Before We Begin...

All links can be found on:
https://uwb-acm.github.io/Git-Gud-Workshop/

1. Install Git (command line)

   https://git-scm.com/downloads

2. Create a GitHub account if you don't have one already.

   Use a private email (not your school email)

   https://github.com/

3. Open this URL for later:

   https://github.com/UWB-ACM/Git-Gud-Workshop

```
$ git gud
git: 'gud' is not a git command. See 'git --help'.

The most similar command is
        gui
```

Git Gud

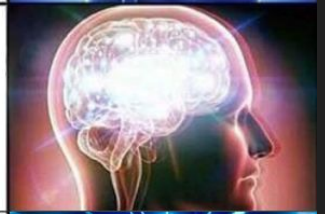
Workshop

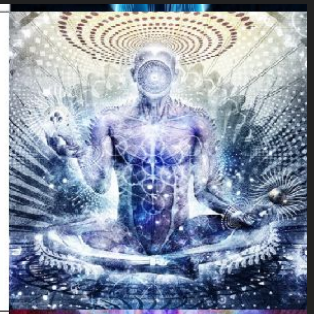
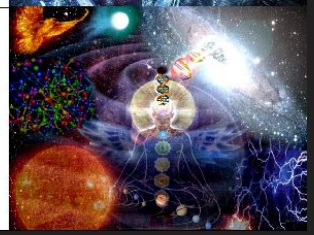
By your friends at

UWB ACM

Agenda

- Introduction and Definitions
- Configuration
- Workshop #1: First Commit
- Workshop #1: Syncing with Remote
- Workshop #3: Working with Files
- Workshop #4: Contributing to Projects
- Flow, GUIs, and Further Topic
- Additional Resources

Git	
Google Drive	
Uploading the code to an external hard drive via USB	
Mailing the code	
Printing the code	

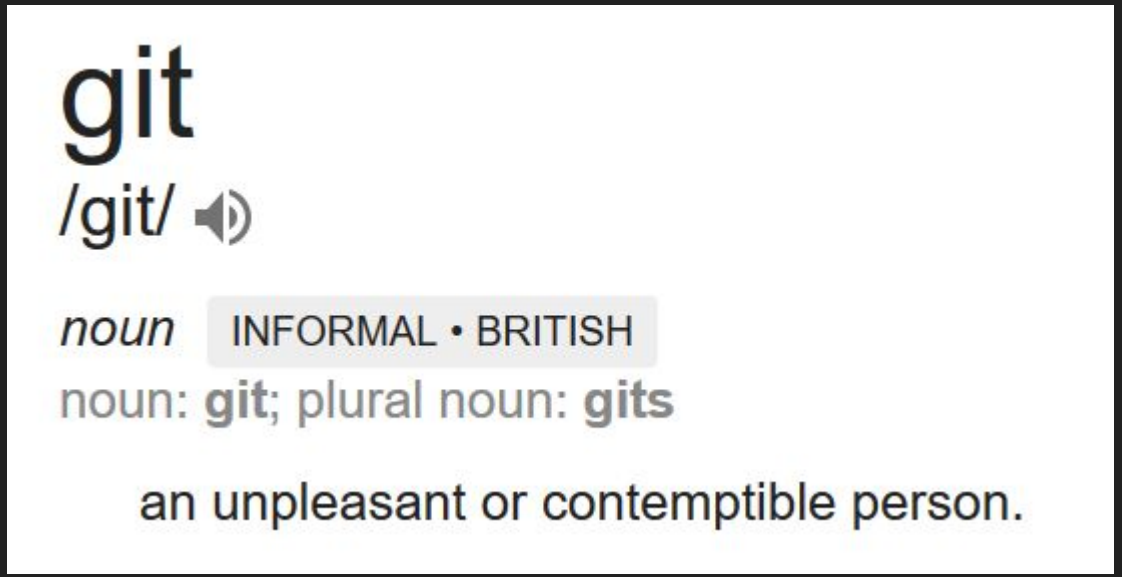
Using a single file on a shared computer	
Placing a video camera recording the keyboard	
Keeping the IDE open permanently and using ctrl-z to go back to previous versions	
Wiping the hard drive and re-writing the program	

What's Version Control?

(If you've had CSS 360, this may ring a bell)

- Tracks revision history
- Allows users to switch between versions
  - "Roll-back"
  - Teams can develop against "stable" version
- Implemented in cloud drive platforms
  - e.g. Google Drive

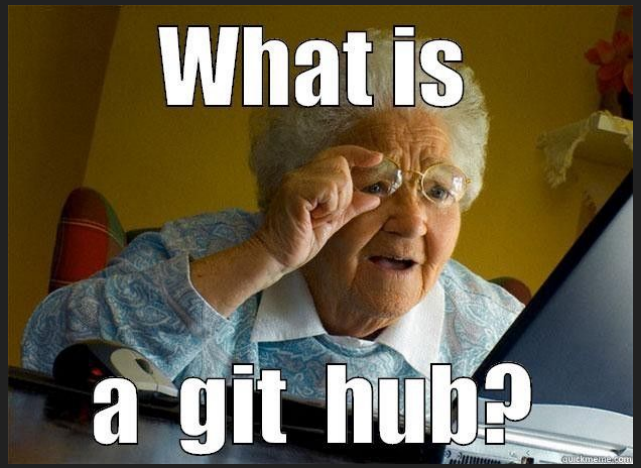
What’s Git?


git
/git/
noun
INFORMAL • BRITISH
noun: git; plural noun: gits
an unpleasant or contemptible person.

Who? What?

What is Git?

- Version Control Software!
- Tool for tracking revisions of files
- Enables synchronization for distributed teams



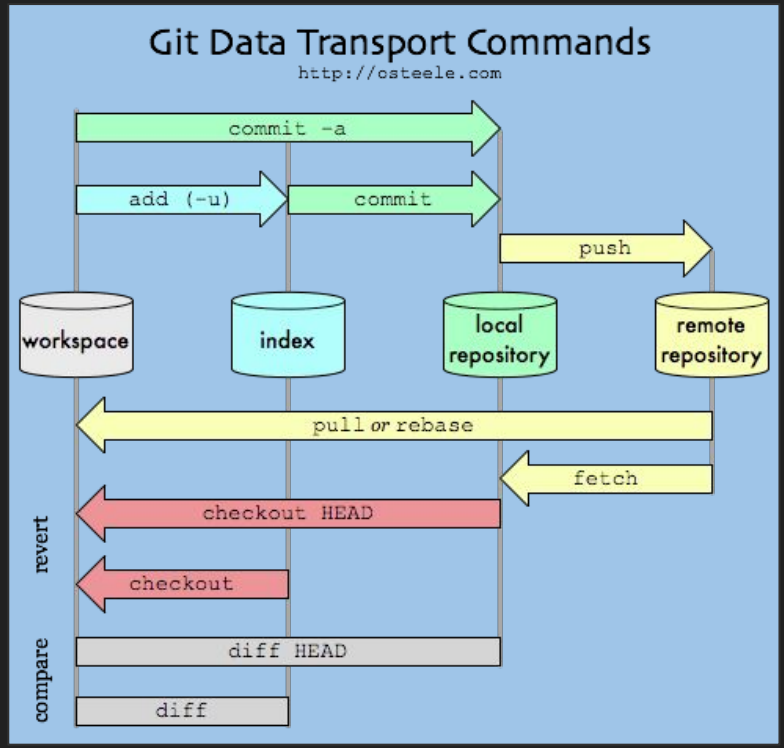
Who uses Git?

- Amazon, Microsoft, Facebook, Netflix, Twitter, Linux, Android, Eclipse, Starbucks, Google, Hashicorp, eBay…
- Over 31 million users on GitHub. (Git != GitHub)
- Your classmates! (Solo, or for group projects)

Why Git?

- Keep files in sync!
  - Among workstations
  - Among teammates
- Backup source code
- Distributed collaboration
  - Work remote!
- Manage complex workflows

It’s great for code!



What *isn't* Git?

Git != GitHub

GitHub *uses* Git*

Git != Google Drive

(Git doesn't sync in real-time)

Git != GitHub

Git (Launched April 2005)

- Version Control Software
- Independent of any online services
  - Doesn't even have to connect online

GitHub (Launched April 2008)

Popular online service for hosting Git repositories, public or private

Add features that enhance an open-source software workflow

- Issues
- PRs
- Support for Continuous Integration / CD

Version Control Vocabulary

Repo / Repository

A set of files, containing commits.

Commit

A set of changes applied to files, linked to the commit before it, with a message and author.

Diff

Changes between current state and last commit.

Branch

A set of commits that are linked to one another.

Remote

A server that hosts a Git repository.

Push

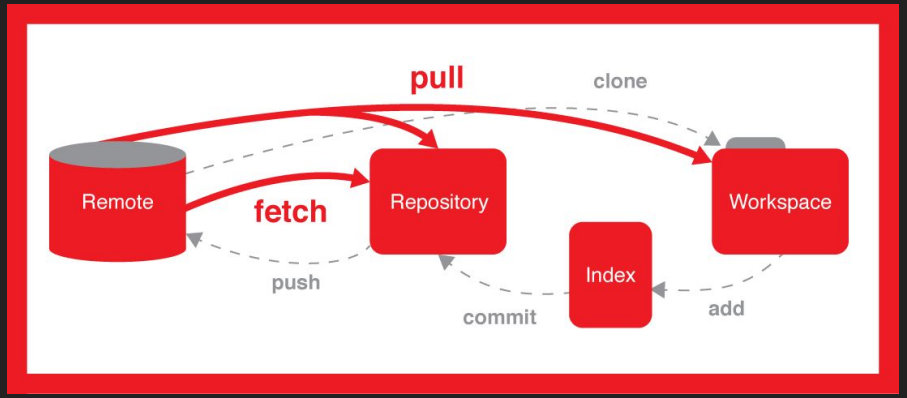
Sync local changes to a branch with a remote.

Pull

Retrieve changes from remote.

Ways to use Git

- Locally: Repo only on your machine
- Synced: Repo synced to remote
  - Remotes are often hosted*



- Command Line Interface (Default)
  - Fully featured
  - Steeper learning curve
- Graphical User Interface
  - May not be fully featured
  - Easy to use
  - IDE Plugins, GitHub Desktop, etc.

