ECE271: Microcomputer Architecture and Applications — University of Maine

Gitlab Setup/Usage

by Yifeng Zhu
modified by Vince Weaver and Pascal Francis-Mezger
February 7, 2022

Background

- We will submit our labs in ECE271 via git to the department gitlab server.
- Git is a widely used version control system for source code maintenance. There is a free online book about Git here: `http://git.scm.com/book/en/v2`. Chapter 2 gives a rundown of basic Git commands.

Workflow Overview

Basic workflow:

- **Clone** a repository
- Write your code
- **Add** the files you have changed
- **Commit** your changes and give a commit message
- **Push** all your commits to your server

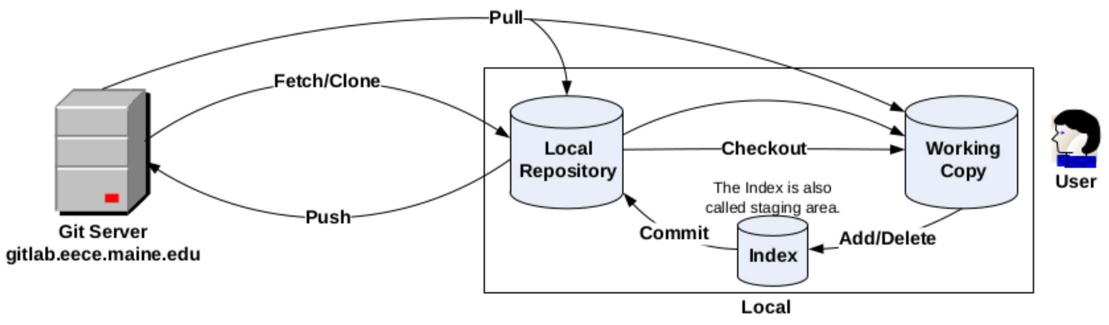

Figure 1: Diagram of what git is doing. Don't worry, for some reason most diagrams about how git work are fairly incomprehensible.

Best Practices

- Commit early, commit small, commit often
  This is especially important when working on a group project as changes to the same file can lead to a dreaded "merge conflict"

- Always commit before you do a pull
- On group projects, always pull to check for new changes before you push your changes
- Do not commit files that can be regenerated from source.
  For example, there's no need to commit .o files, executable files, or anything that is autogenerated at build time.
- Be sure to choose meaningful commit messages.

Setting up your Gitlab account

- Log into the gitlab server `https://gitlab.eece.maine.edu`. Use the LDAP login option, with your MaineStreet username and password.
- You need to log in once to get into the system so the TAs can properly add your account to the various ECE271 groups.

Installing Git

If you are on Linux, you likely do not have to install anything and can skip forward to the next section, Generating a Public Key. For Linux users you will be using the Linux terminal for any of the instrutions instead of Git Bash

1. Install git for Windows
   (a) Download the latest git for windows from `https://gitforwindows.org/`
   (b) Choosing the default options for everything during installation will work fine. If you have specific requirements, there may be some options such as default text editor, or only using Git from the Git Bash command line if you would prefer your Windows path settings to be unaltered.

Generating a Public Key

You will need to generate a public key for your computer to authenticate to your Gitlab account. This is done through a simple command line program. If you are on Windows, open the Git Bash software that was installed in the previous step. On Linux, just open a terminal.

