Intro to Git/Github

Source control for research software development.

Presented by HMS WARP DevOps

HARVARD MEDICAL SCHOOL | Research Computing

What I hope you take away.

- The value of Source control
- A github.com repo
- A basic understanding of the technology and tools
- Ability to update files in a repo
- Create and merge branches.

Topics

- General
  - What is source control?
  - Why source control?
  - What is git?
  - What is github?
- Working with Git
  - Tools
    - git command line
    - github website
- Demo and walkthrough

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.

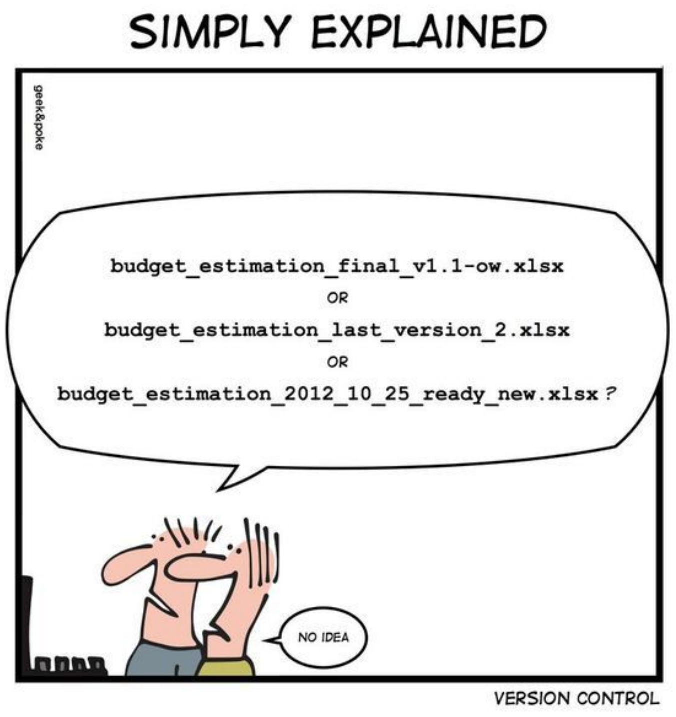
https://xkcd.com/1597/

What is source control?

“A component of software configuration management, version control, also known as revision control or source control, is the management of changes to documents,” - wikipedia (version control)

- track changes
- account for changes
- undo changes

Why source control?


SIMPLY EXPLAINED
geek&poke
budget_estimation_final_v1.1-ow.xlsx
OR
budget_estimation_last_version_2.xlsx
OR
budget_estimation_2012_10_25_ready_new.xlsx ?
NO IDEA
VERSION CONTROL

Idea from Jen Simmons and John Albin Wilkins during episode #40 of "Web Ahead" about Git: http://5by5.tv/webahead/40

Versioning

What is the most current copy

History

The value of change over time

Accountability

Who made what changes when

Context

Why was a change made

What is git?

Git is a distributed version control system.

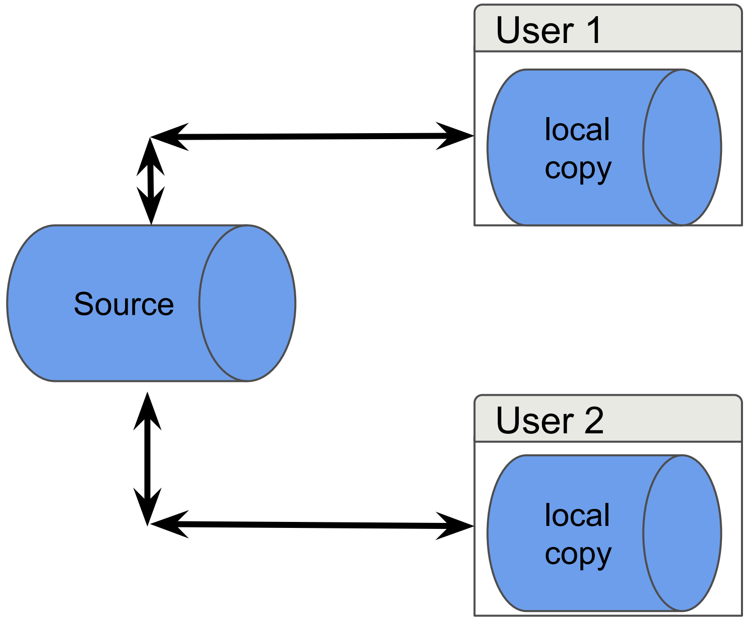
Central repository is not required

Every user gets a full copy of the repo including history

Changes can be branched and merged from any point

Collaboration is built into the structure

Originally created by Linus Trovalds to manage the linux kernel code



What is github?

