GitHub @ TechHive

...

How to use version control for your projects

https://github.com/margaret

Presentation

1. Motivation
2. What is git?
3. Basic git commands (through example)

~ Intermission ~

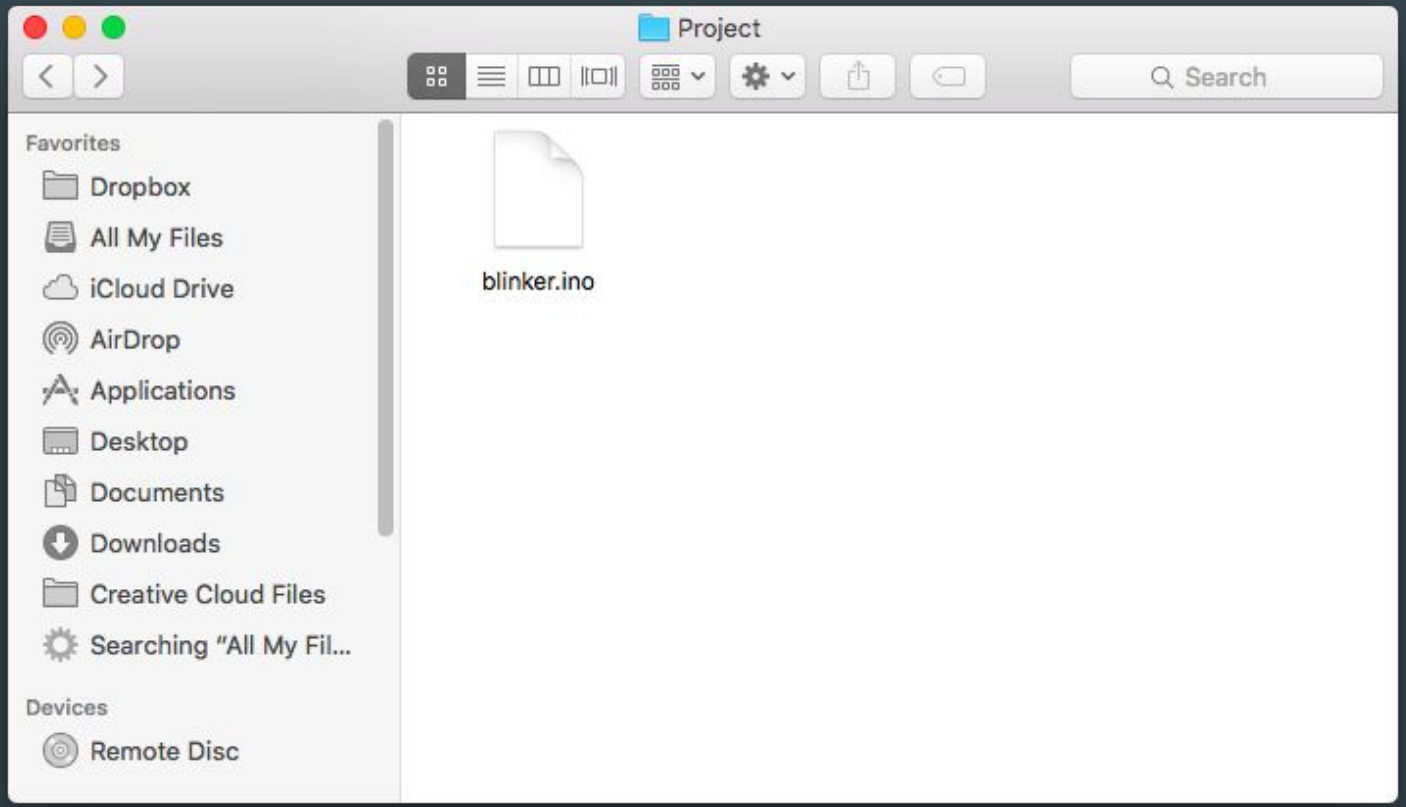
4. Branches
5. LHSTechHive GitHub organization

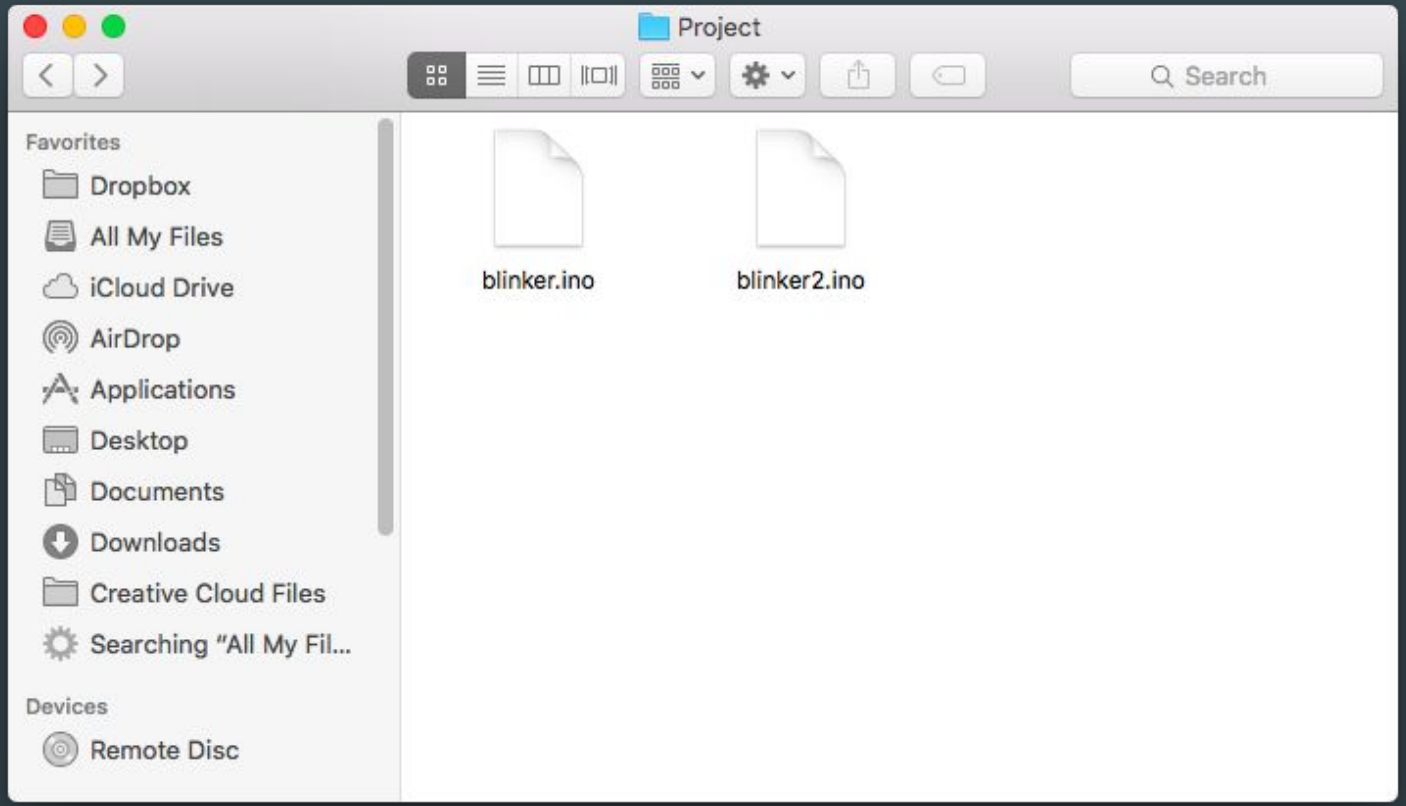
Next time:

