Faculty of Science

Course:	CSCI 4100U: Mobile Devices
Lab Assignment:	#1
Topic:	Git

Note: The purpose of this lab assignment is to ensure that you use Git (and either GitHub or BitBucket) regularly for your projects (and, ideally, for your assignments also). The final group project must be submitted via a GitHub or BitBucket URL.

Introduction

Version control software allows you to save your changes periodically, not just the current versions of files. This allows you to go back and undo some changes that are difficult from which to recover. Version control also makes it easier to work as a team, or to synchronize between multiple development machines (e.g. home and office). Without version control, since each team member could have different versions of the files, there is a strong possibility that work will be done to match old versions of files. Often, the result is that difficult changes need to be combined by multiple users, which can be very slow.

Centralized Vs. Distributed

There are two types of version control software in use today, centralized and distributed. In a centralized version control package, there is one repository, where all changes are saved. Each user performs a *checkout*, to obtain the latest versions of the files, and periodically issues an *update* command to refresh those files, ensuring they are the latest versions. When a user makes changes locally to some files and wants it integrated into the team's work, they *commit* those changes, which sends the new data to the repository. SVN and CVS are examples of centralized version control systems.

In distributed version control, each user has a local repository that stores all files' changes. Initially, users do a *clone*, which downloads a remote repository to their local file system. When changes to the files should be integrated into the project, the user does a *commit*, which updates the local repository. Often, the user will then follow this up with a *push*, which sends the changes to the remote repository. At this point, their changes are visible to other teammates who will perform a *pull* from that same remote repository. Git and Mercurial are examples of distributed version control systems. In this lab, we will use Git.

Getting Started

First, download Git from `http://git-scm.com/downloads` and install it. Next, create a remote repository, by signing up at `https://bitbucket.org/` and choosing `Repositories → + Create Repository`. Ensure that your repository is private. BitBucket should show you the clone request you need, an example of which is shown in the table below.

Note: *If you already have an account on GitHub, you are welcome to use that instead if you wish. The instructions are quite similar, but the URLs will be a bit different.*

Command	**Description**
`git clone https://bsmith@bitbucket.org/bsmith/sample.git`	Downloads the entire repository from the remote server

`git add somefile.txt`	Adds a new file to the list of files that are tracked by Git.
`git commit -m "Description of what I just did"`	Updates the local repository to match the file system (i.e. adds your new changes to the repository)
`git config --global user.name "bsmith"`	Specifies the username to be used when doing push/pull commands to remote servers
`git config --global user.email "barb.smith@uoit.ca"`	Specifies the E-Mail address to be used when doing push/pull commands to remote servers
`git push -u origin master`	Updates the remote repository to match your local repository
`git pull`	Updates your local repository to match the remote repository

Instructions

1. Create your repository
2. Clone the repository to a local directory (your choice)
3. Create a number of files, and add them to the local repository
4. Commit those changes, along with a message
5. Push those changes to the server
6. Pull those changes back to your repository (not necessary, but a job you'd often do in industry since other people would make changes)