Lab 3: Source Code Control

Due: 11:59 PM, Monday, February 15, 2016

In this week's lab exercises, we will be experimenting with one of today's more popular source code control systems:

Git.

You should work in your design project groups. If you don't know the basics, don't let your team get ahead of you before you understand what is going on!

Getting Set Up

Create a document where you will record your answers to the lecture assignment and lab questions. If you use plain text, call it "`lab3.txt`". If it's a Word document, you can call it whatever you'd like, but when you submit, be sure you convert it to a PDF document "`lab3.pdf`" before you submit it.

Version Control Basics

Many Resources have been provided on Blackboard under Lectures.

Question 1: Even if working independently on a software project, describe briefly why a source code control system would be helpful to manage development. Consider both multiple development environments (*e.g.*, a portable computer as well as an office desktop computer, both used in development) and the need for revision history. (4 points)

Git and GitHub

If you are not familiar with Git, read quickly through some of the resources provided on Blackboard.

Examining a sample GitHub repository

In this part of the lab, you will look at a previously-constructed repository on GitHub named "Bootstrap".
Load https://github.com/twbs/bootstrap in your favorite browser. We will examine several aspects of its contents.
First, notice the three buttons at the upper right of the page: "Watch", "Star", and "Fork".

Question 2: Investigate and briefly describe the purpose and functionality of each. (4 points)

In the main table of project files on the page, find the `README.md` file, and click on it to examine its contents.

Question 3: Based on the contents of this file, briefly summarize the purpose of the project.(4 points)

Return to the main Bootstrap page and click the "commits" link. Then scroll down to find the link (under "Commits on Sept 24, 2015") for "Merge pull request **#17693** from **rclai/patch-1** ", and follow that link. Note: be sure to click on the "Merge pull request" part of the entry, not the number.

Question 4: How many changed files are involved? How many additions and deletions were committed? (4 points)

Question 5: What color coding is used to indicate additions and deletions to the files. (4 points)
Return to the main Bootstrap page and click the "Pull Requests" link. You should see a tabbed area with entries for "Issues", "Pull requests", "Labels", and "Milestones". Browse through some of the entries under each category.

Question 6: Briefly describe the purpose of issues and pull requests, and how they differ. (4 points)
Click on one of the links to see the details of one of the Issues on the "Issues" category. Then click on the "Pulse" tab off to the right (shown as a small icon that looks like a heart monitor display).

Question 7: Describe the information shown under "Pulse". (4 points)

Creating your own GitHub Repository

Next, your group will create its own repository on GitHub.
Here are the steps:
1. Make sure each member of the group has and knows his or her GitHub login credentials.
2. One group member should be in charge of creating the repository, and that person should log into GitHub with his or her credentials.
3. Click on the "+" icon next to your username, and choose "New Repository".
    - Give your repository a meaningful name such as "`cs507s15lab5`" and include a description.
    - Be sure to choose the "Public" option, and check the box for "Initialize this repository with a README."
4. Add your teammates as Collaborators
    - Click the "Settings" button on the right.
    - Select "Collaborators" on the left menu.
    - Add each teammate as a collaborator.
    - Back in "Settings", check the "Restrict editing to Collaborators only".
5. Create a Java program `Hello.java` in your repository, which prints your name when it is run.
    - In the main view of your GitHub repository, you can click the "+" after the name of your repository above the file list.
    - Enter `Hello.java` for the file name at the top.
    - Type in your Java code in the main editor window, and commit it to the repository with the "Commit new file" button at the bottom.

Question 8: Give the URL of your repository. Creation of the repository is worth 28 points.

Using your Group Repository

Finally, other team members will access and alter the repository. Each team member other than the one whose account created the repository should perform these steps. It is best if one team member at a time works through these steps.
You can do these either by installing the GitHub desktop application for Windows or Mac, or by using the command-line from `mogul.strose.edu`. These instructions apply to mogul.
1. Log into mogul.
2. Create a new directory into which you can clone your group's repository and `cd` to that directory.
3. Clone the repository with a command such as

```
4. git clone https://github.com/USERNAME/REPONAME.git
```

where you would replace `USERNAME` with the GitHub username of the team member who created the repository and `REPONAME` with the repository name.
This should result in a directory with the team's files.
5. To be certain your clone of the repository is up to date with the master on GitHub, issue a `pull` command from inside the cloned repository:

```
6. git pull
```

7. Edit the `Hello.java` file to add a line or two (could be code, could be comments). Use Emacs or your favorite editor.

Question 9: Issue the command `git status`. What does this print? (4 points)
8. Now issue the command:

```
9. git add Hello.java
```

Question 10: Issue the command `git status` again. What does this print now? (4 points)
10. Your change to `Hello.java` is ready to commit to your cloned repository. Issue the command:

```
11. git commit
```

and add an appropriate message in the text file that comes up. These messages are important to remind yourself and tell others about what you just did.

Question 11: Issue the command `git status` again. What does this print now? (4 points)

You have now committed the change to your cloned repository, but it is not back in the master on GitHub yet.
12. Let's push our change back to the master, with the command:

```
13. git push
```

You will be prompted for your GitHub username and password. Enter your own, not that of the team member whose account was used to create the master repository.

Question 12: Issue the command `git status` again. What does this print now?(4 points)
So your change should now be back in the master copy of the repository. You can verify this back on GitHub's web site for your repository.

Question 13: A successful update to your `Hello.java` by each member is required. (28 points)
There is much more `git` can do, but hopefully this gives you the basics of a typical workflow.

Submitting

Before **11:59 PM, Monday, February 15, 2016**, submit your lab for grading by uploading to Blackboard.