Getting Going with Git

GitHub Private Repositories for CS314 and Using them with Eclipse

Original document prepared by Geri Georg, Colorado State University, January, 2014 for CS314, Spring 2014. Adapted for Fall 2014 by Sudipto Ghosh

Introduction

Git is a source control system that is frequently used for collaborative development, and GitHub.com is one of several hosting sites for both public and private Git repositories that can be accessed over the internet. CS314 was able to get a free private repository area for student teams to use this semester.

Git is available in multiple forms. There is an Eclipse plug-in for it, and there are multiple GUI and command line versions. The command line version is installed on the CS computers, and the Eclipse plugin is included in the general Kepler Eclipse release.

The download site for the standalone versions is: http://git-scm.com/downloads

The download site for the Eclipse plugin is: http://download.eclipse.org/egit/updates (Note: **EGit is INCLUDED In the Keplar Eclipse release!** If you need to install it, you should do this from Eclipse and the Help->Install new Software menu. You can tell if the plugin is installed by going to Help->About Eclipse, and looking at the icons: EGit looks something like a cup with a spout and two sets of '+' and '-' floating above it – when you mouse-over it will say Eclipse EGit.)

The main Git site has a good reference on-line book at: http://git-scm.com/doc

The GitHub site has good help and tutorials too: https://help.github.com/, and their 'bootcamp' at: https://help.github.com/categories/54/articles

Regarding setting up Eclipse and Git, Lars Vogel has a tutorial on this and other general Git tutorials. The Eclipse one can be found at: http://www.vogella.com/tutorials/EclipseGit/article.html

Finally, the GitHub folks have a list of Git commands that you can use on command lines at: http://cheat.errtheblog.com/s/git

This document is the result of the adventures of making a simple remote Java project repository work on a Windows system and documents many of the issues and answers I encountered. It is not intended to be a complete reference, but rather an introductory tutorial to a tool useful for collaborative work.

1. Creating a Repository in the CS314 GitHub Area

We created an organization, CSU-CS314-FA2014, on GitHub. To be added to a team in this organization, you must first create an account for yourself at https://github.com. Do not create any repository – these have to be created in the CS314 organization in order to be private. You then need to send your GitHub user name to cs314@cs.colostate.edu. You will then be added to your appropriate team in the organization, and have read/write access to your team's repository. Your personal GitHub account user name and password will be needed to interact with the GitHub repository to clone it, fetch/pull information from it, and push to it.

Repositories have to be created by the organization admins (Dr. Ghosh, Amila, and Upulee), who also have to set the team as collaborators for the repository. A team can have multiple repositories as needed; you just need to request one of the owners to create it for you and make your team the collaborators for it.

2. Creating a Local Copy of the GitHub Repository

Once you have a GitHub repository, you need to make a local copy of it on the computer you will use. This repository will be linked with your Eclipse project. To do this, you need to have access to Git on your local machine. For Windows, it is easy to download the latest Windows version of Git, which gives you a Bash terminal window that you can use to enter command lines. If you are using a department machine, command line Git is already installed.

You need to create a directory/folder where you want the repository to be placed, and navigate to that location. For example, in the Bash terminal:

```
cd
mkdir gitRep
cd gitRepo
```

On the GitHub repository page, you will see a section that is labeled **HTTPS** clone URL and below it a field that can be copied that says something like:

https://github.com/CSU-CS314-FA2014/<repoName>.git

where <repoName> is the name of the repository. Copy this link, and use these commands to clone the repository to your local system:

```
git clone  https://github.com/CSU-CS314-FA2014/<repoName>.git
```

You will have to supply your GitHub user name and password for this command. Once it completes, if you change directories to `<repoName>`, you will see any files that are in the initialized repository. We will add only one upon creation, a README file that describes the repository.

3. Adding the Repository to Eclipse

In Eclipse, open Window-> Show View-> Other -> Git Repositories. You should see several links, including 'Add an existing local Git repository'. Choose this, and click on the 'Browse' button to the right of the 'Directory:' field. Navigate to the directory, e.g. `gitRepo/<repoName>` and click OK, then Finish.

4. Create a Java Project

Next create a simple Java project, including packages, classes, etc.