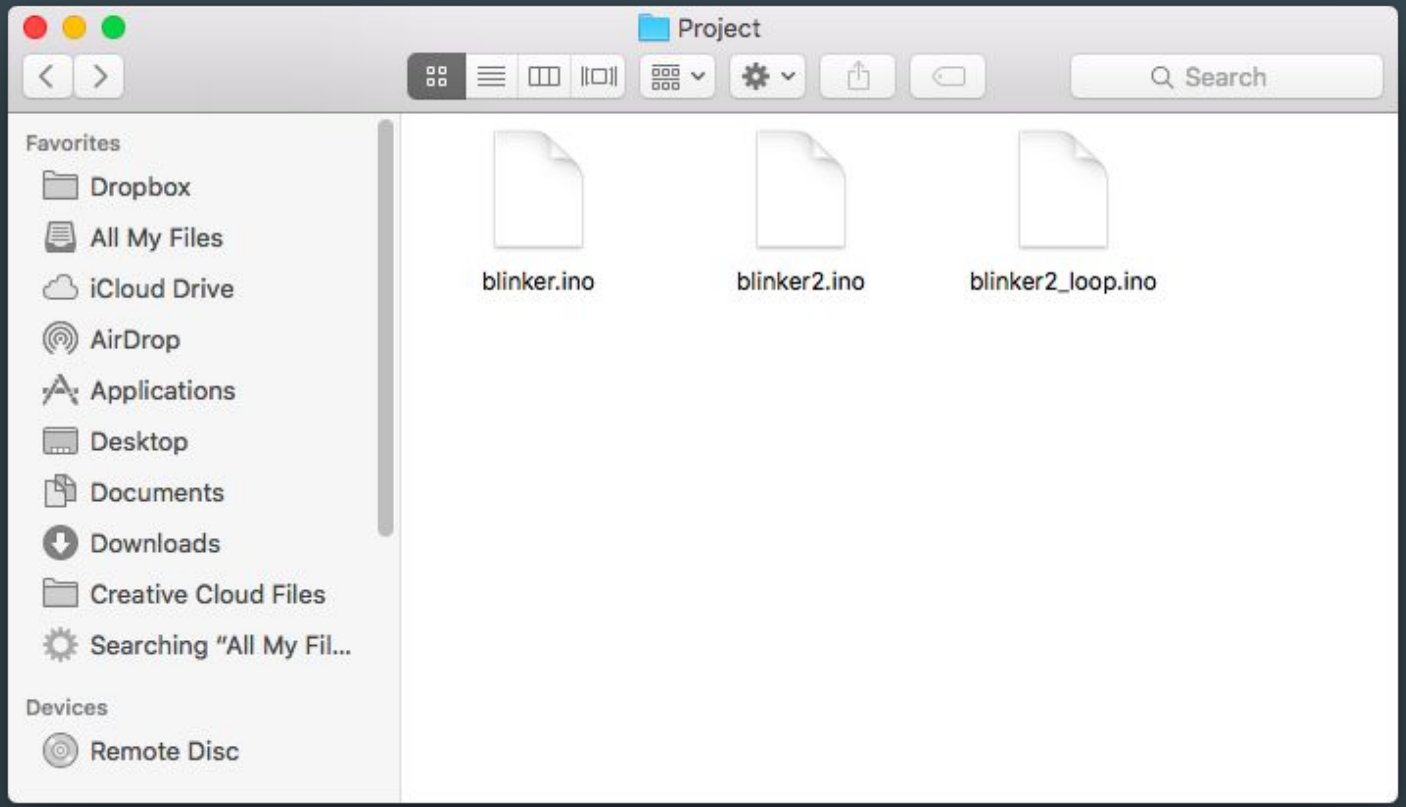
6. Using git and GitHub collaboratively (an exercise)

