Using GitHub as an IBM Integration Bus Code Repository

Paul Faulkner and Jim Berube - Published on February 13, 2018 / Updated on September 30, 2020

Overview

Having a code repository is something that every developer and development team should have. This is a place where code can be saved, versioned, and shared that's independent from local development machines. There are many vendors available; like CVS, SVN, and Rational to name just a few. All of these products require a separate machine to be setup somewhere accessible on the developer's network. Then, of course there's the set up and administration of the software and management of the server itself. GitHub as a repository offers the advantage of already being setup and available as far as the software and hardware goes. All that's required is an internet connection, a GitHub account, and for someone to create and administer the desired repositories for your organization to access them.

In this tutorial, we will show how to setup a GitHub repository and how to use that repository as an IBM Integration Bus (IIB) code repository within Eclipse. We will demonstrate the steps necessary to create a GitHub organization and repository. The installation of the Git for Windows software and the eGit plugin that's required on the client machine will be covered, including making the connection to a previously created repository via the Eclipse toolkit. Additional topics include; how to check-in and check-out code from the repository, creating branches and managing code merges. This tutorial should demonstrate enough of the basic concepts of utilizing a GitHub repository that a developer or development team should be able to create and manage their own repository.

GitHub or Enterprise GitHub

These instructions can be used with a standard GitHub account from https://github.com/ or a Corporate Enterprise GitHub account. A public repository on standard GitHub is open to anyone who has an account on GitHub. (It is possible to have a private repository on standard GitHub, with a paid account.) Enterprise GitHub public repositories are only available to members of that Corporation or outside members who are invited to join your specific Organization. This tutorial uses IBM's internal Enterprise GitHub as the default location for any repositories created, for demonstration purposes.

Local Software Install

Some software installation and configuration is necessary to use GitHub as a repository as described in this tutorial. The Git-for-Windows software must be installed and the eGit plug-in must be installed in your IIB Eclipse toolkit.

Software used in this tutorial:

- IBM Integration Bus (Developer version link provided)
- Git-for-Windows Git-2.15.1.2-64-bit.exe
- Git Releases eGit version 4.2 (http://download.eclipse.org/egit/updates-4.2/)
- Git Releases eGit version 4.4 (http://download.eclipse.org/egit/updates-4.4/) For IIB FP 16 and above
- Git Releases eGit version 5 (https://download.eclipse.org/egit/updates-5.0/) For ACE (IIB v11)
- Git Releases eGit version 5.8.1 (https://download.eclipse.org/egit/updates-5.8.1/) For ACE (IIB v11)
- GitHub http://github.com, (or your Enterprise GitHub location)

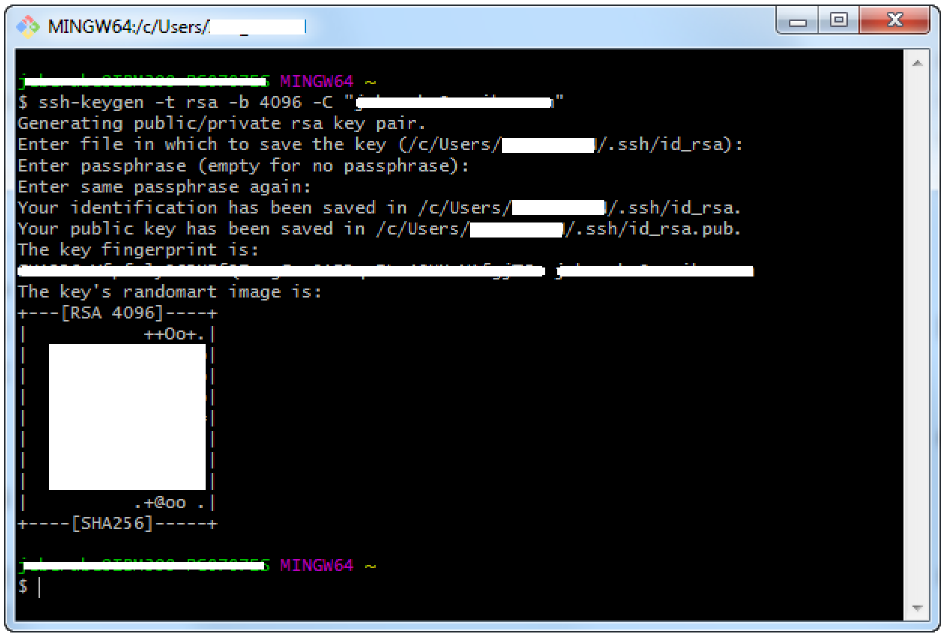
Aside from the standard IIB v10 install (not covered here), the additional software needed will be Git-for-Windows and the eGit plugin. The Git-for-Windows software can be downloaded from the link specified in the software sidebar. The Git-for-Windows software includes the git-bash utility needed when creating your ssh key. The other piece of software needed is the eGit plugin for the Eclipse Toolkit, the link will be used in the toolkit, when installing this plugin, the software does not need to be downloaded in advance. The plugin software needs to be compatible with Eclipse version 4.2.2, until IIB version 10.0.0.15.

For IIB version 10.0.0.16 and above, the Toolkit's Eclipse version was updated to 4.4.2.

So, the eGit Plugin needs to be version 4.4 for IIB version 10.0.0.16 and above.

For App Connect Enterprise (ACE), which is IIB version 11, the Toolkit's Eclipse version is Luna, so you'll need to use version 5 of the eGit Plugin with that release. I experienced issues with eGit version 5.0 on ACE version 11.0.0.10, but version 5.8.1 installed correctly. I have not checked each fixpack, so you might need to try more than 1 version.

1. **For IIB, version 10**, in the git-bash window, enter the following command, substituting your email address: `ssh-keygen -t rsa -b 4096 -C "youremail.com"`
2. **For ACE, IIB version 11**, in the git-bash window, enter the following command, substituting your email address:
   `ssh-keygen -m PEM -t rsa -b 4096 -C "youremail.com"`
3. Click enter to accept the default file name `'id_rsa'`, which should be a location under your user profile directory like: `'C:\Users\<User Profile>\.ssh\id_rsa'`
4. Next, enter a passphrase and make sure it's something you can remember. Then enter that passphrase again to confirm. At the end, you should see something like this:

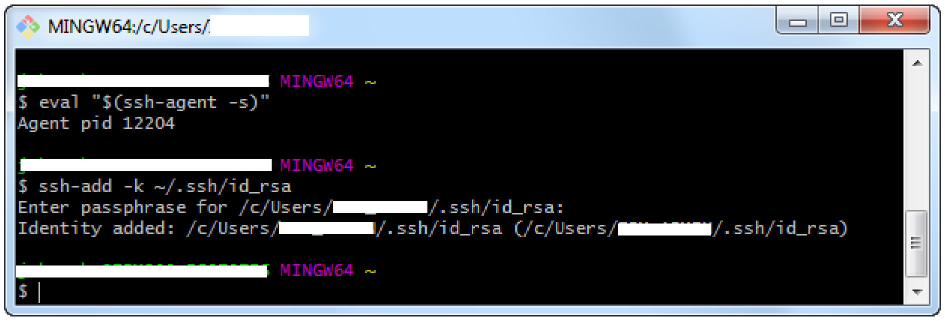


MINGW64:/c/Users/[redacted]

```
[redacted] MINGW64 ~
$ ssh-keygen -t rsa -b 4096 -C "[redacted]"
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/[redacted]/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/[redacted]/.ssh/id_rsa.
Your public key has been saved in /c/Users/[redacted]/.ssh/id_rsa.pub.
The key fingerprint is:
[redacted]
The key's randomart image is:
+---[RSA 4096]----+
|          ++Oo+.|
[redacted]
|          .+@oo .|
+----[SHA256]-----+

[redacted] MINGW64 ~
$ |
```

5. Next, the key needs to be added to the ssh agent. Check to see if the ssh agent is running by executing the following command:
   `eval "$(ssh-agent -s)"`
   This command should return a process id. If you are running on mac or need additional information, please see GitHub's help on this topic here.
6. Now, add the key to the agent by executing the following command:
   `ssh-add -k ~/.ssh/id_rsa`
7. You will need to enter your passphrase when prompted. When complete, you should see something like below:

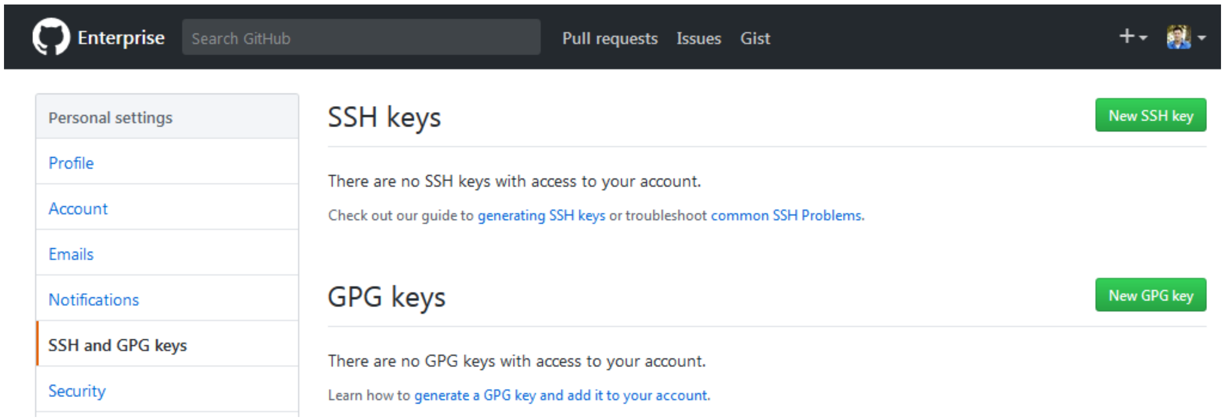


```
[redacted] MINGW64 ~
$ eval "$(ssh-agent -s)"
Agent pid 12204

[redacted] MINGW64 ~
$ ssh-add -k ~/.ssh/id_rsa
Enter passphrase for /c/Users/[redacted]/.ssh/id_rsa:
Identity added: /c/Users/[redacted]/.ssh/id_rsa (/c/Users/[redacted]/.ssh/id_rsa)

[redacted] MINGW64 ~
$ |
```

8. The next step is to add your ssh key to your Enterprise Git Hub account. Login to your Corporation's Enterprise GitHub page. Once logged in, under your profile, go to "Settings". Then click "SSH and GPG Keys" and you should see the following:



SSH keys

There are no SSH keys with access to your account.

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

GPG keys

There are no GPG keys with access to your account.

Learn how to generate a GPG key and add it to your account.

9. Click the "New SSH key" button and enter a title for this ssh key. Keep it simple and identifiable.

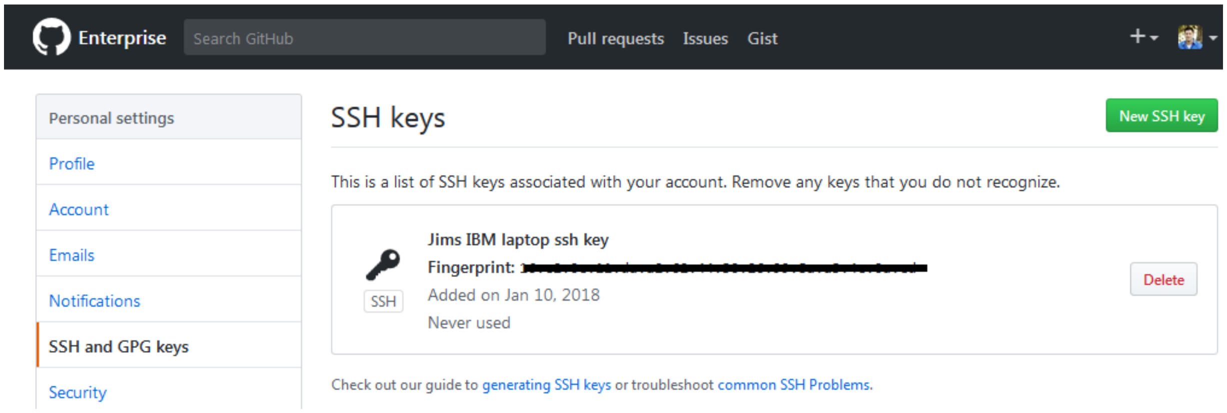
   Go to your git-bash session and enter the following command to copy your key:

   `pbcopy < ~/.ssh/id_rsa.pub`

   or (`pbcopy` doesn't work for me, so I need to use `clip`)

   `clip < ~/.ssh/id_rsa.pub`

10. Now paste the contents into the space on your Enterprise GitHub page under "key" and click "Add SSH key". The result should look something like this:



SSH keys

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.