www.github.com

Repo management service

Free for individual users and educational groups

Additional tools for automation and collaboration

There are many alternatives to github including self hosted services but github is by far the most popular.
Github.com is currently owned by Microsoft.

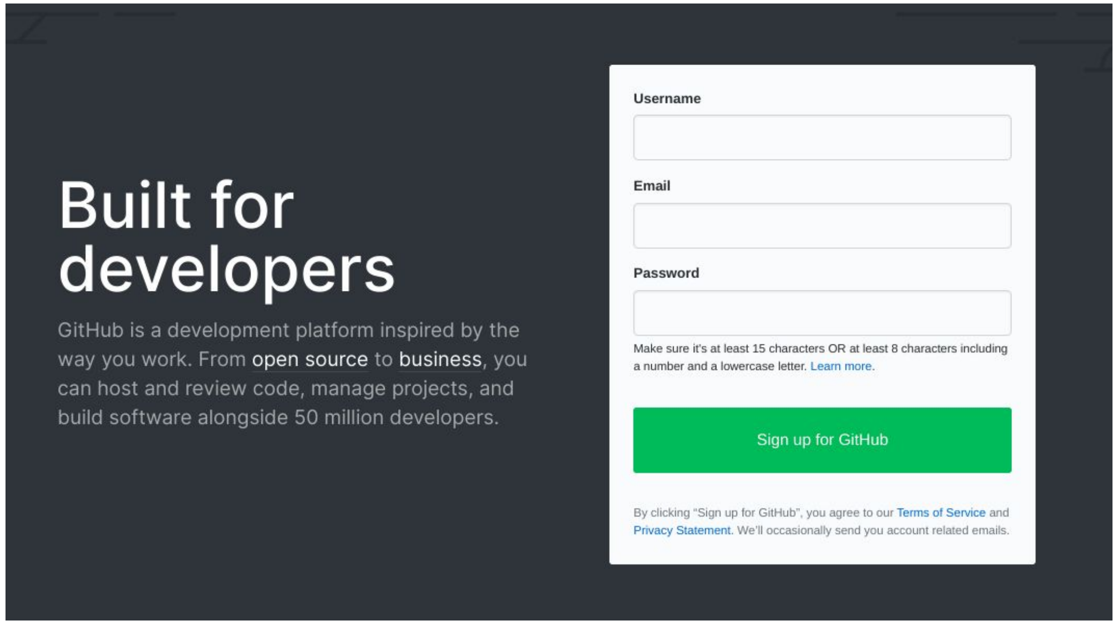
Git tools

Where to get git.

- https://git-scm.com/downloads
- `` `homebrew install git` ``
- `` `sudo yum install git` ``
- `` `sudo apt-get install git` ``
- Already available on all o2 nodes
- https://hub.github.com/ - command line enhancements
- https://git-scm.com/downloads/guis - graphical git

Github.com

- Largest single repo host
- Used for all sorts of projects
- Easy to collaborate
- Free and paid models



Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside 50 million developers.

Username

Email

Password

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. Learn more.

