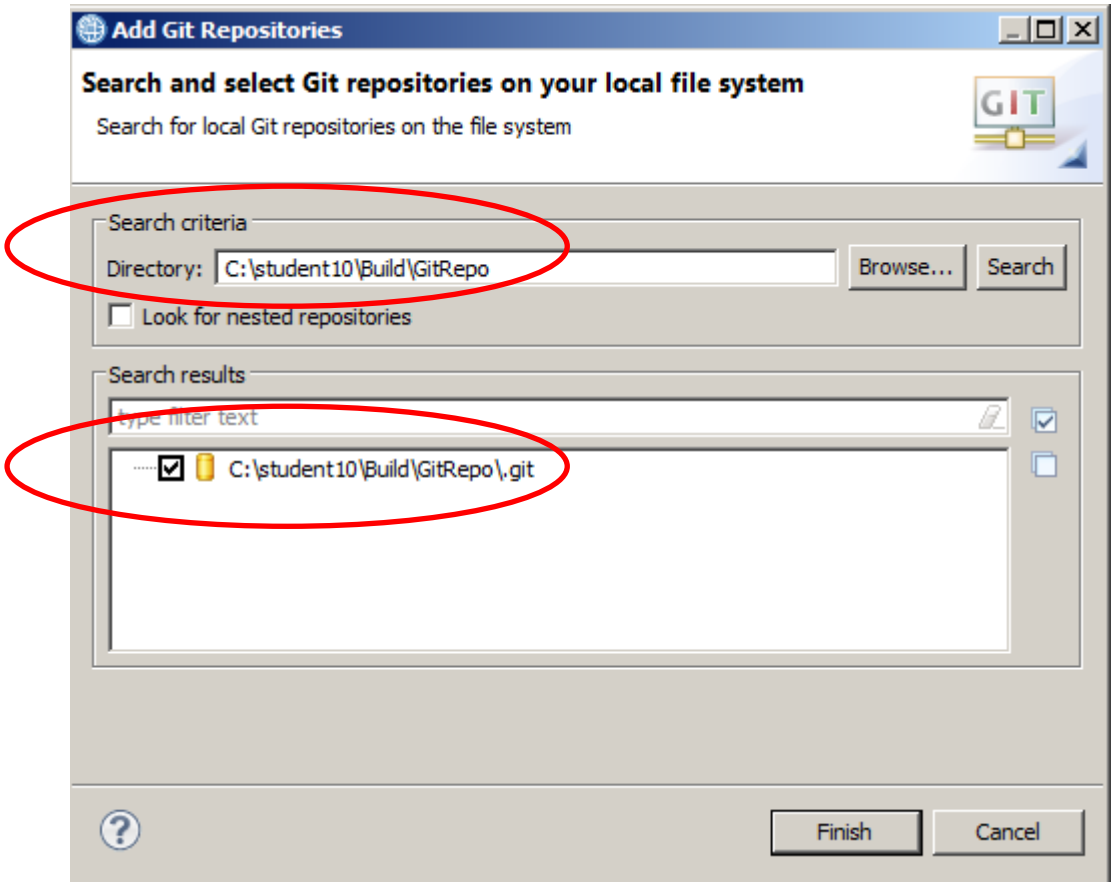


2. Click "Add an existing local Git repository".

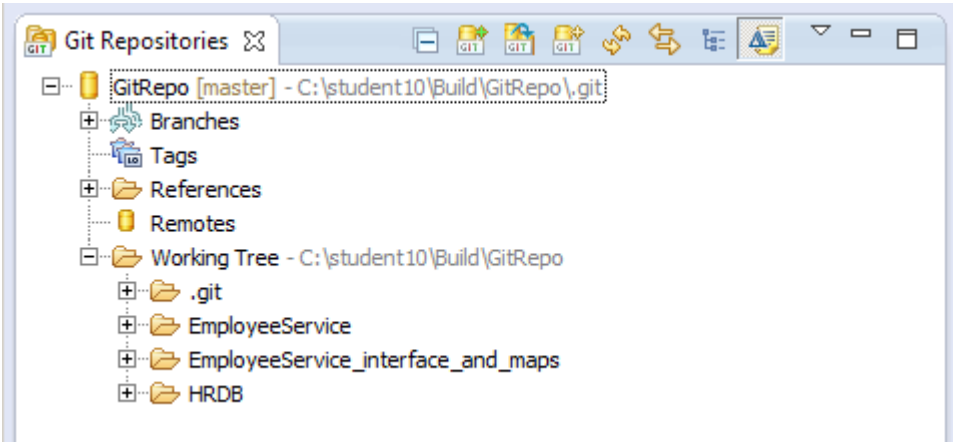


3. Using the Browse button, set the Directory to c:\student10\Build\GitRepo.

   Select c:\student10\Build\GitRepo\.git in the checkbox, and click Finish.



