CMPSC 280
Principles of Software Development
Fall 2015

Laboratory Assignment One: Version Control with Git and Bitbucket

Introduction

Practicing software developers normally use a version control system to manage most of the artifacts produced during the phases of the software development life cycle. In this course, we will always use the Git distributed version control system to manage the files associated with our laboratory assignments. In this laboratory assignment, you will learn how to use the Bitbucket service for managing Git repositories and the `git` command-line tool in the Ubuntu operating system. After connecting to the course's Git repository and creating your own repository, you will work in a team to create Markdown-based documentation of command-line programs that exist in Ubuntu.

Configuring Git and Bitbucket

During this laboratory assignment and subsequent assignments, we will securely communicate with the Bitbucket.org servers that will host all of our projects. In this laboratory assignment, we will perform all of the steps to configure the accounts on the departmental servers and the Bitbucket service. Throughout the assignment, you should refer to the following Web site for additional information: `https://confluence.atlassian.com/display/BITBUCKET/Bitbucket+101`. As you will be required to use Git in the remaining laboratory assignments and during the class sessions, please be sure to keep a record of all of the steps that you complete and the challenges that you face. You are also responsible for working with your team to ensure that everyone is able to successfully complete each of the steps outlined in this assignment.

1. If you do not already have a Bitbucket account, then please go to the Bitbucket Web site and create one—make sure that you use your `allegheny.edu` email address so that you can create an unlimited number of free Bitbucket repositories while you are a student.

2. If you have never done so before, you must use the `ssh-keygen` program to create secure-shell keys that you can use to support your communication with the Bitbucket servers. Follow the prompts to create your keys and save them in the default directory (press "Enter" after you are prompted: "`Enter file in which to save the key ... :`", then press "Enter" twice if you do not wish to create a passphrase at this time or type your selected passphrase if you do). Type `man ssh-keygen` and talk with your partner to learn more about this program. What files does `ssh-keygen` produce? Where does this program store these files by default?

   Once you and your team have created your ssh keys, you should raise your hand to invite either a teaching assistant or the course instructor to help you with the next steps. First, you must log into Bitbucket and look in the right corner for an account avatar with a down arrow. Click on this blue link and then select the "Manage account" option. Now, scroll down until you found the "SSH keys" option and upload your ssh key to Bitbucket. You can copy your SSH key by going to the terminal and typing "`cat ~/.ssh/id_rsa.pub`" command.

3. Now, you need to test to see if you can authenticate with the Bitbucket servers. First, show the course instructor that you have correctly configured your Bitbucket account. Now, ask your instructor to share the course's Git repository with you. Open a terminal window on your workstation and change into the directory where you will store your files for this course. For instance, by opening a terminal window and typing the command "`mkdir cs280F2015`" you could make a `cs280F2015/` directory that will contain the Git repository that the instructor will always use to share files with you. Once you have done so, please type the following command "`git clone git@bitbucket.org:gkapfham/cs280f2015-share.git`".

   If everything worked correctly, you should be able to download all of the files that you will need for this laboratory assignment. Please resolve any problems that you encountered by first reviewing the Bitbucket documentation. If you are still not able to run "`git clone`", then please see the course instructor or a talk with a member of your team.

4. Using your terminal window, you should browse the files that are in this Git repository. In particular, please look in the `cs280f2015-share/labs/lab01/` directory and use `gvim` to study the Markdown files that you find. Throughout the semester, we will use Markdown (and, later, LaTeX) to write documentation for software systems — so, for today's laboratory assignment you should review these files to learn some of the basics of Markdown. Additionally, remember, the "`cd`" command allows you to change into a directory. So, you could type the following commands to go to the directory that contains today's files.

```
cd cs280F2015
cd cs280f2015-share
ls
cd labs
ls
cd lab01
ls
```

Creating a New Personal Repository

Now that you have cloned an existing Git repository, you should make a new repository in the `cs280F2015/` directory that you previously created. First, make a new directory called `cs280F2015-<your user name>` using the "`mkdir`" command in your terminal window. If you opened a new terminal, then you could type the following commands to create the needed directory; again, make sure that you understand each of these steps, discussing them with your neighbor, a teaching assistant, a departmental tutor, or the course instructor if you are confused.

```
cd cs280F2015
mkdir cs280F2015-<your user name>
cd cs280F2015-<your user name>
```

Once you have changed into this directory you can type the command "`git init .`". Then, you can use the "`mkdir labs`" command to make a new directory and "`cd labs`" to change into it. Next, you should again use the "`mkdir`" command to create an additional directory called `lab01`. Please make sure that you completed each of these steps inside of the parent directory called `cs280F2015-<your user name>`; if you are stuck, ask the course instructor or a team member for help. Also, make sure that you review all of the commands that you have typed to see if you input them exactly as they are written on this assignment sheet.