*Remote hosting

- GitHub
- GitLab
- Bitbucket
- SourceForge
- Kiln
- AWS CodeCommit
- etc...

Git Commands Reference

Check out a cheat sheet here:

https://education.github.com/git-cheat-sheet-education.pdf

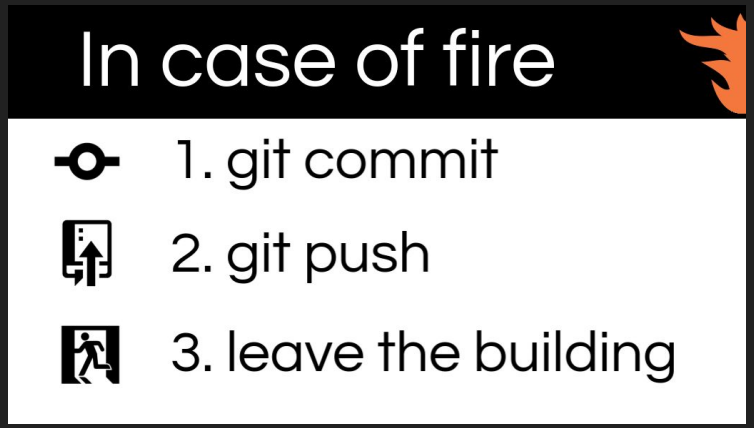
Shortlink:

https://bit.ly/1LHKj5a

Git Commands Reference

`git init` : Makes the current directory into a Git repo.

`git clone [url]` : Downloads a repo from the given URL.



`git add [files]` : Adds the specified files to the staging area.

`git rm [files]` : Removes the specified files from the staging area.

`git commit` : Associates a message with the changes in the staging area, and add these changes to the repo's history.

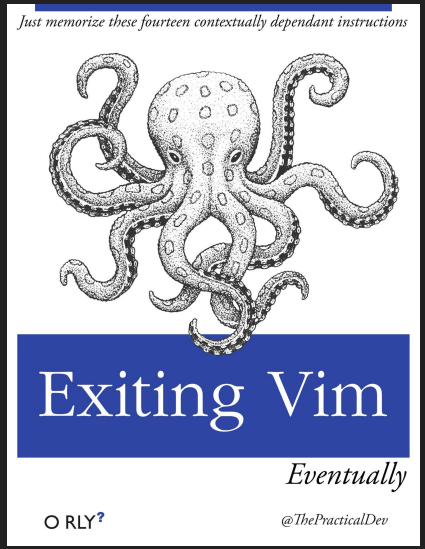
`git checkout [reference]` : Gets the contents of the repo from a specific commit.

`git push` : Uploads changes.

`git pull` : Downloads changes.

Git Config (First-Time Setup)

Git needs to know your username and e-mail to associate your work with you.



Use the following commands:

```
git config --global user.name “Bobby Tables”
git config --global user.email bobby@tables.com
git config --global core.editor "nano"
```

Workshop #1: Our First Commit

1. Create a local repo

```
git init
```

2. Create new files

```
echo “Hi!” >> hello.txt
```

(or equivalent)

3. Add files to our index

```
git add hello.txt
```

4. Commit changes to local repo

```
git commit
```

Workshop #1: Example

1. Initialize repo
2. Create text file
3. Add file to index
4. Commit file to repo

```
12:13:20:spacekatt@spacekatt-Latitude-E6540:~/ACM/test$ git init
Initialized empty Git repository in /home/spacekatt/ACM/test/.git/
12:13:22:spacekatt@spacekatt-Latitude-E6540:~/ACM/test$ echo "Hello, World!" >> hello.txt
12:13:33:spacekatt@spacekatt-Latitude-E6540:~/ACM/test$ git add hello.txt
12:13:38:spacekatt@spacekatt-Latitude-E6540:~/ACM/test$ git commit -m "Add hello world doc"
[master (root-commit) 93e019a] Add hello world doc
 1 file changed, 1 insertion(+)
 create mode 100644 hello.txt
12:13:50:spacekatt@spacekatt-Latitude-E6540:~/ACM/test$
```

Writing Good Commit Messages

Qualities of a good commit message:

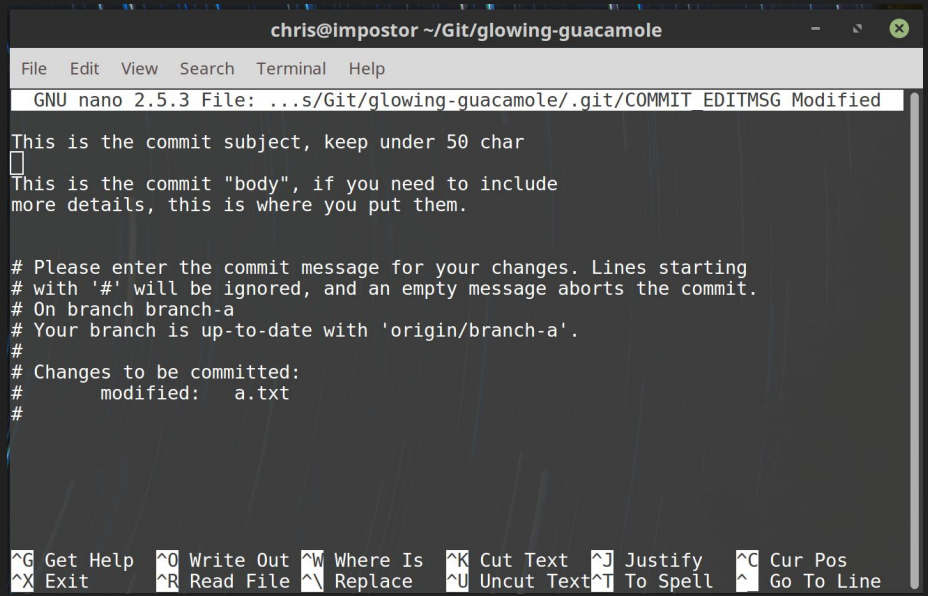
- Use the imperative mood
  - “Add config file”
  - “Fix nginx SSL bug”
- Subject is clear and descriptive, while under 50 characters in length.
- Body adds detail and explanations
- Avoid using `git commit -m`
  (do as I say, not as I do)

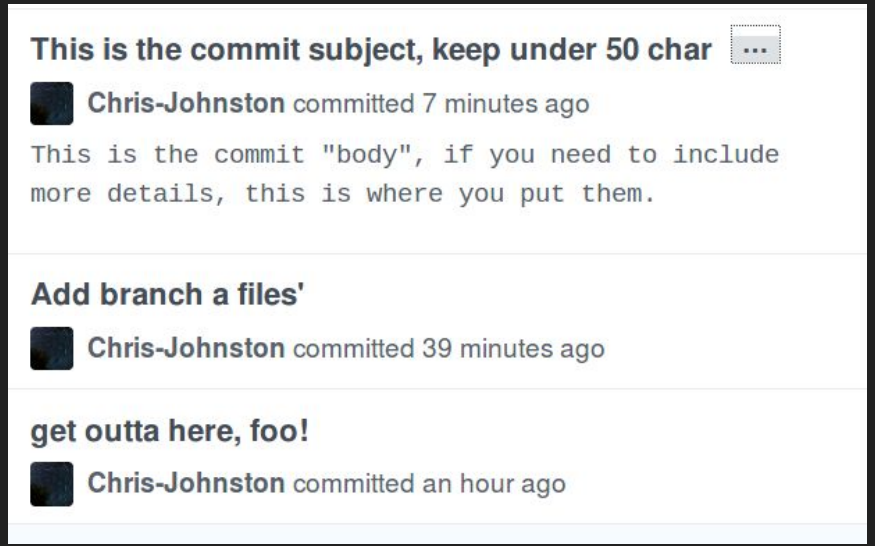
Further reading:

https://help.github.com/articles/closing-issues-using-keywords/

https://code.likeagirl.io/useful-tips-for-writing-better-git-commit-messages-808770609503

Commit Message Example





How others will see your message in GitHub

Viewing the log

View previous commit messages with...

```
git log
```

Benefits of a well-kempt log...

- Workflow transparency
- Process traceability
- Bug tracking
- Smaller commits
- Easier merging

```
commit 3199695f77bdb35fd9d510208b8ee95f04e7f3a3
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 18:56:23 2018 -0800

    Update HTML style

commit 08fa02f8155d73dedbec152e05e3640b5632e408
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 16:49:47 2018 -0800

    Fix bug where service starts before loading profile

    Since AWS credentials are provisioned at deployment, I believe
    the gunicorn service was being started by systemd before the
    credential files existed, causing an error (sometimes). Restarting
    the service after Terraform provisions everything seems to fix it.

commit b0299d535f0b1013c399a9ecaee75e7f21d722ef
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 13:08:49 2018 -0800

    Add alert message to user for empty input

commit ece5c4048e93913ef26d5cf0a324124190a6297f
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 12:56:30 2018 -0800

    Handle ommisions in query field

commit 29011a7ab3923bfa5d07ddd1bcf8dcf77ecfd172
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 12:34:01 2018 -0800

    Add all attributes to table

commit ac282eaac3871148bb52e7bbbd87dc2f001254fa
Author: Thomas Kercheval <spacekattpoispin@gmail.com>
Date:   Thu Nov 8 11:45:08 2018 -0800

    Add simple HTML table in React
```

Workshop #2: Syncing to remote (GitHub)