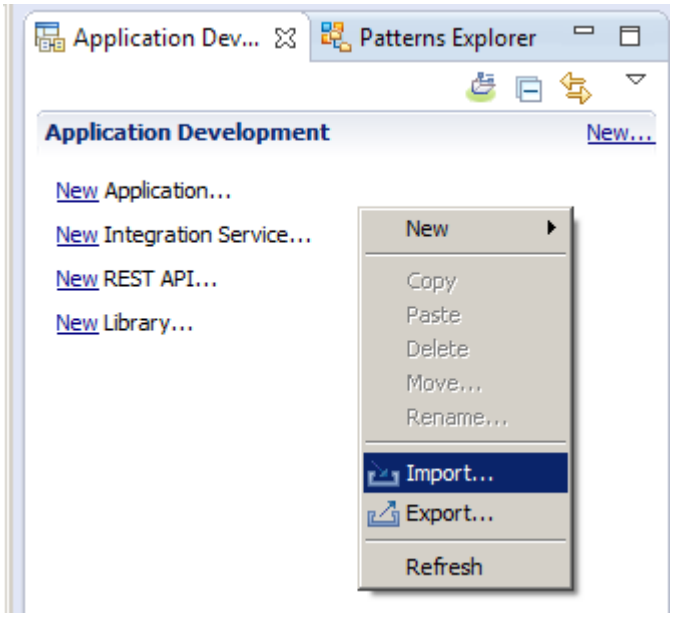
4. Expanding the navigator will show the content of the GitRepo master. The stored projects are shown under the Working Tree folder.



5. Now import these projects into the current IIB workspace.

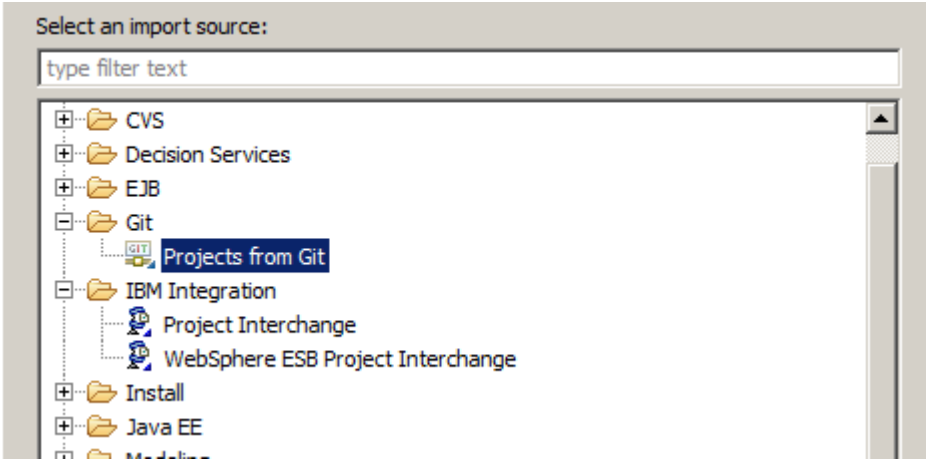
   Switch back to the Integration Development perspective.

   In the navigator, right-click in a blank area, and select Import.