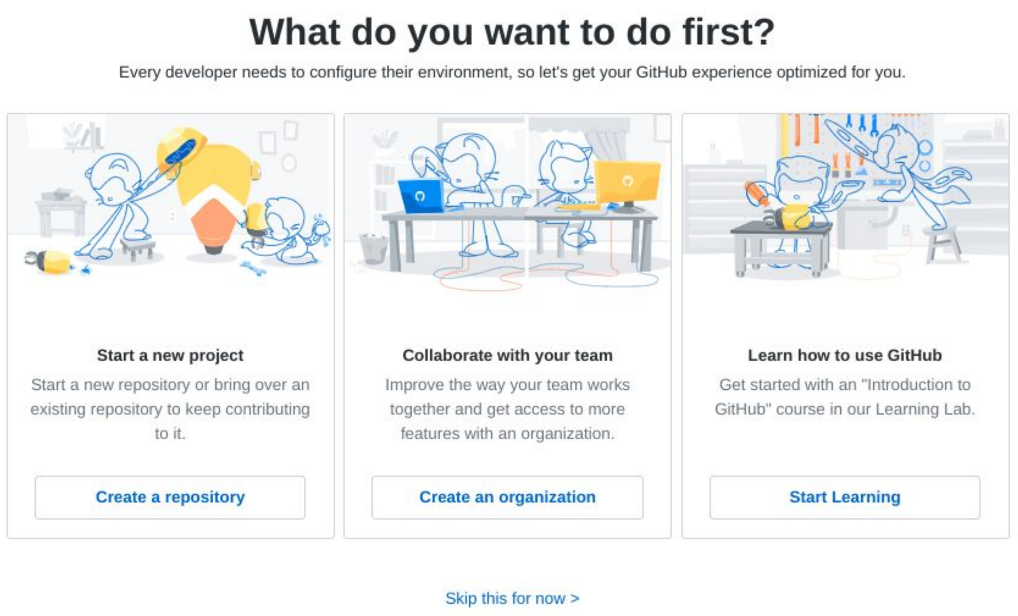
By clicking "Sign up for GitHub", you agree to our Terms of Service and Privacy Statement. We'll occasionally send you account related emails.

Creating an account

- Live walkthrough

  …

- What's next?



What do you want to do first?

Every developer needs to configure their environment, so let's get your GitHub experience optimized for you.

Start a new project

Start a new repository or bring over an existing repository to keep contributing to it.

