# Intro to Git & GitHub

## Announcements

- Degree Plan Due tonight @ 6PM (NO EXTENSIONS without consulting me first!)
- TigerHacks Next Weekend!

## Co-Enrollment

- New Assignment on Canvas.
- Opportunity to get first dibs on classes.
    - Like 2050, Calculus, Physics, etc.
- Co-enrollment is optional, filling out the form is MANDATORY. (Leave it balnk if you don't want to co-enroll)
- Due October 12 @ 6PM
- Rank your Options 1-6 (1 is highest)
- Must have 4 in same class to co-enroll. (Discuss with your compatriots)
- Can do max of 3 Classes.

## First things first

- You made a GitHub account, right????
- Download the Github Desktop Client
- Check your email and accept invitation

## Git

- Version Control Software
- "Easy" way to store of multiple “copies” or versions of code

- Allows revisions and rollbacks to previous versions
- Works both on your computer and over the internet

## Basic things you can do

- Repository
    - Project Folder
    - Where all the code is stored
- Clone
    - Make a copy of an existing Repository onto your computer
    - You can edit/make changes git clone [url]
- Commit (Like a save)
    - Done locally
    - Commits are saved in a “log”
    - Saves full code file for each commit git commit -a -m "Your Commit Message
- Push & Pull (Both over internet)
    - Push – Send entire commit log to server git push origin master git push
    - Pull – Get entire ommit log stored on server git pull origin master git pull
- Branch
    - Create another copy that can be separate from the original version
    - Typically used for:

        - Collaboration – Each member gets their own branch
        - Concurrency – Maintaining multiple versions of same software
            - (e.g. Windows 8 & 10 or Yosemite and Sierra would be different branches)
    - Main branch is called the “master”
    - Typically the release version or the “working copy”
    - Also might have a “nightly” version with the latest and greatest (unstable)
    - Can branch off of branches
    - Branches may or may not be merged back with original
- Merge
    - Bringing two different versions or branches together
    - Keeps track of line changes, adds, deletes
    - If there’s too big of a change, it might fail and cause a “Merge conflict”
        - These must be resolved manually
- Fork
    - A Fork is like a branch (kind of) but it’s usually not done by the same team that made the original.
    - Not typically merged back with original
    - Much more restrictive
    - Examples: Different Flavors of Linux (Ubuntu, RedHat) are Forks of the pure Linux kernel
- .gitignore

    - Special file in repo that tells git which files to not keep track of.
    - Executables a.out
    - Weird stuff .DS_STORE

# GitHub 

- What's the difference between Git and GitHub?
    - GitHub is (kind of) a social network for programmers that uses Git at the core.
    - Manages commits, merges, etc.
    - Like Cloud storage for code

## Workflow

1. Pull (Sync) git pull
2. Make Code Changes
3. Commit git commit -a -m "Message"
4. Push git push
5. Solve Merge Conflicts (if any)
6. Repeat

# Activity

1. Go to my GitHub account
    - Github.com/holtwashere
    - Follow Me
    - Click on FIG-Sample-Project
2. Clone the Repository
    - Click on Green "Clone or Download" button and select "Open in Desktop"

- The desktop client should open
- Save it wherever (Preferably in a GitHub folder)

3. Make a Change

    - If you have Atom or VSCode Installed, right click on repo name or click on Atom icon in upper right corner.
    - If not, Right click on repo name (left side) and select open in explorer (or Finder)
    - Open main.md in your text editor of choice (brackets, sublime, notepad, etc) Don't use the 1050 server
    - Add your name and save

4. Commit Changes

    - Go to desktop client, make sure you're on the changes tab.
    - Your change should show up
    - Add a commit message and commit to master.

5. Push Changes

    - After commit is made, click Sync in upper right corner
    - If it doesn't work, let me know (You probably didn't send your github username to me!!!)

# How to post stuff from the 1050 Server to Github

Inside your 1050 folder

```
# Initializes the Repository
git init

# Tells git to not track a.out files
echo "*.out" > .gitignore
```

```
# Creates a README file (You can change it later)
echo "# I just made a git repo :)" > README.md

# Tells git to stage all files in current directory for
committing
# You only need to type this if you added a new file
git add .

# Commits (Saves)
# -a everything that you added
# -m "Commit message"
git commit -a -m "Init Repo"

# Links Local Git Repo to GitHub Repo
# Replace Url with your own repo URL
git remote add origin
https://github.com/holtwashere/Test.git

# Sends all local commits to the GitHub server. (Puts
it online)
# --set-upstream tells git where to push in the future
# master is the main branch
# Enter GitHub Username and password after this line
executes
git push --set-upstream origin master
```

After the first time, you can follow the workflow.

```
# Get any changes from GitHub
git pull

# Make Code changes
# Save Changes
```

```
git commit -a -m "Message"

# Send to github
git push
```