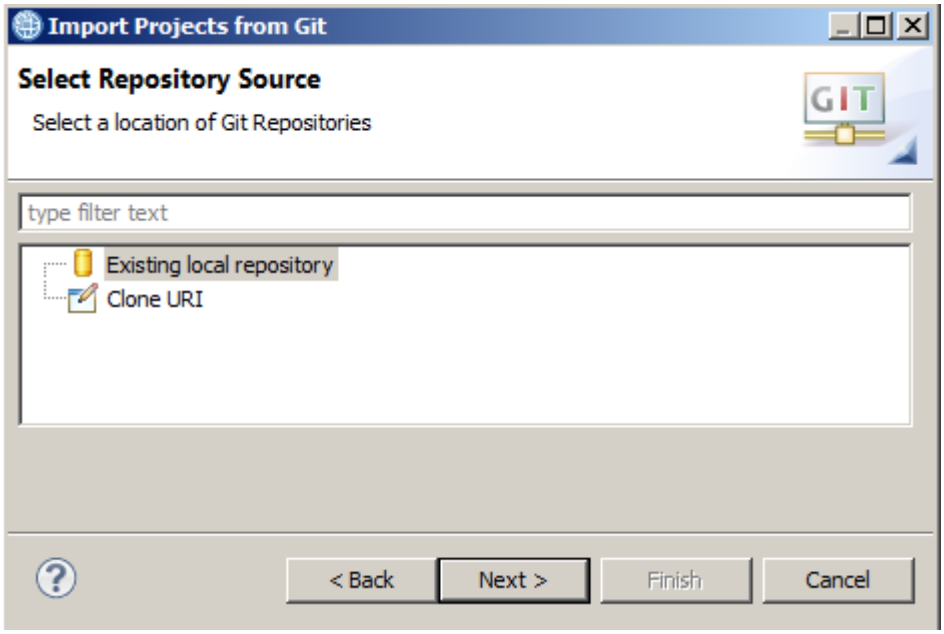


6. In the import wizard, expand **Git**, and select **Projects from Git**.

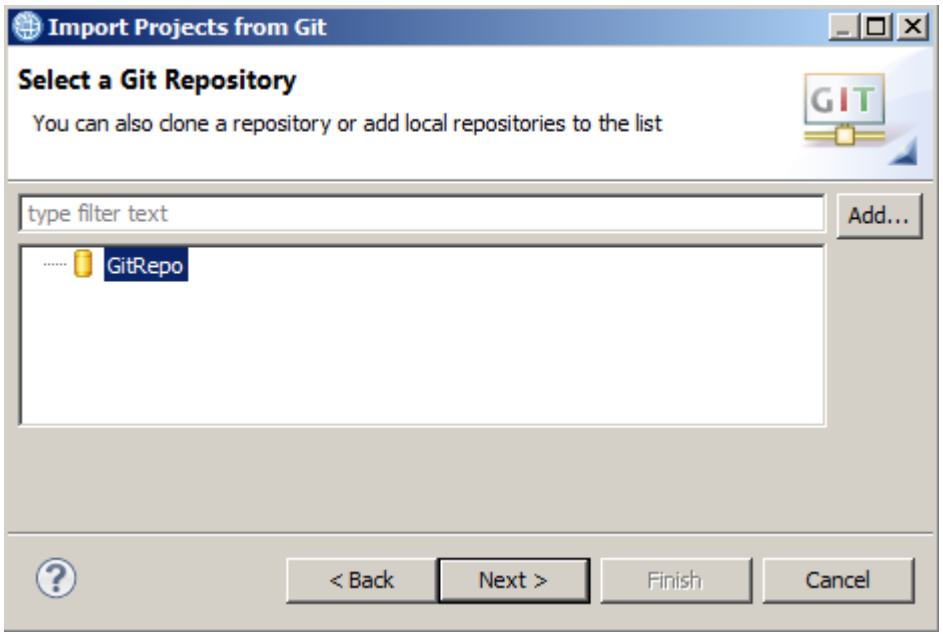
   Click Next.



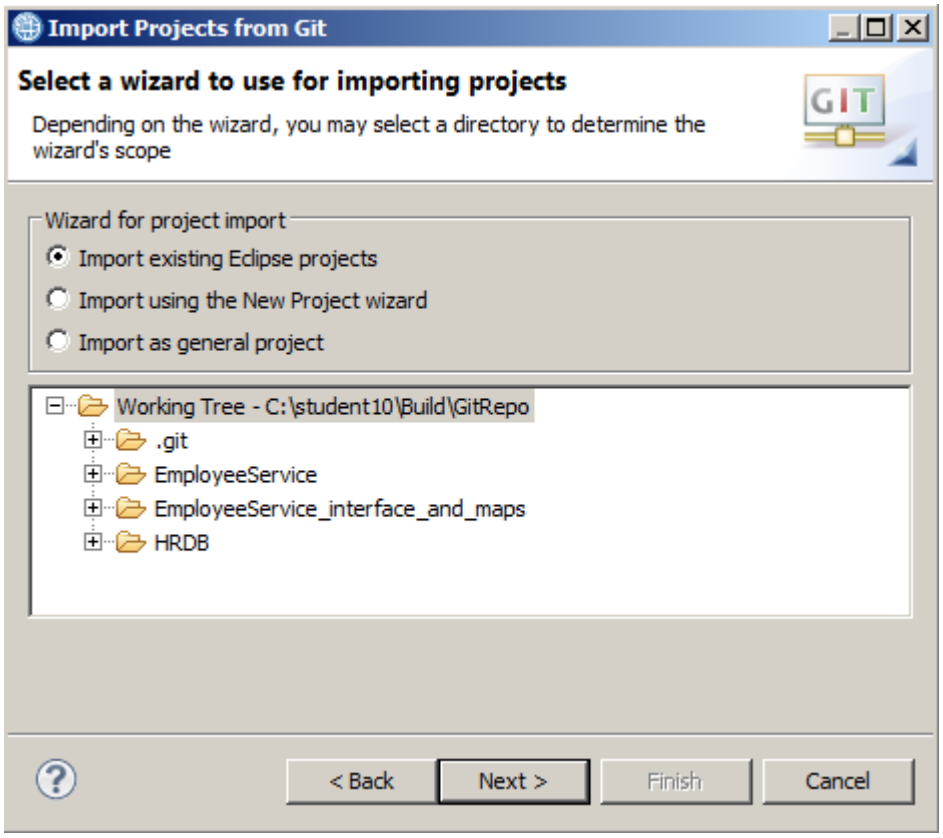
7. Select **Existing local repository**, and click Next.