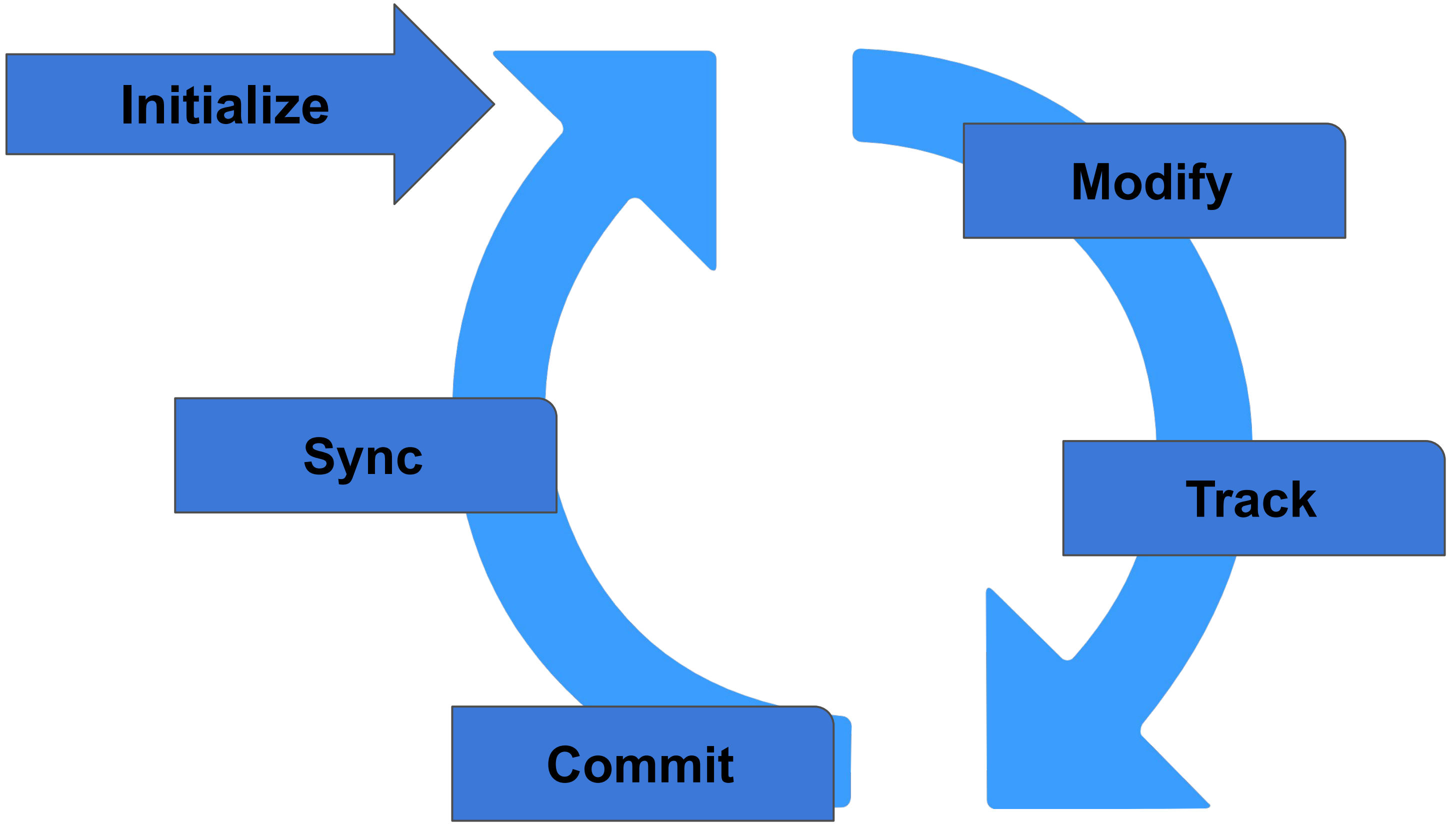
Collaborate with your team

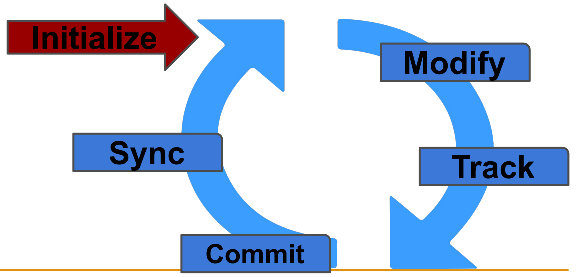
Improve the way your team works together and get access to more features with an organization.

Learn how to use GitHub

Get started with an "Introduction to GitHub" course in our Learning Lab.

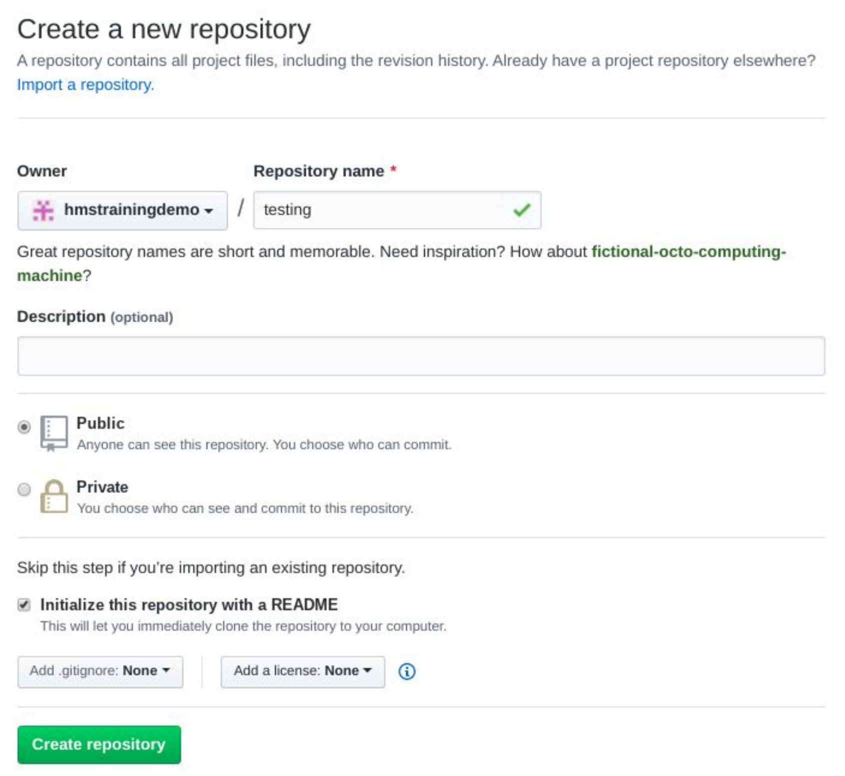
Version contorl lifecycle





First repo!