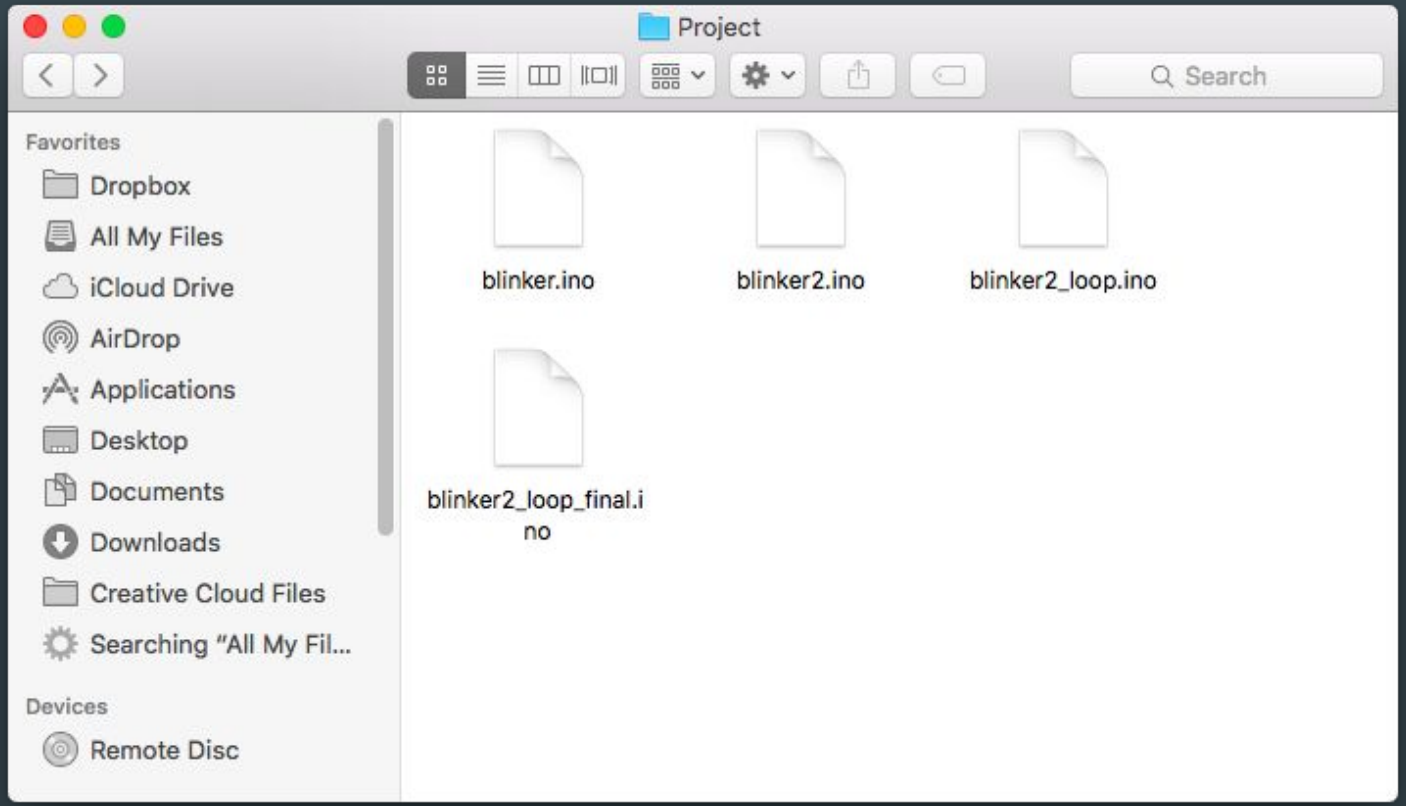
Part 1: Motivation

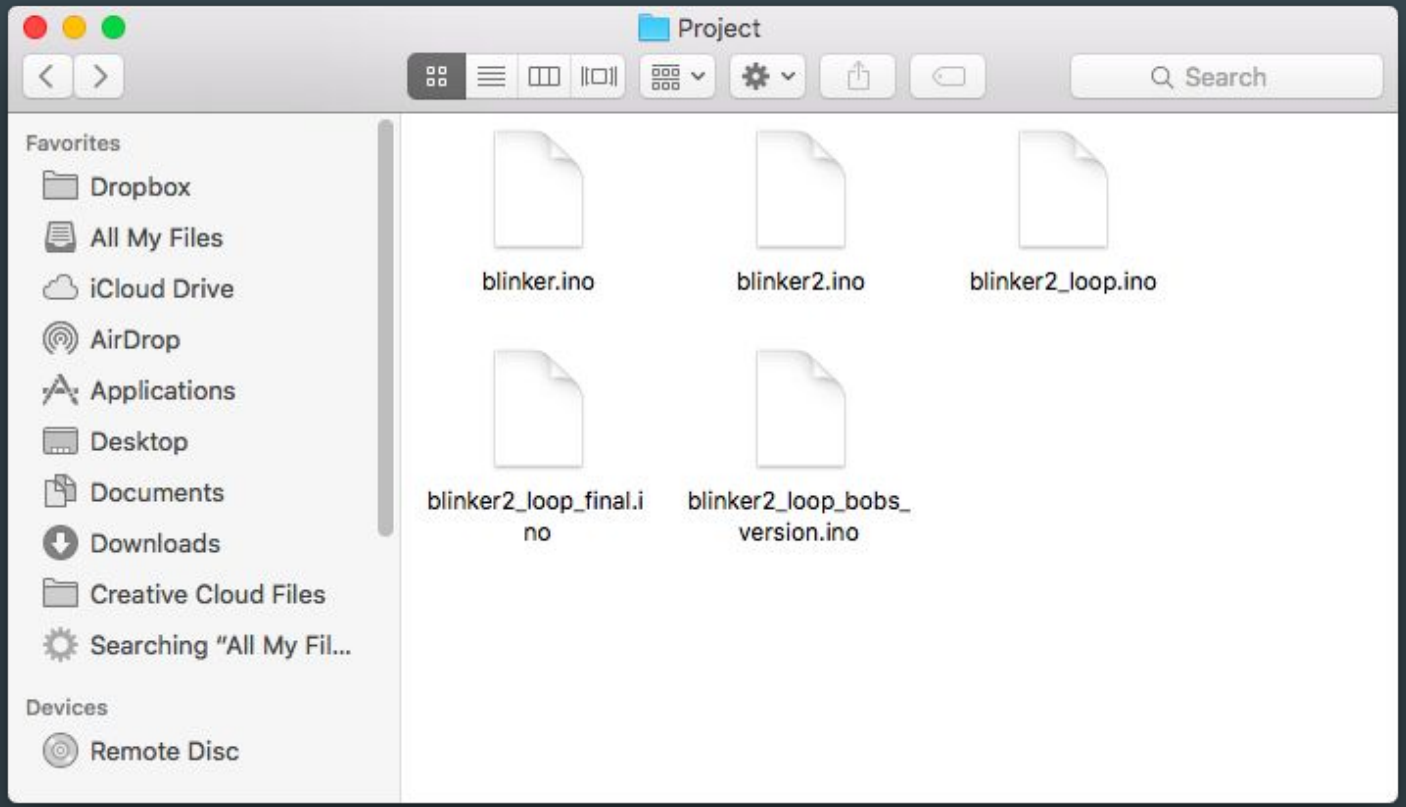
A Scenario

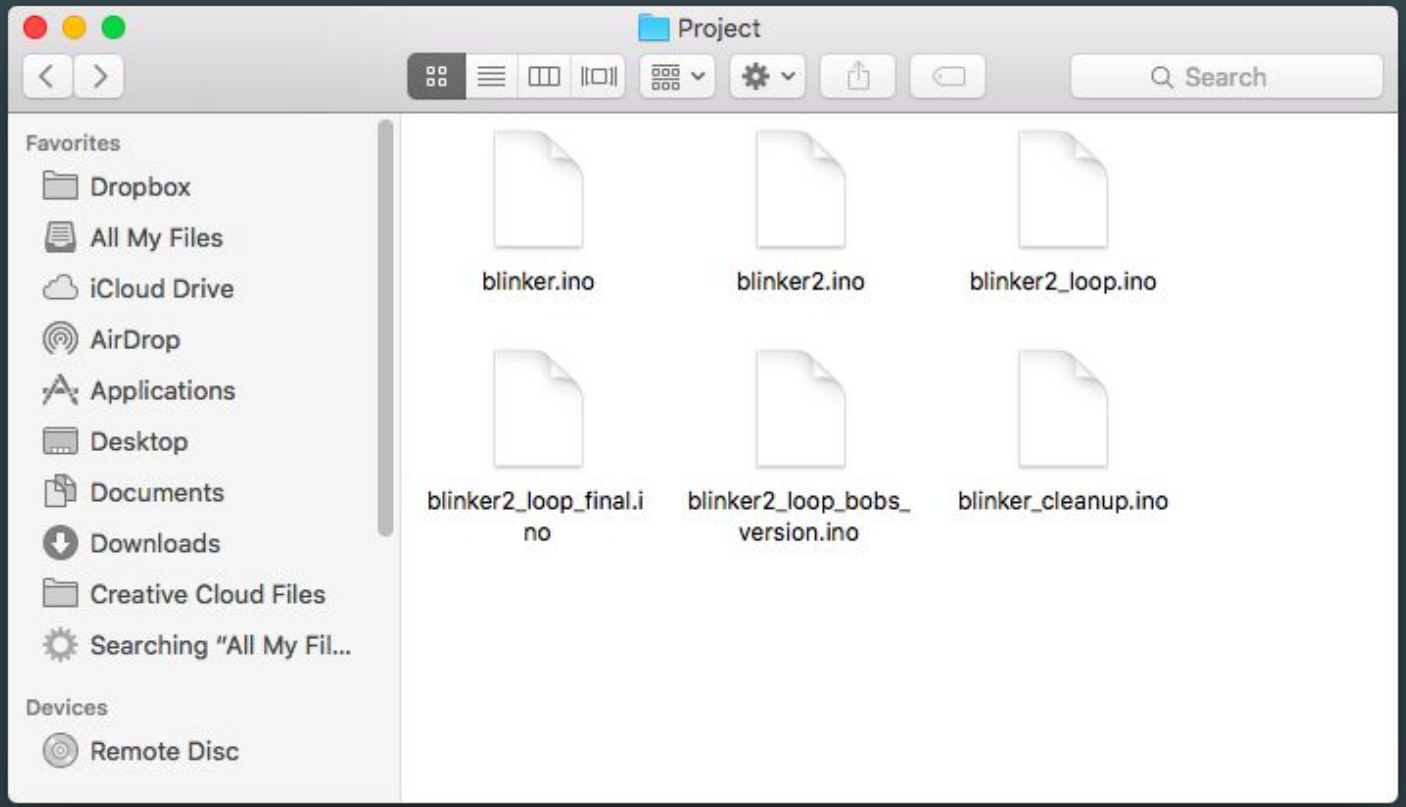

Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino

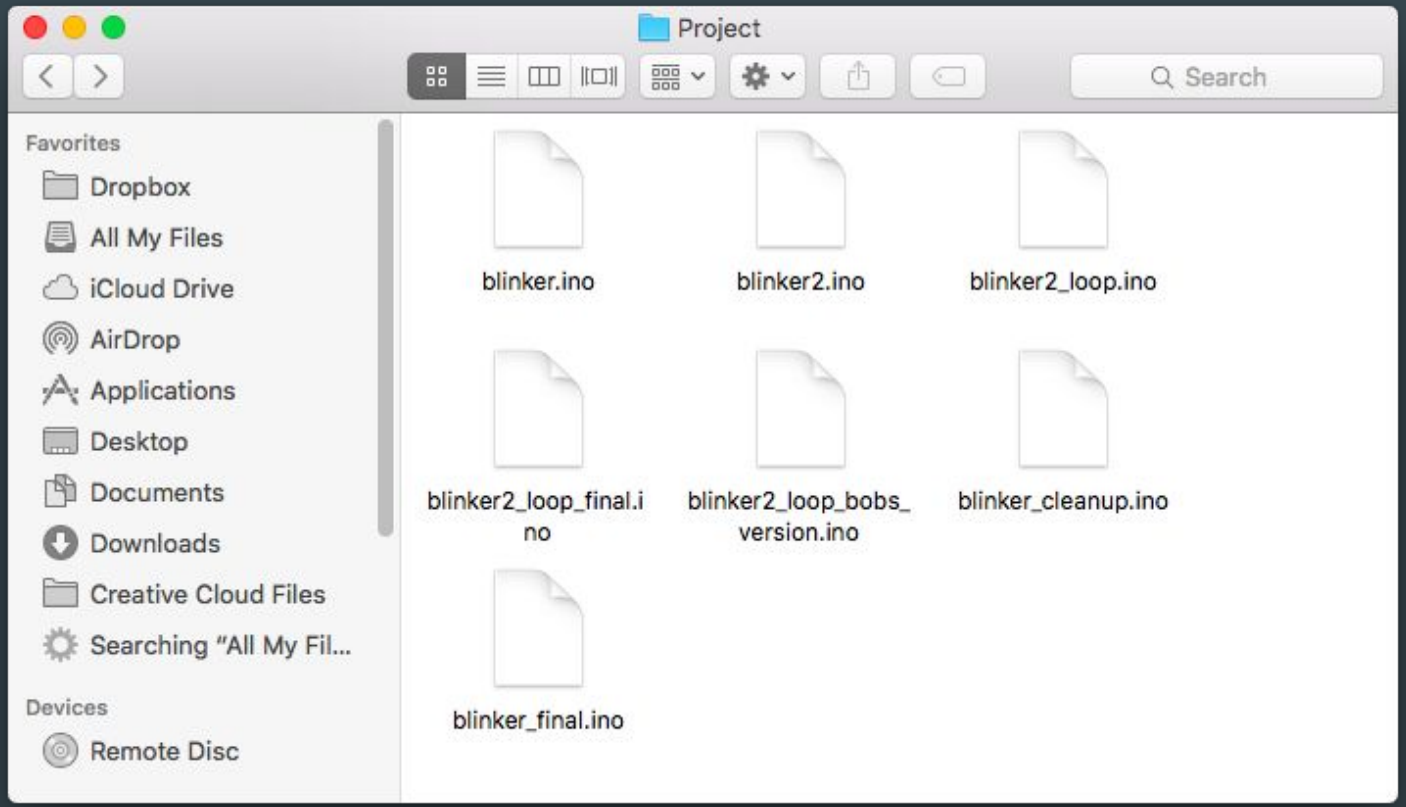

Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino
blinker2.ino

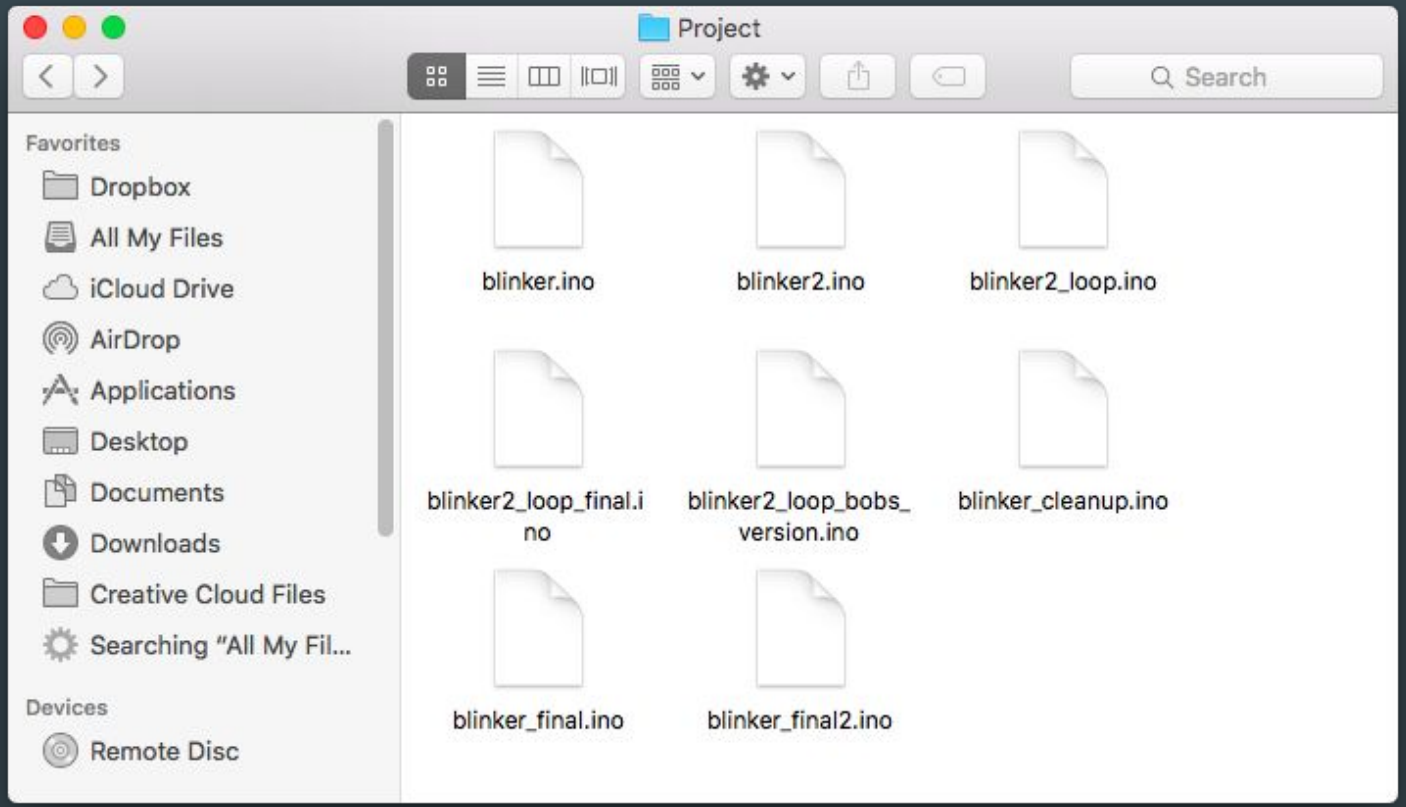

Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino
