Tutorial 2 – GitHub Tutorial

The purpose of this tutorial is to provide an overview of using GitHub. This tutorial should be completed collectively with your software project teams. The tutorial goes through the basic use of git and sets up a collaborative source repository.

1. Configure Git

Run the following commands to configure your git environment:

Replace content in {}'s and the {}'s with your information:

```
$ git config --global user.name "{Firstname Lastname}"
$ git config --global user.email {my@email.com}
$ git config –global core.editor gedit
```

You may choose a preferred editor.
Pico and nano are two text-based editors that are easy to use.

Check that the settings are ok:

```
$ git config --list
```

Check out the git manpage:

```
$git help config
```

2. Copy the "sample_maven_web_app" Netbeans project for this tutorial

Make a complete copy of the sample_maven_web_app.

In Ubuntu:
Recursively copy the full project directory to a new directory:

From /home/ubuntu:

```
cp -R sample_maven_web_app new_app1
```

Purge the existing ".git" directory under the new project:

```
rm -rf /home/ubuntu/new_app1/.git
```

Now this would be as if we had created a brand new maven project (like the one for your group project), and we are now adding it to github.

In netbeans, open this project, and rename it to “new_app1”
Right click on the project, go to properties, under the general tab

For teams using Java maven projects, this project could be used as your main working repository for the group project.

3. Create a new project in github

Change into the new project's directory:

```
cd /home/ubuntu/new_app1
```

Next at the command line initialize a new repository (“repo”) for this project:

```
$git init
```

Next we will add all of the files

```
$git add *
```

Now commit the new project to your local repository (not the cloud):

```
$git commit
```

The gedit editor will popup and ask for a commit message.

Commit messages are important.
This is where the developers provide a brief description for the changes they are about to commit. These comments help others understand what changes have been made to the code. Often it is not unusual to go through source code repositories to find when and where a particular change has been committed, in case there is a bug or question regarding a change. This also supports answering “who dun it?” for better-or-worse.

If you don’t save the comment file, your commit will abort.
You must save and close the editor to complete the commit.

4. Change a file and commit the change

Next, let’s change a file.

Edit the src/scripts/post_test.sh
For example, try commenting out an ENDPOINT and enabling another, then save

the file.

Next try the git status command which shows the status of all files:

From /home/ubuntu//new_app1:

```
$git status
```

Notice that git reports the change to src/scripts/post_test.sh even though we are in /home/ubuntu

A shorter version of git status is:

```
$git status -s
```

Now commit this change.

5. Ignoring binary files

Git can be configured to ignore certain types of files, such as binary files. Traditionally these are not committed to source code repositories. They are binary (compiled) files so they can be large. Both transferring and storing them is unnecessary since they can easily be regenerated by compiling/building the project's source code.

Create a .gitignore file using a text editor under the root directory of the project.

The following github repository has a set of standard .gitignore files for each language. These provide a good starting point:

https://github.com/github/gitignore

Save the file, and add it to your repository.

```
$git add
```

Then commit,

```
$git commit
```

6. Other git commands

Two other git commands to be aware of are:

```
$git log
```

This command provides a log of commits to the repository.
Check the many ways to search and filter git logs:

```
$git log --help
```

Try browsing a condensed listing:

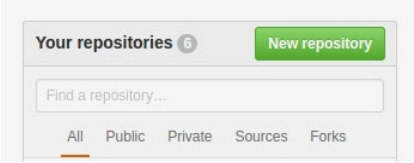
```
$git log –online
```

Go ahead and try out some other log filters.

7. Pushing changes to the cloud

Now, let’s push the changes to the cloud in github:

First, you will need to create a new repository on github.comf
Log into github.com and click on the GREEN “New repository” button.



Create a new public repository called “new_app1”.

Next, add an “origin” repository at github.com:

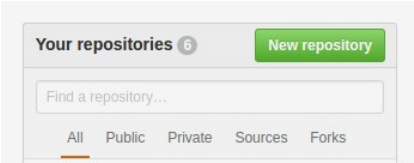
```
git remote add origin git@github.com:{your-username}/new_app1.git
```

Now, push the code to the “origin” repository.
Here “origin” is the name of the repository, and “master” is the name of the code branch:

```
git push --set-upstream origin master
```

Now, let’s demonstrate how you can have multiple repositories for the same codebase.

Create another new public repository on github. The GREEN button again:



This time let’s call it “new_app_other”.

Now let’s add a new remote called “other”:

```
git remote add other git@github.com:{your-username}/new_app_other.git
```

And now let’s commit the code to the “other” repository

```
git push –set-upstream other master
```

Now let’s add a new text file to the repository.
Create a file called readme.txt under /home/ubuntu/new_app1 using a text editor.