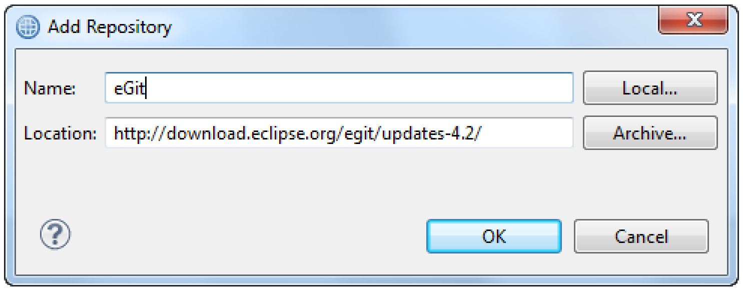
Jims IBM laptop ssh key
Fingerprint: [redacted]
Added on Jan 10, 2018
Never used

Check out our guide to generating SSH keys or troubleshoot common SSH Problems.

11. Next, we'll add the eGit plugin to the IIB Eclipse Toolkit.

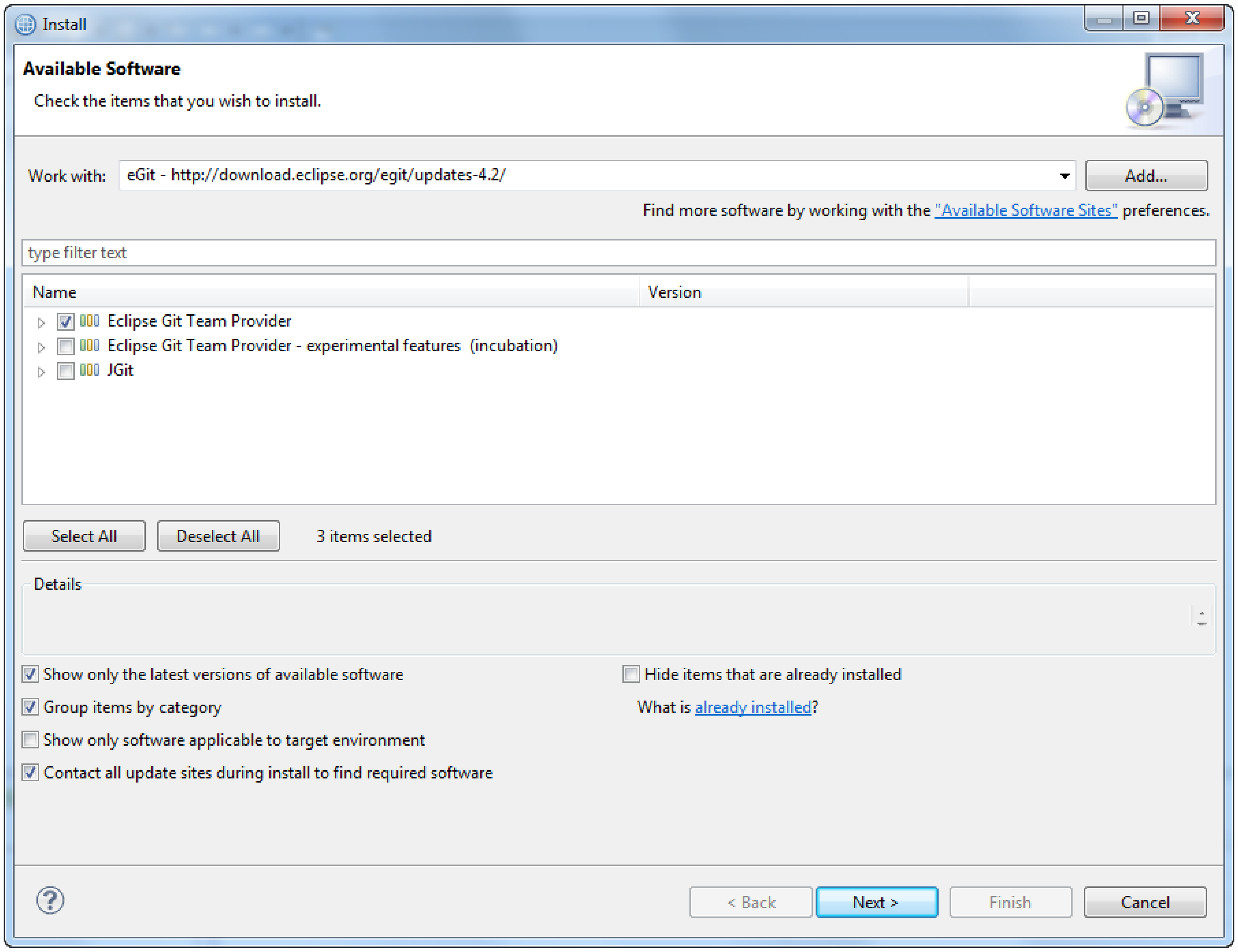
    Note: The Toolkit will need to be restarted at the end of this process, so make sure your work has been saved prior to beginning this activity if your Toolkit was already open.

12. Open your IIB Eclipse Toolkit and go to Help –> Install New Software
13. In the interface that pops up, click the Add button in the top right corner so we can add the site for the eGit plugin. Give it a name, maybe something like "eGit" and use this address: http://download.eclipse.org/egit/updates-4.2/, http://download.eclipse.org/egit/updates-4.4/ for IIB 10.0.0.16 and up, http://download.eclipse.org/egit/updates-5.0/ or http://download.eclipse.org/egit/updates-5.8.1/ for ACE Toolkits, then click "OK"



Name: eGit
Location: http://download.eclipse.org/egit/updates-4.2/

14. Select just the top item "Eclipse Git Team Provider" or it might say "Git integration for Eclipse" and click "Next".



Available Software
Check the items that you wish to install.

Work with: eGit - http://download.eclipse.org/egit/updates-4.2/

Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
[✓] Eclipse Git Team Provider	
[ ] Eclipse Git Team Provider - experimental features (incubation)	
[ ] JGit	

3 items selected

Details

[✓] Show only the latest versions of available software
[✓] Group items by category
[ ] Show only software applicable to target environment
[✓] Contact all update sites during install to find required software
[ ] Hide items that are already installed
What is already installed?

15. Click "Next" again, then Accept the license terms and click "Finish".

The IIB Eclipse toolkit will need to be restarted when complete.

Once restarted, you can open the Git perspective by doing the following:

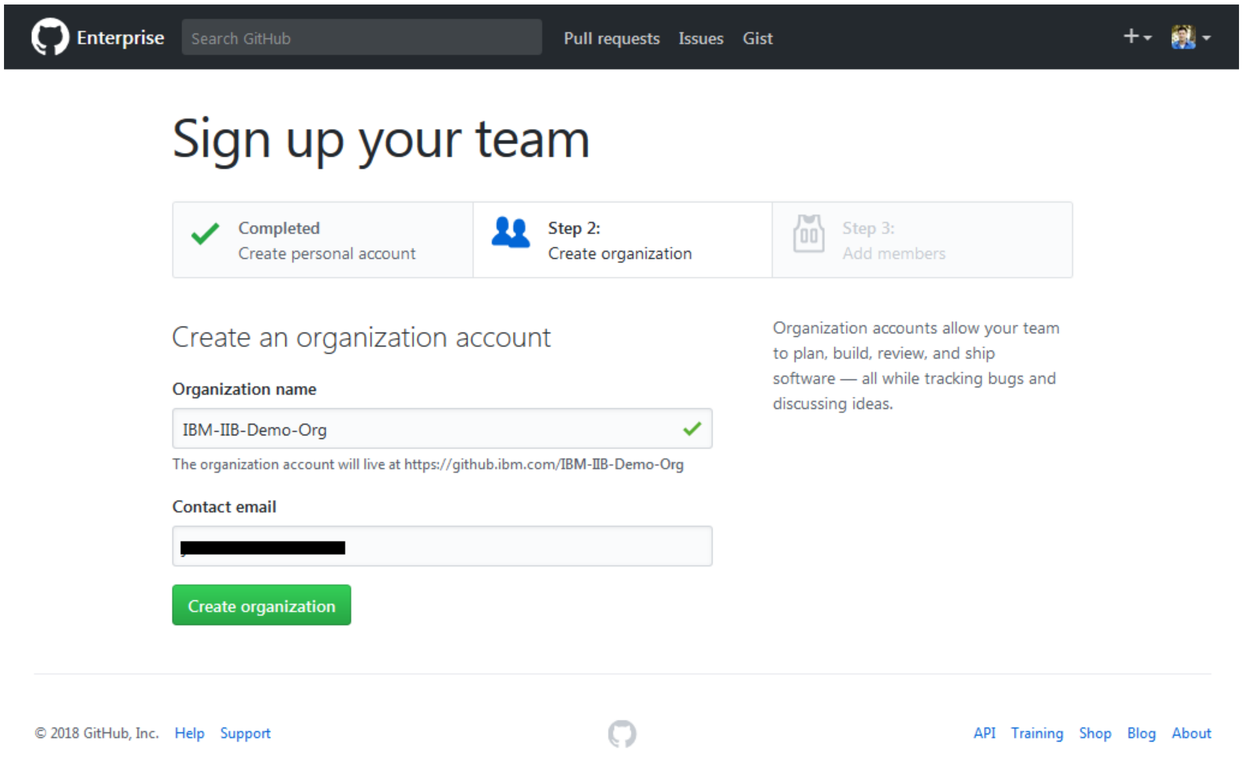
16. Go to Window –> Open Perspective —> Other
17. Then select "Git" from the list.

Setting up an Enterprise GitHub Repository

Now we'll create a repository to store code in. This activity does not rely on the local software setup, but you will not be able to connect your IIB Eclipse toolkit to your GitHub repository until that has been completed.

In GitHub, you can create repositories directly under your profile, which you will own and can add other members to. Another way is to create an Organization and create your repositories under that instead. In this tutorial, we'll create an Organization first. Creating an Organization allows you to assign different privileges to the members you invite to the Organization.

1. To start with, go to the Organizations tab under Settings in GitHub Enterprise. Then click on the "New Organization" button in the top right corner. Enter the name of your Organization and your contact email address, as shown:



Sign up your team

Completed
Create personal account

Step 2:
Create organization

Step 3:
Add members

Create an organization account

Organization accounts allow your team to plan, build, review, and ship software — all while tracking bugs and discussing ideas.

