Getting Started with Git CLI

This tutorial is meant to help you get started with Git’s Command-Line Interface (CLI) and using Git with GitHub. Before you start, make sure you have a GitHub account and have created a repository on GitHub. Also, make sure you have installed the Git CLI, which is normally done on UNIX systems (e.g., macOS and Linux) with:

|  |
| --- |
| $ sudo <package manager> install git |

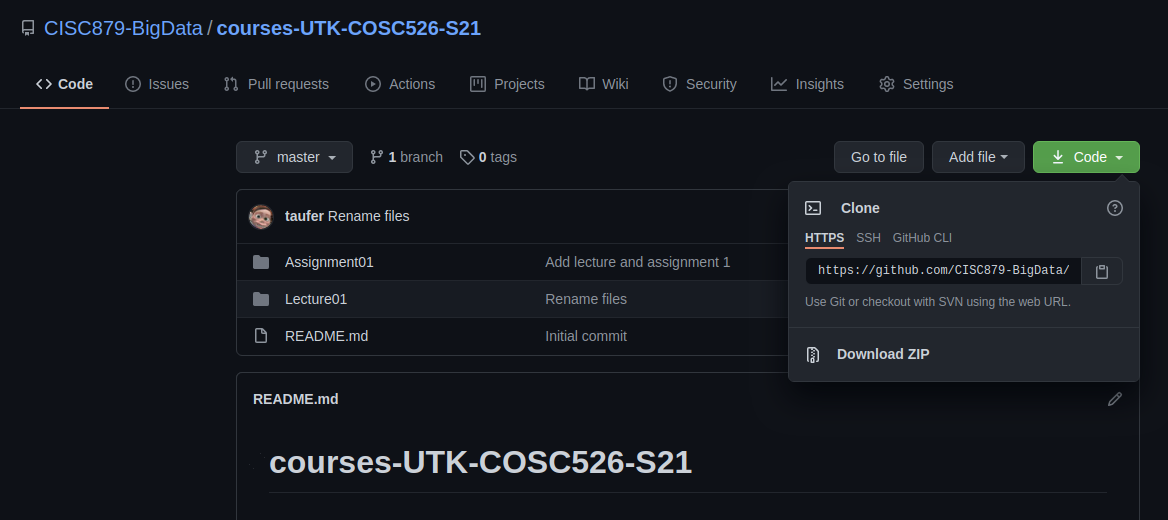
# Basic Git CLI Workflow

When working with a single repository, there is a basic workflow that most people follow. This workflow consists of 3 steps:

1. Cloning the GitHub repository
2. Adding changes to your local repository and pushing them to GitHub
3. Fetching changes from GitHub that other people made

## Cloning the Course and your Personal GitHub Repositories

The first step when using the Git CLI is to “clone” the GitHub repository. Cloning is a process in which you create a local copy of the GitHub repository. After cloning, you will make changes in your local repository before sending them to the repository on GitHub. To clone a GitHub repository, you will first need to get a cloning URL. You can get this URL from the GitHub webpage for the repository you want to clone. As shown below, the repository’s webpage has a green “Code” button (used to say “Clone”). When you click that button, a small pop-up will appear with three options and a URL. For basic use, it is recommended that you select the HTTPS option. The SSH and GitHub CLI options are for more advanced use. The URL below these options is the cloning URL for the repository.



Once you have this URL, open your terminal and go to the directory that you want to be the parent of your local repository (i.e., if you want your repository to be ~/courses/courses-UTK-COSC526-S21, go to ~/courses). Once there, run the following command:

|  |
| --- |
| $ git clone https://github.com/CISC879-BigData/courses-UTK-COSC526-S21.git |

If you want the cloned repository to be in a different directory than the repository name, add the desired directory name to the end of the above command. Once this command is finished, a new directory will be created containing your local repository.

## Adding and Pushing Local Changes

Once you’ve cloned your repository, you’ll want to make changes to its contents. You do this by simply editing the

# Managing Synched GitHub Repositories

There are two primary ways to manage multiple synchronized GitHub repositories from your local device:

1. Using Multiple Directories
2. Using Git Remotes

## Using Multiple Directories

This method for managing multiple synchronized repositories is the simplest, but also causes unnecessary duplication of files and file movement. To use this method, simply clone every repository to a separate directory. For example,

|  |
| --- |
| # Clone repo 1 $ git clone <repo1 url> # Clone repo 2 $ git clone <repo2 url> |

This will create two separate directories: one for repo 1 and another for repo 2. To copy files from one repository to the other, simply copy the file between directories and push to GitHub. For example, if repo 1 is in the “repo1” directory and repo 2 is in the “repo2” directory, you can move the file “test.txt” from repo 1 to repo 2 with:

|  |
| --- |
| # Copy file from repo1 to repo2 $ cp repo1/test.txt repo2 # Enter repo2 $ cd repo2 # Add test.txt to a commit $ git add test.txt # Create the commit $ git commit -m "Your commit message goes here" # Push to GitHub $ git push # Can also use git push origin <branch name> |

## 

## Using Git Remotes

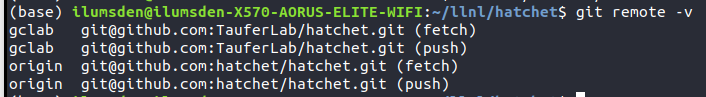
The other way to manage multiple synchronized repositories is to use Git remotes. Remotes can be thought of as links to GitHub (or other distributed Git service) repositories. Most Git commands allow you to specify what remote you want to operate on. For example, if you want to pull from one remote and push to another, you can use the following:

|  |
| --- |
| # Pull from a remote named "origin" $ git pull origin # Can also use git pull origin <branch name> # Push to a remote named "personal" $ git push personal # Can also use git push personal <branch name> |

When you first clone a GitHub repository, a remote will automatically be created with the name “origin”. You can add other remotes using the following:

|  |
| --- |
| # Add the remote "personal" using a GitHub provided link $ git remote add personal https://github.com/hatchet/hatchet.git |

The URL used in this command is the same as the URL you would use to clone a repository. To see all the remotes your local Git repository has, use git remote -v. This will produce an output like the example below:



In this output, the left column shows the names of the remotes, while the right column shows the URLs associated with those remotes. When using git fetch or git pull on a particular remote, the fetch URL will be used. When using git push on a particular remote, the push URL will be used.