8. Select **GitRepo**, and click Next.

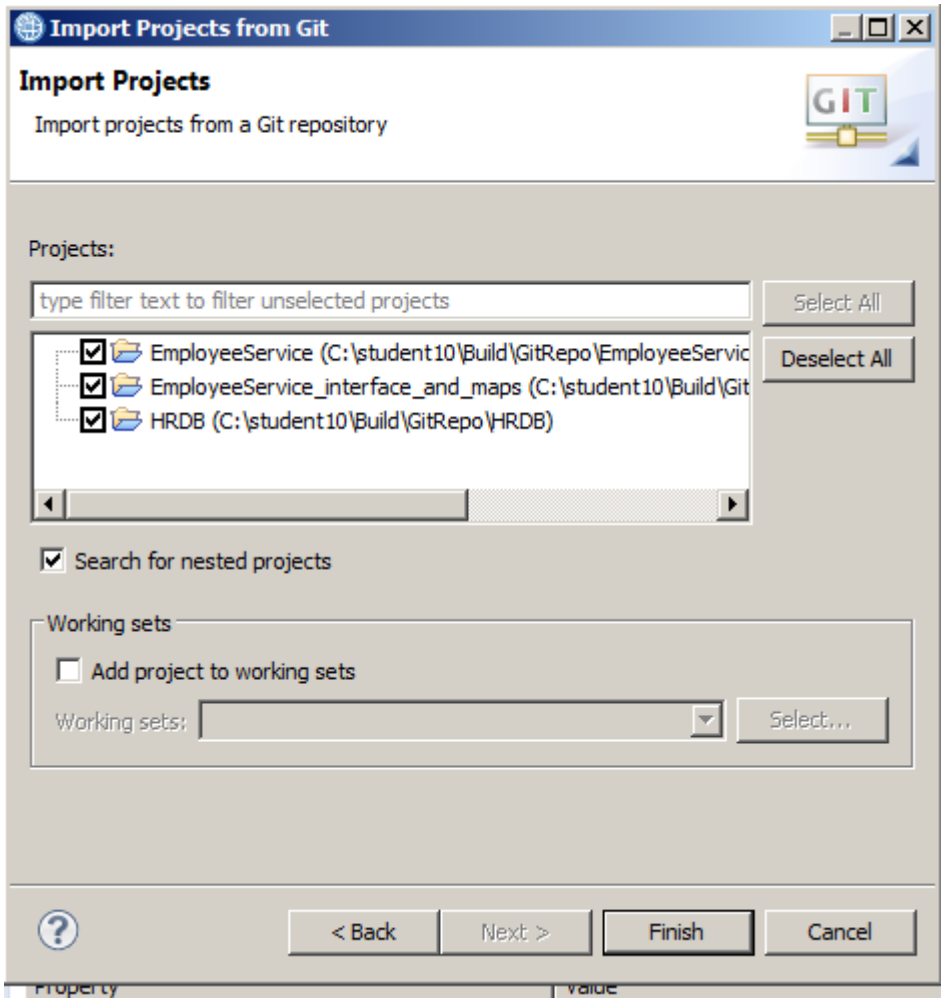


9. Select "Import existing Eclipse projects" (should be the default choice), and click Next.

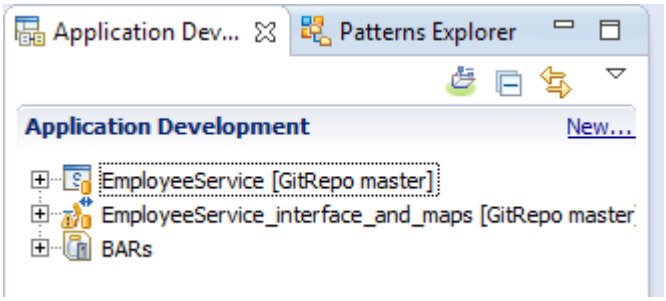


10. Ensure all three projects are selected.

    Click Finish.



11. The projects will be imported into the workspace. Note that the HRDB project will not be shown in the navigator, since it is a project included by reference from the library project.

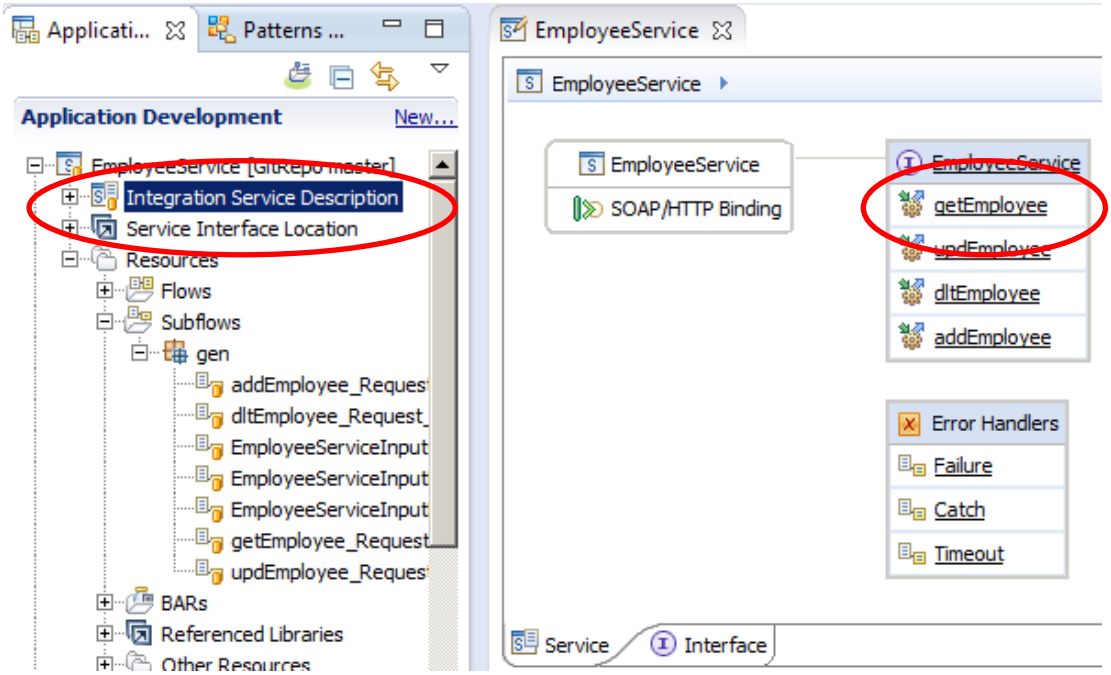


5.2 Update the Integration Service

1. In this example, you will make a change to the integration service to demonstrate that changes can be automatically built, deployed and tested.