Organization name

IBM-IIB-Demo-Org

The organization account will live at https://github.ibm.com/IBM-IIB-Demo-Org

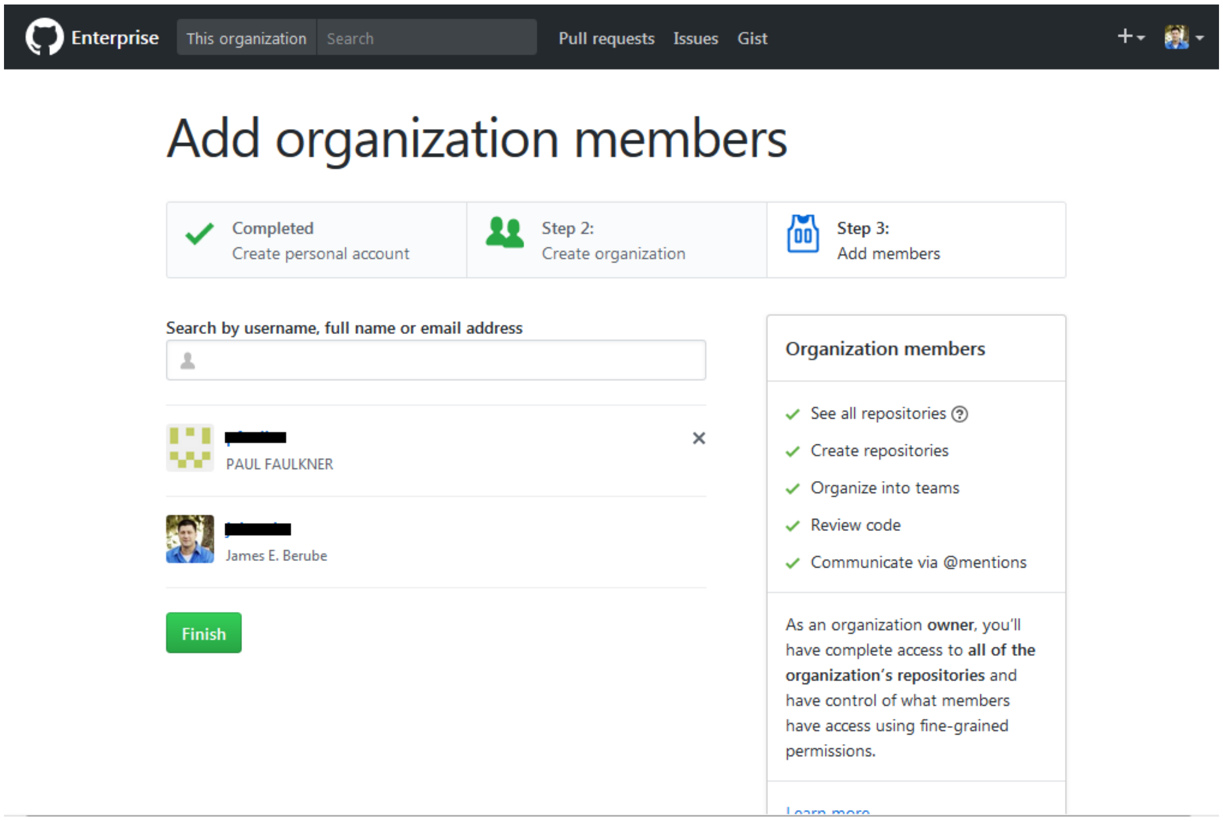
Contact email

[redacted]

Create organization

© 2018 GitHub, Inc. Help Support API Training Shop Blog About

2. Then click the "Create organization" button.
3. Once that is complete, you can add members to your Organization. Just search for members and click their information to add them. GitHub members outside your Enterprise can be added


Enterprise This organization Search Pull requests Issues Gist

Add organization members

Completed
Create personal account

Step 2:
Create organization

Step 3:
Add members

Search by username, full name or email address

[redacted]
PAUL FAULKNER

[redacted]
James E. Berube

Finish

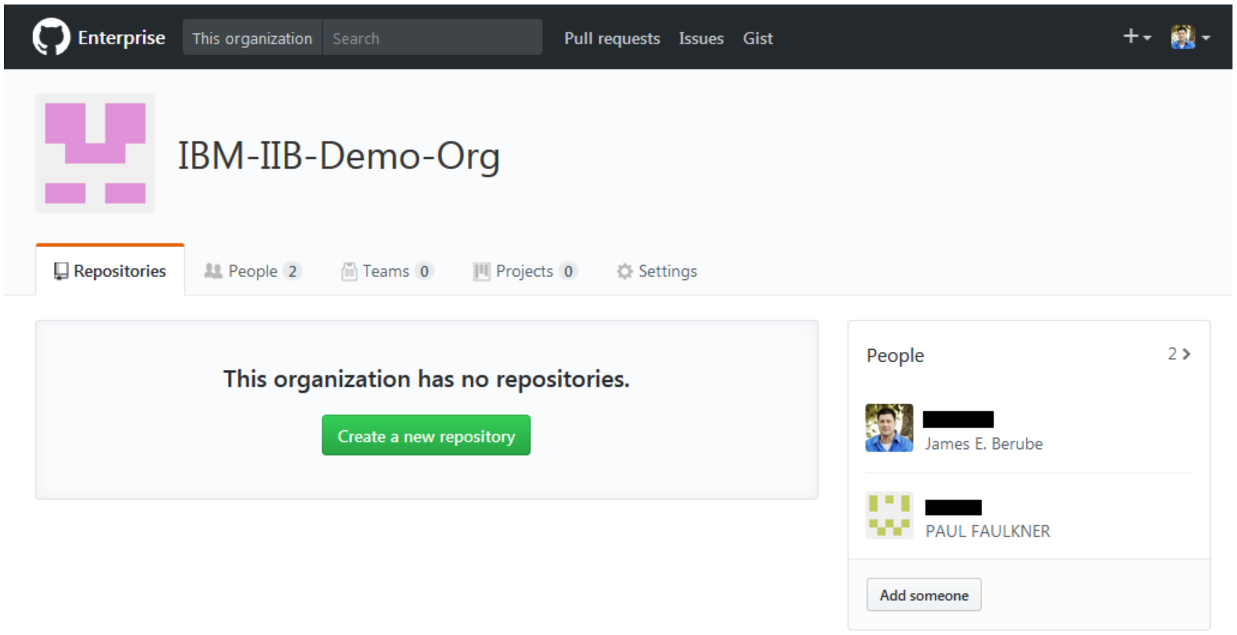
Organization members

- ✓ See all repositories
- ✓ Create repositories
- ✓ Organize into teams
- ✓ Review code
- ✓ Communicate via @mentions

As an organization **owner**, you'll have complete access to **all of the organization's repositories** and have control of what members have access using fine-grained permissions.

4. Once all members have been added, click the "Finish" button.

   You should now see the home page for the Organization that was just created. There are several tabs available; to create and manage the Organization's repositories, manage the members and their access, create and manage teams, create and manage projects and the general settings for the organization.


Enterprise This organization Search Pull requests Issues Gist

IBM-IIB-Demo-Org

Repositories | People 2 | Teams 0 | Projects 0 | Settings

This organization has no repositories.

Create a new repository

People 2

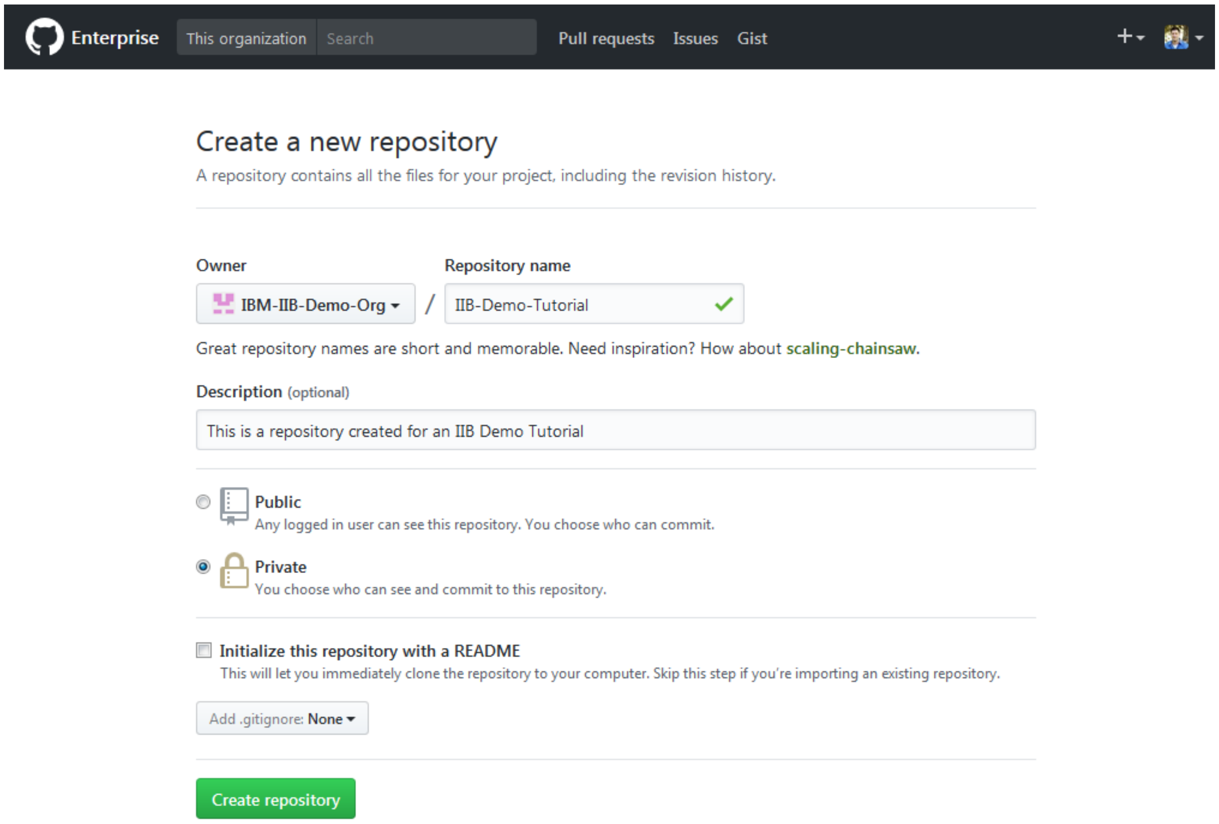
[redacted]
James E. Berube

[redacted]
PAUL FAULKNER

Add someone

   For now, we're just going to focus on creating a repository for our organization.

5. Click the button to "Create a new repository".
6. Give your repository a name and a description. We'll be creating this as a Private repository, which means it won't be seen by anyone who's not a member of the Organization.


Enterprise | This organization | Search | Pull requests | Issues | Gist

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: IBM-IIB-Demo-Org / Repository name: IIB-Demo-Tutorial ✓

Great repository names are short and memorable. Need inspiration? How about scaling-chainsaw.

Description (optional): This is a repository created for an IIB Demo Tutorial

Public
Any logged in user can see this repository. You choose who can commit.

[✓ Private]
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

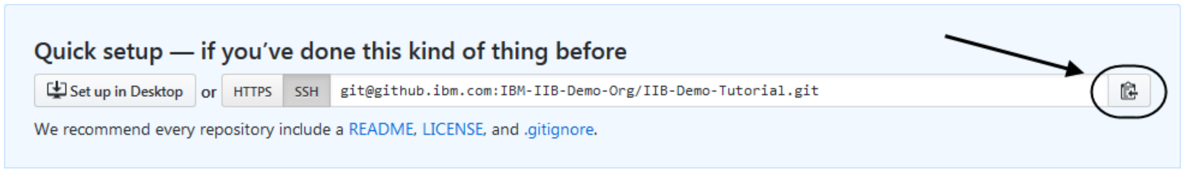
Add .gitignore: None

Now that a repository has been created, we'll add it to the IIB Eclipse Toolkit.

Connecting to the Repository

The Git software has been installed and configured. A GitHub repository has been created. Now, you need to connect your IIB Eclipse toolkit to the repository so you can start checking code in.

1. Go to your Enterprise Organization Repository page and click the button to copy the information in the first box, under "Quick setup".

