ACM Officers Git Workshop

Getting Started

- Go here to install Git onto your machine: https://git-scm.com/downloads
- Create a GitHub account here: https://www.github.com
- Leave a comment here to be added as a collaborator to the repo: https://github.com/UWB-ACM/Git-Gud-Spring2020/issues/1
- For more in-depth installation instructions for your specific OS, go here: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

What is Git?

Git is a tool used to track changes that are made to a system of files in order to integrate changes made to a single file while multiple people are working on them simultaneously. This type of tool is known as a version control system (VCS). There are two types of version control systems: centralized and distributed. Git is distributed, meaning the full file system is maintained on every developer's local machine, including the full version history. Centralized version control systems such as Subversion and Perforce maintain one "central" copy of a file system (oftentimes located on a local server) that each developer commits changes to.

What is it used for?

The use of version control systems such as Git offers a number of advantages, including:
- Allowing developers to make changes to the same file system simultaneously
- Prevents developers from overwriting other developer's changes
- Maintains a history of all versions of a file system

Git can be used to track changes within a variety of different file systems, including code, documents, websites, applications, etc.

Git vs. GitHub

Although Git and GitHub complement each other, there are significant differences between the two. Git is installed locally on the developer's system and maintains a local copy of the file system on each developer's individual computer, while GitHub is an online service that can be used to upload and download changes to a file system that are tracked by Git to the GitHub cloud. GitHub offers some additional features over Git, such as the ability to fork a repository, branch protection, tracking of project-related tasks, online file editing, and third-party tool integration such as CI/CD automated build tools.

Configuring Git

The gitconfig tool is used to set identity variables and customize the Git environment on your system. From a Linux or Mac OS, start by opening a Terminal instance. From a Windows OS, open the Git Bash application that should now be installed on your desktop.

To set your environment variables, run the following commands:

```
git config --global user.name “Crobbi McCrobberson”
git config --global user.email “crobbi@gmail.com”
```

Initializing the Repo

Begin by navigating to and creating the folder where you would like your project files to be stored using the `cd` and `mkdir` commands.

Once you are within the appropriate directory, run the `git init` command.

To check and confirm the status of your newly initialized repository, run `git status`.

Modifying Files and Committing Changes

To make changes to the file from the command line, run the following command:
```
echo "am crob. pls gib fry. caw" >> hello.txt
```

To add the file to the staging area, run `git add hello.txt` (Note: `git add --all` can also be used to quickly add all files that were changed to the staging area instead of individually)

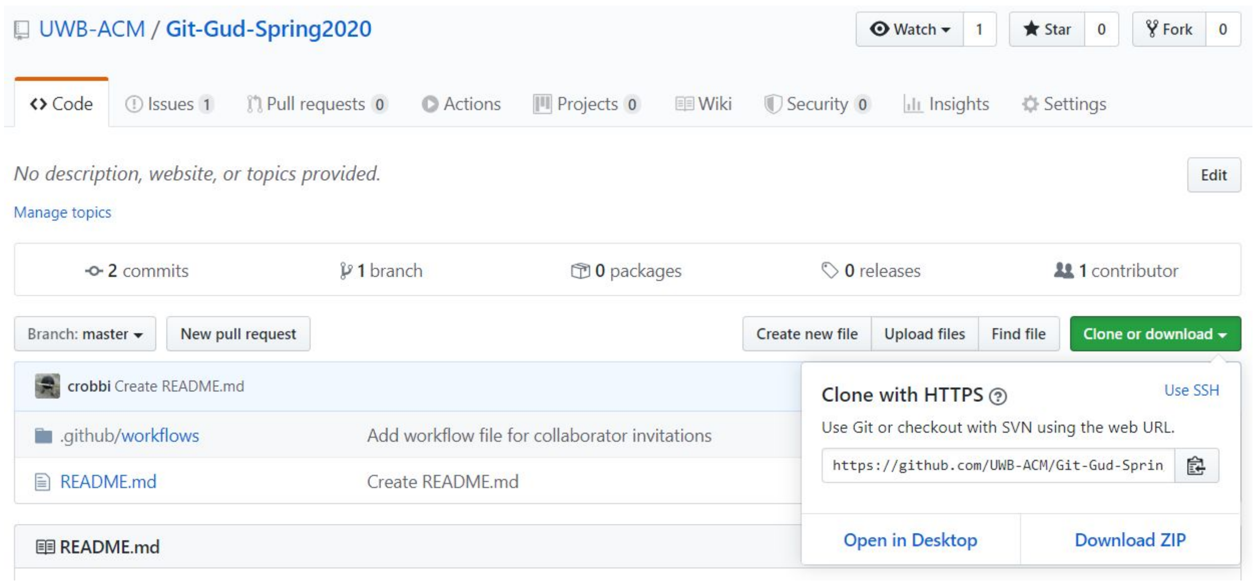
To confirm the changes were successfully made to your file, run `git status`.