Now add this file:

```
git add readme.txt
```

And commit to your local repository-name

```
git commit -a
```

And finally push changes to the “origin” repository:

```
git push --set-upstream origin master
```

Now using the github.com web interface, check for the existence of the readme.txt file.

Is the new file in the “new_app_other” repository?
In “new_app1”?

Using git, it is possible to set up local your very own local github server, where you might commit code.

To see a list of your configured “remotes” type:

```
git remote -v
```

or simply just

```
git remote
```

8. More git commands!

Now let’s review a few other commands.
No need to try them now, just be aware of them for future reference:

The “git rm” command can be used to remove file(s) from the repository.

```
$git rm <filename>
```

To clone a repository, which means to checkout the source files from github.com for the first time:

```
$git clone <repository name>
```

as in:

```
$git clone https://github.com/wjlloyd/sample_maven_web_app.git
```

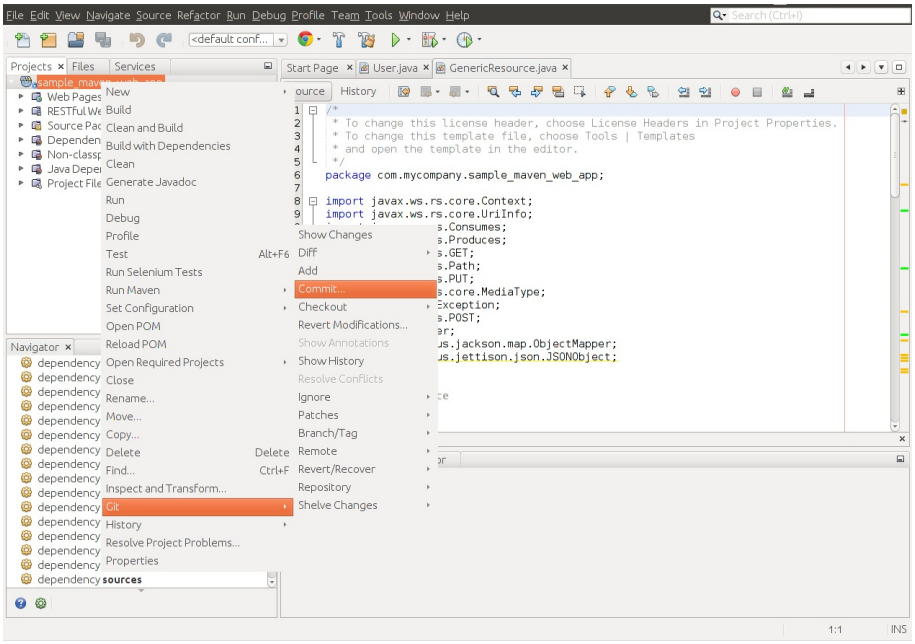
9. Git from the Netbeans IDE

Git hub interaction is supported directly through the Netbeans IDE.

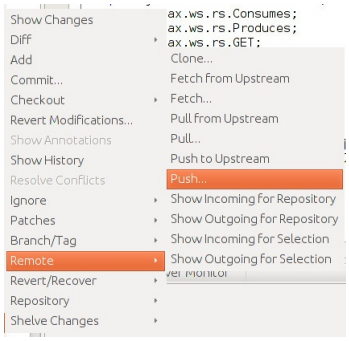
Right click on your project name, then find the “Git” menu:

Now, let’s try editing the readme.txt file that you’ve added.
In NetBeans, click on “Files”, and find the readme.txt.
Double click to open it for editing.
Edit the file.

Now let’s commit the changes to your local repository from within Netbeans.
Select the commit option as below:



Now, let’s push these changes. From the “Remote” menu, select “push”:



Select the remote repository for the commit.
If necessary, you may be asked for authentication. If so, you click on the radio button and specify your RSA key as opposed to using a password.

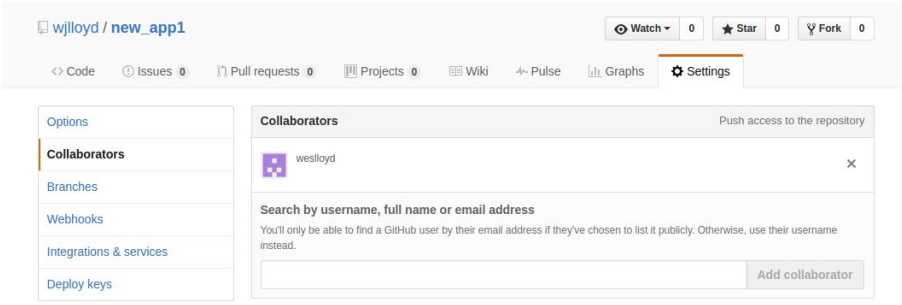
10. Group Collaboration with Git

By adding project collaborators, other users can directly check out code from a single repository. The master code trunk can be branched into separate copies for individual developers. These copies are called “branches”, and the master is called the trunk.

