CSE 220: Systems Programming

Lab 01: Introduction to Git

Introduction

This recitation will help you become familiar with `git`, GitHub, and the GitHub Classroom interface that we will be using. You will also use Autograder to submit your results. You will be required to join a GitHub Classroom assignment, check code out of GitHub using the git command line tools, modify and commit a changed file, push it to GitHub, and then submit the file to Autograder.

Using Git, and using it well, *will absolutely improve your experience in this course.* You should focus not only on achieving the points in this lab (for which full credit is required to pass the course), but also on learning something about Git. Using git well will almost certainly improve your final grade in this course!

This document assumes some familiarity with the UNIX command line. If you are not yet comfortable with the command line, just follow the instructions as best you can, and ask your TA when you get stuck or become confused. By the end of the semester, these sorts of command line shenanigans will be second nature!

1 Getting Started

The first thing you need to do is sign up for a GitHub account if you do not already have one. You can do this by navigating to github.com and filling out the form on the front page. If you already have a GitHub account and wish to use it for this course, that's fine.

Once you have a GitHub account, you will need to follow the GitHub Classroom link that should have been posted on Piazza. If you are not yet fully enrolled in this course or you do not have access to Piazza, your TAs will provide you with the link.

1.1 Setting up SSH Access to GitHub

In order to check a Git repository out of GitHub onto your computer, you will have to use the `ssh` command and *ssh keys* to log into the GitHub server. SSH keys are a method of authenticating your system to a remote machine via cryptography. This section will walk you through creating ssh keys and uploading the resulting key to GitHub. Note that this is only worthwhile on a machine that has persistent storage (that is, your home directory will be around for a long time), and only *safe* on a machine that you trust (*e.g.*, the virtual machine image you'll be using for this course, or your personal laptop).

To create an ssh key, open up a command prompt and run the command `ssh-keygen`. It will prompt you for both a key file and a *passphrase*. Use the default filename, and select a good passphrase. A good passphrase should be moderately long, yet memorable; a series of unrelated words (unrelated, that is, to other people), a long password-like string, or similar. Creating good passphrases is outside of the scope of this course, but there are good techniques available online [2, 1], and there are tools to manage such things for you. After you have chosen a passphrase, `ssh-keygen` will generate a key (which may take some time, particularly on a virtual machine) and print out a report:

```
eblanton@westruun:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/elb/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/elb/.ssh/id_rsa.
Your public key has been saved in /home/elb/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:x44gY3VWXQDTVUEMyqZXvqDaUoJHEt8A6hUCCJMfA30 eblanton@westruun
The key's randomart image is:
+---[RSA 2048]----+
```

```
|*=.. o.   ++.=*+o|
|o.+ E... ...+  . |
| . = .+ =  + .   |
|  o .o = oo o    |
|   .+ = S.oo .   |
|   . + + =o . .  |
|      . +..  .   |
|       .o        |
|       ...       |
+----[SHA256]-----+
```

Once your key has been generated, log into the GitHub web site, drop down the profile menu in the top right (it will be your profile picture with a dropdown arrow beside it), and select Settings. From the settings page, select "SSH and GPG keys" on the left, and then click the green[1] "New SSH key" button at the top right.

From the terminal where you created your ssh key, run the command `cat ~/.ssh/id_rsa.pub` (the `.pub` extension is important!), or whatever key you just generated, then select the resulting line of output and paste it into the "Key" field on the GitHub new key page. It should look something like this (note that the output will be one *very long* line):

```
eblanton@westruun:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC6DdUioe8FwnzPPNf+Rl2 [...]
```

Give the key a descriptive title (for example, "CSE 220 VM key") and hit the green "Add SSH key" button.

From here on out, the first time you use your ssh key after each boot, you should be prompted to enter its passphrase. After that, ssh access to GitHub will proceed without having to authenticate manually!