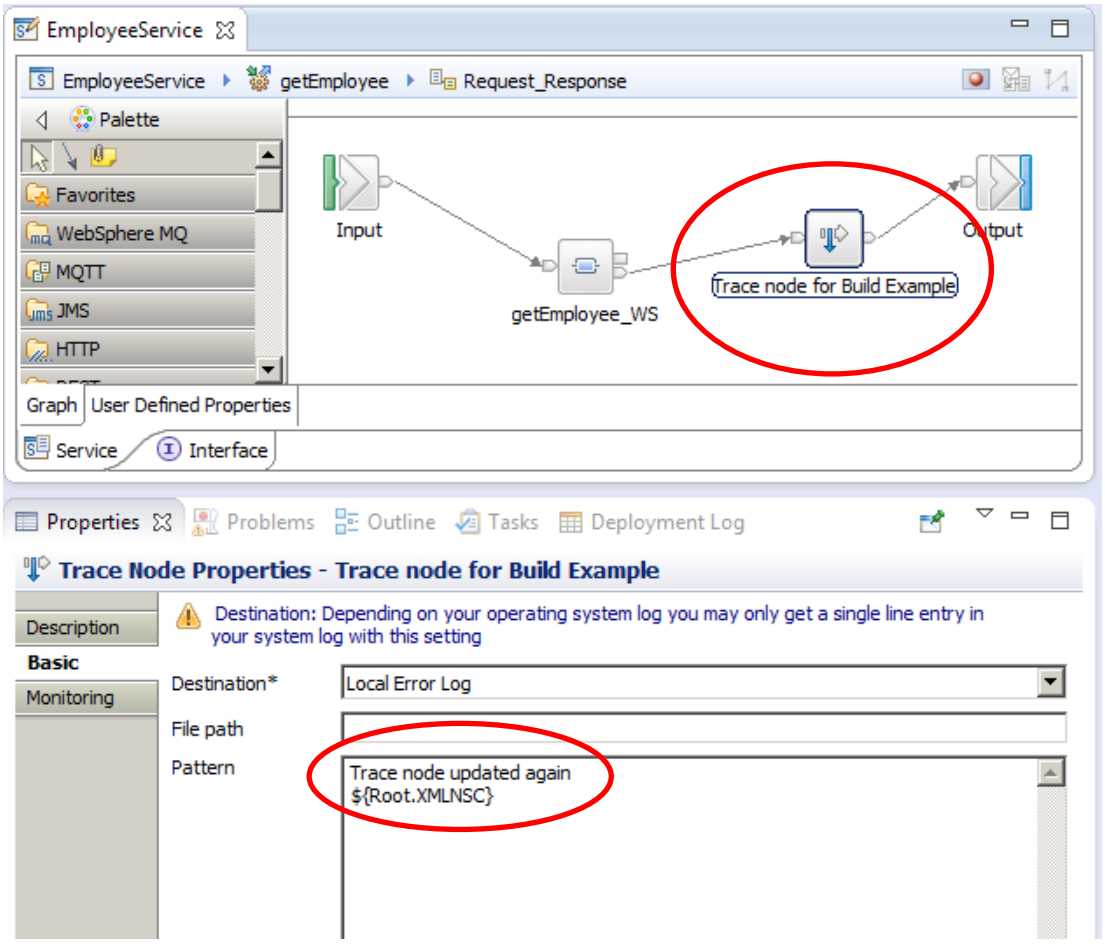
   Expand the EmployeeService integration service, and double-click "Integration Service Description".

   Click the getEmployee operation.



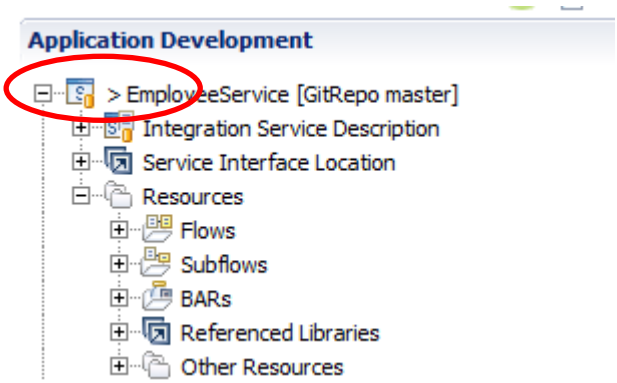
2. In the subflow, highlight the Trace node.

   In the properties of the Trace node, change the Pattern text to something else, for example "**Trace node updated again**".