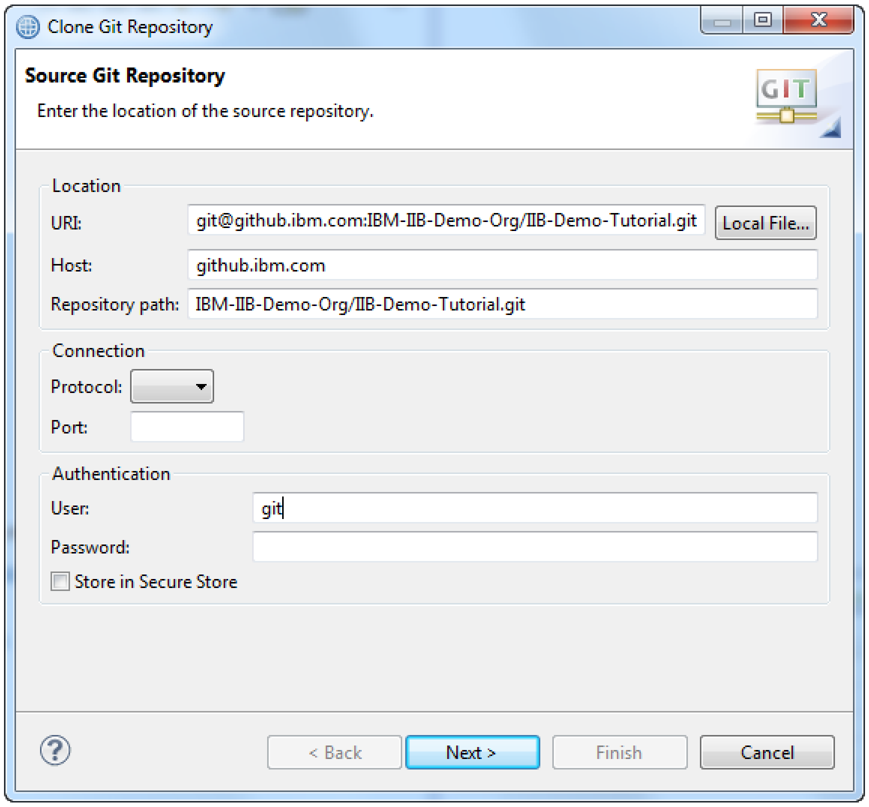

Quick setup — if you've done this kind of thing before

```
git@github.ibm.com:IBM-IIB-Demo-Org/IIB-Demo-Tutorial.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

2. Open your IIB Eclipse Toolkit and go to the Git perspective.
3. Click the link to "Clone a Git repository" and a new window will open.
4. Click Paste.

   All of the information should be auto-populated in the correct places in the window to add your repository link.


Clone Git Repository

Source Git Repository

Enter the location of the source repository.

Location

URI: git@github.ibm.com:IBM-IIB-Demo-Org/IIB-Demo-Tutorial.git

Host: github.ibm.com

Repository path: IBM-IIB-Demo-Org/IIB-Demo-Tutorial.git

Connection

Protocol:

Port:

Authentication

User: git

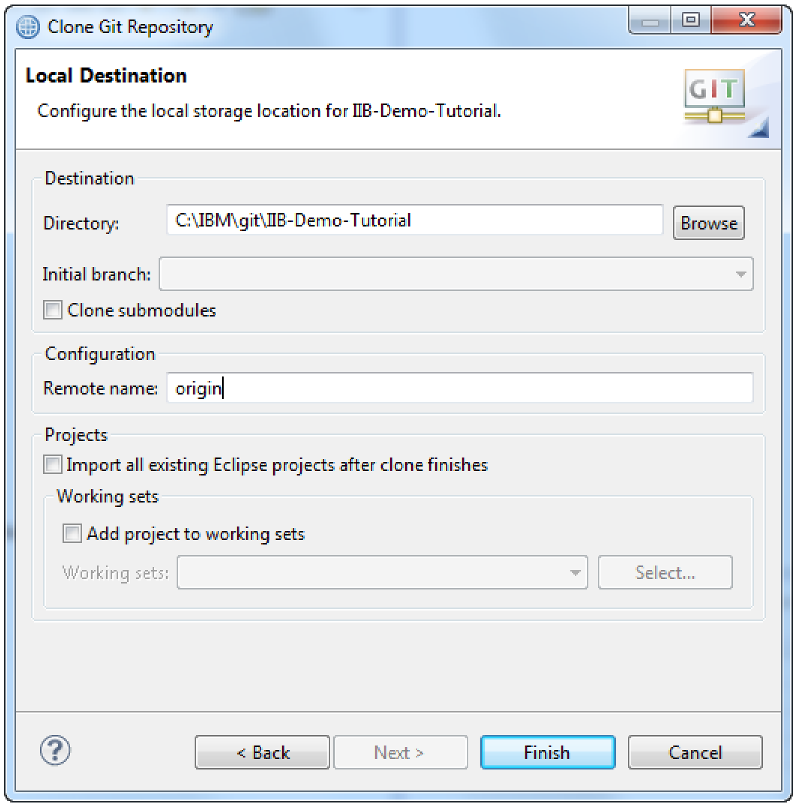
Password:

Store in Secure Store

5. Click "Next". If you get a popup about the site not being trusted yet, click "Yes" to accept and "Yes" again to create a known hosts file. Enter the passphrase for the SSH key you created earlier and click "OK".
6. You will see a message that the Branch is currently empty under Branch selection. This is fine, since we haven't added anything to the repository yet. Click "Next".

   On the next screen, you are being asked where you'd like to store your local version of the repository on your file system. The location defaults to your user profile. If you don't want to keep it there, it would be a good idea to at least keep the Git directory and the name of your repository if you change anything.

7. If other users are adding this connection later, after code has been added to the repository, we strongly recommend clicking the checkbox to "Import all existing Eclipse projects after the clone finishes" when making this connection. If they do not, they will not see anything added to their workspace.


Clone Git Repository

Local Destination

Configure the local storage location for IIB-Demo-Tutorial.

Destination

Directory: C:\IBM\git\IIB-Demo-Tutorial

Initial branch:

☐ Clone submodules

Configuration

Remote name: origin

Projects

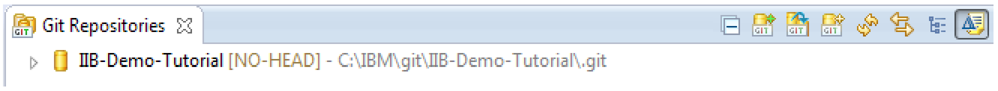
☐ Import all existing Eclipse projects after clone finishes

Working sets

☐ Add project to working sets

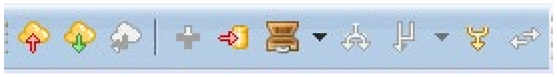
Working sets:

8. Then click "Finish".
9. You should now see a connection to your repository listed under Git Repositories.


Git Repositories

IIB-Demo-Tutorial [NO-HEAD] - C:\IBM\git\IIB-Demo-Tutorial\.git

10. There is a Git toolbar that can be added to the Git perspective. To add the Git commands. Open the Git perspective, then go to the Window menu bar and select Customize Perspective.
11. Go to the Command Groups Availability tab, then scroll down and check the Git command groups.
12. The Git commands will be added as a toolbar.

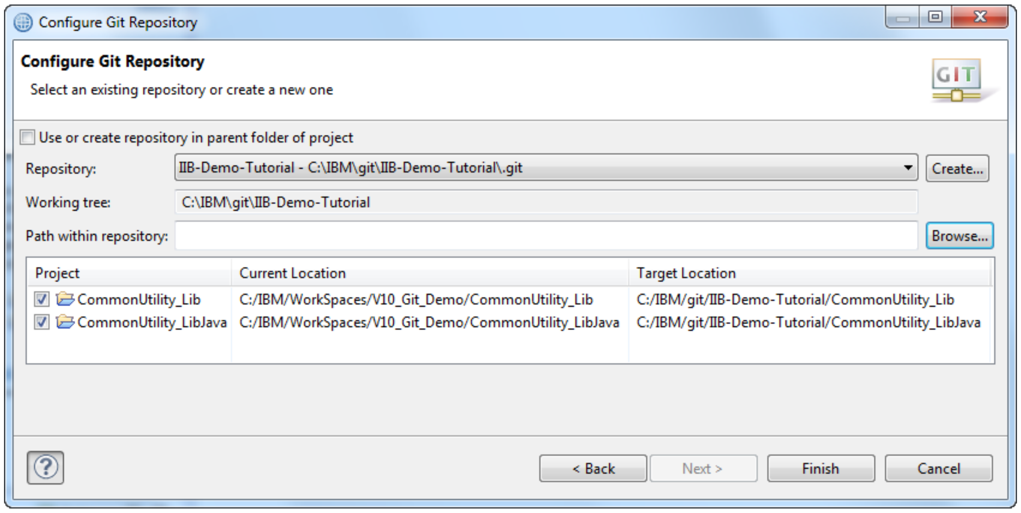


Initial Code Check-in

Your IIB Eclipse toolkit is now connected to the GitHub repository and it's time to start checking code in for the first time. This section describes the process to check code in that doesn't yet exist in your repository.

1. To initially check code into the Repository, go to the Integration Development Perspective.
   We recommended that you close any open objects that are going to be checked in. The files will still appear in your workspace, but the physical location of the files will change. The files will be removed from the workspace on your file system and moved to your local git repository file system location.

2. Right click on the code that you'd like to check in and select Team –> Share Library (or Application) in the window that pops up.
3. Share the code and select "Git" for the repository and click "Next".


Configure Git Repository

Configure Git Repository

Select an existing repository or create a new one

☐ Use or create repository in parent folder of project

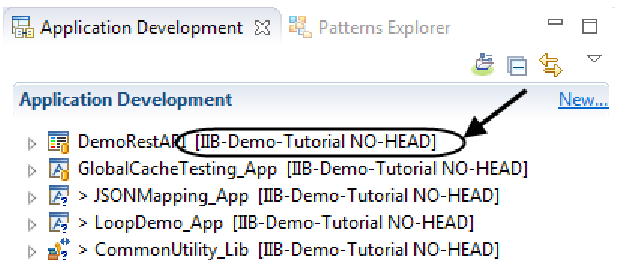
Repository: IIB-Demo-Tutorial - C:\IBM\git\IIB-Demo-Tutorial\.git

Working tree: C:\IBM\git\IIB-Demo-Tutorial

Path within repository:

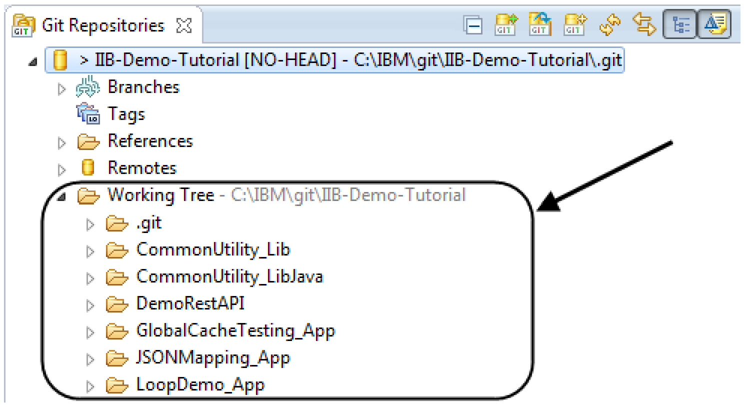
Project	Current Location	Target Location
☑ CommonUtility_Lib	C:/IBM/WorkSpaces/V10_Git_Demo/CommonUtility_Lib	C:/IBM/git/IIB-Demo-Tutorial/CommonUtility_Lib
☑ CommonUtility_LibJava	C:/IBM/WorkSpaces/V10_Git_Demo/CommonUtility_LibJava	C:/IBM/git/IIB-Demo-Tutorial/CommonUtility_LibJava

4. In the window that opens, select your Git repository from the drop down and make sure your projects are checked, then click "Finish".
5. Once you've shared the project(s) with your repository, they will be moved, under the covers, from your workspace to your local git repository directory, indicated by the Target Location directory. Now, when opening the toolkit, you will still go to the same workspace, but all the code will actually be physically located in the target location.
6. Now, in your toolkit, notice that all of the code has markers indicating it's associated with a repository.



DemoRestAPI [IIB-Demo-Tutorial NO-HEAD]
GlobalCacheTesting_App [IIB-Demo-Tutorial NO-HEAD]
> JSONMapping_App [IIB-Demo-Tutorial NO-HEAD]
> LoopDemo_App [IIB-Demo-Tutorial NO-HEAD]
> CommonUtility_Lib [IIB-Demo-Tutorial NO-HEAD]

7. Switching over to the Git perspective, the code that was just shared will now be visible under Working Tree.

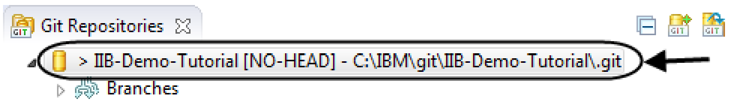


> IIB-Demo-Tutorial [NO-HEAD] - C:\IBM\git\IIB-Demo-Tutorial\.git
Branches
Tags
References
Remotes
Working Tree - C:\IBM\git\IIB-Demo-Tutorial
.git
CommonUtility_Lib
CommonUtility_LibJava
DemoRestAPI
GlobalCacheTesting_App
JSONMapping_App
LoopDemo_App

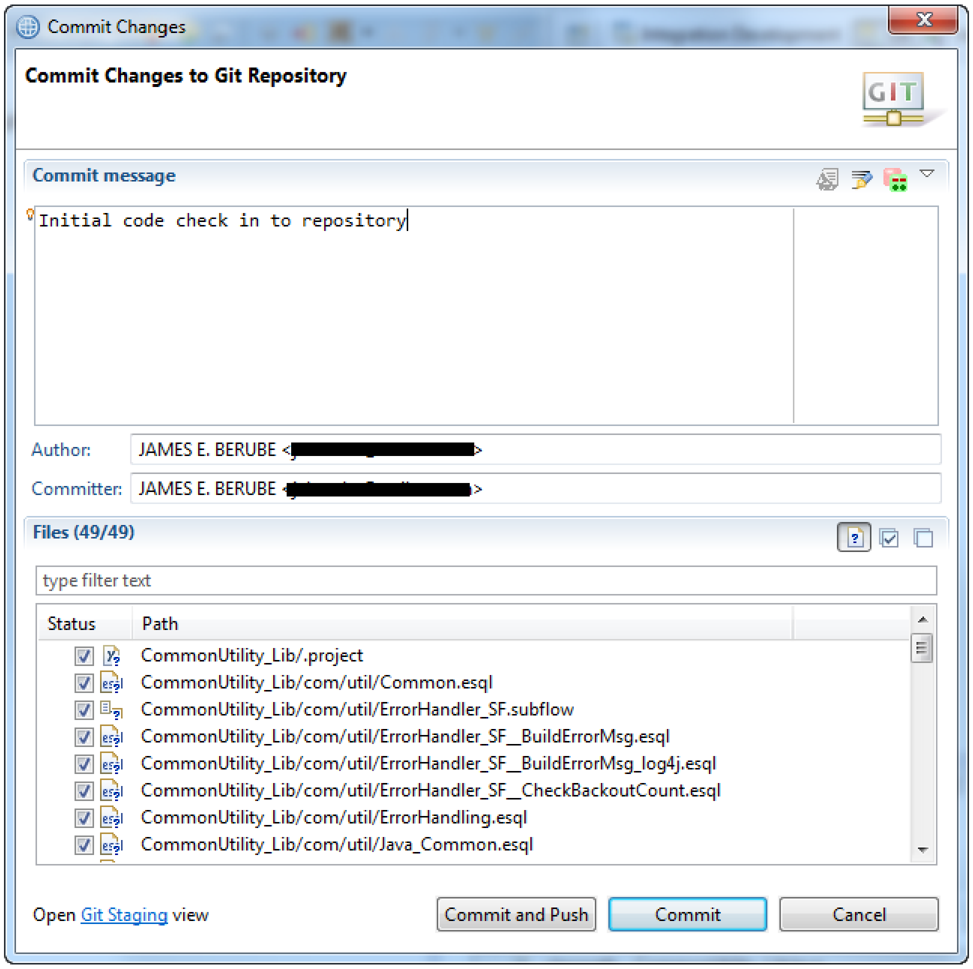
The code still needs to be Committed and Pushed to get it to the actual GitHub repository as all work is always done locally first. Using a Git repository, you're operating in a disconnected mode until you request a connection to the remote repository to push your changes or when pulling or fetching changes from the remote GitHub repository. The Commit is to your local repository, where the push is sending your changes up to the GitHub remote repository.

8. There are multiple ways to accomplish this. Either by clicking the "Commit" icon on the Git toolbar, right clicking on the repository and selecting "Commit" from the dropdown menu, or going to the "Git Staging"

panel in the lower right hand portion of the toolkit and pressing the "Commit" or "Commit and Push" buttons. We'll be doing the commit based on right clicking the repository. From the Git perspective, right click on your repository name and select "Commit".


Git Repositories
> IIB-Demo-Tutorial [NO-HEAD] - C:\IBM\git\IIB-Demo-Tutorial\.git
Branches

9. This will open up a new window, where you will select the files to be committed and add a check in comment. The checkmark button in the lower half of the window on the right will allow you to select all files. If there are any files that should not be checked in (like java class files or anything that's generated by the toolkit), make sure to deselect them.