1.2 GitHub Classroom

After following the GitHub Classroom link, you will need to log in to your GitHub account (if you're not already logged in) and then click the green "Accept this assignment" button. A message that the repository is being set up and the starter code imported should show up. It may take a few minutes to complete. If it seems to be taking a long time, you can open GitHub in another tab and see if your repository has already been created; it will be at:
`https://github.com/ub-cse220-s21/lab01-github-YourGitHubUserID`

You will likely find that a link is present on your GitHub front page in the box labeled "Repositories" on the left side. Clicking on the repository should show a `README.md` file with some instructions. If the repository is empty, let a TA know so that they can fix it. Once you can find this repository on GitHub, you may proceed with the assignment.

1.3 Checking Out Your Repository

In this assignment, you will use the `git` command line interface. For the rest of this course, you may use whatever Git frontend (often called "porcelain") you prefer. You will probably find that the course staff can only assist you with the command line.

Navigate to the GitHub repository that GitHub Classroom created for you when you accepted the assignment invitation. It will be called something like `ub-cse220-s21/lab01-github-userid`, where `userid` is your GitHub username. You should see a green "Code" button toward the right side of the screen; drop it down and copy the clone URL from the resulting dialogue box. There may be more than one possible clone URL. If so, select "SSH" and copy that URL, which should start with `git@github.com:`.

Open a command prompt, navigate to a convenient place in your filesystem (I keep all of my projects separated by directory; for example, everything for this course is in `~/work/buffalo/cse220`. When I was a student, I would have used `~/class/cse220`) and run the command `git clone $URL`, where `$URL` is the URL you copied from GitHub. If you want to clone into a directory other than the default (which will be `lab01-github-userid` for this

[1] For those who may be (in particular) red-green colorblind: sorry, most of the GitHub action buttons are green. The text should make it obvious. I don't know why they did that.

project, where `userid` is once again your GitHub username), you can add it to the command line at the end, as `git clone $URL $dir`, where `$dir` is the desired directory name. *E.g.*:

```
eblanton@westruun$ git clone eblanton@ub-cse220-s21/lab01-github-eblanton
```

This command would clone the project into a directory called `lab01-github-eblanton`.

You may need to enter your SSH key passphrase or your GitHub password at this point. If you have previously entered your SSH key passphrase during this login session, it may complete without prompting for authentication.

2 Requirements

Having checked out your GitHub repository, you are required to:

1. create a file with specific content and commit it to your local git repository,
2. push the changes to that file to your GitHub repository,
3. then submit the *file itself* to Autograder.

The specific file that you must create is named `userinfo.txt`, and it should contain exactly two lines of text, each terminated by a single UNIX newline character, with no extraneous whitespace. Autograder will verify these requirements. The two lines that it should contain are your UBITName and your GitHub username, in that order. There is a file `README.md` in the handout repository that has an example.

Once you have created this file, you should use `git add` to add it to the *index* for your git workspace. Please review `git help add` or `man git-add` for more details on the add command, which you will want to learn. Having added it, you must *commit* using `git commit` to save your changes to the local repository, and then *push* your changes to the GitHub remote repository with `git push`. Like the add command, you can view the documentation for these with `git help` or `man`, and you should do so. (Note that, for this assignment, the `add`, `commit`, and `push` commands that you will use are very simple.)

Only after your changes are pushed to GitHub, submit your `userinfo.txt` — and only your `userinfo.txt` — to Autograder as your solution for Recitation 1. Autograder will verify that your GitHub username matches your submission by consulting your GitHub repository, and then pass or fail your submission based on the above requirements.

3 Troubleshooting

The Autograder scripts will provide you with some feedback if your submission is not entirely correct. You should look at the output of the Autograder execution to see this feedback. In this course, you should be in the habit of looking at the Autograder feedback for your projects, as it may help you improve your scores.

4 Grading

A correct, complete commit to GitHub accompanied by a matching submission to AutoGrader will receive full credit (5 total points). Any other submission will receive no credit. Note that the Autograder score sheet may show some points even if no points will be ultimately awarded.

References

[1] The Electronic Frontier Foundation. EFF dice-generated passphrases. https://www.eff.org/dice.

[2] Bruce Schneier. Choosing secure passwords. https://www.schneier.com/blog/archives/2014/03/choosing_secure_1.html.