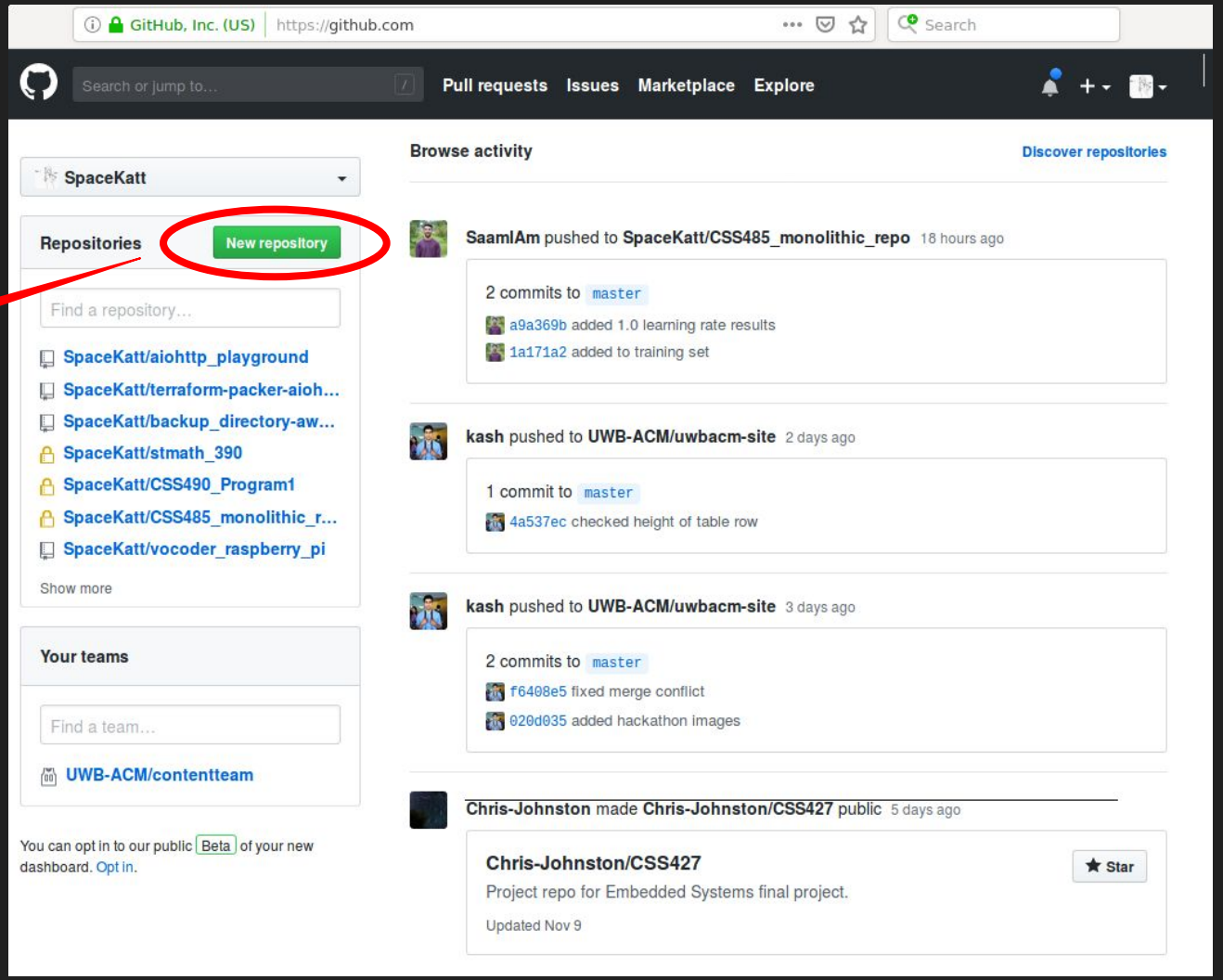
Using the repository from Workshop #1...

- Create remote repo (on GitHub)
- Add reference to remote (URL) locally
- Sync local changes to remote

```
<Create repo on GitHub>

git remote add origin <remote url>

git push -u origin master
```

Create New Repository

Click da' button!

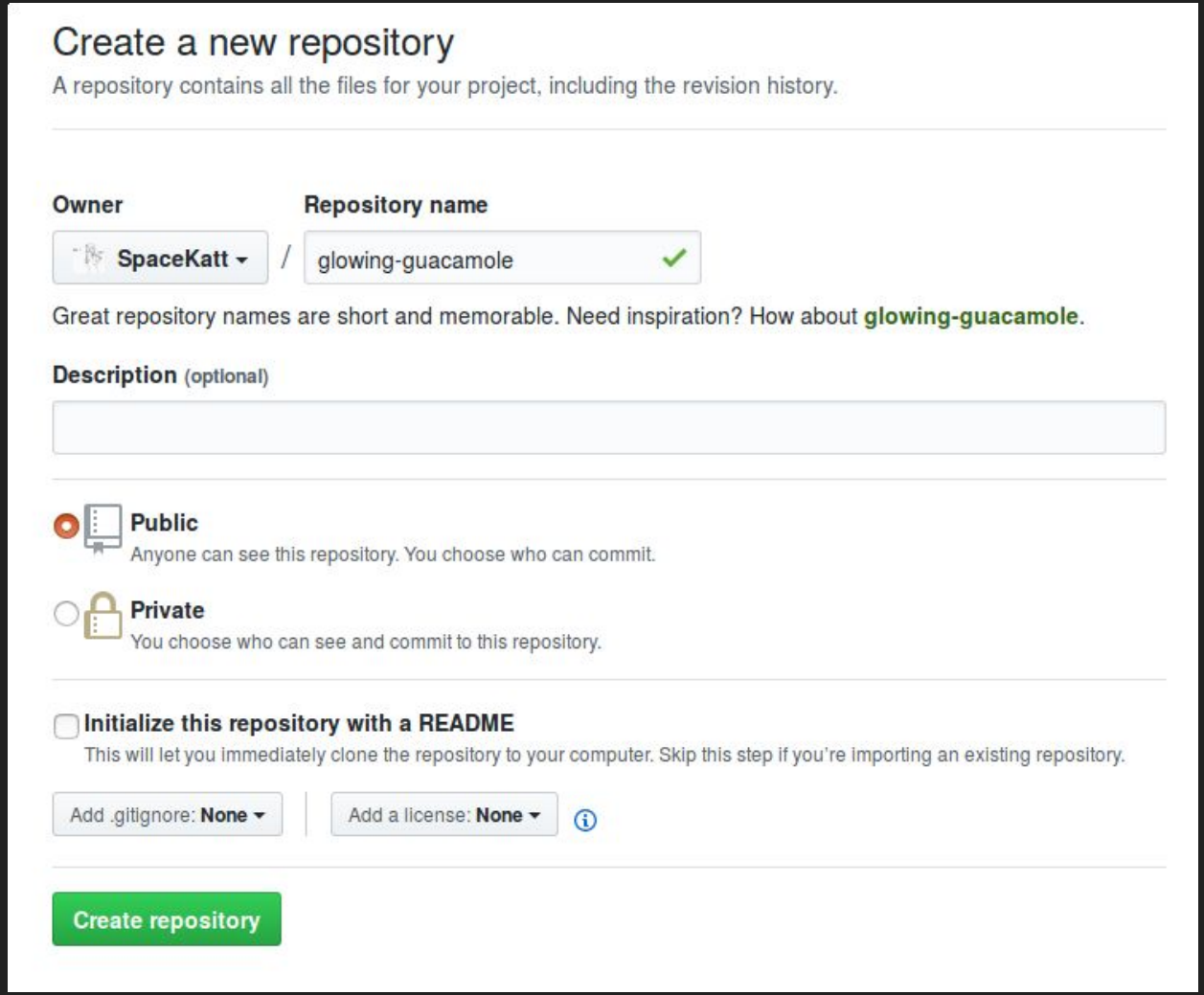


Name New Repository

Give your new repository some name…

If you're hosting a cool project, then use a descriptive name!

Don't include a README, .gitignore, or license yet.

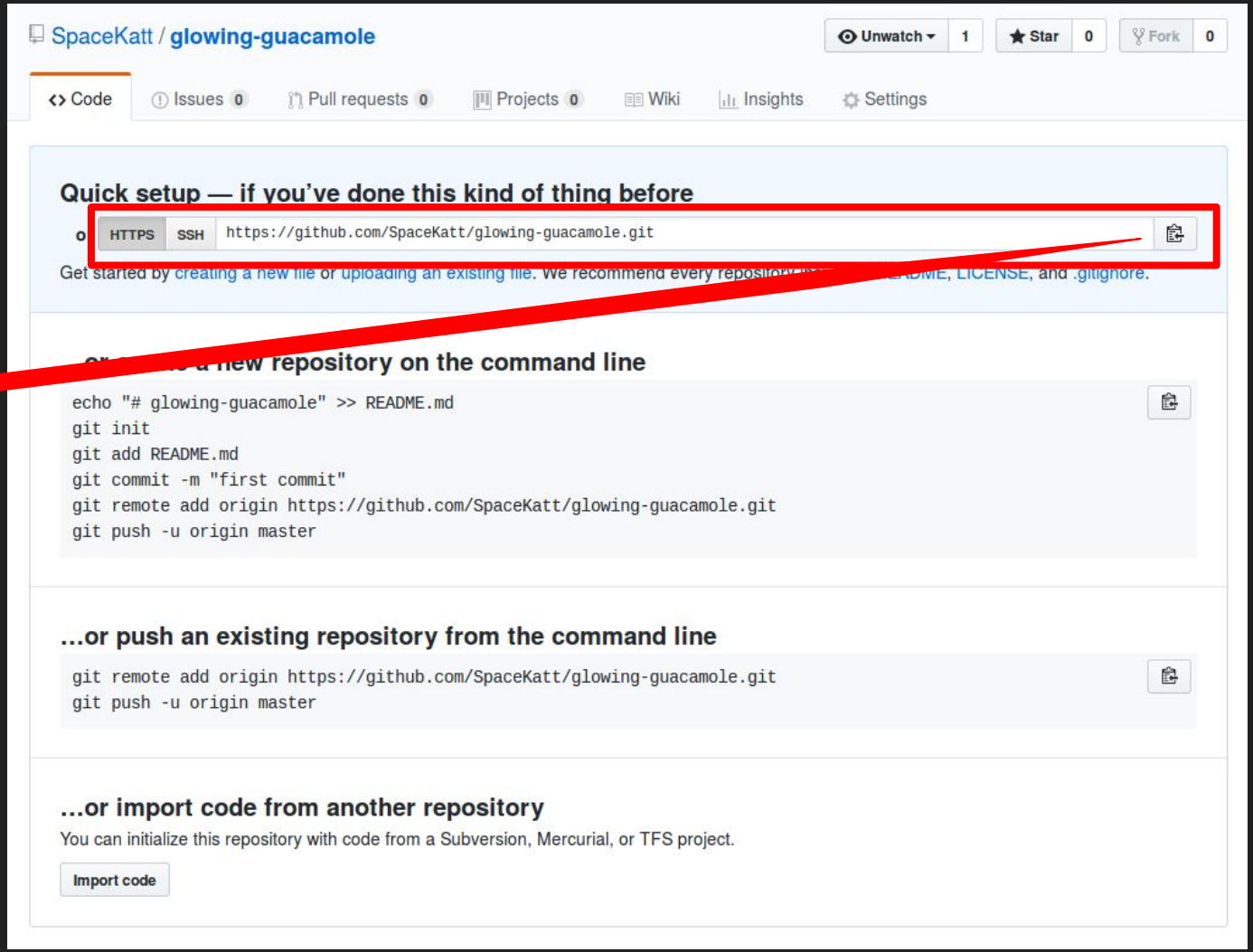


Copy URL of repo

Click da' button to copy to your clipboard

Keep this page open

We will refresh it after the next step!