blinker2.ino
blinker2_loop.ino

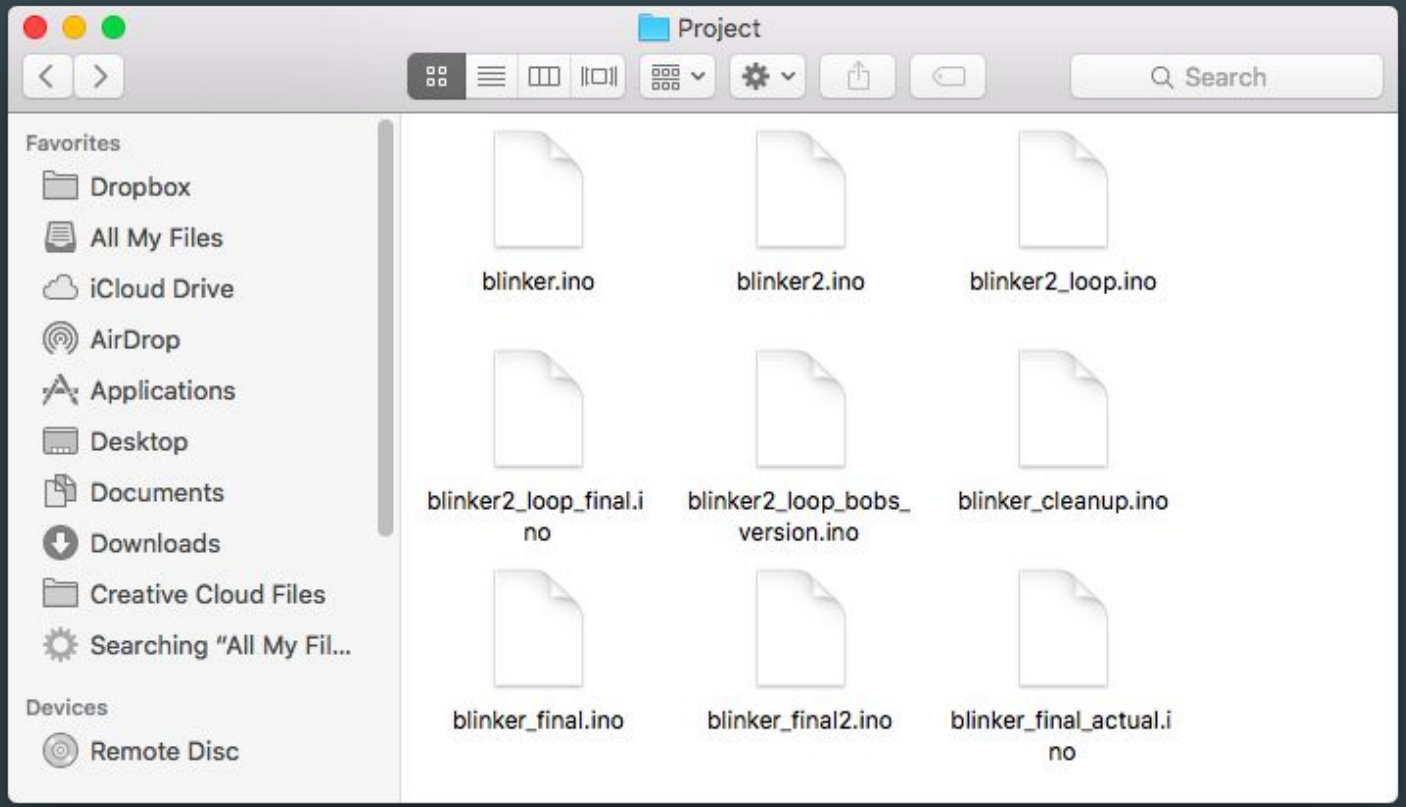

Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino
blinker2.ino
blinker2_loop.ino
blinker2_loop_final.ino

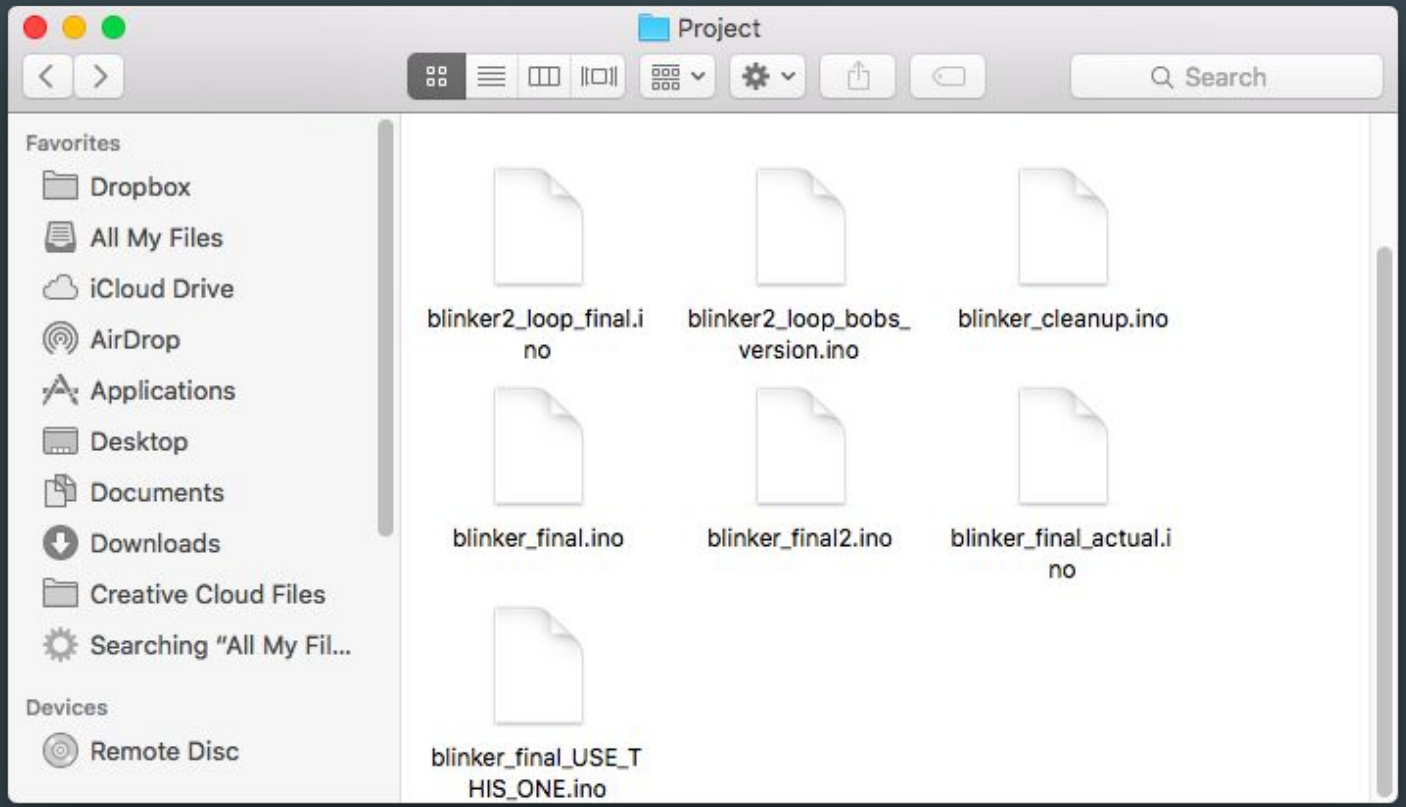

Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino
blinker2.ino
blinker2_loop.ino
blinker2_loop_final.ino
blinker2_loop_bobs_version.ino


Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino
blinker2.ino
blinker2_loop.ino
blinker2_loop_final.ino
blinker2_loop_bobs_version.ino
blinker_cleanup.ino


Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino
blinker2.ino
blinker2_loop.ino
blinker2_loop_final.ino
blinker2_loop_bobs_version.ino
blinker_cleanup.ino
blinker_final.ino


Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino
blinker2.ino
blinker2_loop.ino
blinker2_loop_final.ino
blinker2_loop_bobs_version.ino
blinker_cleanup.ino
blinker_final.ino
blinker_final2.ino


Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker.ino
blinker2.ino
blinker2_loop.ino
blinker2_loop_final.ino
blinker2_loop_bobs_version.ino
blinker_cleanup.ino
blinker_final.ino
blinker_final2.ino
blinker_final_actual.ino


Project
Search
Favorites
Dropbox
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Creative Cloud Files
Searching "All My Fil...
Devices
Remote Disc
blinker2_loop_final.ino
blinker2_loop_bobs_version.ino
blinker_cleanup.ino
blinker_final.ino
blinker_final2.ino
blinker_final_actual.ino
blinker_final_USE_THIS_ONE.ino

VERSION CONTROL

Part 2: What is git?

Dictionary

Enter a word, e.g. "pie"

ver·sion con·trol

noun COMPUTING

the task of keeping a software system consisting of many versions and configurations well organized.

📚 👀 https://www.atlassian.com/git/tutorials/what-is-version-control

Tags

A tag is a keyword or label that categorizes your question with other, similar questions. Using the right tags makes it easier for ot

mercurial

mercurial × 7728

a fast, open-source DVCS (Distributed Version Control System).

5 asked this week, 13 this month

mercurial-hook × 135

a mechanism to customize and extend functionalities of the Mercurial DVCS.

7 asked this year

mercurial-subrepos × 122

Questions related to the subrepository feature in the Mercurial distributed version control system.

8 asked this year

Tags

A tag is a keyword or label that categorizes your question with other, similar questions. Using the right tags makes it easier for o

svn

svn × 25399

for questions about SVN (Subversion), a centralized open-source revision control system distributed under the Apache License.

20 asked this week, 80 this month

tortoisesvn × 4511

a Subversion client for Windows, implemented as a shell extension.

8 asked this week, 25 this month

git-svn × 1761

a bidirectional bridge between git and Subversion allowing to use a remote Subversion repository as a local Git

7 asked this month, 97 this year

Questions Developer Jobs Tags Users Search...

Tags

A tag is a keyword or label that categorizes your question with other, similar questions. Using the right tags makes it easier for othe

git

git × 99140

an open source distributed version control system (DVCS). Use this tag for questions related to Git usage and workflows. Do not

29 asked today, 275 this week

github × 28692

a web-based hosting service for software development projects that use Git for version control. Use this tag for questions specific to

8 asked today, 97 this week

gitlab × 5116

an Open Source Git repository manager with issue tracking and wiki as well as continuous integration features. Use this tag for

58 asked this week, 188 this month

How does git work?

- Git stores lists of the changes you make to your project.
  - It doesn't have to store a whole copy every time you make a change, which is efficient.
  - Internally, it does a lot of really clever things with tree data structures.
- As you work, instead of using "save" to overwrite an entire file, you "commit" the *changes* you make as you make them.
- Other commands coordinate keeping track of these commits across everyone's computers.

📚 👀 https://www.atlassian.com/git/tutorials/what-is-git

Cool. So how do you use it?

⚠ □ Before we get started

- I am going to go *really fast*.
- 🙋 If you have questions, raise your hand and ask!
- 📥 I will send this presentation out after the meeting.
- ⁉ □ Version control can get really confusing. This is normal. I guarantee that any software developer who uses git professionally still has to google how to do things in git on a regular basis.
- The goal of this presentation is to make you aware of concepts in the general workflow of using git so that even if you can't remember exactly what to type to accomplish X, you know enough to google around and find an answer.

Part 3: Git basics

Command line

- The terminal (or powershell on Windows) is a program that allows you to interact with your computer using text commands rather than clicking on things.
- I will be showing commands that you type in the terminal of your computer.
- If you prefer, you can download a GUI to interact with git. There are a whole bunch of different ones, but they all will use these same concepts. These can look quite different from each other, so to keep things streamlined, I will be showing all the examples here on the Command Line Interface (CLI) for git.
- Find the GUI that is right for you here ➡ https://git-scm.com/downloads/guis/
  - https://desktop.github.com/ is a GUI that works with GitHub
  - Your fancy IDE might have a plugin for git.

Command line

- What directory am I in exactly?
  - `pwd` “print working directory (directory you are currently in)”
- What files are in this directory?
  - `ls` “list files”
  - `ls -a` “list all files, including hidden files (dotfiles)”
- Changing directories
  - `cd ..` “change directory to the parent directory”
  - `cd /exact/path/to/directory` “change directory to /exact/path/to/directory”
  - `cd ~/path/relative/to/home/directory`
  - `cd -` “change directory to previous directory”

Get started: create a repo with `git init`

`git init` tells git to start tracking the current directory as a project

```
08:00:24 margaret:~/Projects/TechHive $ mkdir GitDemo

08:00:46 margaret:~/Projects/TechHive $ cd GitDemo
```

Get started: create a repo with `git init`

`git init` tells git to start tracking the current directory as a project

```
08:00:53 margaret:~/Projects/TechHive/GitDemo $ git init

Initialized empty Git repository in /Users/margaret/Projects/TechHive/GitDemo/.git/

08:01:03 margaret:~/Projects/TechHive/GitDemo $ ls -a

.    ..    .git
```

What happened? `git status`, `git diff`, `git log`

`git status` print the state that git knows about

`git diff` show how are the files different from the last time you committed

`git log` show a list of commits (to exit, press ‘q’)

These will be used throughout the following example.

What happened? `git status`, `git diff`, `git log`

```
08:05:25 margaret:~/Projects/TechHive/GitDemo $ touch README.md
```

The above creates the empty file. Before the next line, we add text to README.md using a text editor

```
08:06:07 margaret:~/Projects/TechHive/GitDemo $ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README.md

nothing added to commit but untracked files present (use "git add" to track)
```

Save changes: `git add` and `git commit`

```
08:10:50 margaret:~/Projects/TechHive/GitDemo $ git add README.md
08:10:56 margaret:~/Projects/TechHive/GitDemo $ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   README.md
```

Save changes: `git add` and `git commit`

```
08:11:08 margaret:~/Projects/TechHive/GitDemo $ git commit -m
"first commit - README"
[master (root-commit) e92049b] first commit - README
 1 file changed, 16 insertions(+)
 create mode 100644 README.md

08:11:18 margaret:~/Projects/TechHive/GitDemo $ git status
On branch master
nothing to commit, working directory clean
```

Save changes: `git add` and `git commit`

```
08:11:35 margaret:~/Projects/TechHive/GitDemo $ git log

commit e92049b81a2d98ca4006eefae858b5868697e2d0

Author: Margaret Sy <margaretsy1016@gmail.com>

Date:   Sat May 26 20:11:18 2018 -0700

    first commit - README
```

Each commit has an associated hash (usually referred to as SHA, short for Secure Hash Algorithm). This is a unique ID that refers to the commit.

Save changes: `git add` and `git commit`

Why are there two steps to save your changes? Wouldn't it be simpler if git commit automatically added all the changes in your working directory by default?

Having `add` as a separate step allows you to pick what commit more granularly. You can have changes in a bunch of files and only commit a subset of them. You can also do things like add only some changes within a single file (`git add -p`)

A quick note on `vim` and default text editors

If you commit without using the `-m` flag, git will open up a text editor in the terminal for you to type the commit message in. Unless you have set it otherwise, it is probably an editor called `vi` or `vim`. It is an ancient and powerful text editor that has a very high learning curve and is probably *not* what you want to start out using.

Press the `esc` key, then type a colon, then press `q` to exit. This will abort the commit.

You can set the editor git uses for editing. Follow the instructions here:

➡ https://help.github.com/articles/associating-text-editors-with-git/

Undoing things: `git revert` and `git reset`

```
08:12:52 margaret:~/Projects/TechHive/GitDemo $ git diff
diff --git a/README.md b/README.md
index 6eec9e3..b6b1006 100644
--- a/README.md
+++ b/README.md
@@ -14,3 +14,5 @@ To use this repository, you must
 ## Contact

 [Margaret](https://github.com/margaret) is the maintainer of this repo. Please
create an issue if you have suggestions for it.
+
+THIS IS A TERRIBLE MISTAKE
```

Here, we have added a terrible mistake to our README.md file.

Undoing Things: `git revert` and `git reset`

```
08:13:42 margaret:~/Projects/TechHive/GitDemo $ git add README.md

08:13:49 margaret:~/Projects/TechHive/GitDemo $ git commit -m "a
terrible mistake"

[master cfa5d3a] a terrible mistake

 1 file changed, 2 insertions(+)
```

We accidentally committed the terrible mistake!

