SE-2030 Lab 3: Intro to Git - using Git with Bitbucket

Objectives

- Learn to use the Git Version Control System to maintain files related to software development in a Bitbucket remote repository.
- Learn to use a Git client (mainly Git bash, but also Git GUI)

Bitbucket account

You should have received an email from Atlassian/Bitbucket inviting you to use a Bitbucket repository. This email was generated automatically when your instructor added you to the repository's list of authorized users. If you haven't set up a Bitbucket account profile yet, do that now.

Use your real name for your account, as well as your MSOE email. Note that the password you choose is completely separate from your MSOE password.

Git installation

You should already have **Git bash** and **Git GUI** installed on your computer. However, if not, consult the instructions posted on the course web page. Be sure to follow the steps in **sections 2 and 3** so that you avoid pushing unnecessary files (for example .class files) up to the repository. Accidentally adding the wrong file type both slows down access and creates extra work every time you commit changes.

If you are using an IDE other than Eclipse, consult your instructor for additional settings you may need to make to your .gitignore file.

Also, be sure to follow the instructions **in section 5** to setup and install a **public key** so that you can authenticate yourself to Bitbucket whenever you use the Git client.

Assignment

As a student, you already have a remote Bitbucket repository created for you and your team. Besides your instructor, only your team has access to your repository. Caution: Your instructor can view (and modify) any file you place in your repository - even after you delete it from the repository. Don't place any inappropriate files in the repository (e.g. passwords to your bank account, etc.). The url to your repository takes the following form: **git@bitbucket.org:TEAM/REPO.git**, where you replace **TEAM** and **REPO** with values supplied by your instructor. Note that the entire url is case-sensitive.

There are multiple parts to this assignment. The first parts are to be completed during lab, while the remaining parts may require outside work on your part.

Be sure to follow the directions in this lab precisely! Most of the problems with using tools like Git is in not being careful about things like the number of dashes in a command or putting spaces in the right place. Carpenters say “measure twice, cut once”. An equivalent saying for users of new software tools might be “double-check and run once” as opposed to typing wrong commands over and over.

Part 1: Creating a local working folder

Every member of the team should do this part.

To begin with, create new, **EMPTY** project folder where you'll be eventually storing the source code and other files comprising the WordCounter application (e.g. C:\MyProjects\SE2030\WordCounter). Note: you can choose any filepath you like for this folder - it does NOT have to be the same path that your teammates use.

From within Windows File Explorer, right-click on the **WordCounter** folder. From the context menu that appears, select **Git Bash Here**. This will open a Git Bash console where you'll be entering Git commands. Type the bash command **ls -al**, a very common command which lists all the files and directories in the WordCounter folder. The output you see should indicate that the WordCounter folder is empty.

Part 2: Cloning the remote repository

Copy the following command into your bash window (NOTE THE PERIOD at the end of the command):

```
git clone git@bitbucket.org:TEAM/REPO.git .
```

where **TEAM** and **REPO** are filled in with appropriate values for your team.

This command creates an empty LOCAL repository on your laptop and copies the contents of the **remote** repository to your local repository and into your WordCounter folder.

Since there is nothing in either your local or remote repository yet, your WordCounter folder will still be empty, EXCEPT for a (normally hidden) .git folder that was created as the result of the command above. This folder essentially contains your local Git repository. Do not delete or modify this folder (that would destroy your local repository). You can see this hidden folder by typing **ls -al** at the bash command prompt.

STOP Demonstrate this to your instructor before proceeding.

Part 3: Adding files to the repository

Choose one person on your team.That person should create a **src** and a **docs** subfolder in their WordCounter directory. The rest of the team gets to observe at this point to make sure it gets done correctly. These folders will eventually contain the code your team produces for the WordCounter application.

That person should next create a text file in the docs subfolder, using their name as the name of the text file. Use the following commands to add the files to the local repository. Ask your instructor for help!

```
git status
git add .
git commit -m “added my text file to my local repository”
```

At this point, that person has committed a file to their local repository. Once this is done, the local repository must be synchronized, or “pushed” to the remote repository.

Type

```
git push origin master
```

This copies the contents of their local repository to the remote repository. Now the file is safely stored in the remote repository, where it can be retrieved to the other team members' local repositories. Note that Git may warn you that you have to type in another command in order to establish the correct “connection” to the remote repository – if it does, just follow the instructions that appear on the screen.

STOP Demonstrate this to your instructor before proceeding.

Part 4: Synchronizing the remote repository to a local repository

The rest of the team does this part; the person who did Part 3 watches: The remaining people on the team should type

```
git pull
```

Note that Git may warn you that you have to type in another command in order to establish the correct “connection” to the remote repository – if it does, just follow the instructions that appear on the screen.

This fetches the contents from the remote repository into your local repository. You should now have a **docs** subfolder in your WordCounter folder that contains your teammate’s text file.

Next, create your own text file in your docs subfolder. Type the commands from step 3 to add your file to your local repository and synchronize to your remote repository.

NOTE: You may encounter some synchronization warnings. Ask for help!

STOP Demonstrate this to your instructor.

Part 5: Branching the repository

Your instructor has created a lab3_practice branch on your team's remote repository. Before using that branch, you must make sure your current work is completed and committed to the master branch. To do this, repeat the usual 3-steps for staging and committing your files:

```
Git add .
Git commit -m <commit comment goes here>
Git push
```

To fetch the lab3_practice repository locally and switch to it, type the following commands:

```
git fetch && git checkout lab3_practice
```

You are now on the lab3_practice branch, and the files in your working folder have been replaced with the files on that branch.

Part 6: Merge practice

You and your teammates should select a single common file to edit simultaneously. Using an editor, add your name to the bottom of that file. Close the file and add, commit and push. If you were not the first one to push, you will get errors when you try to push since one of your teammates pushed before you. Git should print some information regarding merging to the bash console. Try to follow those to successfully merge your changes into the repository. Ask for help if needed!

Repeat Part 6, but let someone else make the first push so that they can experience the merge conflict.

Before proceeding to Part 7, everyone should **git pull** to get back in sync.

Part 7: More merge practice

Next, you and your teammates should edit your own text files. . Change the file by adding your name (or some other minor edit). Close the file and add, commit and push. If you were not the first one to push, you will get errors when you try to push since one of your teammates pushed a different file before you and changed the

remote repo.. Again, Git should print some information regarding merging to the bash console. Try to follow those to successfully merge your changes into the repository. Ask for help!

Edit

Lab Submission

The files in your remote repository constitute your submission!

Your grade will be based on the following aspects:

- Demonstration of the creation and files committed to both the local and remote repositories
- Correct configuration of Git on your computer - including .gitignore and .gitconfig files.