To commit all changes currently in the staging area, run: `git commit -m “Add hello.txt”`

NOTE: for the sake of consistency, Git commit messages should always be in the imperative form, should always start with a capital letter, never end with punctuation, and ideally, not exceed 50 characters. Commit often to avoid having to make excessively long commit messages. For more info, see: https://chris.beams.io/posts/git-commit/#seven-rules

Cloning a Remote Repository

To clone an existing remote repository, first locate the GitHub URL(always ends in `.git`):



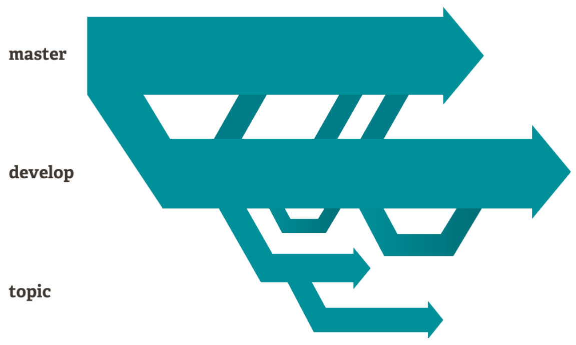
To clone the repository to be used in this workshop, run the following command:

```
git clone https://github.com/UWB-ACM/Git-Gud-Spring2020.git
```

Now that we have cloned the repository, everyone has their own local copy of all the files contained in the remote repo, including a file called `README.md`.

Creating a Branch

By default, repositories contain only one branch, called the master branch. Creating a branch essentially makes a second copy of the master branch; any changes made to the new branch will not be reflected in the original master branch until the developer chooses to merge the new branch back into the master branch. Developers may find it useful to make an additional branch for testing purposes, experimentation, creating new features, etc.



Let's try out a few useful branching commands:

`git branch -a` displays a list of all branches in the repo
`git branch [branch name]` creates a new branch with the given name
`git branch -d [branch name]` deletes the branch with the given name

Committing to a Branch

By default, git will commit changes to the master branch. to create a new branch and switch to it in a single command, use `git checkout -b`

NOTE: In order to switch to switch to an already created branch, run `git checkout [branch name]`

Now, any commits or changes you make to the repo will only be reflected in your new branch until you choose to merge the new branch into the master branch.

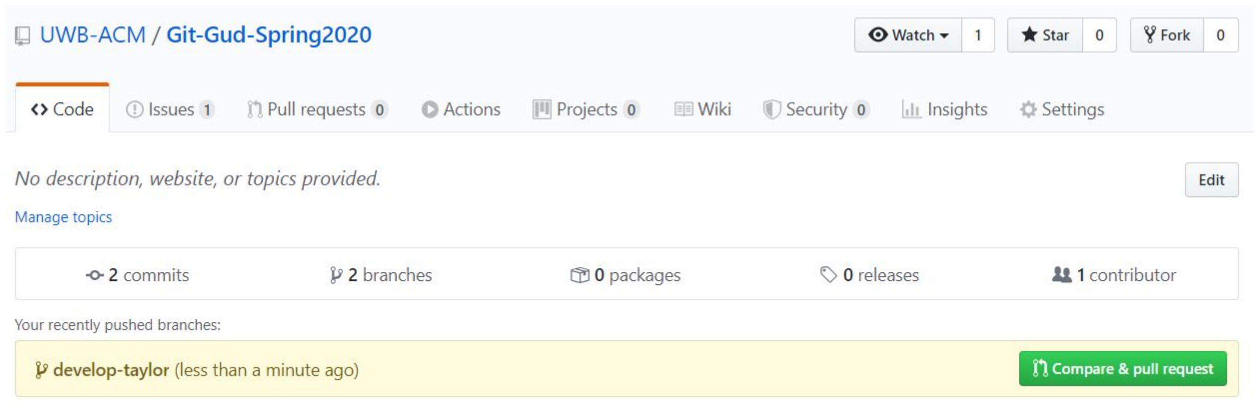
Pushing a Branch to Remote Repo

Once you have made changes to the branch and are ready to push your branch to the remote repo, run the following command: `git push origin [branch name]`

NOTE: this is NOT the same as merging the branch with master; a developer may choose to push the branch before merging for review purposes, if others are working off the same branch and need to pull changes, etc.

Creating a Pull Request

Pull requests (PRs) are used to request a review of your changes from other members of your team. Once the appropriate number of approving reviews is left on your PR, you will have the ability to merge the changes made on your develop/feature branch into the main master branch.



No description, website, or topics provided.

2 commits · 2 branches · 0 packages · 0 releases · 1 contributor

Your recently pushed branches:

develop-taylor (less than a minute ago)