```
Initial code check in to repository
```

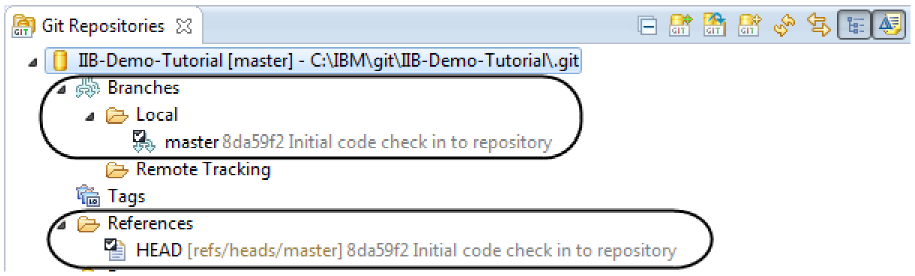
Author: JAMES E. BERUBE <[redacted]>

Committer: JAMES E. BERUBE <[redacted]>

Files (49/49)

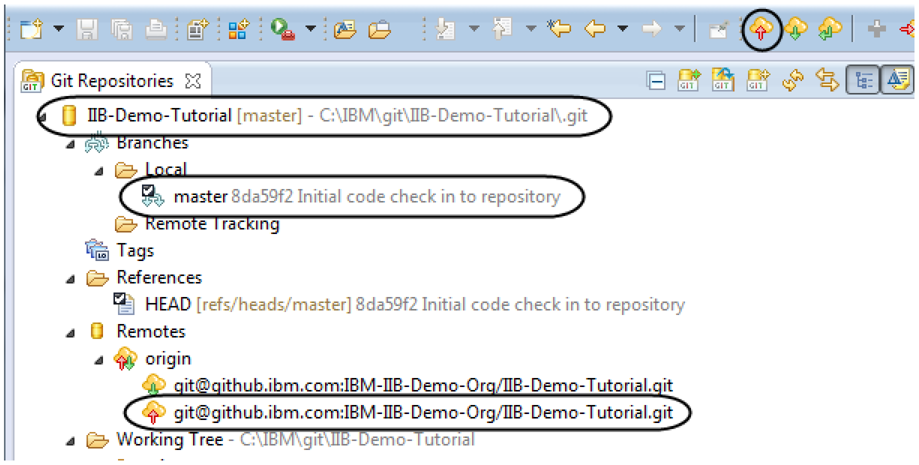
Status	Path
✓	CommonUtility_Lib/.project
✓	CommonUtility_Lib/com/util/Common.esql
✓	CommonUtility_Lib/com/util/ErrorHandler_SF.subflow
✓	CommonUtility_Lib/com/util/ErrorHandler_SF__BuildErrorMsg.esql
✓	CommonUtility_Lib/com/util/ErrorHandler_SF__BuildErrorMsg_log4j.esql
✓	CommonUtility_Lib/com/util/ErrorHandler_SF__CheckBackoutCount.esql
✓	CommonUtility_Lib/com/util/ErrorHandling.esql
✓	CommonUtility_Lib/com/util/Java_Common.esql

10. Once you've added your comment and selected the files, click the "Commit" button.
11. Now that the code has been checked in, you'll see that 2 new entries have been added to the Git Repositories view, under Branches –> Local and References.

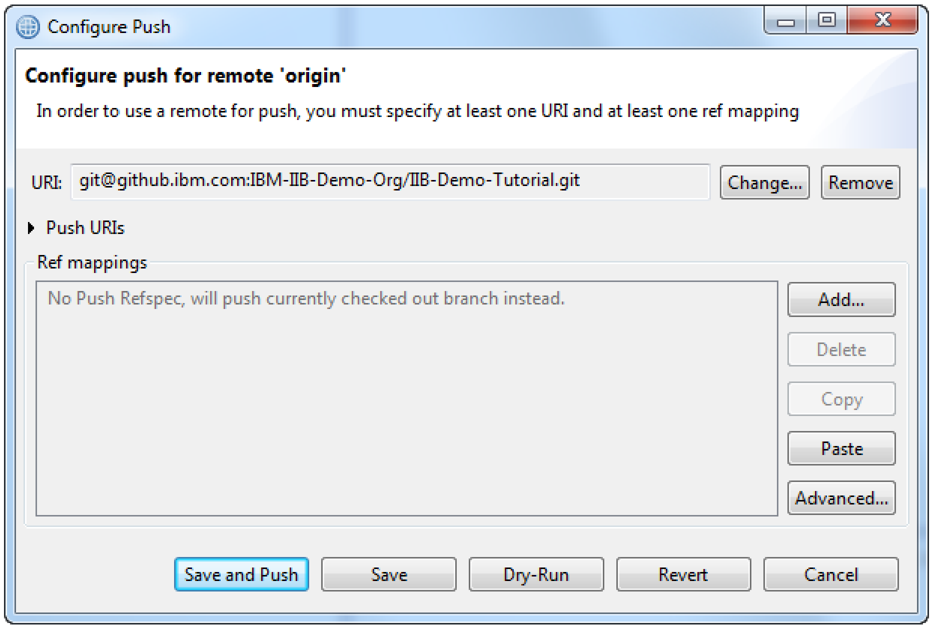

Git Repositories
IIB-Demo-Tutorial [master] - C:\IBM\git\IIB-Demo-Tutorial\.git
Branches
Local
master 8da59f2 Initial code check in to repository
Remote Tracking
Tags
References
HEAD [refs/heads/master] 8da59f2 Initial code check in to repository

12. This has committed the changes from your workspace to your local copy of the repository. In order to get code from your local repository up to the GitHub repository, the code still needs to be "pushed". To push

code to GitHub, there are multiple ways to do that as well. By right clicking on the Repository name and selecting "Push Branch to master", right clicking on the Local Branch and selecting "Push Branch", clicking the toolbar button for push, or by right clicking on the red arrowed line to push under Remotes –> origin and selecting "Push".


Git Repositories
IIB-Demo-Tutorial [master] - C:\IBM\git\IIB-Demo-Tutorial\.git
Branches
Local
master 8da59f2 Initial code check in to repository
Remote Tracking
Tags
References
HEAD [refs/heads/master] 8da59f2 Initial code check in to repository
Remotes
origin
git@github.ibm.com:IBM-IIB-Demo-Org/IIB-Demo-Tutorial.git
git@github.ibm.com:IBM-IIB-Demo-Org/IIB-Demo-Tutorial.git
Working Tree - C:\IBM\git\IIB-Demo-Tutorial

13. Since this is the first time we're pushing code to GitHub, we'll want to configure the Push first. So, let's start by right clicking on the "red arrow" line under Remotes –> origin and selecting "Configure Push". This will bring up a new window.


Configure Push

Configure push for remote 'origin'

In order to use a remote for push, you must specify at least one URI and at least one ref mapping

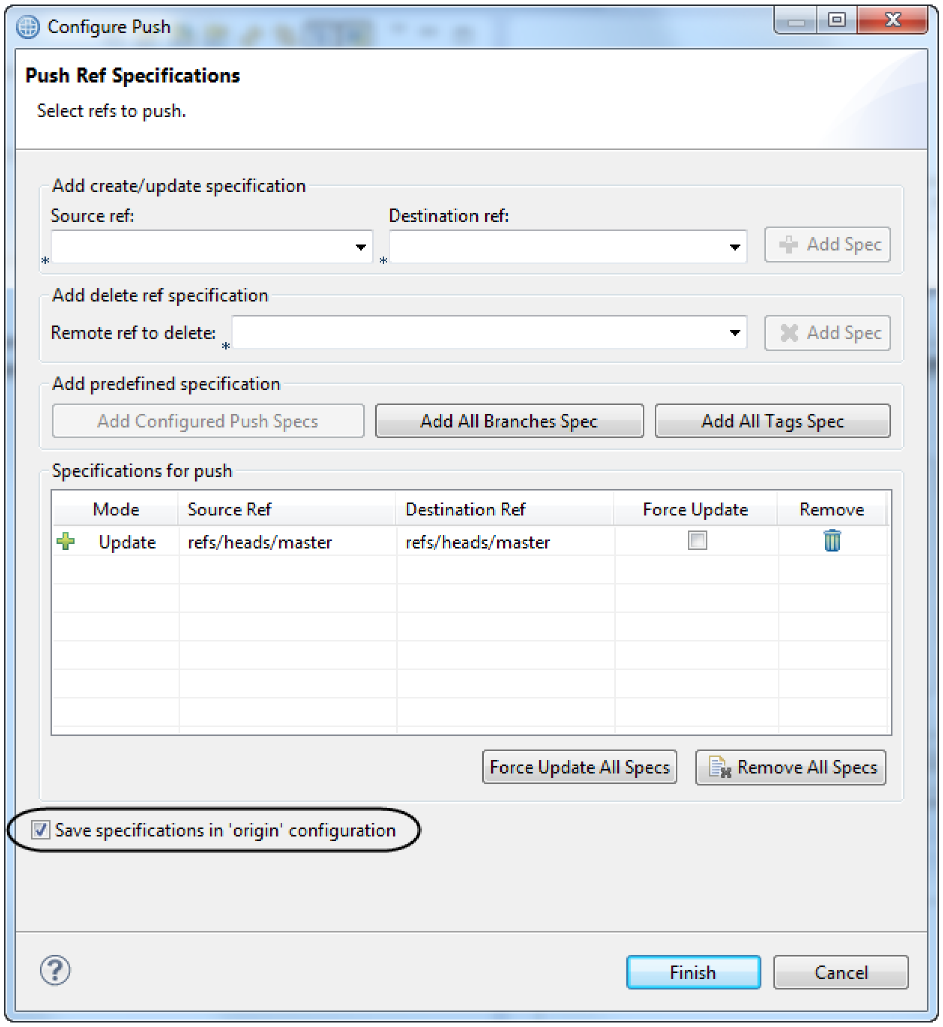
URI: git@github.ibm.com:IBM-IIB-Demo-Org/IIB-Demo-Tutorial.git

Push URIs

Ref mappings

No Push Refspec, will push currently checked out branch instead.

14. As you can see, no "Ref mappings" have been configured yet. You can click "Add" and manually enter all the information required or click on the Advanced button and use the wizard. We're going to click Advanced. (If prompted, enter your ssh password created earlier.)
15. In the Push Ref Specifications, select the Source ref as your master. the Destination should be auto populated. Then click the Add Spec button. The spec will be added under "Specifications for push". Also check the "Save specifications in 'origin' configuration" checkbox before clicking Finish.


Configure Push

Push Ref Specifications

Select refs to push.

Add create/update specification

Source ref: | Destination ref:

Add delete ref specification

Remote ref to delete:

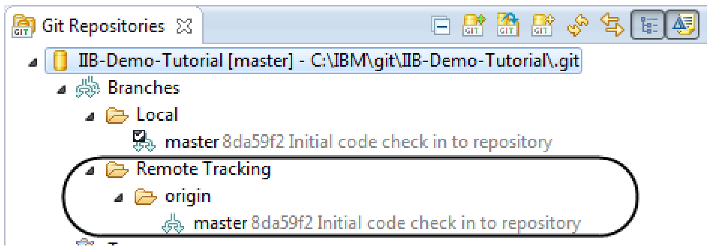
Add predefined specification

Specifications for push

Mode	Source Ref	Destination Ref	Force Update	Remove
Update	refs/heads/master	refs/heads/master	☐	🗑

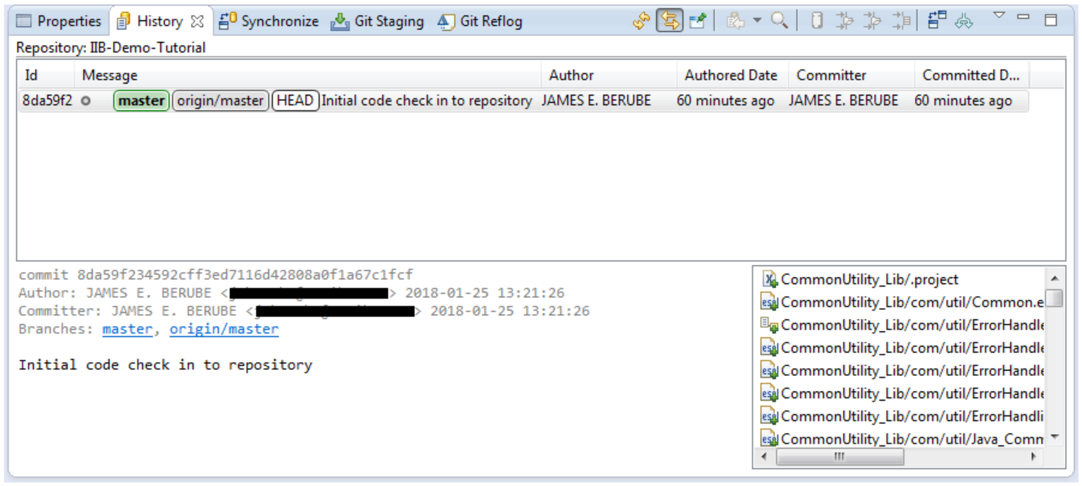
[✓] Save specifications in 'origin' configuration

16. Now, from the "Configure Push" window, click the "Save" button to save this configuration. (Save and Push is an option, and would be a shortcut, I want to show them individually here.)
17. Now that the configuration has been saved, right click on the same line under Remotes –> origin and select "Push".
18. A dialog will appear giving the status of the code being pushed to the repository and a Push Results window will appear when complete. (If there are any merge conflicts, you would see them here. There shouldn't be any now, since this is our initial code check in.) Click "OK" to exit. You should now see something created under Branches –> Remote Tracking


