Introduction to Git

In this course you will get experience working with the industry standard ***version control system***. A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information. In computer software engineering, revision control is any kind of practice that tracks and provides control over changes to source code. Software developers sometimes use revision control software to maintain documentation and configuration files as well as source code.

As teams design, develop and deploy software, it is common for multiple versions of the same software to be deployed in different sites and for the software's developers to be working simultaneously on updates. Bugs or features of the software are often only present in certain versions (because of the fixing of some problems and the introduction of others as the program develops). Therefore, for the purposes of locating and fixing bugs, it is vitally important to be able to retrieve and run different versions of the software to determine in which version(s) the problem occurs. It may also be necessary to develop two versions of the software concurrently: for instance, where one version has bugs fixed, but no new features (branch), while the other version is where new features are worked on (trunk).

At the simplest level, developers could simply retain multiple copies of the different versions of the program, and label them appropriately. This simple approach has been used in many large software projects. While this method can work, it is inefficient as many near-identical copies of the program have to be maintained. This requires a lot of self-discipline on the part of developers and often leads to mistakes. Since the code base is the same, it also requires granting read-write-execute permission to a set of developers, and this adds the pressure of someone managing permissions so that the code base is not compromised, which adds more complexity. Consequently, systems to automate some or all of the revision control process have been developed. This ensures that the majority of management of version control steps is hidden behind the scenes.

Git

Git is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

Git was created by Linus Torvalds in 2005 for development of the Linux kernel, with other kernel developers contributing to its initial development.

Git in the Classroom

It is difficult to simulate real world practices in the classroom and version control is no different. The code bases we will be working on will be small, our work is not team oriented and there may not be multiple versions of a single program.

That being said, any experience you can get get with a versioning system will benefit your development as a programmer and prepare you for employment or complex project work. With that in mind, we will be using Git and GitHub as the means of downloading and submitting assignments.

GitHub
GitHub is a global company that provides hosting for software development version control using Git. It is a subsidiary of Microsoft, which acquired the company in 2018 for $7.5 billion. It offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its own features. It provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project.

As a student who is working toward entering the tech workforce, GitHub is a great place to host a portfolio. Building up a portfolio (a collection of your work) is essential. Many employers will require it before they consider you for a job. Take the time you need to produce something that will impress them--it'll really pay off. Portfolios are perfect for self-marketing and allow your work to speak for itself. In many modern interview situations, it is expected that interviewees are able to link to an online repository that highlights their work.

You can take this opportunity to familiarize yourself with the basic functionality of git while also beginning the arduous task of building your portfolio.

Task One: Install Git. If you arrived in this class via CSCI160 and went through the steps to install GitBash (if you are windows user) you already have Git installed

⇒ **https://git-scm.com/downloads**

Task Two: Create a GitHub account: https://github.com/

- It is important that this account looks professional. Do not name your account ***SuperStud*** or ***WaifuVixen***. The account should reflect your real-life name and identity. Remember, this is something you may potentially send employers to view. Put your best foot forward and take it seriously. You are more than welcome to create as many GitHub accounts as you like. **For this class, one reflecting your actual identity is required**
- Once your GitHub account has been created, create a ***private repository*** for this class. It is important that the repo be private as this will prevent undesired access to your work. Please name the repository ***CSCI165_Spring21_LastName***. Replace "LastName" with your actual last name. My repository for this class would be named ***CSCI165_Spring21_Whitener***

  ◦ **https://docs.github.com/en/free-pro-team@latest/github/getting-started-with-github/create-a-repo**

- Once you have completed these steps you will need to add me as a collaborator. This will allow me to be able to ***clone*** your repo to my system and execute ***pulls*** to download and view your work.

  ◦ **https://docs.github.com/en/free-pro-team@latest/github/setting-up-and-managing-your-github-user-account/inviting-collaborators-to-a-personal-repository**

Git Basics
There are many free online resources for learning git, but it only really sinks in if you experiment with it. That is the goal of the workflow in this class. I generally don't like the signifier "for dummies" but it is what it is, and these articles usually are pretty decent. This one is no different.

http://wiki.freegeek.org/index.php/Git_for_dummies#The_basics:_What_you_need_to_know

This Git Tutorial is a good one: https://www.tutorialspoint.com/git/index.htm

Great introductory Video: https://www.youtube.com/watch?v=SWYqp7iY_Tc

Important Terms: The following terms will be used throughout this course. For now, just make sure you have a general understanding.