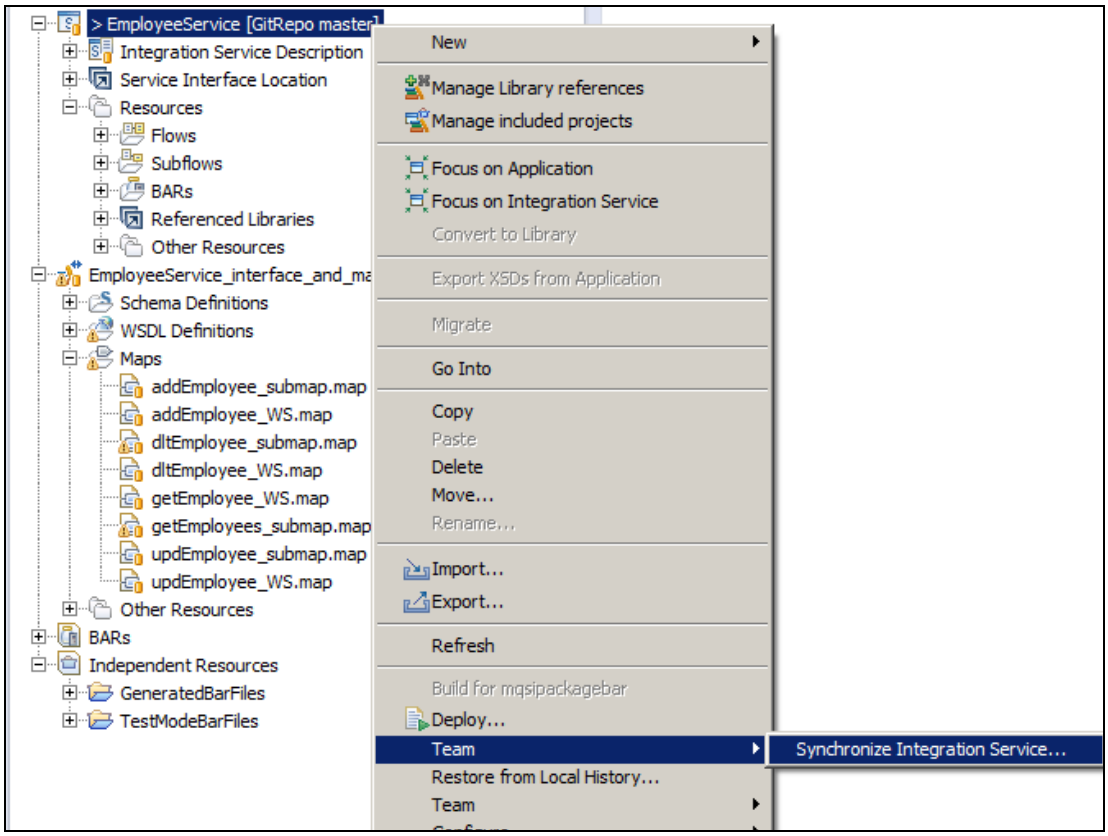


3. Save the updated subflow (Ctrl-S). When you save the subflow, the navigator will show that the integration service has been updated, but not committed to the Git code repository.

   This is indicated by the small arrow (>) on the project line.

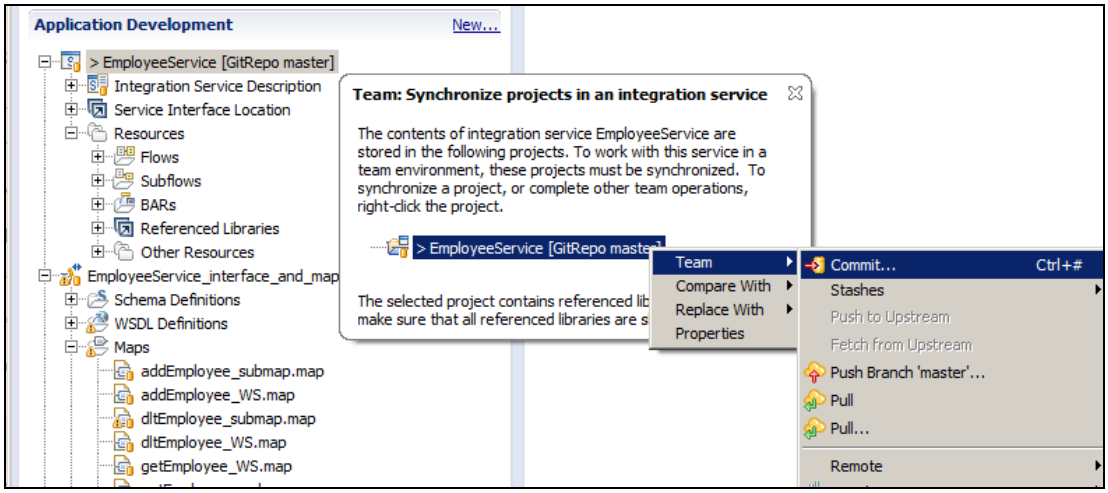


4. The change needs to be committed to Git. To do this, right-click the EmployeeService project, click Team, then Synchronize Integration Service.