The command to run is:

```
ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

The command will give a few prompts. You should leave the path as default unless you know what you are doing (On Windows this should be `/c/Users/yourusername/.ssh/id_rsa`), but you should change the id_rsa part to be something more descriptive of what this public key is for. For example, ece271_git would be reasonable. Type the full path (`/c/Users/yourusername/.ssh/ece271_git`) to change the name and then hit enter.

Utilizing the Public Key

Now that you have created a pubic key, you need to add it to your Gitlab account so Gitlab knows to allow your interactions.

1. Go back to your account at `https://gitlab.eece.maine.edu` and sign in
2. Click on your user icon in the top right, and select preferences from the list
3. On the left hand side, select SSH Keys. You should now see a large box where you can paste in your public key
4. Go back to the Git Bash window
5. Open the public key file you created, so you can copy and paste your key. If you are on Windows, you can navigate to the `C:/users/yourusername/.ssh/` folder, and then right click on your public key file (something like ece271_git.pub) and edit/open with something like Notepad. If you are on Linux, or just want to use the Git Bash command line, you can open the file with a terminal text editor. For example,

   ```
   vim C:/Users/yourusername/.ssh/ece271_git.pub
   ```

   would work if you alter the path to match where you generated your key. In Vim you could then copy by typing :%y+ and then hitting enter. To exit Vim just type :x and hit enter. If you are using a normal text editor such as notepad, just highlight all of the text, right click on it, and choose copy.
6. Go back to the Gitlab Window and paste the public key into the box
7. Choose a name in the box at the bottom relevant to the computer you are using, so if you use multiple computers in the future and need to add another public key you can tell them apart. Also give an expiration date that is after the end of the semester, then hit add key.

Create an SSH Config File

The SSH config file will tell your SSH connection to the Gitlab server what credentials to use. Without setting this up properly you will get a connection denied error in later steps with interacting with the Gitlab server.

1. Create a new blank file in the .ssh folder (the same folder where your public key is) named config. You can do this with notepad (just open a blank notepad file, and then do a save as in the .ssh folder and name it config, removing the automatically added .txt extension). You can also do this from the Git Bash command line or terminal by navigating to the folder (cd .. to go up a directory, cd foldername to go into a directory) and then use

   ```
           vim config
   ```

   to create a blank file called config. To enter the text below, in vim just type i to go into insert mode to type text.

2. Enter the following into the config file and then save it

   ```
   # gitlab.eece.maine.edu account
   Host gitlab.eece.maine.edu
   HostName gitlab.eece.maine.edu
   User your.name
   IdentityFile ~/.ssh/your_public_key_name
   ```

   In vim to save, hit escape to exit insert mode, type :w and hit enter to save, and then :x and enter to exit.

You should now be ready to start interacting with the Gitlab server. All the previous steps should only need to be done once to initially set up your connection to Gitlab.

Create a New Project in Gitlab

You will be using a single Gitlab project for all of your labs this semester, so you will only need to do this step once. Each lab will have its own folder inside of this project.

1. Log in to your Gitlab account at `https://gitlab.eece.maine.edu`
2. Choose the New Project option at the top right
3. Choose to create a blank project
4. Give the project the name ece271
5. Leave the project as private
6. Click on create project

Clone Your Gitlab Repository

Now that you have a project to interact with, and your computer is authenticated with Gitlab, you can start interacting with files on the Gitlab server. For this class you will create and modify files on your computer, and then push them to the Gitlab server. The first step will be to clone the Gitlab project you created to somewhere on your computer, and use that as your master.

1. Decide where on your computer you want your labs. A reasonable place would be in your My Documents. Generally it would be advised against doing this directly in C:, or in a cloud managed folder, due to potential administrative rights issues.
2. In the Git Bash window, navigate to where you have decided to clone the project. You would navigate by using cd .. to go up a directory, or cd foldername to go into a directory.
3. When you are in the location you decided on above, you can clone the repository. The command for this is:

```
git clone git@gitlab.eece.maine.edu:your.name/ece271.git
```

If you receive an error about permissions, go back and make sure you completed all of the previous steps.

You will now have a new folder on your computer in the location you selected, named ece271.

Utilize Git with Push and Pull

The folder you got from cloning in the previous step will be the folder you will utilize throughout the semester for git. You can access it from your normal file browser, or from the command line. Do a trial run to make sure everything is working correctly by following the steps below. You will be setting up a test for lab 1, but the same directions would be used for any future labs.

1. Go into the the ece271 folder that you created in the previous step in Git Bash. You can do this by right clicking in your file explorer and choosing the Git Bash Here option, or by opening Git Bash and navigating to the location with cd
2. Run the command

   ```
   git pull
   ```

   Pulling will ensure you have the newest files from the server before you start modifying anything. With you being the only person utilizing your git project, this isn't generally necessary, but it is a very important habit to get into. If you start modifying files without doing a git pull first, if there are changes on the server that you do not have it can be a lot of work to sync things back up. If git pull did not find any differences on the server, you should just get a message saying "Already up to date."
3. Create a new folder in the ece271 folder, called lab_1. You can do this from the file explorer by right clicking and choosing new folder, or from the command line with

   ```
   mkdir lab_1
   ```

4. Copy your lab 1 files into this folder. Drag and drop them in the file browser, or use the cp command on the command line. cp requires the file path to move the files to, so look up the usage on the internet if you need help.
5. Add files to be tracked by git using git add. You should not add compiled files, only the source files. For example, you should add any .c files, and any .h files. You can do them one by one using their filename, or use a wildcard character (*) to add any matching file of that file type. For example, you can add any file that ends in .c using

   ```
   git add *.c
   ```

If you do not add the file, when you do a git push to send the files to the server, they will not actually be uploaded. Only added files will be sent to the project on the Gitlab server.

6. Commit your changes using

   ```
   git commit -m "Some message describing the changes you made"
   ```

   Even if a file has been added, changes will not be uploaded until you commit your changes. The -m signifies you want to add a message to your commit. You should always put a relevant message in the quotes to tell others that may view your commits what your goal was, but also to help yourself remember what you have done.

7. Send you changes to the server using

   ```
   git push
   ```

   This will send all of the committed changes to the added files to the gitlab server

Git for the Rest of the Semester

1. For each lab create a folder in the ece271 folder named for that lab (lab_1, lab_2, etc)
2. Download the base files for that lab, and put them in that folder. Push them to git.
3. Perform your lab. It is recommended to commit changes relatively frequently to get used to using git, rather than just once when you start the lab and once when you finish.
4. Create a README.md file in the lab folder to answer the post lab questions
5. Push all of your code and the README to the server before the due date. Check that it has been properly submitted by reading the messages output by git push and by logging into the Gitlab server and viewing the files are there.