Add remote and push!

Add reference to the remote repository

Sync remote with local repo with push

```
03:18:23:spacekatt@spacekatt-Latitude-E6540:~/ACM/test$ git remote add origin https://github.com/SpaceKatt/glowing-guacamole.git
03:18:25:spacekatt@spacekatt-Latitude-E6540:~/ACM/test$ git push origin master
Username for 'https://github.com': spacekatt
Password for 'https://spacekatt@github.com':
Counting objects: 6, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 506 bytes | 0 bytes/s, done.
Total 6 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/SpaceKatt/glowing-guacamole/pull/new/master
remote:
To https://github.com/SpaceKatt/glowing-guacamole.git
 * [new branch]      master -> master
03:18:41:spacekatt@spacekatt-Latitude-E6540:~/ACM/test$
```

Refresh Repo Page on GitHub

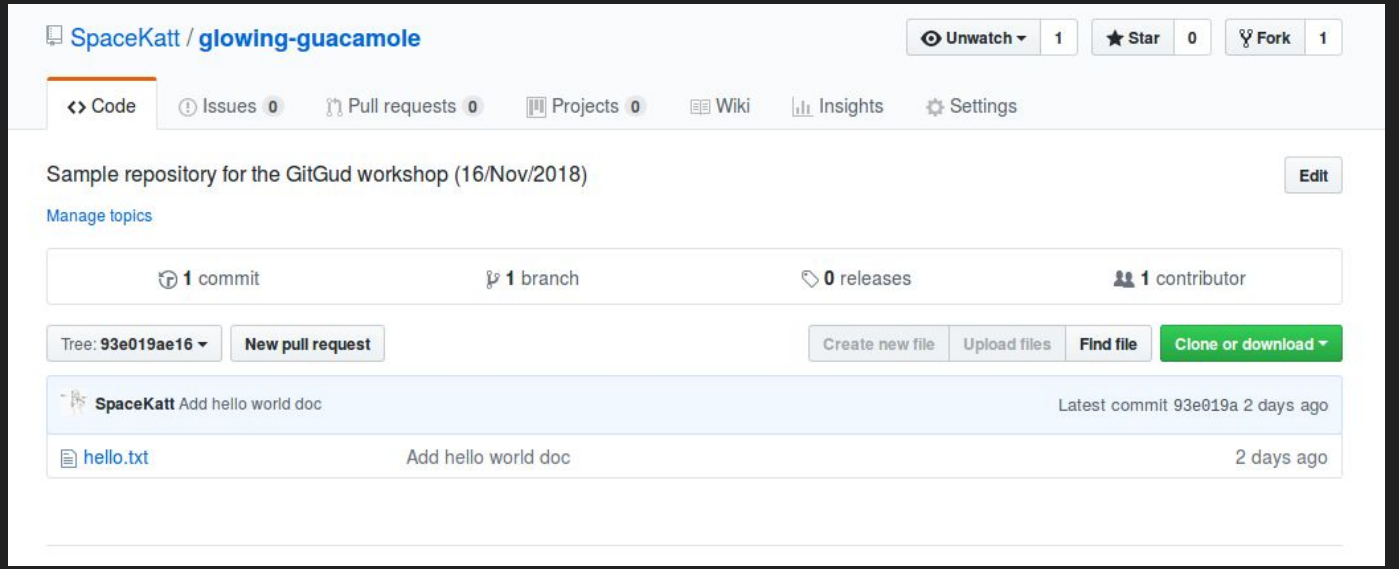
We now see the file we pushed to our remote, `origin`

Exciting!

Much open source!

wow

Heckin' `origin`



Workshop #3: Working with files

We are going to create a new file, commit it, and upload it to GitHub.

Then, once it's uploaded, we are going to delete that file.

We can also use `git mv` in pace of `git rm`, if we desire to move a file, instead of removing it

```
echo "bar" >> foo.txt

git add foo.txt

git commit

git push

git rm foo.txt

git commit

git push
```

Workshop #3 Example: Sync File to Remote (i)

Create a new file	`echo “bar” >> foo.txt`
Add file to index	`git add foo.txt`
Commit it	`git commit`
Sync file to remote	`git push`

```
chris@impostor ~/Git/glowing-guacamole $ echo "bar" >> foo.txt
chris@impostor ~/Git/glowing-guacamole $ git add foo.txt
chris@impostor ~/Git/glowing-guacamole $ git commit -m "I pity the foo.txt"
[master 8bef996] I pity the foo.txt
 1 file changed, 1 insertion(+)
 create mode 100644 foo.txt
chris@impostor ~/Git/glowing-guacamole $ git push
```

Workshop #3 Example: Tell git to Remove File (ii)

Tell git to remove file `git rm foo.txt`

Check file's status `git status`

```
chris@impostor ~/Git/glowing-guacamole $ git rm foo.txt
rm 'foo.txt'
chris@impostor ~/Git/glowing-guacamole $ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    foo.txt
```

Workshop #3 Example: Sync Change to Remote (iii)

All we have to do now is synchronize!

Commit it `git commit`

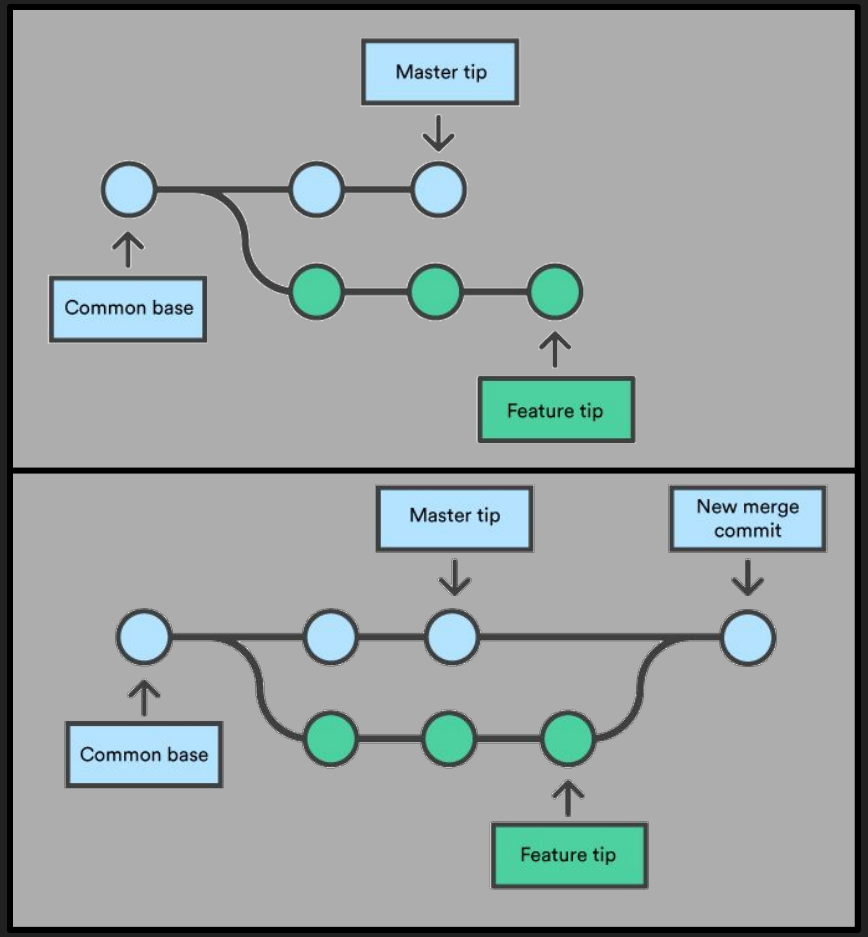
Push it to GitHub. `git push`

```
chris@impostor ~/Git/glowing-guacamole $ git commit -m "get outta here, foo!"
[master e774ea7] get outta here, foo!
 1 file changed, 1 deletion(-)
 delete mode 100644 foo.txt
chris@impostor ~/Git/glowing-guacamole $ git push
```

Branches?

Branches keep track of changes, from a shared starting point.

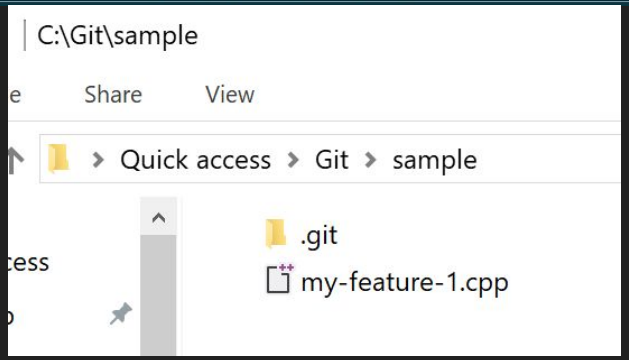
Branches can:
- Separate from another branch
- Merge into another branch
- Contain different contents from another branch



Branches (1 of 8)

```
chris@ComputerMachine:/mnt/c/Git/sample$ echo "my-feature-1.cpp" >> my-feature-1.cpp
chris@ComputerMachine:/mnt/c/Git/sample$ git add my-feature-1.cpp
chris@ComputerMachine:/mnt/c/Git/sample$ git commit -m "Add Feature 1"
[master (root-commit) 23ed8fd] Add Feature 1
 1 file changed, 1 insertion(+)
 create mode 100644 my-feature-1.cpp
chris@ComputerMachine:/mnt/c/Git/sample$ git status
On branch master
nothing to commit, working directory clean
```

```
chris@ComputerMachine:/mnt/c/Git/sample$ git log
commit 23ed8fdffbd307fa3e3197f9429caa8ced3ca9f1
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 00:51:19 2018 -0800

    Add Feature 1
chris@ComputerMachine:/mnt/c/Git/sample$
```



Let’s use a new repo as an example.