- Repo name
- Private repo
- Create Readme
- Click Create repository


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: hmstrainingdemo / Repository name *: testing

Great repository names are short and memorable. Need inspiration? How about fictional-octo-computing-machine?

Description (optional)

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

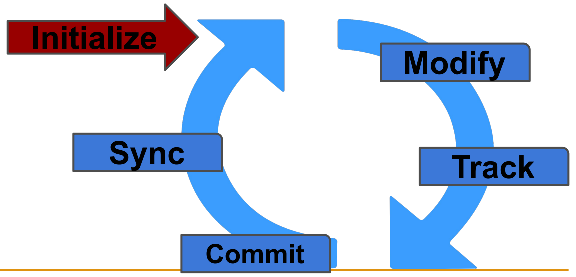
○ Private
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☑ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None

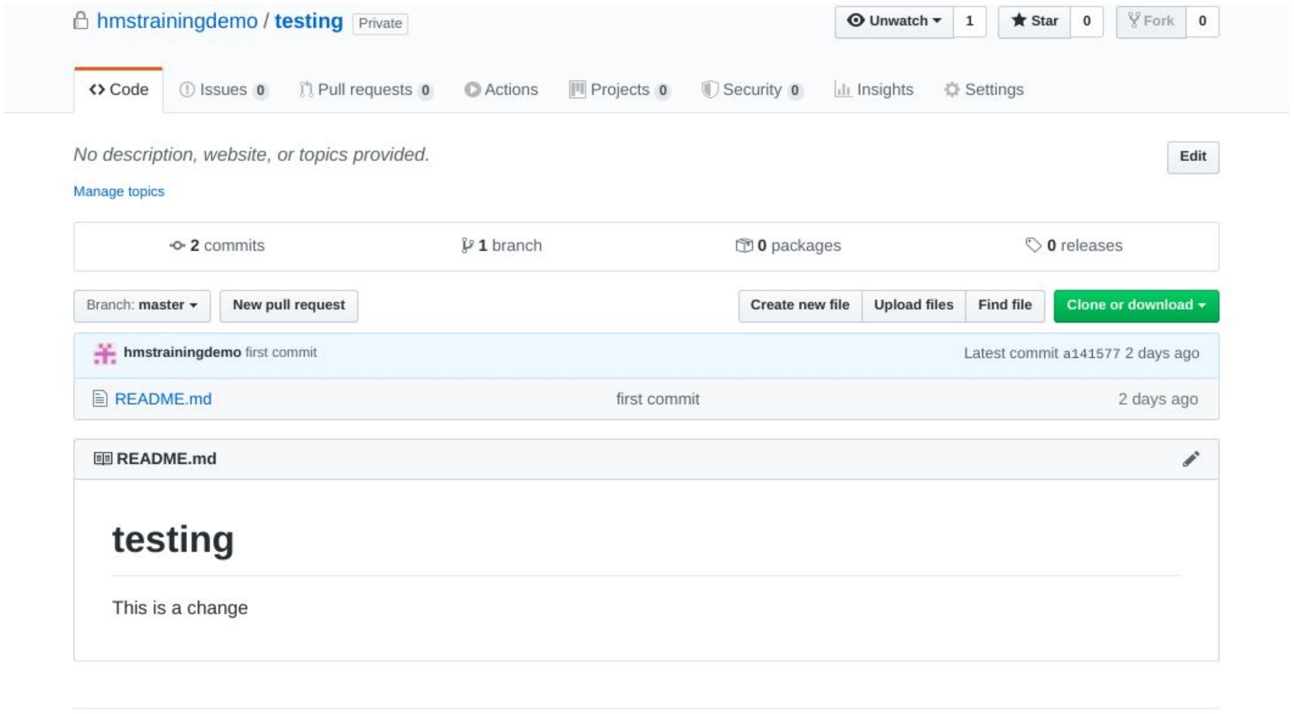
Create repository



The repo interface

- Tour!