Undoing Things: `git revert` and `git reset`

```
08:14:07 margaret:~/Projects/TechHive/GitDemo $ git log
commit cfa5d3a516b23f198119fa92020bc7ca4ccd8c25
Author: Margaret Sy <margaretsy1016@gmail.com>
Date:   Sat May 26 20:14:04 2018 -0700

     a terrible mistake

commit e92049b81a2d98ca4006eefae858b5868697e2d0
Author: Margaret Sy <margaretsy1016@gmail.com>
Date:   Sat May 26 20:11:18 2018 -0700

     first commit - README
```

Undoing Things: `git revert` and `git reset`

```
08:14:08 margaret:~/Projects/TechHive/GitDemo $ git revert
cfa5d3a516b23f198119fa92020bc7ca4ccd8c25

[master b2662e4] Revert "a terrible mistake"

 1 file changed, 2 deletions(-)
```

The SHA of the commit we want to undo

Undoing Things: `git revert` and `git reset`

```
08:14:50 margaret:~/Projects/TechHive/GitDemo $ git log
commit b2662e4d4a61075e90cf9b4c0a5d2a5fb2c6041b
Author: Margaret Sy <margaretsy1016@gmail.com>
Date:   Sat May 26 20:14:45 2018 -0700

      Revert "a terrible mistake"

      This reverts commit cfa5d3a516b23f198119fa92020bc7ca4ccd8c25.

commit cfa5d3a516b23f198119fa92020bc7ca4ccd8c25
Author: Margaret Sy <margaretsy1016@gmail.com>
Date:   Sat May 26 20:14:04 2018 -0700

      a terrible mistake

commit e92049b81a2d98ca4006eefae858b5868697e2d0
Author: Margaret Sy <margaretsy1016@gmail.com>
Date:   Sat May 26 20:11:18 2018 -0700

      first commit - README
```

The revert adds a commit (with an autogenerated commit message).

It reverses all the changes in the commit that you are reverting.

Undoing Things: `git revert` and `git reset`

```
08:15:20 margaret:~/Projects/TechHive/GitDemo $ git reset
e92049b81a2d98ca4006eefae858b5868697e2d0
08:15:24 margaret:~/Projects/TechHive/GitDemo $ git log
commit e92049b81a2d98ca4006eefae858b5868697e2d0
Author: Margaret Sy <margaretsy1016@gmail.com>
Date:   Sat May 26 20:11:18 2018 -0700

    first commit - README
```

The SHA of the most recent commit we want to keep.

Or instead of reverting, you can reset *to* a commit. This removes every commit after that commit.

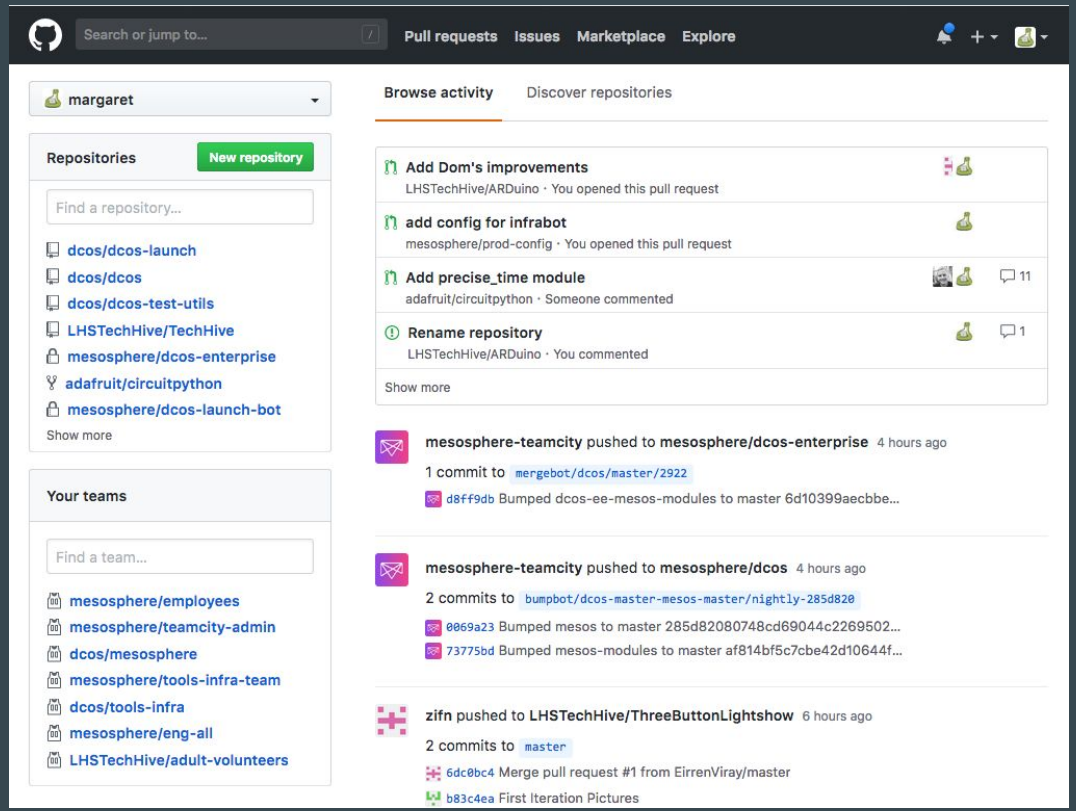
Undoing Things: `git revert` and `git reset`

- It is safer to revert a commit than to reset to a previous one because it is more truthful in how it tracks the changes.
- You can revert a commit even if you've made a bunch of commits after it.

Working with remotes: git and GitHub

- To use git with other people, there needs to be a server somewhere that people can push (upload) their changes to so that other people can pull (download) them, since someone else can't get your changes from your computer directly.
  - Similar to how you can share files with people through Google Drive, but you put the files on Drive which is hosted on a server by Google, and the other person gets the new files from that server rather than your computer
- GitHub is a website that provides this service.
  - There are other websites or services that you can use to work with git. Two other services are BitBucket and GitLab. GitHub is the most popular.

Working with remotes: git and GitHub



Working with remotes: `git remote` and `git clone`

A remote is a pointer to hosted repository. We are going have our remote hosted on GitHub.

You can have more than one remote for a repository. In this example we are just working with a single remote.

Working with remotes: `git remote` and `git clone`

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner **Repository name**

LHSTechHive / GitDemo

Great repository names are short and memorable. Need inspiration? How about **reimagined-goggles**.

Description (optional)

Introductory exercise for working with git and GitHub

Public
Anyone can see this repository. You choose who can commit.

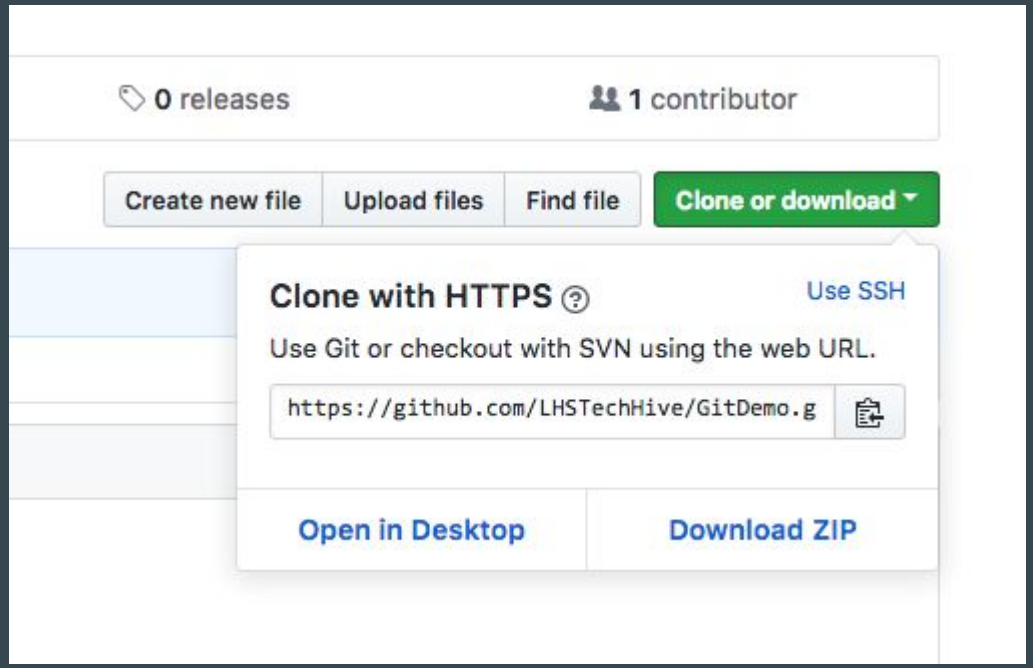
Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Create repository

Working with remotes: `git remote` and `git clone`



Working with remotes: `git remote` and `git clone`

```
08:15:43 margaret:~/Projects/TechHive/GitDemo $ git remote add origin
https://github.com/LHSTechHive/GitDemo.git
08:20:53 margaret:~/Projects/TechHive/GitDemo $ git remote -v
origin    https://github.com/LHSTechHive/GitDemo.git
 (fetch)
origin    https://github.com/LHSTechHive/GitDemo.git
 (push)
08:20:55 margaret:~/Projects/TechHive/GitDemo $ git branch
* master
```

The name of this remote. It can be anything, but 'origin' is standard

Working with remotes: `git remote` and `git clone`

If you are starting from an existing repository, instead of `git init` you would do:

```
08:15:43 margaret:~/Projects/TechHive $ git clone
https://github.com/LHSTechHive/GitDemo.git
08:20:53 margaret:~/Projects/TechHive $ cd GitDemo
```

Cloning sets the remote automatically.