1. **Repository:** Git repository is just a file location where you are storing all the files related to your project. Let's say if you are developing an application, whatever you are coding, all the different modules for your application will be dumped in your Git repository once you ***commit & push***. When you are working with Git, you have two repositories - Local & Remote.

2. **Local repository:** It is just a file location (folder) residing in your system. All source code and related files will be saved in this location. When you ***commit*** your code, a version/snapshot is created in your local repo.

3. **Remote repository:** A remote repository generally lies somewhere outside your system, on a remote machine. This is important when you are working with multiple people and is the place where everyone will be sharing their code. A remote repo also serves as a great backup scheme for your work and will make it available to any system with an Internet connection.

4. **Clone:** Git command line utility which is used to target an existing repository and create a clone or copy of the target repository.

Saving Changes: commit, add, diff, status

When working in Git, or other version control systems, the concept of "saving" is a more nuanced process than saving in a word processor or other traditional file editing applications. The traditional software expression of "saving" is synonymous with the Git term "committing". A ***commit*** is the Git equivalent of a "save". Traditional saving should be thought of as a local file system operation that is used to overwrite an existing file or write a new file. Alternatively, Git committing is an operation that acts upon a collection of files and directories.

Git commits are captured and built up locally, then pushed to a remote server as needed using the **git push** command.

The commands: **git add, git status,** and **git commit** are all used in combination to save a snapshot of a Git project's current state.

6. **Add:** The ***git add*** command adds a change in the working directory to the ***staging area***. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded

until you run git commit. In conjunction with these commands, you'll also need git status to view the state of the working directory and the staging area.

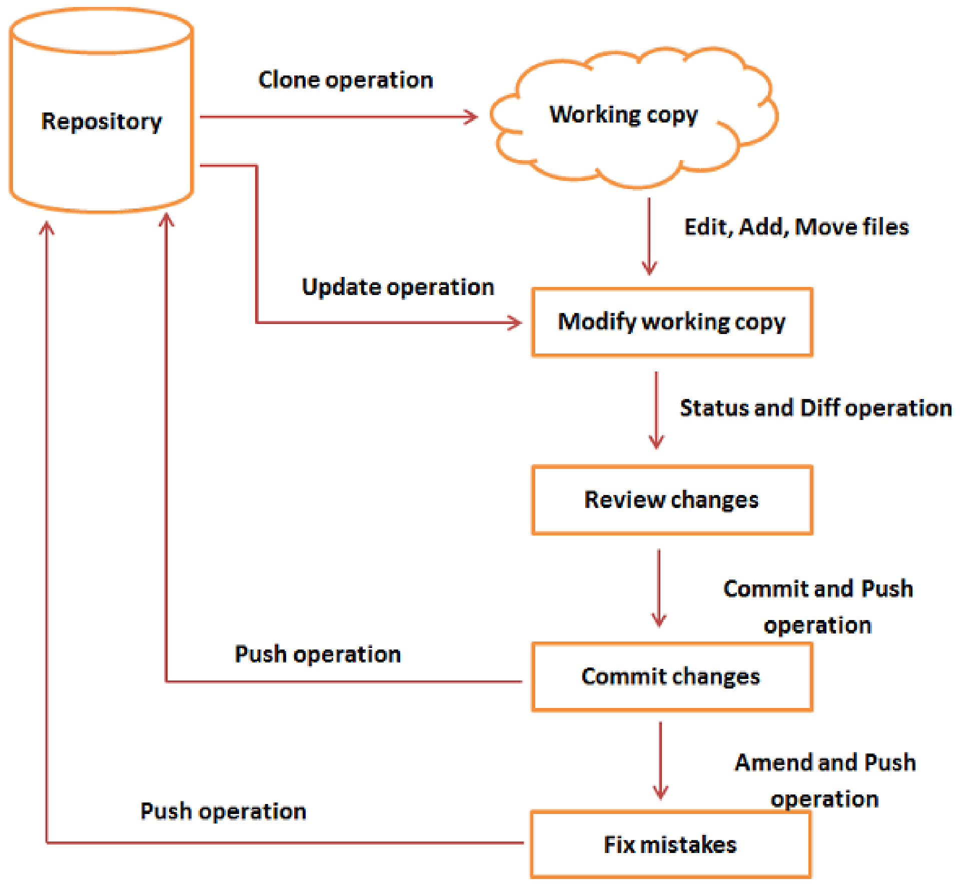
7. **Commit:** Adds the latest changes to [part of] the source code to the repository, making these changes part of the ***head revision*** of the repository. The ***git commit*** command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as "safe" versions of a project—Git will never change them unless you explicitly ask it to. Prior to the execution of git commit, the git add command is used to promote or 'stage' changes to the project that will be stored in a commit. These two commands git commit, and git add are two of the most frequently used.