If you know how to do so, then simply use the terminal window to copy all of the Markdown files from the share repository to your own repository. But, if you do not know how to do this step in the terminal window, then you can complete the remainder of these instructions. Next, you should click on the file icon in the upper left corner to load the Ubuntu file browser. Once the first window displays, you should click the "File/New Window" menu item to load a second graphical file browser. At this point, you should have two file system browsers that are displaying the same content. In the first window, which we will call the "source" window, you should navigate to the `cs280f2015-share` directory by clicking on the correct folder icons. Now, please find the Markdown file(s) that are stored inside of the `cs280f2015-share` repository in the "source" window. In the second window, which we will call the "destination" window, you should follow the same process to first navigate to the `cs280F2015-<your user name>` directory, ultimately finding `lab01/`.

The next step in this laboratory requires you to use the graphical browser to copy the Markdown files from the "source" browser to the "destination" browser, ensuring that the file is transferred from the `cs280f2015-share` repository to the `laboratory01/` directory of the `cs280F2015-<your user name>` repository. Once the files are in your own Git repository, please use the `git add` and `git commit` commands to add them correctly. You can learn how to use the `git add` and `git commit` commands in the terminal window by reviewing the "Git Cheatsheet", discussing them with your team members and course instructor, or searching on the Internet.

Next, you should use the Bitbucket Web site to create a repository that has the same name as the local directory and local repository. You must follow Bitbucket's instructions to push the code and tags in your local repository to the remote one. When you are finished with this step, you should see in your Web browser that the Bitbucket servers are storing the Markdown files. Once the Git repository contains the correct files, you should share your Bitbucket repository with the course instructor. In this course your instructor's Bitbucket user name is "gkapfham".

At this point, you can learn more about Git by consulting Web sites like `http://try.github.io/` and `http://gitimmersion.com/`. After discussing them with a class member and a teaching assistant, you should ensure that you have a basic understanding of the following Git commands:

1. `git init`
2. `git status`
3. `git add`
4. `git commit`
5. `git push`
6. `git pull`

Modifying an Existing Markdown File

At this point, you should have the Markdown files in your own repository. Now, you can make some small changes to each of the files, regularly using `git` to commit your changes to the repository. As you do this, please focus on writing descriptive commit messages so that the course instructor can easily understand the changes that you are making to the Markdown files. Students who use `vim` or `gvim` should investigate the installation of the "Fugitive" plugin available from `https://github.com/tpope/vim-fugitive` to support the seamless interaction with the Git repository through the Vim text editor. Students who are using other text editors to modify the existing Markdown files can look for similar plugins or use the `git` program in the terminal window.

Documenting Command-Line Programs

Section 1.3 of your textbook poses the question "What is good software?" and then explains different ways in which this question can be answered. For the Unix and Linux operating systems, there are a wide variety of command-line programs that people have been using for decades. While some may argue that these tools do not evidence the characteristics of high-quality software, others claim that they are exceptional and thus use them regularly. For the final part of this laboratory assignment, you and you team members will develop your own understanding of the word "quality" as it applies to computer software. With a particular focus on command-line applications, you should define the word quality and then evaluate the quality of two command-line programs. One of your applications should be `git` and the other can be chosen by your team. Examples of other programs for study include `ag`, `grep`, `ps`, and `top` — or, alternatively, any other program installed on our computers.

To write this commentary on software quality, your team should create a new repository and share it with the course instructor. Then, make sure that all of the members of your team can also access the Git repository. Please use a descriptive name for your repository that includes the name of our course and a numerically-based name for your team. Again using Markdown to write your documents, please create one file to define the meaning of the word quality and then two additional files, one for each of the command-line programs that you select. Please divide up the work for this part of the assignment so that everyone makes their own well-defined contribution.

Once you have finished the aforementioned tasks, you and your team members should further investigate how to use the two programs that you have chosen. Then, you should write user documentation for both of the programs. Instead of duplicating what is already available in the "man" pages for these programs, you tutorials should provide examples and value-added advice about how to use them in the most effective manner. As you complete this part of the assignment, you can imagine that you are writing a blog post to explain to others how these command-line programs work. In particular, your documentation should focus on the inputs, outputs, and behavior of the chosen programs. To complete this final part of the assignment, please again collaborate with your team members to write Markdown files to document the programs; like before, all of these files should be stored in your team's Git repository. Please ask the course instructor if you have questions about how to complete the final phrase of this laboratory assignment.

For submission, please use `pandoc` to convert all of your Markdown files to PDF documents.

Summary of the Required Deliverables

This assignment invites you to submit printed and signed versions of the following deliverables:

1. The modified Markdown files that you produced individually.
2. A single copy of the document describing your definition of the word "quality".
3. A single copy of the documents that evaluate the quality of two programs.
4. A single copy of the documents that give a usage tutorial for two programs.