- Edit the Readme file
- Commit changes
- View history



No description, website, or topics provided.

hmstrainingdemo first commit — Latest commit a141577 2 days ago

README.md — first commit — 2 days ago

README.md

testing

This is a change

Git config

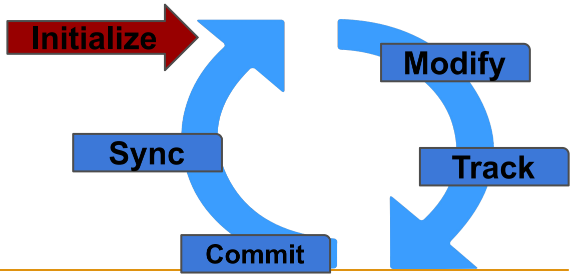
Get config with `` `git config --list` ``

Setting a user

```
`git config --global user.name “hms training”`
`git config --global user.email hmstrainingdemo@gmail.com`
```

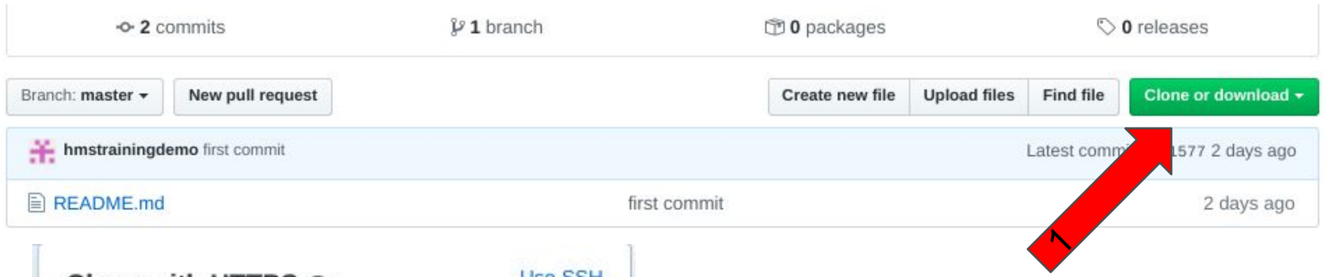
Sets user info for your changes

The `.gitconfig` file in your home directory stores this info

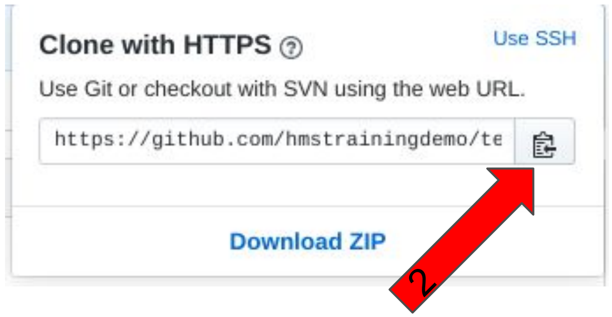


Cloning your repo

Click "Clone or download"

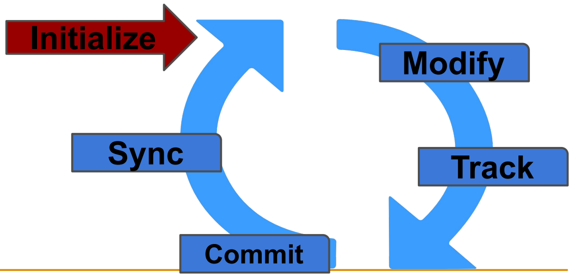


Click the clipboard icon


Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/hmstrainingdemo/te
Download ZIP
2