The **git add** and **git commit** commands compose the fundamental Git workflow. These are the two commands that every Git user needs to understand. They are the means to record versions of a project into the repository's history.

The primary function of the **git add** command, is to promote pending changes in the working directory, to the **git staging area**. The staging area is one of Git's more unique features. It helps to think of it as a buffer between the working directory and the project history. The staging area is considered one of the "three trees" of Git, along with, the working directory, and the commit history.

8. **Push:** The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to ***git pull***, but whereas pulling imports commits to local branches, pushing exports commits to remote branches. Remote branches are configured using the git remote command. Pushing has the potential to overwrite changes, caution should be taken when pushing. These issues are discussed below.

9. **Pull:** The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration workflows. The git pull command is actually a combination of two other commands, git fetch followed by git merge. In the first stage of operation git pull will execute a git fetch scoped to the local branch that HEAD is pointed at. Once the content is downloaded, git pull will enter a merge workflow. A new merge commit will be created, and HEAD updated to point at the new commit.



Your Workflow for This Class

1. I will be pushing all readings, assignments and code to this repository:

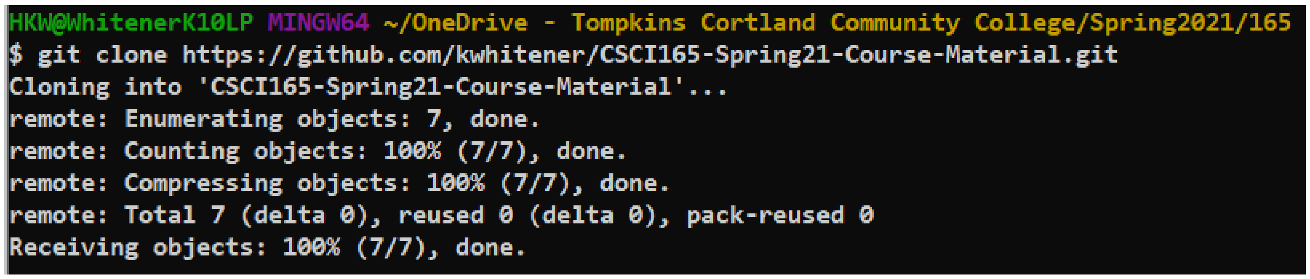
    1. https://github.com/kwhitener/CSCI165-Spring21-Course-Material.git.
    2. You are required to interact with this repository **via the command line** and you will have to submit screen shots proving your compliance to this directive. Trust me, it is for your development as a programmer and I will help you become comfortable with this process.

2. **Task Three:** This will only need to be done once. ***Clone*** the repository to your system. On your machine, open a terminal session. Navigate to a location in which you will be saving all of your course related files.

    From the terminal execute the following:

    ```
    git clone https://github.com/kwhitener/CSCI165-Spring21-Course-Material.git
    ```

    You should see result similar to the following (depending on the status of the repo when you clone)



```
HKW@WhitenerK10LP MINGW64 ~/OneDrive - Tompkins Cortland Community College/Spring2021/165
$ git clone https://github.com/kwhitener/CSCI165-Spring21-Course-Material.git
Cloning into 'CSCI165-Spring21-Course-Material'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
```

This will clone the repo ***to the directory from which you execute this command***. The result of this command will be a local copy of my remote repository folder. This will only need to be done once. As more materials are pushed to the online repo, you will be ***pulling*** those new files . . . not cloning the entire repo again. The clone process simply establishes your local repository. Each week a new folder of course material will be pushed to this repository. You will begin each week by opening a terminal, navigating into the **CSCI165-Spring21-Course-Material** folder and executing **git pull.** This will fetch and merge any new files I have uploaded into your local repository.

3. **Task Four:** Clone your personal git repository to your system. This is how you will be submitting your work. Repeating the process described above, clone your remote git repository to your system. ***You will now have two repos***

   1. **CSCI165-Spring21-CourseMaterial:** You will execute ***pulls*** from here to get new material each week
   2. **CSCI165_Spring21_LastName:** This is your private repository. You will be submitting your work here. This will require you to ***stage changes*** using ***git add*** and then performing a ***commit*** and a ***push.*** Those steps will upload your new files to your remote git repository hosted on GitHub. I will then clone your repo and execute a ***pull*** whenever I go to grade your work.