```
Git Repositories
IIB-Demo-Tutorial [master] - C:\IBM\git\IIB-Demo-Tutorial\.git
  Branches
    Local
      master 8da59f2 Initial code check in to repository
    Remote Tracking
      origin
        master 8da59f2 Initial code check in to repository
```

Notice that the hash values for each of these are all the same. In the case shown, `8da59f2`.

19. Right click on your code, select Show In, then select History. In the History view, you can see the history of your check-ins as well as the id associated with the branch.



Repository: IIB-Demo-Tutorial

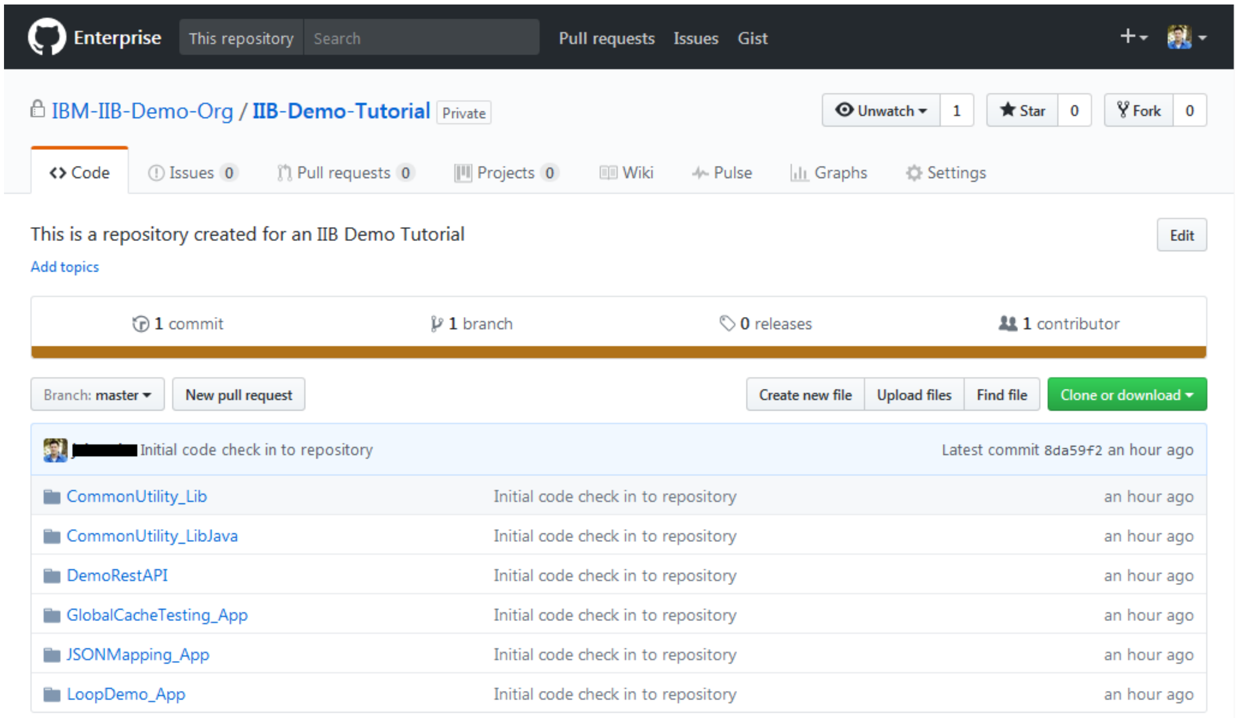
Id	Message	Author	Authored Date	Committer	Committed D...
8da59f2	master origin/master HEAD Initial code check in to repository	JAMES E. BERUBE	60 minutes ago	JAMES E. BERUBE	60 minutes ago

```
commit 8da59f234592cff3ed7116d42808a0f1a67c1fcf
Author: JAMES E. BERUBE <[redacted]> 2018-01-25 13:21:26
Committer: JAMES E. BERUBE <[redacted]> 2018-01-25 13:21:26
Branches: master, origin/master

Initial code check in to repository
```

CommonUtility_Lib/.project
CommonUtility_Lib/com/util/Common.e
CommonUtility_Lib/com/util/ErrorHandle
CommonUtility_Lib/com/util/ErrorHandle
CommonUtility_Lib/com/util/ErrorHandle
CommonUtility_Lib/com/util/ErrorHandle
CommonUtility_Lib/com/util/ErrorHandli
CommonUtility_Lib/com/util/Java_Comn

20. Now, looking at GitHub, you can also see that the code has been successfully pushed to the repository.



IBM-IIB-Demo-Org / IIB-Demo-Tutorial Private

This is a repository created for an IIB Demo Tutorial

Add topics

1 commit · 1 branch · 0 releases · 1 contributor

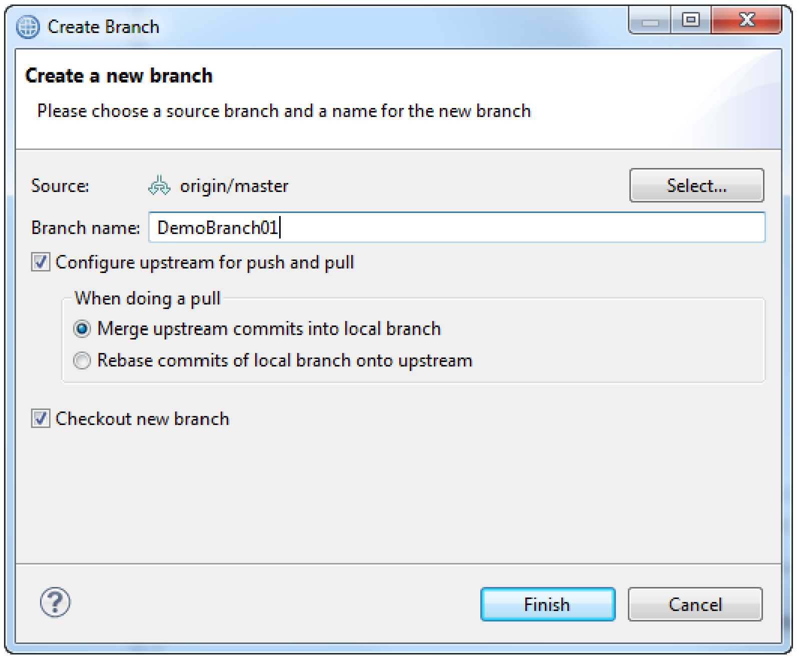
[redacted] Initial code check in to repository — Latest commit 8da59f2 an hour ago

CommonUtility_Lib	Initial code check in to repository	an hour ago
CommonUtility_LibJava	Initial code check in to repository	an hour ago
DemoRestAPI	Initial code check in to repository	an hour ago
GlobalCacheTesting_App	Initial code check in to repository	an hour ago
JSONMapping_App	Initial code check in to repository	an hour ago
LoopDemo_App	Initial code check in to repository	an hour ago

Creating Branches

Branches are the mechanism used to check-out the codebase locally from the remote GitHub repository. This does not impact the remote code in any way. Branches can be created for future work, experimental code, bug fixes, etc. The Branch can either be merged back into the origin/master later or discarded. Branches are not made for individual files or projects, but everything in the repository. Branches are very lightweight in GitHub because GitHub is not creating copies of all the files to modify, it's a snapshot of what the full repository looked like at the time the Branch was created.

1. To create a Branch, return to the IIB Eclipse Toolkit, right click on the "master" branch under Remote Tracking –> origin and select "Create Branch …". Give the new Branch a name, We'll call ours "DemoBranch01". It's good to use something short but descriptive and meaningful here. The first checkbox indicates that you intend for the code to be available for merge later. (Rebasing won't be covered here.) The second checkbox will create the Branch in your local repository, if that's not checked, then you'll have to fetch and add the code to your workspace separately. It's a good idea to make sure this is checked. Click "Finish".


Create Branch

Create a new branch

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

Source: origin/master

Branch name: DemoBranch01

[✓] Configure upstream for push and pull

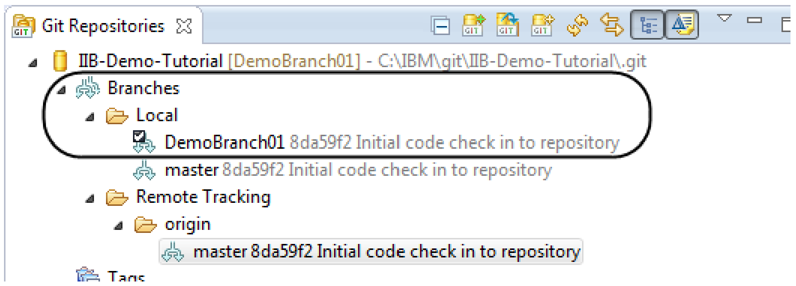
When doing a pull

(●) Merge upstream commits into local branch

( ) Rebase commits of local branch onto upstream

[✓] Checkout new branch

2. Once the new Branch has been created and checked out, you'll see a new Local Branch has been created under Branches –> Local


Git Repositories