5. Adding the Java Project to the Repository and Pushing it to GitHub

i Right click on the project name, go to Team-> Share Project-> Git, and press the Next button.

ii A window will come up titled 'Configure Git Repository'. There is a checkbox below this 'Use or create repository in parent folder of project', and field labeled 'Repository:' just below this. The field is probably shaded, with nothing in it but a down arrow on the right side. Click in this field, and the repository you just added should be a choice. Click on it.

iii Press the 'Finish' button at the bottom of the window. You should notice that the project now has information about the repository in square brackets, e.g. *GitTest1 [InstrRepo master]*

You now need to show the Git Staging view, which will show the repository name, and areas labeled 'Unstaged Changes', 'Staged Changes', 'Commit Message', your name as 'Author:' and 'Committer:', and two buttons labeled 'Commit and Push', and 'Commit'.

At this point, as far as git is concerned, you have created a bunch of files that need to be added to the local repository, but you haven't done anything beyond creating them. Thus, they should be listed in the 'Unstaged Changes' area.

i Select all of the unstaged file names, and drag them to the 'Staged Changes' area.
ii Next, add a commit message, e.g. 'first version of java project files'.
iii You can then click either button. If you click 'Commit and Push', another window will come up with a title, e.g. 'Push Results: InstrRepo – origin'. It will also say 'Pushed to InstrRepo – origin', and list the files to be transferred to the remote. Click the OK button to make the push.

Now if you make a change to a class file and save it, Eclipse will ask what file, and when you choose the original name (default) it will ask if you want to replace the existing file. Select 'Yes'. Again you will see this in 'Unstaged Changes', and you can commit the changes and push to the remote as before.

6. Committing Files to the Local Repository and Manually Pushing to GitHub

If you have problems with the 'Commit and Push' process from Eclipse, you can still commit the changes to your local copy of the repository and then push them to the remote using the command line interface. For example, from the repository directory, e.g. `gitRepo/<repoName>`, you can use this command:

```
git push origin master
```

to send the changes to the remote GitHub repository.

7. Verifying changes on GitHub

You can navigate through the repository on GitHub to the actual Java source file you added and changed if you did that. Double click on the source file and it will display the contents.

8. Recommended Collaboration Work

Scott Chacon has a good description of how they use Git at GitHub:

http://scottchacon.com/2011/08/31/github-flow.html - read the part that describes the work flow they actually use, starting at the section titled 'How We Do It'. He recommends each team member work on their own branch, pushing frequently to the remote repository, for backup and so that others can see what they are doing. Only when changes are working together properly do they go onto the master branch.

Branching is fairly easy to do; you make a branch on the GitHub site, and this branch will contain everything on the master at that time. You need to copy the branch down to your local site to see and work on it. You can do this with the git fetch command. For example, if you create a branch called 'myBr' in your repository on GitHub. The following commands will get a copy of the branch in your local repository:

```
cd
cd gitRepo/<repoName>
git fetch origin myBr
```

Now in Eclipse you need to change to that branch and make your changes there. To do this, right click on the project and go to Team-> Switch To-> new branch. A new window will come that is titled 'Create Branch'. There is a field with the label 'Source ref:' that may have something like 'refs/heads/master' in it. Click on the down arrow on the right side of this field, and choose the one that says 'refs/remote/origin/myBr'. When you do this, the 'Branch name:' field should say 'myBr', and the 'Pull strategy' should switch to 'Merge'. Click the 'Finish' button. You should see that the square bracket

repository name next to the project name has been changed to the branch name. Make a change to your source file, save it, stage it, add a commit message and commit and push it.

If you go the GitHub site and navigate down the branch to the source file, it should have changed, while the version on the master did not.

NOTE: If you have to push from the command line, make sure you push to the branch using the command:

```
git push origin myBr
```

This will push your changes onto the branch.

9. Merging Changes

Whether you work on branches or everyone on your team works on the master, you need to periodically pull changes that have been put onto the master into your work and merge them in so that you can get other people's work and reconcile any problems. After merging, changes can be put on the master so that everyone can get them. This can be a fairly involved adventure, but a key is to do it often so that the base of your changes does not vary greatly from what other people are using. There are many good discussions and tutorials on GitHub and the Git site on-line book (http://git-scm.com/doc).