In the master branch, we can add a new file for “Feature 1”.

Branches (2 of 8)

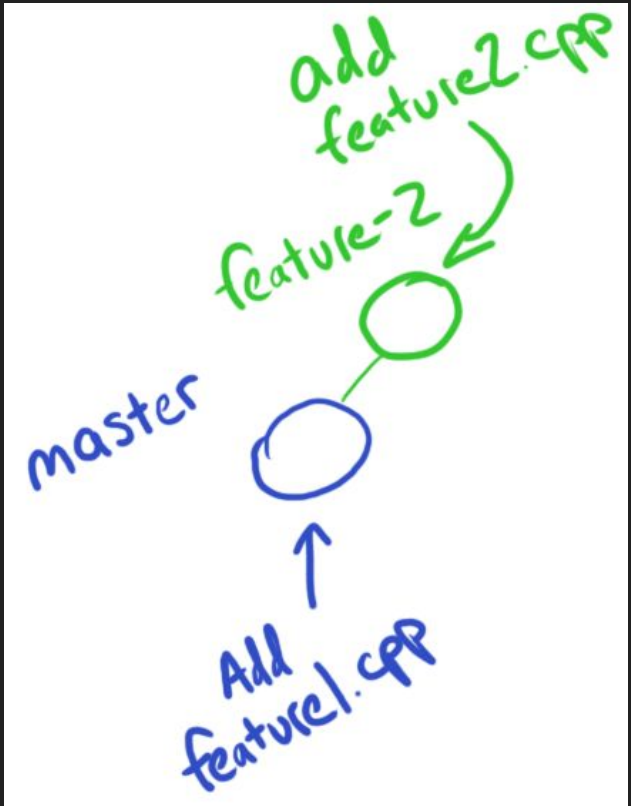
```
chris@ComputerMachine:/mnt/c/Git/sample$ git checkout -b feature-2
Switched to a new branch 'feature-2'
chris@ComputerMachine:/mnt/c/Git/sample$ echo "my-feature-1.cpp" >> my-feature-2.cpp
chris@ComputerMachine:/mnt/c/Git/sample$ git add my-feature-2.cpp
chris@ComputerMachine:/mnt/c/Git/sample$ git commit -m "Add Feature 2"
[feature-2 889c9f1] Add Feature 2
 1 file changed, 1 insertion(+)
 create mode 100644 my-feature-2.cpp
chris@ComputerMachine:/mnt/c/Git/sample$ git status
On branch feature-2
nothing to commit, working directory clean
chris@ComputerMachine:/mnt/c/Git/sample$
```

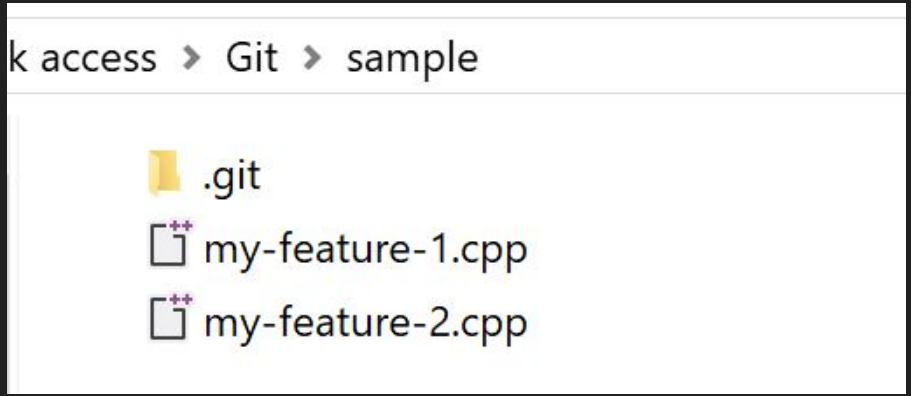
```
chris@ComputerMachine:/mnt/c/Git/sample$ git log
commit 889c9f101079ae707a2f07e84c66cc72b6d001c3
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 00:51:50 2018 -0800

    Add Feature 2

commit 23ed8fdffbd307fa3e3197f9429caa8ced3ca9f1
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 00:51:19 2018 -0800

    Add Feature 1
```

Let's branch off into a new branch, called "feature-2".

In this branch, we'll add a new file.

Branches (3 of 8)





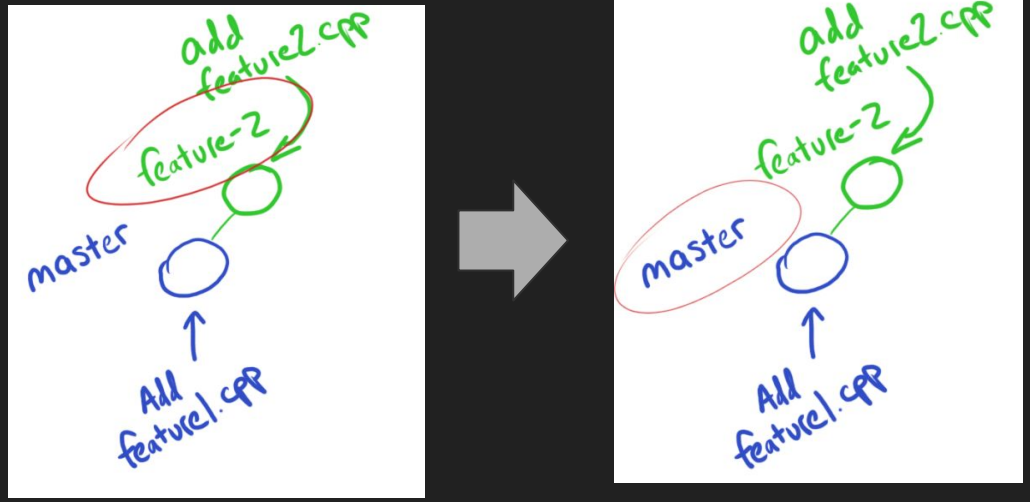
Branches (4 of 8)

Uh-oh! Your boss just stopped by and wants to see your progress, but “Feature 2” is breaking the build! What do you do?

How do you show your boss the work you’ve done, but without discarding the hard work you’ve done on “Feature 2”?

Branches (5 of 8)

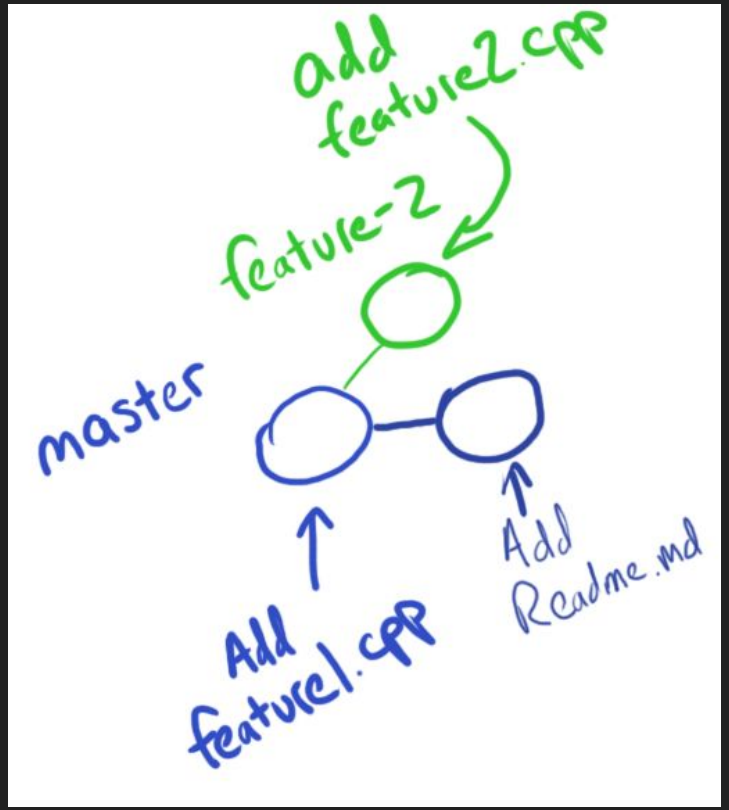
```
chris@ComputerMachine:/mnt/c/Git/sample$ git checkout master
Switched to branch 'master'
chris@ComputerMachine:/mnt/c/Git/sample$ git log
commit 23ed8fdffbd307fa3e3197f9429caa8ced3ca9f1
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 00:51:19 2018 -0800

    Add Feature 1
```



Good for us, we're using git!

We can use the git checkout command to switch back and forth without losing any progress.

Branches (6 of 8)

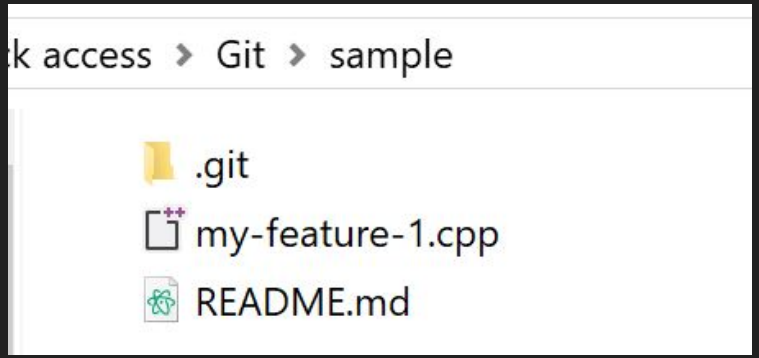


Your boss asked you to create a Readme file to describe your project, so you added it to the master branch.

