HTTP 5302 Lab 1: Git Foundations

Objective

To get an understanding of git and how it works. By the end of this lab, you should be able to add a project to a git repository with meaningful commits that other developers can understand and follow the development.

Description

This lab will have you download git to your developer environment as well as set up a remote repository to contain git-tracked projects.

Instructions

Git setup

1. Download git

   https://git-scm.com/downloads

2. Register an account with a git repository service

   https://bitbucket.org/product

   https://github.com/

3. Set up an empty repository on your git service account

   On github, once you are logged in there is a "New repository" button coloured in green at github.com/

   You'll be taken to a screen to set up the new repository. Give it a name, preferably something related to this course. Give it a description and make it a public repository. DO NOT initialize the repository with a README and do not add a .gitignore.

4. Initialize a git project on your machine

   Using a command line interface (CLI), have the cli on the directory to contain the project to be tracked by git.

   Use the following git command to initialize a git project. Note, you can also initialize git to existing projects:

   ```
   git init
   ```

   This will create the necessary file(s) that git needs to track your project, .git. There are very few circumstances where you will directly edit this file so sometimes this file is hidden from view (on Windows). Do not remove this file while working with git; doing so removes git from your project.

5. Add a git origin to your git project

   To save your changes remotely, you first need to have an origin to hold those changes. This is the git repository you just set up on your git account. The git command to add an origin:

   ```
   git remote add origin [url containing your remote git repository]
   ```

HTTP 5302 Lab 1: Git Foundations

6. Make your first commit

Now that your local project is tracked by git on your machine and has a remote origin on a server online, let's make a change to this project by adding a README file. Create a markdown file in the same directory as the .git file:

README.md

Add this file to the git tracker:

```
git add README.md
```

You can see what files are added and not added by doing:

```
git status
```

This is command that will be used often.

If you want to add all file additions and changes to the tracker:

```
git add .
```

Then *commit* this change in the project. Commits will take all the files that were added to the tracker and package that change with a commit label of a hash:

```
git commit –m "{Short title/description of the commit}"
```

Commits require a message (-m). You can commit without a message but you will be brought to an unfamiliar interface that will ask you to provide a commit. It is much more convenient to add the commit message at the same time as your commit.

You can perform as many commits as you can locally, but best practice is to *pull* commits from the origin:

```
git pull origin {branch name}
```

This will detect any potential conflicts with the remotely hosted origin and add changes that have been committed remotely to your local project. Conflicts happen when changes are made to the origin while you are making those same changes on the same files. After you've resolved any potential conflicts, you may safely *push* your commits to the origin.

```
git push origin {branch name}
```

By default, each git repository has one branch already set up for you, the master branch. If you want to push to the master branch *specifically* then the command would be like this:

```
git push origin master
```

HTTP 5302 Lab 1: Git Foundations

Git cloning

1. Clone a git project

Navigate somewhere else in your environment, not anywhere inside an existing git project.

Use the following git command to clone a git project from a remote source:

```
git clone {git url}
```

This will create a directory containing the clone repository. Your CLI will not automatically change directory into the newly created directory.

Try this with someone else's remotely hosted git project.

You can view the history of a git repository by viewing it online on a git service or in any downloaded git project by using the git log command.

Branching

1. Create a branch

On large projects tracked by git, development should be done in branches. Use the git command:

```
git checkout {branchname}
```

(optional) `git checkout {commit hash}`

to navigate to the commit that you wish to branch from. If you just checkout the branch, you will be on the latest commit of that branch.

Use the following command to branch from the current commit:

```
git branch {new branch name}
```

These new branches will continue accepting commits in parallel with other branches.

2. Merging branches

Branches can be merged so that commits will follow a shared branch rather than two separate ones. This usually happens when parallel branches no longer need to be parallel so development can continue on the same path. This way, a project can have multiple versions simultaneously and switching between versions is seamless.

Before you merge, pull from the branch you wish to pull from

```
git pull origin {branch name}
```

Once potential conflicts have been resolved, the merging of the branches can occur:

```
git merge {branch name to merge to}
```

HTTP 5302 Lab 1: Git Foundations

Evaluation

Simply upload the url of the repository you used for this lab to Blackboard. What is looked for in this lab is that you understand how to push commits to git.

This lab is worth 5% of your grade.

Grade	Description
0	No url provided.
1	
2	Url provided but contains only a single commit.
3	
4	
5	Url provided with multiple commits, at least one branch, and a merge of branches.