Syncing: `git push` and `git pull`

```
08:22:27 margaret:~/Projects/TechHive/GitDemo $ git push origin master

Counting objects: 3, done.

Delta compression using up to 4 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 519 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/LHSTechHive/GitDemo.git

 * [new branch]    master -> master
```

The name of the remote to push to

The name of the branch to push to

Working with remotes: `git remote` and `git clone`

LHSTechHive / GitDemo

Unwatch 2 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Insights | Settings

Introductory exercise for working with git and GitHub

Edit

Add topics

1 commit | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

margaret first commit - README — Latest commit e92049b 11 minutes ago

README.md — first commit - README — 11 minutes ago

README.md

Git Demo

This repository is for practicing using git with GitHub.

Participants will fork the repository and make a pull request containing a text file introducing themselves (e.g. "Hello, I am ").

Setup

To use this repository, you must

1. Have a GitHub account
2. Have git installed on your computer

Contact

Margaret is the maintainer of this repo. Please create an issue if you have suggestions for it.

Your project is now hosted on GitHub

Pull new changes from the remote

Say someone else has pushed changes to that remote. How do you get those changes down to your computer (so that you are working on the most recent version of the project)?

```
08:23:48 margaret:~/Projects/TechHive/GitDemo $ git pull origin
master
```

Other stuff: .git, and .gitignore, `--help`

- `.git` is a directory that will be created by git in the directory that has your project. This is where it stores all the things it keeps track of. You should not need to modify it. Deleting it will remove git tracking from the project.
- `.gitignore` is a file you can create in your project directory to tell git to ignore certain files. For example, tell it to ignore compiled files by putting `*.o` in the file.
- All git commands take an argument `--help` which will show the manual for the command. To exit the help screen, press 'q'. E.g. typing `git checkout --help` will show all the info about the help command.

Core Commands

`git init` Start tracking a project

`git remote add <url-of-repo>` Set a remote host for your project

`git clone` Get an existing project from a remote to your computer

`git add` / `git commit` Save your changes

`git revert` / `git reset` Undo previous changes

`git push` Upload your changes to the remote host

`git pull` Download latest changes from the remote host

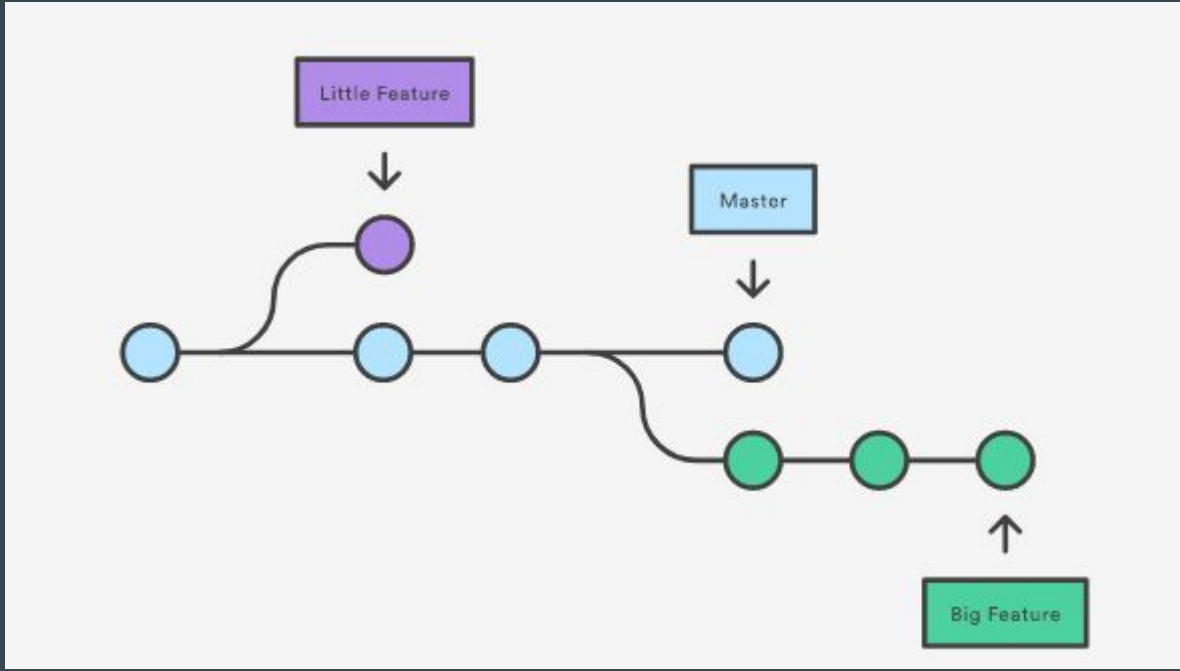
➡ ➡ https://www.atlassian.com/git/tutorials/atlassian-git-cheatsheet ⬅ ⬅

Questions


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

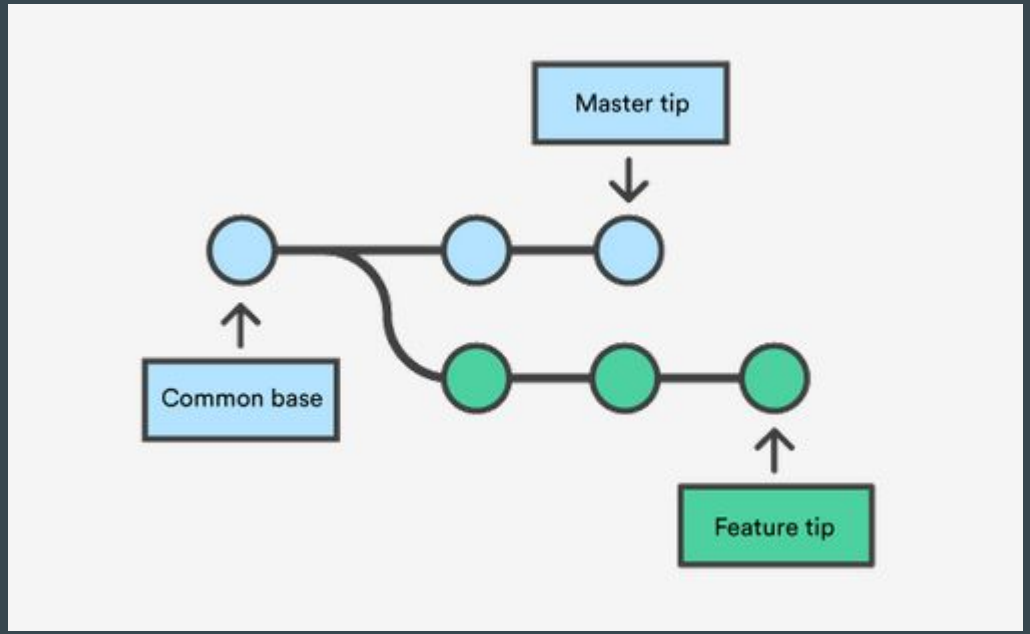
Part 4: Branches

git branch and git merge

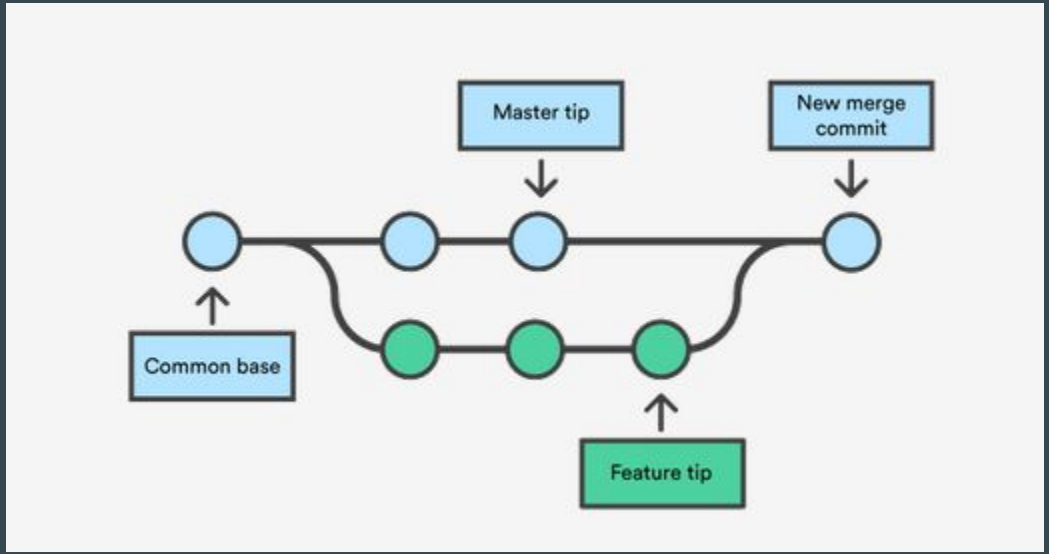


From https://www.atlassian.com/git/tutorials/using-branches/git-merge

git branch and git merge



git branch and git merge



git branch and git merge (local)

```
# view what branch you're on
git branch
# create a branch
git branch <branch-name>
# move to another branch
git checkout <branch-name>
```

git branch and git merge (local)

```
# Start a new feature. Move to (-b also creates the branch you are checking
out) a branch called new-feature
git checkout -b new-feature
# Edit some files
git add <file>
git commit -m "Start a feature"
# Edit some files
git add <file>
git commit -m "Finish a feature"
# Move back to the master branch
git checkout master
# Merge the new-feature branch into the branch you are on
git merge new-feature
# delete the branch
git branch -d new-feature
```