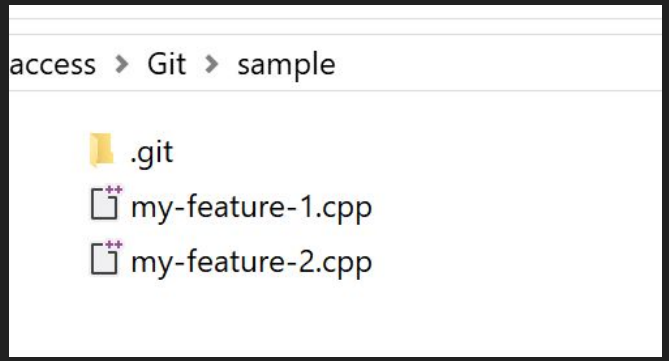
```
chris@ComputerMachine:/mnt/c/Git/sample$ echo "TODO Make a readme" >> README.md
chris@ComputerMachine:/mnt/c/Git/sample$ git add README.md
chris@ComputerMachine:/mnt/c/Git/sample$ git commit -m "Add README"
[master 1e22836] Add README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
chris@ComputerMachine:/mnt/c/Git/sample$ git log
commit 1e228365350283b1af07d666f6aa9f176607c431
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 01:35:32 2018 -0800

    Add README

commit 23ed8fdffbd307fa3e3197f9429caa8ced3ca9f1
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 00:51:19 2018 -0800

    Add Feature 1
chris@ComputerMachine:/mnt/c/Git/sample$
```

Branches (7 of 8)

Master



Feature-2



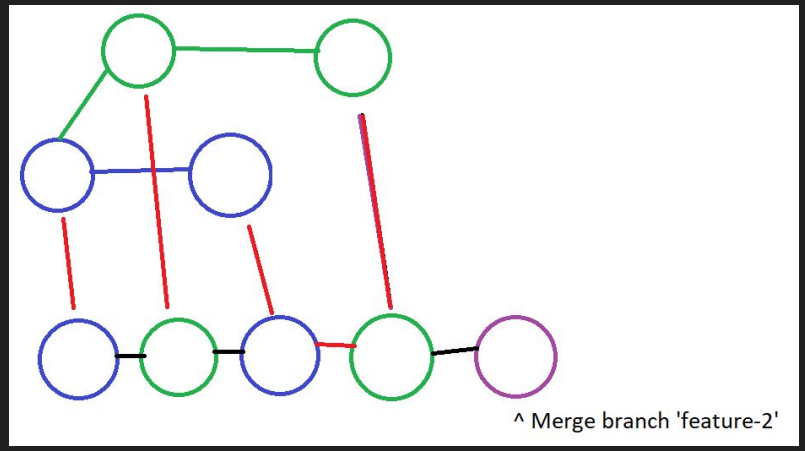
Same folder, different contents!

Branches (8 of 8)

So you've finished your work on Feature 2, and you want to add this back into the master branch. How do you do this? With merging!

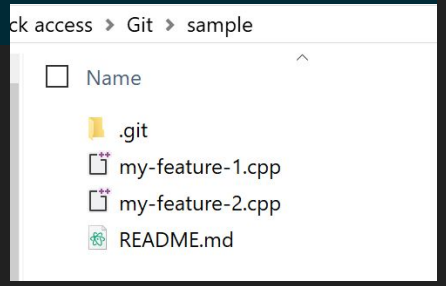
git checkout master

git merge feature-2



^ Merge branch 'feature-2'

```
$ git log
commit 5a37c9f4895796e70d3020180c0d7c7f7f27072b (HEAD -> master)
Merge: 1e22836 ae7619f
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 12:42:05 2018 -0800

    Merge branch 'feature-2'

commit ae7619fb40bf8ceb5408a62153ab38c63f04b170 (feature-2)
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 12:40:53 2018 -0800

    Finished work on Feature 2

commit 1e228365350283b1af07d666f6aa9f176607c431
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 01:35:32 2018 -0800

    Add README

commit 889c9f101079ae707a2f07e84c66cc72b6d001c3
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 00:51:50 2018 -0800

    Add Feature 2

commit 23ed8fdffbd307fa3e3197f9429caa8ced3ca9f1
Author: Chris Johnston <chris@thejohnstons.net>
Date:   Fri Nov 16 00:51:19 2018 -0800

    Add Feature 1
```



https://github.com/UWB-ACM/Git-Gud-Workshop

Workshop #4: Contributing

Using git, we can contribute to a pre-existing codebase.

Fork the sample repo first

We will clone our sample repo, make a new branch, add some commits, upload them to our forked copy, and make a new pull request.

Create a file called:

```
jokes/<FirstName><LastInitial>.txt
```

Include a funny joke inside it, commit, and push it

Then, we open a pull request on GitHub, to signal upstream that we wish them to merge our work in.

[Fork repo on GitHub]

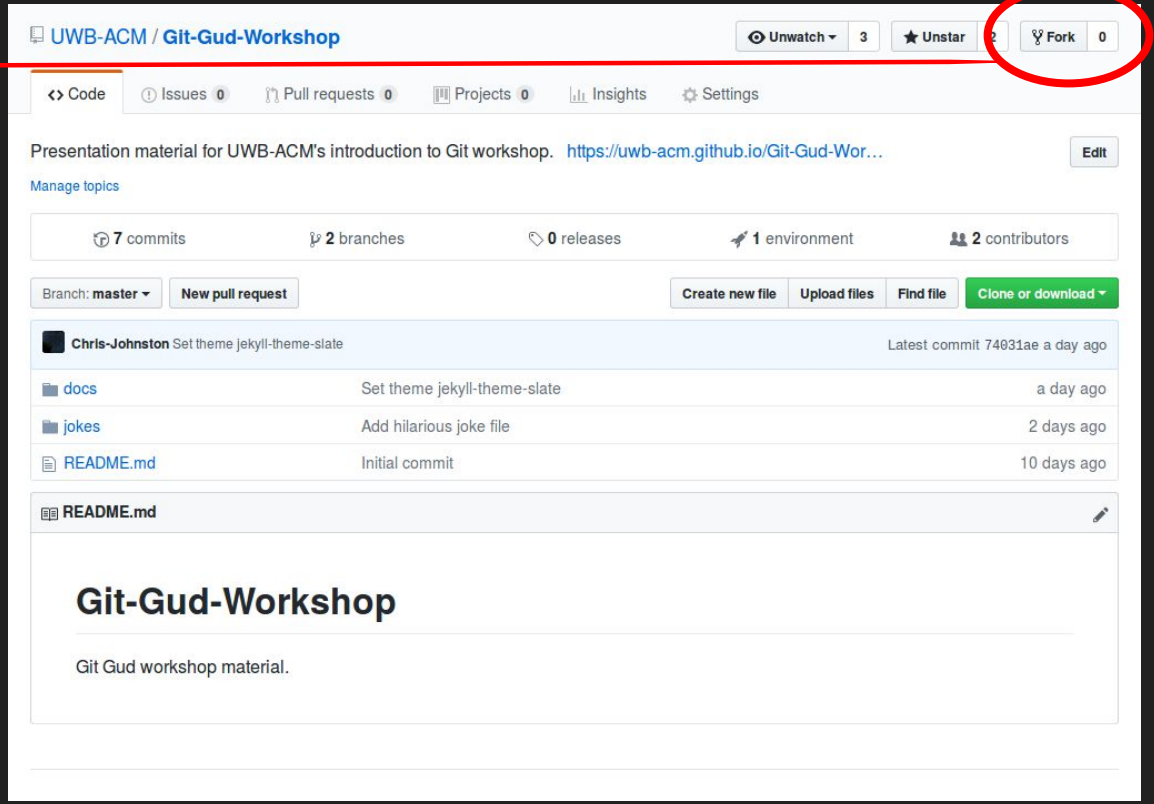
```
git clone <remote_forked_url>

cd folder

git checkout -b <branch_name>

echo “<joke>” >> jokes/BobbyT.txt

git add jokes/BobbyT.txt

git commit -m “<comment>”

git push origin <branch_name>

Open pull request [on GitHub]
```

(1 of 6) Forking the repo

Use the Fork button!

This will give us a copy

Use the forked copy to modify project without interrupting upstream development



(2 of 6) Cloning the repo (part 1)

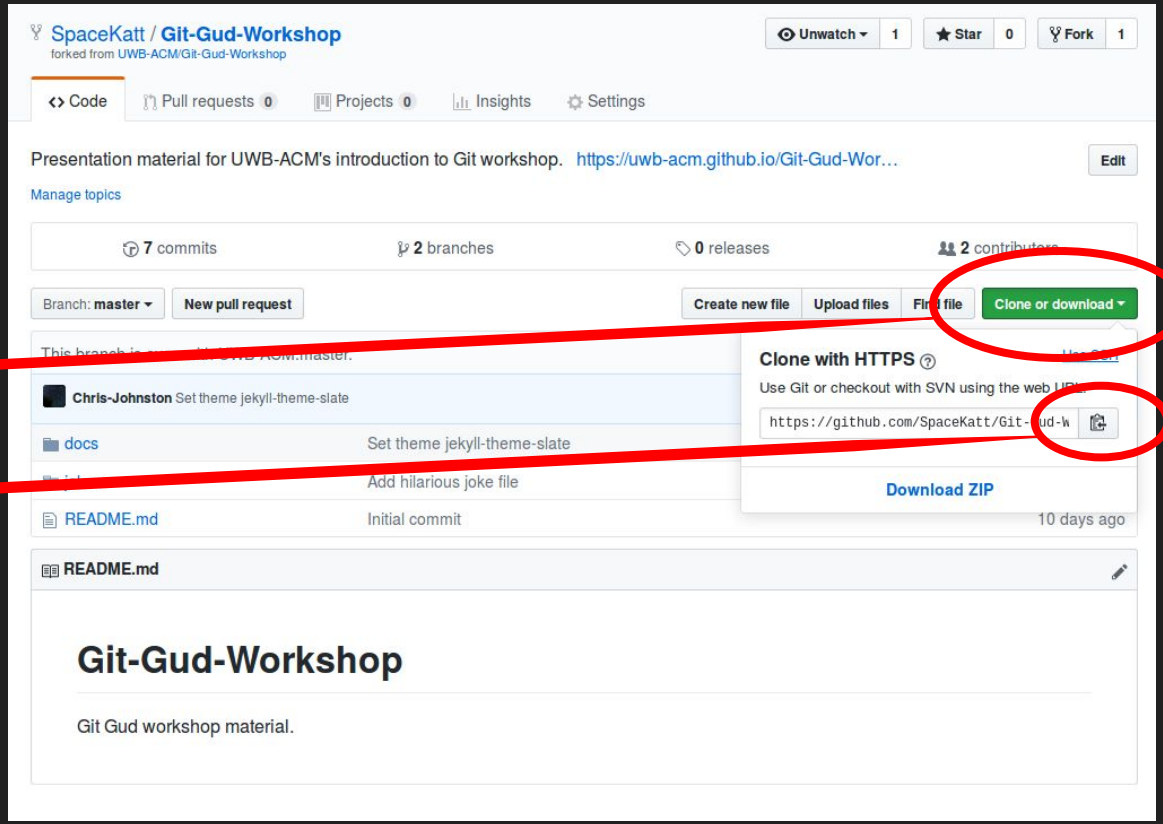
To clone your forked copy...