Individual developer changes are eventually then merged into the trunk. Branching also can be used to support prototypes and experimentation. The master code could be branched to prototype a possible feature, which is later discarded.

Let’s try adding collaborators to your new_app1 repository:

Go to the github.com web client.
From the login page, select the “new_app1” repository.
Click on “settings”.
On the left hand side, select “Collaborators”:



Add everyone from your TCSS 360 project team by username.

They will need to accept the invitation to collaborate, sent by email, to officially become a project collaborator.

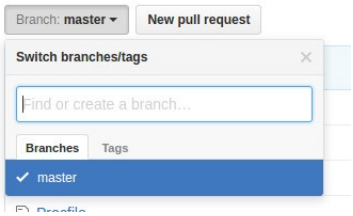
Collaborators are then able to clone the repository directly using the project URL. This is different than if they fork the repository. If they fork it, they make a complete copy of the project to their own github account.

For your group TCSS360 project, someone will need host the group project in github, and everyone else will share as a collaborator.

11. Working with Branches

Now let's try working with branches.

The project owner should now create a branch named after each member of the team on the github.com website:



Type new branch names into the box that says "Find or create a branch"…

Now, have each group member clone their branch.

Branches can be cloned directly in netbeans.

With all projects closed, go to Team | Git | Clone.
A wizard will step through cloning the repository.
One the second step, you can specify to clone each team member's branch and not the master.

On the command line, branches can be cloned with using "-b"

For a "Fred Smith" collaborator to clone their branch:

```
git clone -b "fredsmith" https://github.com/{original-owner-username}/new_app1.git
```

Once each team member has cloned their branch, have each developer add a line of text to the readme.txt in the root directory of the project as follows:

"Hello my name is Fred Smith."

Now, have each group member commit and push this code:

```
git commit -a
git push
```

When you inspect the readme.txt file on github.com, you should be able to select the branch drop down. You'll see how each team member's readme.txt file is now different.

12. Merging the branches

First from the command line, let's see how the "checkout" command let's you switch the active branch.

For one of the users who has committed their readme.txt:

```
$cat readme.txt
```

Note how the text file contains the user's specific addition.

Check out all of the branches:

```
$git branch
```

Now let's switch to the master branch and look at readme.txt

```
$git checkout master
$cat readme.txt
```

This file is different. Where did the text go?

Let's go ahead and merge the two branches:

```
$git merge {user's-branchname}
```

This is a simple merge operation. There was no conflict. Let's see what is in the readme.txt file in the master branch now:

```
$cat readme.txt
```

Go ahead and commit this:

```
$git commit -a
$git push
```

Now merge all of the other branches one by one…
Look at the readme file:

```
$git merge {user's branchname}
$cat readme.txt
$git commit -a
$git push
```

13. Resolving Merge Conflicts

Now, let's introduce a merge conflict.
Select one group member. First they will need to merge with the master branch:

```
$git checkout {user's branchname}
$git merge master
```

Now already there may be a conflict in readme.txt.
If so, edit the file to make it match the original file checked into the master and commit the change:

```
$git commit -a
$git push
```

Now have the user introduce a change into their readme.txt on their own branch.
Have them change one character of the last word in readme.txt.

```
$gedit readme.txt
$git commit -a
$git push
```

Next, in the master, change the same character, but make it something different:

```
$git checkout master
$gedit readme.txt
$git merge {user's-branchname}
```

```
ubuntu@ubuntu-VirtualBox:~/test/new_app1$ git merge weslloyd
Auto-merging readme.txt
CONFLICT (content): Merge conflict in readme.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Now, open the readme.txt file in an editor.
Git has marked the conflict error.
The merge conflict must be manually corrected.

```
<<<<<<< HEAD
Hello my name is Fred SmitA.
=======
Hello my name is Fred SmitB.
>>>>>>> weslloyd
```

The top portion shows the master branch, while the bottom shows the specific user's branch. To correct the merge, the file should be edited to include just one version of the text line. All of the <<< === >>> markings should be removed:

```
Hello my name is Fred SmitA.
```

14. Receiving Credit for this Tutorial

To complete this tutorial and receive a grade, an group assignment has been set up in Canvas. The group should collectively be able to submit the test_app1 project git repository URL to canvas. The repository should be a copy of sample_maven_web_app with the collaboratively built readme.txt file generated through merging.