```
IIB-Demo-Tutorial [DemoBranch01] - C:\IBM\git\IIB-Demo-Tutorial\.git
  Branches
    Local
      DemoBranch01 8da59f2 Initial code check in to repository
      master 8da59f2 Initial code check in to repository
    Remote Tracking
      origin
        master 8da59f2 Initial code check in to repository
  Tags
```

The hash numbers are still the same because the snapshots of both branches are still identical and will remain so, until the new Branch gets committed.

The Checkmark on the Branch indicates which Branch is active in your toolkit. By double clicking on different Branches, that branch will become the active Branch for that code in your workspace. You can see this when you look at your workspace in the Integration Development perspective.

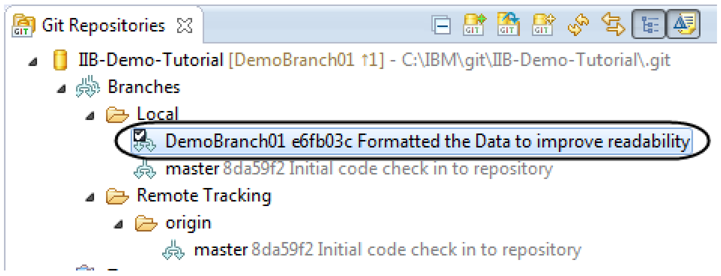
It is possible to have code and projects from multiple different repositories in your workspace at the same time.

When working with a Git repository you're working with a local disconnected copy of the code, it's a good idea to get in the habit of doing a fetch or pull from the remote repository when multiple developers are working in the same repository. This can help to resolve any potential conflicts sooner rather than later. A fetch will update your toolkit with what's on the remote server for the current Branch (or Branch in focus), but it does not actually change that local Branch. A pull will do a fetch first, but will also update your local Branch. So, if "master" is in focus had been updated on the Remote, then a fetch will just update your toolkit to reflect that the Remote "master" Branch has been changed, where a pull will do that but also update your local repository "master" Branch.

Code Merges

A Branch has been created and changes were made and committed locally. Now those changes need to be merged back into the origin/master code on the GitHub remote repository.

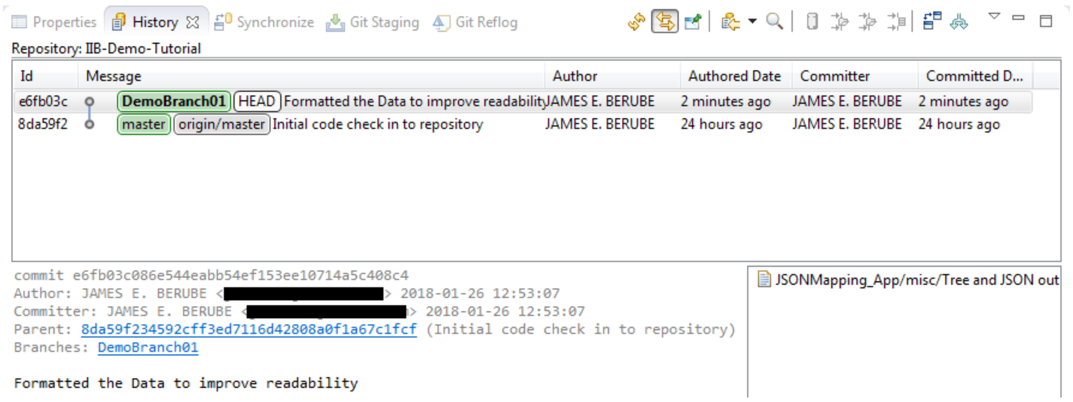
As you can see in the image below, the hash number has changed on my Branch now that it's been committed. Also, the check in comment is displayed as well.



Git Repositories

```
IIB-Demo-Tutorial [DemoBranch01 ↑1] - C:\IBM\git\IIB-Demo-Tutorial\.git
  Branches
    Local
      DemoBranch01 e6fb03c Formatted the Data to improve readability
      master 8da59f2 Initial code check in to repository
    Remote Tracking
      origin
        master 8da59f2 Initial code check in to repository
```

Once the change has been committed, it will also be visible in the History view. When multiple Branches have been created and/or merged, the History view helps to show where different Branches started and were later merged back into the master again, along with comments, dates and who was responsible for the work.



Repository: IIB-Demo-Tutorial

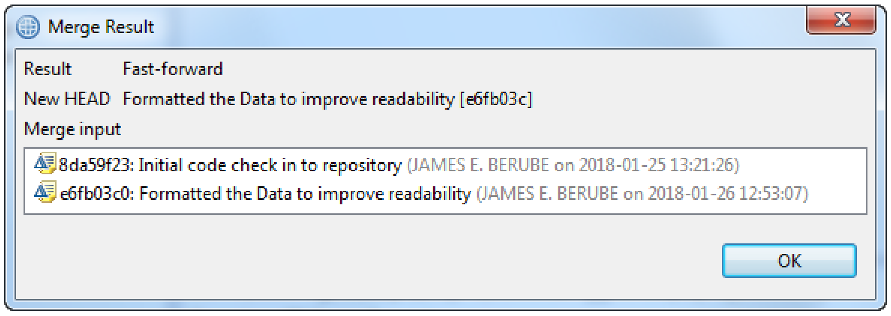
Id	Message	Author	Authored Date	Committer	Committed D...
e6fb03c	DemoBranch01 HEAD Formatted the Data to improve readability	JAMES E. BERUBE	2 minutes ago	JAMES E. BERUBE	2 minutes ago
8da59f2	master origin/master Initial code check in to repository	JAMES E. BERUBE	24 hours ago	JAMES E. BERUBE	24 hours ago

```
commit e6fb03c086e544eabb54ef153ee10714a5c408c4
Author: JAMES E. BERUBE <[redacted]> 2018-01-26 12:53:07
Committer: JAMES E. BERUBE <[redacted]> 2018-01-26 12:53:07
Parent: 8da59f234592cff3ed7116d42808a0f1a67c1fcf (Initial code check in to repository)
Branches: DemoBranch01