Copy remote's URL

View URL

Copy to clipboard

We use this URL on next slide



(3 of 6) Cloning the repo (part 2)

Cloning using HTTPS (Use this!)

```
12:13:38:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp$ git clone https://github.com/SpaceKatt/Git-Gud-Workshop.git
Cloning into 'Git-Gud-Workshop'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 25 (delta 2), reused 14 (delta 0), pack-reused 0
Unpacking objects: 100% (25/25), done.
Checking connectivity... done.
12:13:42:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp$ ls
.  ..  Git-Gud-Workshop
```

[Don’t use this for the workshop] Cloning using SSH (requires setting up SSH keys)

```
12:11:46:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp$ git clone git@github.com:SpaceKatt/Git-Gud-Workshop.git
Cloning into 'Git-Gud-Workshop'...
Enter passphrase for key '/home/spacekatt/.ssh/id_rsa':
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 25 (delta 2), reused 14 (delta 0), pack-reused 0
Receiving objects: 100% (25/25), done.
Resolving deltas: 100% (2/2), done.
Checking connectivity... done.
12:11:52:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp$ ls
.  ..  Git-Gud-Workshop
```

(4 of 6) Making a Joke

Create a new branch, with your first name and last initial

Create a file inside the jokes directory, with your first name and last initial

```
jokes/<FirstName><LastInitial>.txt
```

Inside it, include a funny joke. If it is not funny, you **will** be asked to leave.

Commit it

```
12:19:19:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp$ cd Git-Gud-Workshop/jokes/
12:19:21:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git checkout -b SpaceK
Switched to a new branch 'SpaceK'
12:19:30:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ vi SpaceK.txt
12:19:38:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git add SpaceK.txt
12:19:42:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git status
On branch SpaceK
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   SpaceK.txt

12:19:44:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git commit -m "Add best joke, ever"
[SpaceK 7b52c35] Add best joke, ever
 1 file changed, 2 insertions(+)
```

(5 of 6) Pushing your joke

Push your branch to remote

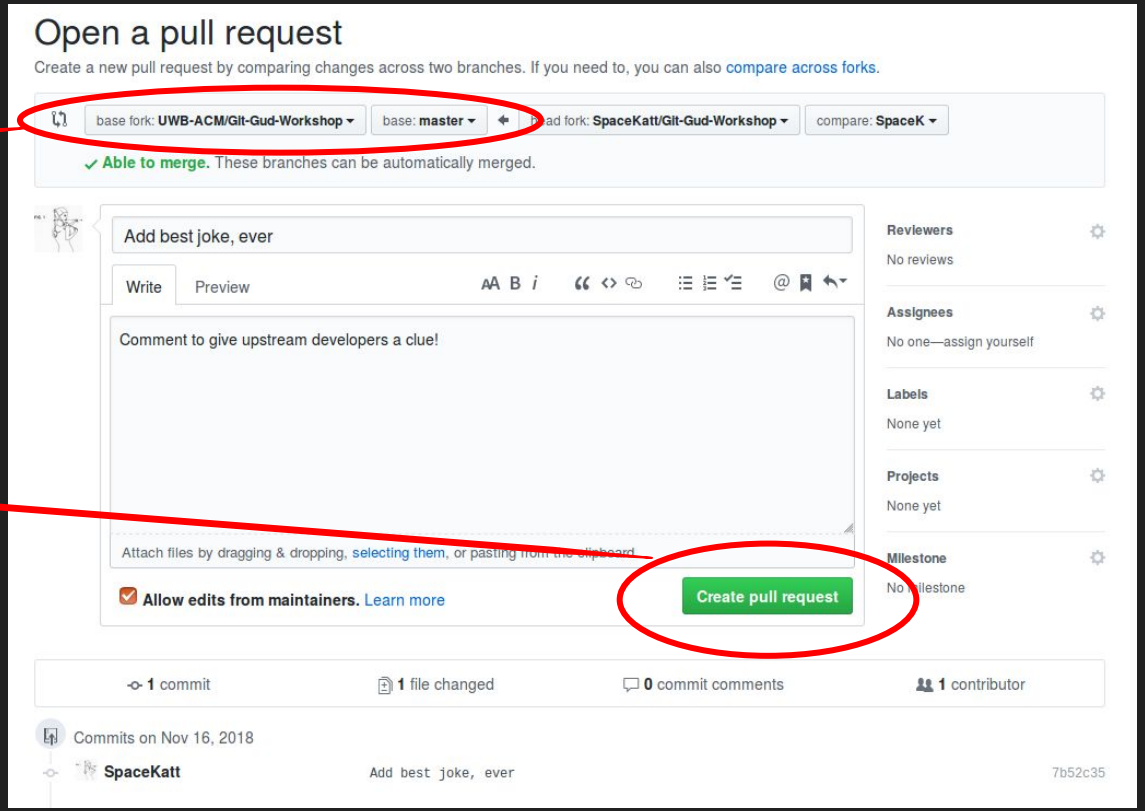
Remote is by default “origin”, from when we cloned the repo

For the next step (making a pull request) visit the URL shown below

```
12:20:01:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git push origin SpaceK
Username for 'https://github.com': spacekatt
Password for 'https://spacekatt@github.com':
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 424 bytes | 0 bytes/s, done.
Total 4 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'SpaceK' on GitHub by visiting:
remote:      https://github.com/SpaceKatt/Git-Gud-Workshop/pull/new/SpaceK
remote:
To https://github.com/SpaceKatt/Git-Gud-Workshop.git
 * [new branch]      SpaceK -> SpaceK
```

(6 of 6) Creating a pull request

- Ensure “base fork” is set to master from “upstream”
  - Upstream is remote you forked repo from
- Leave a comment
- Click “Create pull request”

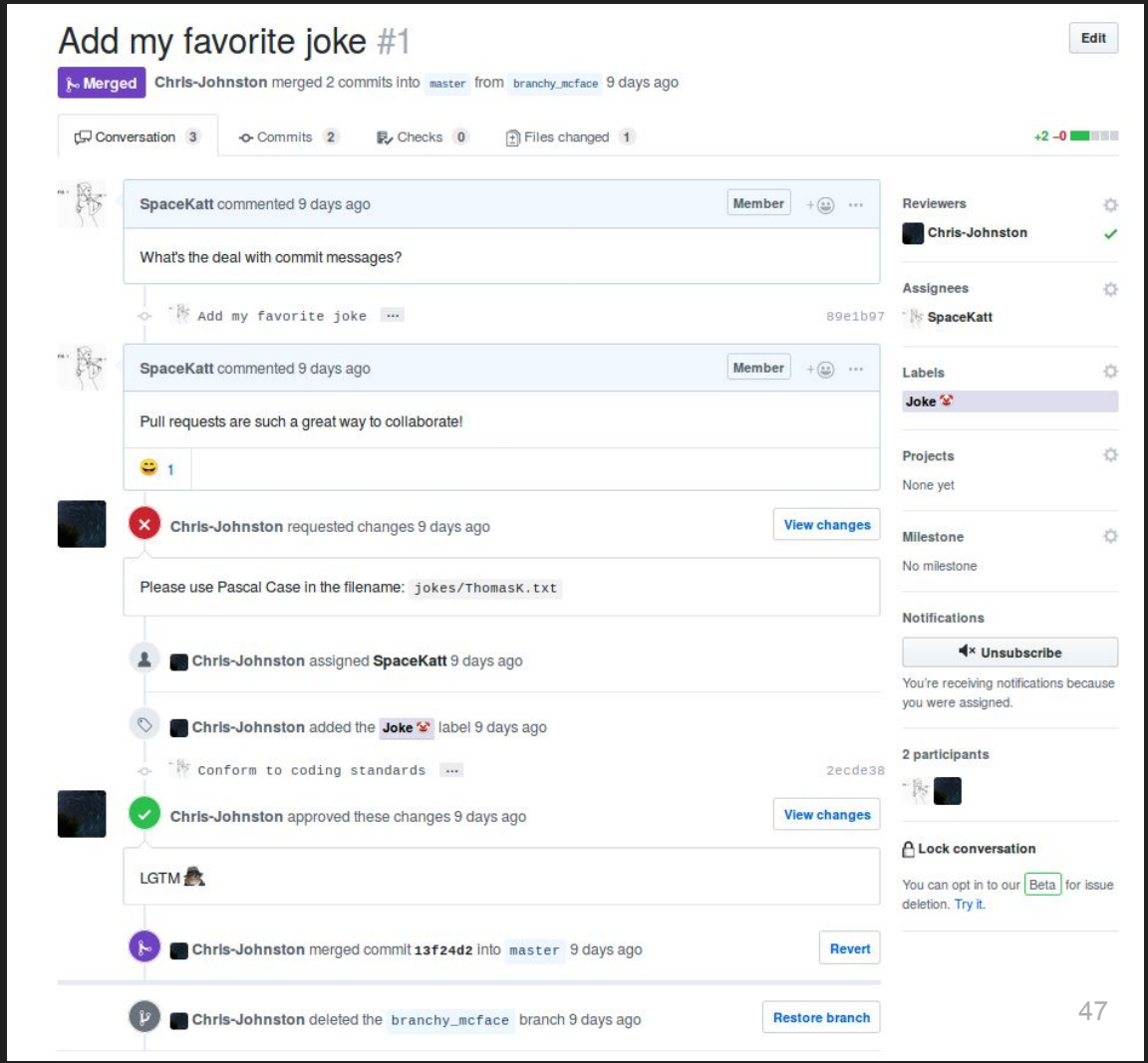


Now others can comment on your changes and approve them for integration into the code base.