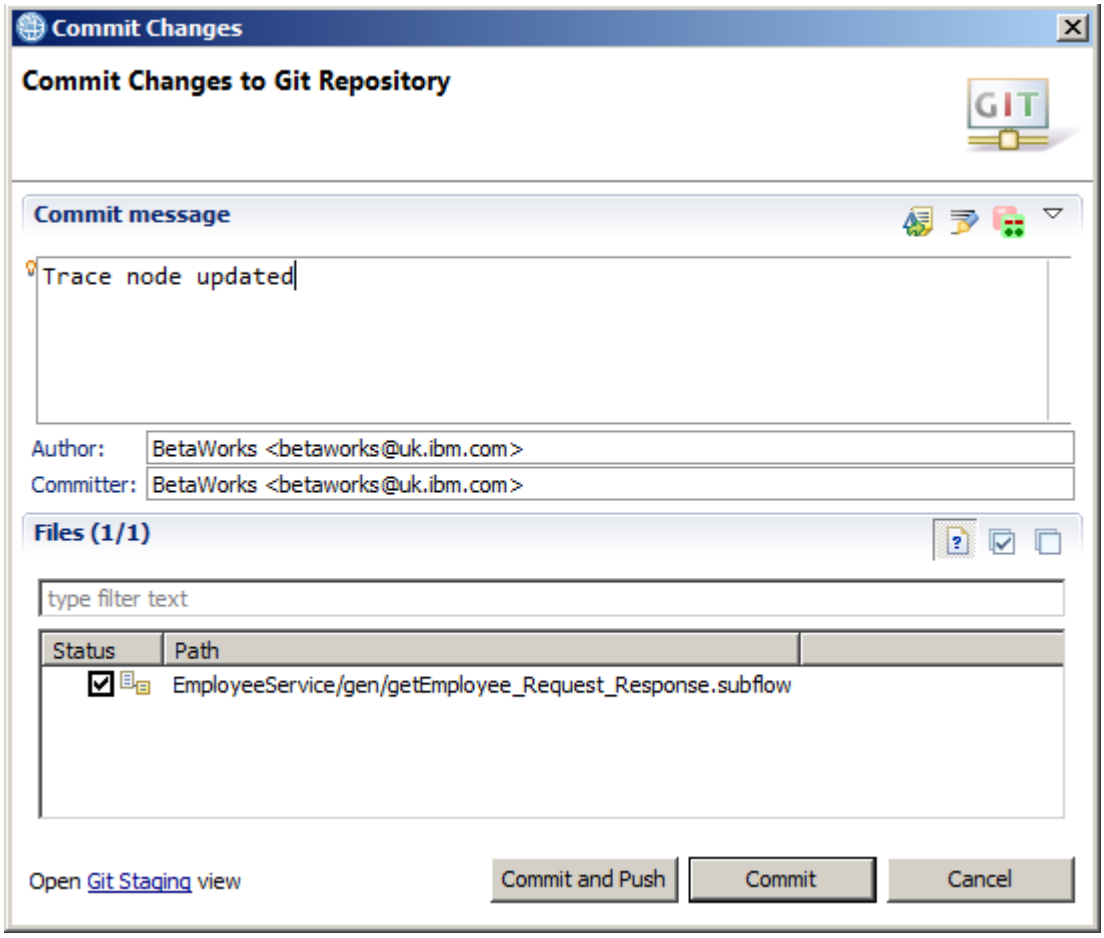
5. The "Team: Synchronize" pop-up window will open. In this example, the integration service is represented in the IIB Toolkit by just a single Eclipse project, and this is shown in the pop-up window. Note - sometimes, an IIB Application, Library or Service can be represented by more than one Eclipse project, perhaps if there is a project reference, or if there is a java project. If this is the case, then each Eclipse project will appear in the pop-up, and each must be committed to the code repository.

   In this case, there is a single Eclipse project, which should be committed now. Right-click EmployeeService and select Team, Commit. This will commit the change to Git immediately (recall we configured the EGit plugin to commit immediately, and bypass the Git staging process).

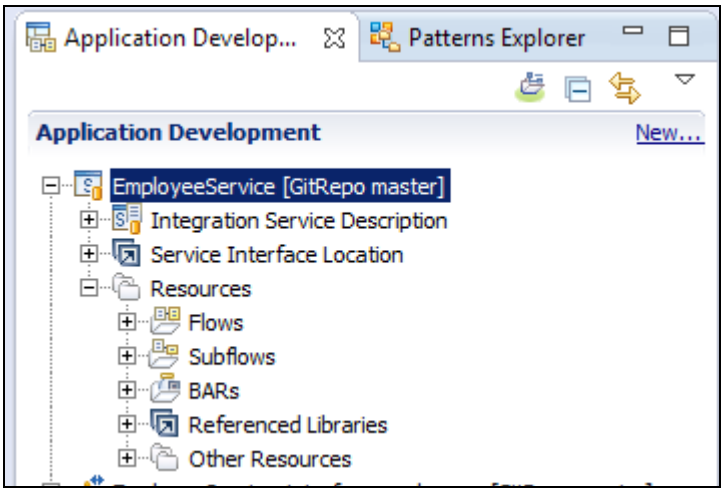


6. In the Commit Changes window, provide a suitable Commit message, ensure the subflow file is selected, and click Commit.

   Note - the Author and Committed fields have been populated by the EGit plugin, which we looked at earlier.