Formatted the Data to improve readability
```

JSONMapping_App/misc/Tree and JSON out

1. The next step is to Merge the Branch you've been working on with your local "master" Branch. To do this, double click the local "master" branch, so the checkbox moves to that Branch. Then, right click on the Branch you've been working on and select Merge. When complete, you'll see a Merge Result window.



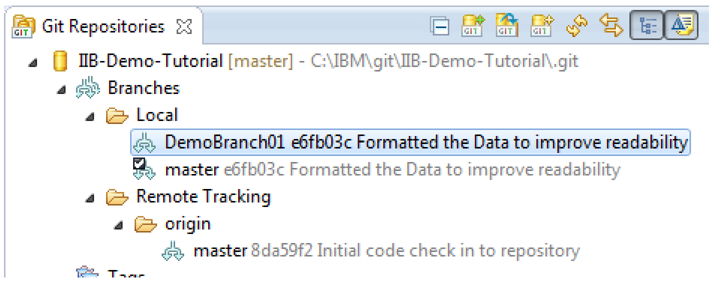
Merge Result

Result	Fast-forward
New HEAD	Formatted the Data to improve readability [e6fb03c]

Merge input

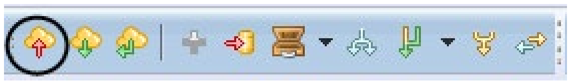
8da59f23: Initial code check in to repository (JAMES E. BERUBE on 2018-01-25 13:21:26)
e6fb03c0: Formatted the Data to improve readability (JAMES E. BERUBE on 2018-01-26 12:53:07)

This indicates that the initial (in this case "master") branch is able to "Fast-forward" to the latest Branch without any conflicts. If there were conflicts, that would be indicated here and would need to be resolved before moving forward. You'll also see that the hash number for the local "master" Branch has now also been updated so it's the same as the Branch that was merged in. However, because the changes have not yet been pushed to GitHub, the remote hash id is still the old id.

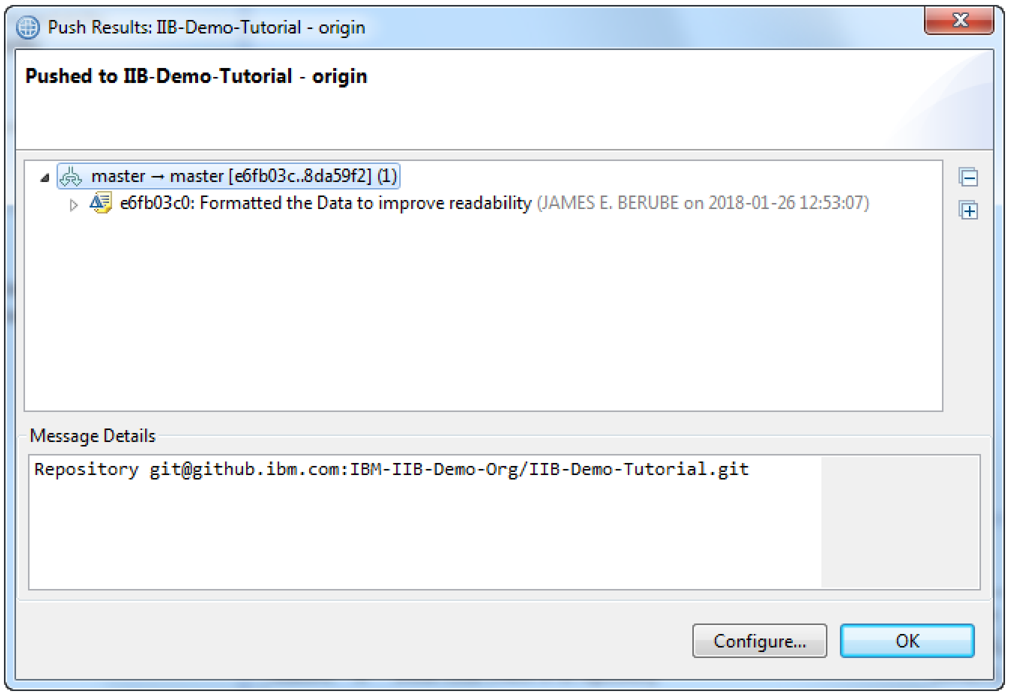

Git Repositories
IIB-Demo-Tutorial [master] - C:\IBM\git\IIB-Demo-Tutorial\.git
Branches
Local
DemoBranch01 e6fb03c Formatted the Data to improve readability
master e6fb03c Formatted the Data to improve readability
Remote Tracking
origin
master 8da59f2 Initial code check in to repository

The next step is to Push the local "master" to the Remote "master". Push is always "Fast-forward", so conflicts must be resolved locally before being able to push the code to GitHub. (This is the reason for performing a fetch or pull prior to attempting to push your code to GitHub.)

2. With the Local "master" Branch in focus (double click it so it has the check mark). Next, click on the "Push" button on the toolbar, or right click on the Branch and select "Push Branch".



3. The "Push Results" window will be displayed. If there is a conflict, the push will be "rejected" and a conflict resolution will need to be performed before being able to successfully push the code again. Otherwise, the operation was a success and your changes should now be reflected with an updated hash number in the remote "master" Branch.

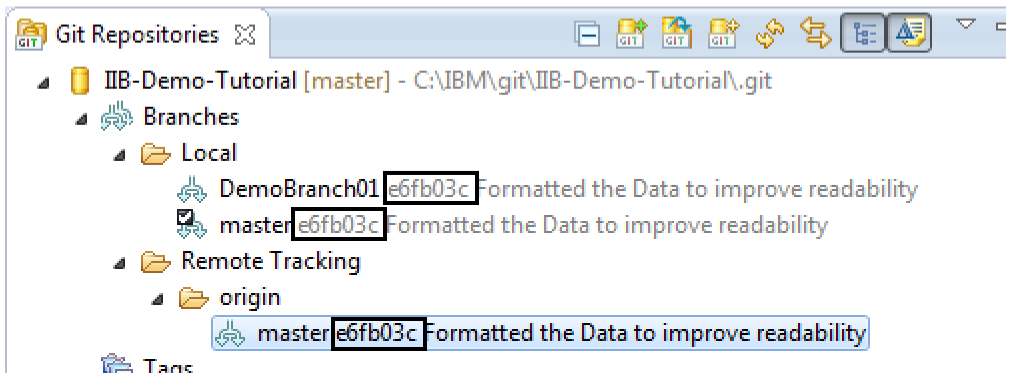

Push Results: IIB-Demo-Tutorial - origin

Pushed to IIB-Demo-Tutorial - origin

master → master [e6fb03c..8da59f2] (1)
e6fb03c0: Formatted the Data to improve readability (JAMES E. BERUBE on 2018-01-26 12:53:07)

Message Details
```
Repository git@github.ibm.com:IBM-IIB-Demo-Org/IIB-Demo-Tutorial.git
```

Looking at all three branches and their hash ids, you can see now that all three hash ids match. This indicates that all three Branches are now pointing to the same snapshot.


Git Repositories

IIB-Demo-Tutorial [master] - C:\IBM\git\IIB-Demo-Tutorial\.git
Branches
Local
DemoBranch01 e6fb03c Formatted the Data to improve readability
master e6fb03c Formatted the Data to improve readability
Remote Tracking
origin
master e6fb03c Formatted the Data to improve readability
Tags

Wrap up

In this tutorial, we covered the basics of how to set up and configure a GitHub repository for use with the IIB Eclipse Toolkit. Additional concepts included code check-in, creating, and merging branches.

About the authors;

PaulFaulkner

Paul Faulkner is a Certified IT Specialist with extensive experience designing and implementing system integration solutions. With 25+ years industry experience Paul has designed and implemented many complex integration patterns and specializes in solutions for IBM Integration Bus (IIB).

25 comments on"Using GitHub as an IBM Integration Bus Code Repository"

1. Rajdeep September 30, 2020

   Hi Team ,
   I am trying to install egit plugin on ACE toolkit version 5.0 as mentioned in IBM site
   ACE Version :
   IBM App Connect Enterprise Toolkit
   Version: 11.0.0.10
   Build id: 11.0.0.10-20200826-0227

   when i add this URL : it provides the options to install , i selected all of them
   but when it is checking for requirement it says : The intsallation cannot be completed as requested
   and in below options:
   will be upgraded and downgraded :
   Git Integration for Eclipse : Source code

   Will be installed :
   Java Implementation of Git : Command line interface

   and when i say Next it will show the installable package as below :

   Java Implementation of Git : Command line interface

Git Integration for Eclipse : Source code

But when i accept the License and say install :
getting below error :
“An internal error occured during : “Install download3”
An internal error occurred during: “Install download3”.
java.lang.NullPointerException

Do i need to add Fixpack for ACE ? Let me know how i can resolve it?

- JimBerube September 30, 2020

  Hi Rajdeep,

  It looks like something has changed again between the Eclipse toolkit and the eGit plugin.
  When installing the eGit plugin, in your Eclipse Toolkit, try using version 5.8.1.
  https://download.eclipse.org/egit/updates-5.8.1
  I found that I got the same error you reported when I tried using version 5.0 or the latest version available in the Marketplace, 5.9. (5.9 is not supported because the Eclipse core runtime is not the correct version.)

  I was able to install version 5.8.1 successfully. I only installed the eGit plugin, basically everything included in the first choice. I did not select the other 2 options.

2. Gen_Maximus June 14, 2020

   I did the following to make it work with ACE.
   1. used egit5.0 because ACE is based on Luna.
   https://download.eclipse.org/egit/updates-5.0/
   2. used following command to get RSA private key.
   ssh-keygen -m PEM -t rsa -b 4096 -C “your_email@example.com”

   - JimBerube June 15, 2020

     Thank for you posting an update and your resolution. I hadn’t had time to look into this myself yet. I will hopefully get that information added into the article this week to help others.

3. Gen_Maximus June 08, 2020

   Hi Jim, I am using 4.4 from ACE. I checked Error Log. It looks strange. I am connecting to our repo, but it ACE tries to connect to the following first.

   com.ibm.etools.mft.pattern.community

   Error

   Connection to the URL https://ot4i.github.io/ace-patterns/no/repo_metadata.json failed. Response code: 404
4. Gen_Maximus June 07, 2020

   Hi,

   I tried this with ACE, but it did not work. To Clone a repo gives:

   Incorrect URL

No network connection.
However, it works well with IIB toolkit 10.0.0.14 and 10.0.0.16
I wonder, if ACE is supported and which Git Release is suitbale for ACE.

- JimBerube June 07, 2020

  I would think that ACE should also be supported, but I have not tried this with ACE yet myself.
  The eGit plugin should be version 4.4, which is the same version used for IIB version 10.0.0.16 and above.
  The Eclipse Platform version on ACE FixPack 8 is 4.4.2.v20150204-1700, which is the same as what I see in IIB version 10.0.0.20.

5. jack li May 11, 2020

   Do you guys have a good gitignore file for IIB projects to use to ignore those files we don't need to commit ? I tried to a few like this one: https://github.com/quantiguous/iib_samples/blob/master/.gitignore, but broken when tried to switch branches.
   Thanks

   - JimBerube May 11, 2020

     Hi Jack,

     I do not have one that I can share at this time. This is a topic I need to explore in more depth.
     In the meantime, this page is a good reference and may help with whatever issues you're experiencing:
     https://git-scm.com/docs/gitignore

6. Pratish P May 09, 2020

   Hi Jim,
   Thanks for the update
   For IIB version 10.0.0.16 and above, the Toolkit's Eclipse version was updated to 4.4.2.
   So, the Git Plugin needs to be version 4.4 for IIB version 10.0.0.16 and above.
   I was attempting to follow the instructions however since I am on IIB 10.0.0.20 ,
   I failed for me
   ….
   14. Go to Window –> Open Perspective —> Other
   15. Then select "Git" from the list. >>>>>>>>>>>>>>>>>>>>>> The "GIT" prospective is not listed"
   However by reinstalling the plugin version 4.4 I was successfully able get the GIT prospective

   Thanks,
   Pratish

   - JimBerube May 09, 2020

     Hello Pratish,

     You're very welcome.
     Thank you for bringing that to my attention, so I could make the appropriate updates.
     Glad that fixed your issue.

7. Vinayaka July 25, 2019

   Team

   I am facing issue at “Connecting to the Repository” step 5 when connecting to repositories “Click “Next”. If you get a popup about the site not being trusted yet, click “Yes” to accept and “Yes” again to create a known hosts file. Enter the passphrase for the SSH key you created earlier and click “OK”.”

   Problem:

   When i click on next , i didnt get any pop up – it is showing ” Couldn’t create temporary repository.”

   Can you help me in resolving the issue

   - PaulFaulkner July 31, 2019

     It appears you may have an issue with permissions. I would suggest you reach out to your Git administrators and ensure the access is setup correct. Unfortunately we are unable to help with individual access issues via this forum but I did Google you error and found several links to potential permission problems. Hope that helps.

8. Robert Tilley March 07, 2019

   We are using MS Git repo with IIB V10 FP 14 and it doesn’t want us to update the master directly and is asking for a Pull request. When I try the Pull option it says nothing has changed. How can I get this code Pushed upstream if it wont let me directly update it?

   - JimBerube March 09, 2019

     Hello Robert, sorry for the delay, I’ve been on vacation this week.

     I’m not sure about MS Git repo.
     This might be something better directed to someone who is an authority on that product.

     I’m not sure I fully understand how you’ve got your environment setup.
     In GitHub, there’s the Remote Master and a Local Master, but you’re typically working in a local Branch, but you’re not working directly on the Remote Master.

9. kranthi February 25, 2019

   iam trying to clone repository from cloud to local but iam getting a error

   possible reasons

   incorrect url
   no network connection
   ssl host could not be verified

   - JimBerube February 27, 2019

Hello Kranthi, sorry for the delay in responding.

I was speaking with Paul about this and this sounds similar to an issue someone else encountered recently.
It may have something to do with the SSH key.
This is what worked for the other person, give it a shot and let us know if it also worked for you.

When you're logged into your GitHub page, where your profile drop down is, there is a "Settings" link at the bottom.
Click on that.
On the page that loads, you should see a button at the bottom of the "menu" list for "Developer Settings"
Click on that.
In the next page that opens, there are 3 tabs on the left, click the one for "Personal access tokens"
Now you should see a button to "Generate new token"
Generate a token with all available options and use that instead of the ssh token that was generated via the command line options.

- Karen Broughton-Mabbitt February 28, 2019

  (Hi Kranthi – I've moved your reply to follow the conversation with Jim)

  Hi JimBerube,
  Thanks for the your suggestion i resolved issue .after creating repository in git hub iam trying to clone the repository into my local pc ..the mistake that i commited is while coping the repository path iam using https instead of ssh ..i switched to ssh in git hub and in tool kit we have use protocol ssh .
  Kranthi

10. Sharma October 12, 2018

    All installation went well, however when trying to connect/clone the GIT repository in IIBs GIT Perspective i couldn't, i am using Enterprise GIT HUB and there is a firewall and all the corporate security, will this security affect connection? i tried SSH, HTTPS however i keep getting invalid URL and Proxy errors. However when using the GIT GUI i could clone the repository successfully. What additional measures should i take to be able to clone in IIB.

    - JimBerube October 12, 2018

      I would try to talk to someone at your company about the corporate security, firewall and/or proxy. It does sound to me like something is blocking your connection. It's interesting that you say the Git GUI isn't being blocked though. I would hope someone on that team might be able to help you find the error and set something up to allow the connection.

11. Eleazar Ponce August 03, 2018

    Hi Jim,

    Which version of IIB toolkit did you have?
    I'm trying with v10.0.0.7 but It fails when I install the eGit plugin.
    I have toolkit on a Linux machine

    - JimBerube August 09, 2018

Hello Eleazar,

I followed these same instructions with multiple fixpack releases of the toolkit for IIB v10, but they were all on Windows.
I haven't attempted to install on a Linux based toolkit.

- Eleazar Ponce August 09, 2018

  Issue fixed. I had to remove all the plugins related to Git on: Help -> About IBM Integration Toolkit -> Installation Details -> Installed Software. After removed, I was able to install the eGit plugin 4.2

  - Karen L October 08, 2018

    I am getting dependency errors when install http://download.eclipse.org/egit/updates-4.2/ to IIB 10.0.0.7. What have you removed from "Installed Software" to make the installation successfully?

  - Karen October 08, 2018

    what need to be removed from installed software?