Collaborate!!!

Pull requests are a great way to collaborate with your teammates

- Code reviews
- Code state management
- Greater visibility of changes



After your changes get ***merged***...

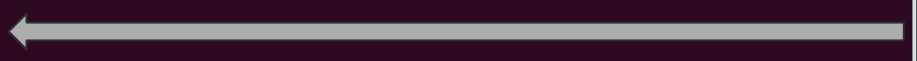
To sync back with master after someone mergers your PR…

`git remote add upstream <upstream_url>` (Add remote reference to fork's upstream)

`git checkout master` (Switch back to master branch)

`git pull upstream master` (Pull changes from remote [GitHub])

```
12:31:11:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git remote add upstream git@github.com:UWB-ACM/Git-Gud-Workshop.git
12:31:22:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
12:31:28:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git pull upstream master
Enter passphrase for key '/home/spacekatt/.ssh/id_rsa':
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From github.com:UWB-ACM/Git-Gud-Workshop
 * branch            master     -> FETCH_HEAD
 * [new branch]      master     -> upstream/master
Updating 74031ae..9634c6c
Fast-forward
 jokes/SpaceK.txt | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 jokes/SpaceK.txt
```



See our new file!?!?

Part II: After your changes get ***merged***...

Update our remote fork after we fetch upstream changes

`git push origin master` (Synchronize our forked remote with changes from upstream)

```
12:31:50:spacekatt@spacekatt-Latitude-E6540:~/ACM/tmp/Git-Gud-Workshop/jokes$ git push origin master
Username for 'https://github.com': spacekatt
Password for 'https://spacekatt@github.com':
Counting objects: 1, done.
Writing objects: 100% (1/1), 655 bytes | 0 bytes/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To https://github.com/SpaceKatt/Git-Gud-Workshop.git
   74031ae..9634c6c  master -> master
```

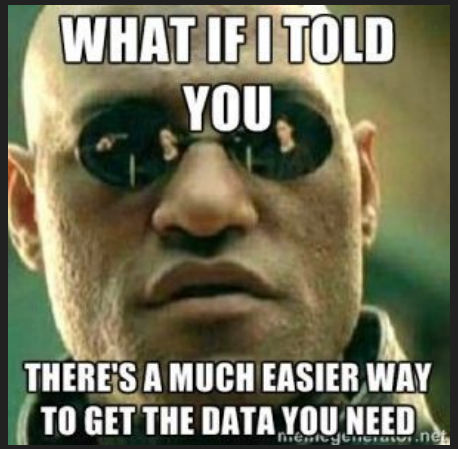
Git Flow

“Git flow” is a pattern for how you can collaborate with others on the same codebase. We just did this!

GitHub explains this better than we can: (open this link)

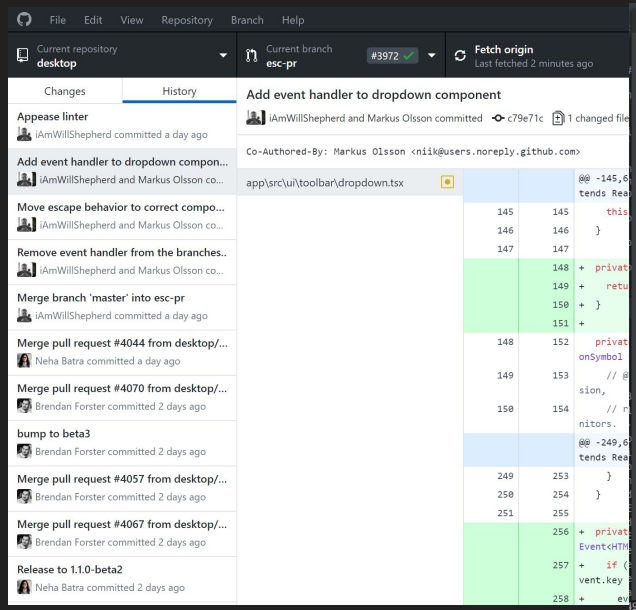
https://guides.github.com/introduction/flow/

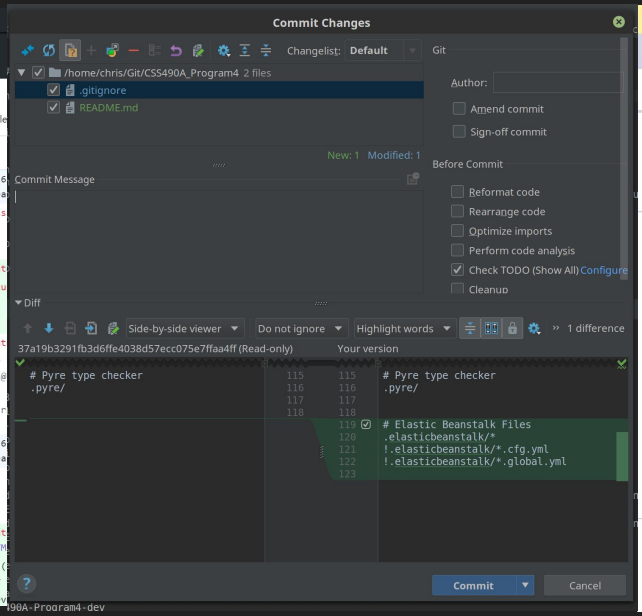
https://goo.gl/jgeusu

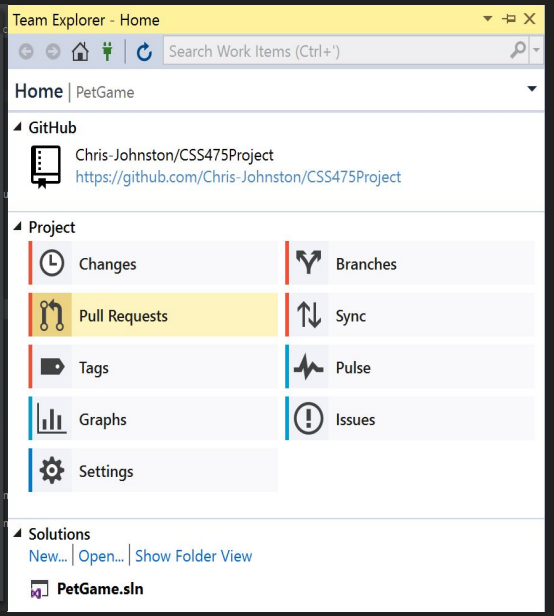


Git GUIs

Nearly every modern IDE will have a Git plugin (so does vim)







GitHub Desktop is a Git client that is oriented around GitHub. It works with non-GitHub repos too.

The JetBrains IDEs (IntelliJ, PyCharm, CLion) have pretty good Git integration.

Visual Studio has a pretty solid Git integration.

Further Topics

GitHub Issues

Markdown & making a good readme

Continuous Integration & Continuous Deployment

Releases / tags

Licensing

In-Depth Merging

Rebasing/merge conflicts

Git Stash

Gitignore

Gitattributes

Github-specific stuff

… and a lot more.

If you have *questions* right now, we can try to give a brief explanation.

We encourage you to learn these on your own! Practice! Try using Git for your programming assignments.

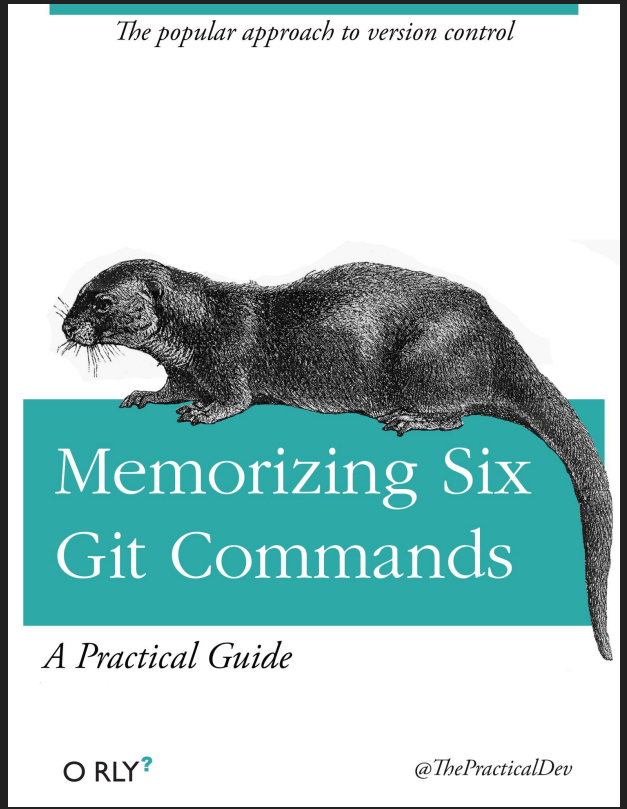
Great Resources for Further Reading

The official Git documentation: https://git-scm.com/doc

GitHub documentation: https://guides.github.com/

Atlassian Git tutorials: https://www.atlassian.com/git/tutorials

And of course: Google, Stack Overflow, and your peers!



Lessons Learned

- Why git is important
- How to configure git CLI
- How to make a commit
- How to push local repository to remote (GitHub)
- How to manage files with git
- How to contribute to open source projects
- Identified more advanced topics for further consideration
- **Git is fun!**

Questions?
Comments?
Concerns?

Anything from the previous slides or b e y o n d?