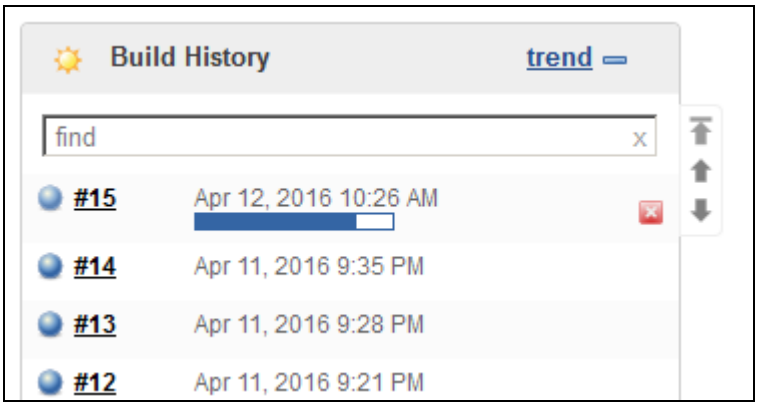
7. The change arrow will disappear from the navigator project, and the change will be committed to Git.



8. Fairly quickly, switch back to the Jenkins page in the Firefox browser.

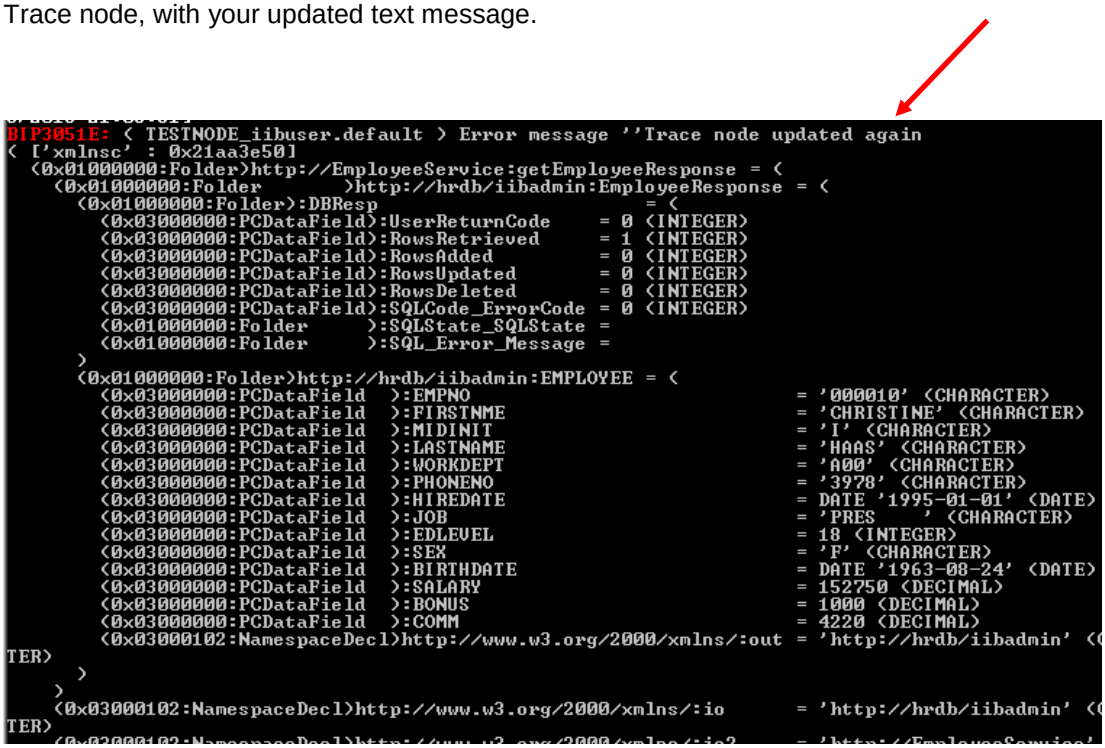
   Because you configured Jenkins to poll Git for changes every minute, it shouldn't be too long before Jenkins notices this change, and invokes the Jenkins build process.

   You will see a new task in the Build History.



9. When the Jenkins build has finished, switch to the IIB Event Log Viewer.

   Remember that the Ant script includes the final stage which invokes the integration service, using curl. So, if the Jenkins build was successful, you should see the output of the updated Trace node, with your updated text message.



6. Appendix

6.1 Configure Integration Bus node to work with DB2

If you have already done Lab 1 in this series (create an Integration Service), this step will already have been done.

To run this lab, the Integration Bus node must be enabled to allow a JDBC connection to the HRDB database.

1. Open an IIB Command Console (from the Start menu), and navigate to

   `c:\student10\Create_HR_database`

2. Run the command

   `3_Create_JDBC_for_HRDB`

   Accept the defaults presented in the script. This will create the required JDBC configurable service for the HRDB database.

3. Run the command

   `4_Create_HRDB_SecurityID`

4. Stop and restart the node to enable the above definitions to be activated

   `mqsistop TESTNODE_iibuser`

   `mqsistart TESTNODE_iibuser`

This will create the necessary security credentials enabling TESTNODE_iibuser to connect to the database.

Recreating the HRDB database and tables

The HRDB database, and the EMPLOYEE and DEPARTMENT tables have already been created on the supplied VMWare image. If you wish to recreate your own instance of this database, the command files **`1_Create_HRDB_database.cmd`** and **`2_Create_HRDB_Tables.cmd`** are provided for this. If used in conjunction with the VM image, these commands must be run under the user "iibadmin". Appropriate database permissions are included in the scripts to GRANT access to the user iibuser.

7. END OF LAB GUIDE