Branches IRL

adafruit / circuitpython
forked from micropython/micropython

Watch 91 | Unstar 455 | Fork 1,746

Code | Issues 92 | Pull requests 4 | Insights

Branch: master

Commits on Jun 2, 2018

Merge pull request #892 from tannewt/clock_w_crystal dhalbert committed 17 minutes ago	Verified	9a6a5ea
Merge pull request #896 from dhalbert/update_frozen_lis3dh kattni committed an hour ago	Verified	47b60fd
Merge remote-tracking branch 'adafruit/master' into update_frozen_lis3dh dhalbert committed an hour ago		0ed946c
update HID, and update seesaw version dhalbert committed an hour ago		27eba65
Merge pull request #895 from dhalbert/update_frozen_lis3dh kattni committed an hour ago	Verified	4186588
Update frozen LIS3DH to 4.2.1 to save space dhalbert committed 2 hours ago		6e6bfc4

Questions


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Part 5: TechHive on GitHub

Housekeeping / Best Practices

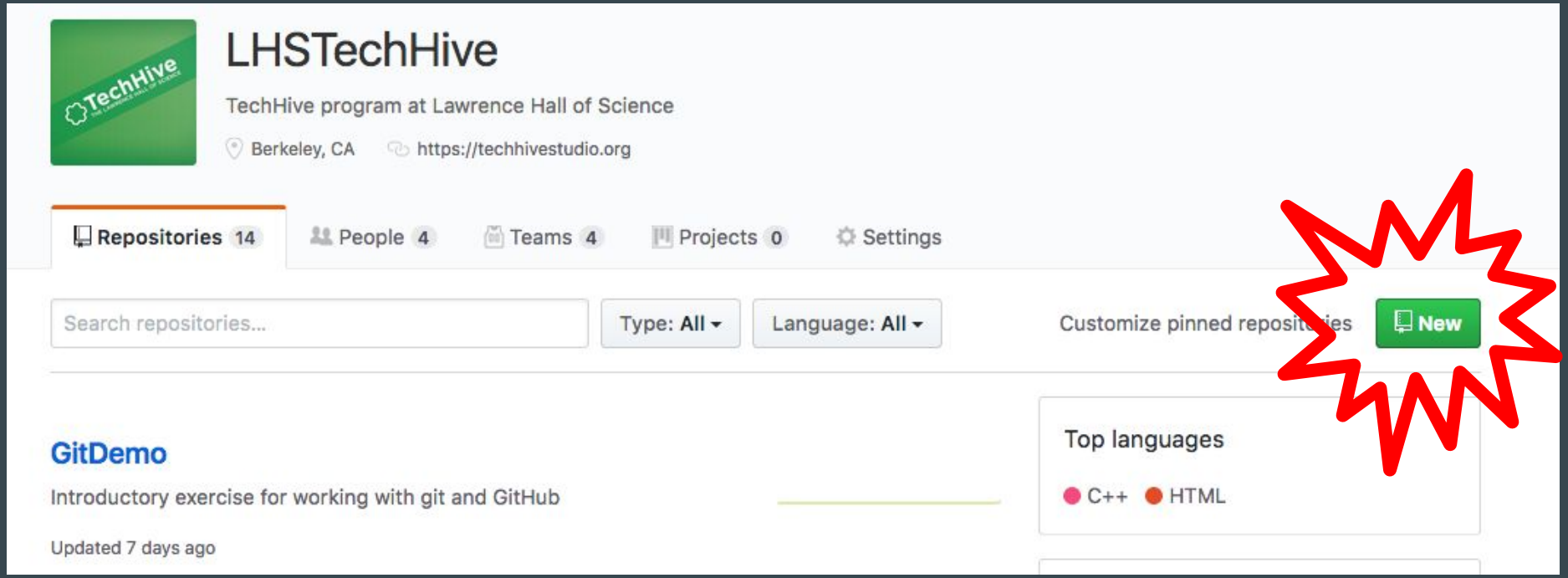
LHSTechHive organization

- https://github.com/LHSTechHive is an organization. You can add users to an organization, but you can't log in as an organization.
- Organizations allow you to group users into different teams. You can set different permissions for teams, e.g. one team has write access to X repository but another only has read access, or change settings so that members cannot delete repositories.
- I have added staff and adult volunteers who are comfortable with git as *admins*, and everyone else (interns, new volunteers) should be *members*. Members currently can clone repositories and make pull requests, but not push directly to them.
- Admins can add and remove members and change settings for the organization.

Contributing to existing projects

- Contributors should fork a repo and then make a pull request to it to merge changes in. (This will be covered in the exercise next week.) Do this even if you are the primary maintainer and the repo only contains one file. This way, you can document why changes were added in the pull request description before merging it, and it's easier for people looking through the project's commit history to view changes by looking through closed pull requests.
- The corollary of making pull requests instead of pushing to the master branch is that we should use branches for developing new features instead of creating new repos with version names.
- *The master branch should always work* (or at least, should have the most-working version of the code if the project is Work in Progress). Keep WIP changes to your own branch until you are sure they will work when merged.

Creating new projects



Documentation: READMEs

Every new project should have a README.md file to describe what it does. There is a template for this in https://github.com/LHSTechHive/TechHive/tree/master/templates that has sections for describing different aspects of a project.

The idea is that anyone who has a general familiarity with the tech stack used in a project should be able to get started working on it after reading a README file.

Project Name

High-level description of the repo. What does this do? E.g. "This is the code for the PIdestal component of the TechHive ARduino Box. It uses an Arduino Uno connected to blah, blah, and blah to rotate the camera mount (pedestal) clockwise or counterclockwise if the red or blue buttons are pressed, respectively. It should not rotate more than 360 degrees in either direction."

If a repository is deprecated, note that here. There is also an archive feature on GitHub now which you could also apply if a repo is dead.

Related repositories

E.g. if this is the repo for the PIdestal, probably want to link to the PIcam code. If there is a higher level organizing document, link to that here (or if it's an internal doc, at least mention its existence and who to contact for access to it).

Schematics

If applicable. If you upload a photo, take the time to label it (just something quick with microsoft paint or OSX preview is fine). Label parts and circle details that might not be obvious.

Branches

If applicable. Is there more than one version of this that is being worked on? Write a quick description of each of the branches here. The master branch, unless otherwise stated, should always work if you follow the setup steps.

Dependencies

What do you need to get this code to do what the description says?

Hardware

List each part including version number.

Documentation: Pull Request templates

Each repository should have a pull request template. This is a feature that is used by GitHub that pre-populates the description box that contributors will fill out. There is one in https://github.com/LHSTechHive/TechHive/tree/master/templates that you can copy into new projects.

Title here

High level description of what this change does goes here.

Related issue(s) ↔️

Links to related issues go here.

Testing 🔥

Describe how this was tested here. Be specific! What versions of hardware and software were used?

Alternatively, justify why this does not need testing (e.g. "Only changed documentation").

Documentation 📝

- [ ] Associated documentation to this change was updated or created.

Review 👀

- [ ] At least one person (who is not the person who made the pull request) has reviewed this. This is consciously not programmatically enforced, so use your best judgement.

Documentation: commit messages

You're spending all this energy Fully Utilizing Github, so make sure you effort is helpful for future contributors / your future self.

Write specific commit messages. E.g. instead of “fixed stuff” say “increased the speed of the rotation to 11 in the blah function” or “fixed a typo in the docs for ”.

Goes without saying, but no profanity in commits (we are family-friendly 😸).

Documentation: Branch names

Ideally you should name your branch something descriptive.

If you fixed something related to an issue where a servo movement was jerky, call the branch `smooth_servo_movement` instead of just `improvements.`

Documentation: wikis

Each repository comes with an empty wiki. You can use this to add extra notes about the project, but generally you should try to put everything in the README for visibility and to keep track of changes. Wikis are not version controlled.

There is a general wiki for TechHive at https://github.com/LHSTechHive/TechHive/wiki . You can document things that don't necessarily correspond to a repository there.

DO NOT COMMIT YOUR PASSWORD TO VERSION CONTROL

Miguel Grinberg - Oops! I Committed My Password To GitHub! ➡ 📺

This video uses example code in Python, but the concepts can be applied in any programming language.

Questions


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Questions

- Google is your friend
  - “How to undo previous commit”
  - “How to fetch all branches from remote”
- Before pasting the first command in the first StackOverflow result into your terminal, try and understand the concept behind the command you are using.