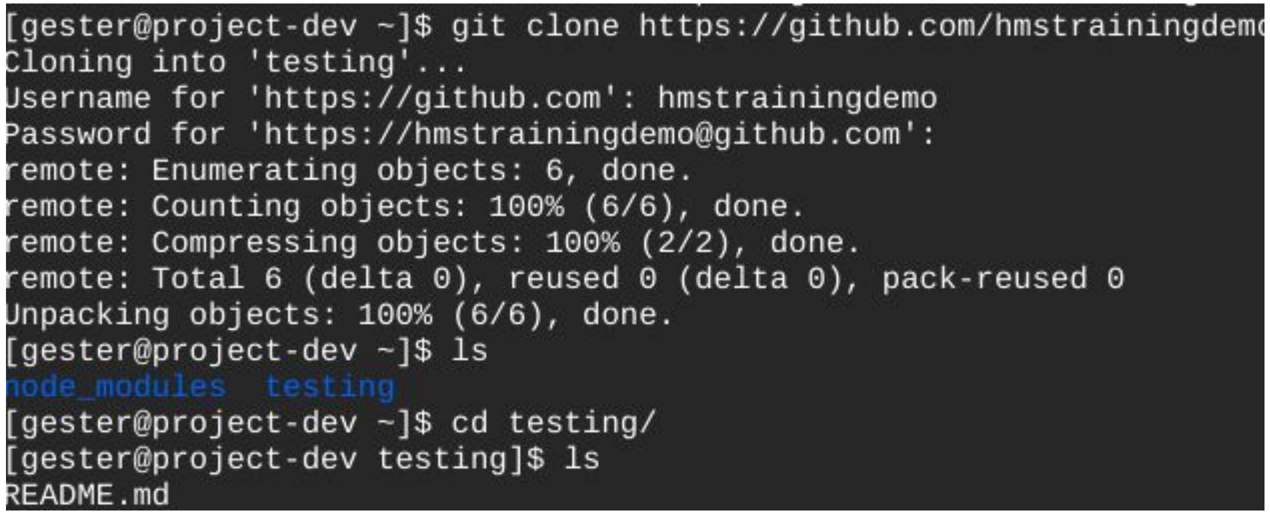
Switch to your terminal and type `git clone `

and paste in your url (ctrl+v)



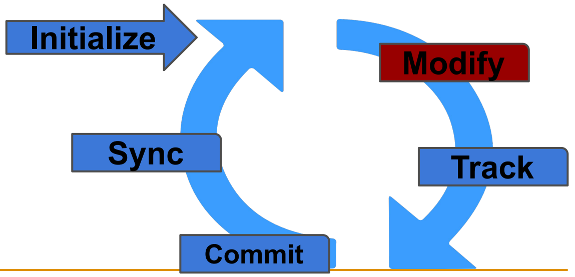
Cloning your repo (cont)

You should see something like this…



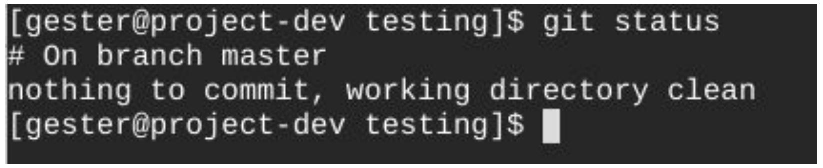
```
[gester@project-dev ~]$ git clone https://github.com/hmstrainingdem
Cloning into 'testing'...
Username for 'https://github.com': hmstrainingdemo
Password for 'https://hmstrainingdemo@github.com':
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
[gester@project-dev ~]$ ls
node_modules  testing
[gester@project-dev ~]$ cd testing/
[gester@project-dev testing]$ ls
README.md
```

you can see your new directory, in this case “testing”.
You should also be able to see your `README.md` file inside it.



Git status and making changes

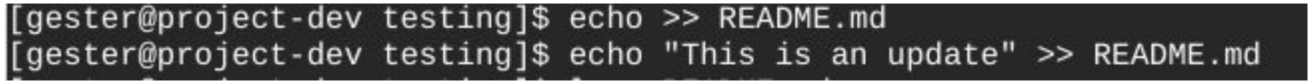
Shows the current state of your copy of a git repo



```
[gester@project-dev testing]$ git status
# On branch master
nothing to commit, working directory clean
[gester@project-dev testing]$ █
```

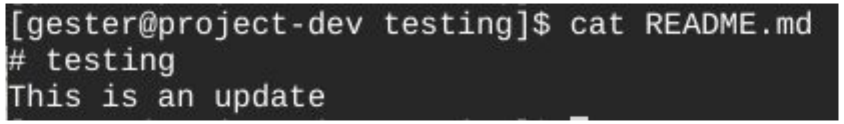
Lets update the `README.md` file with;

```
echo >> README.md
echo "This is an update" >> README.md
```



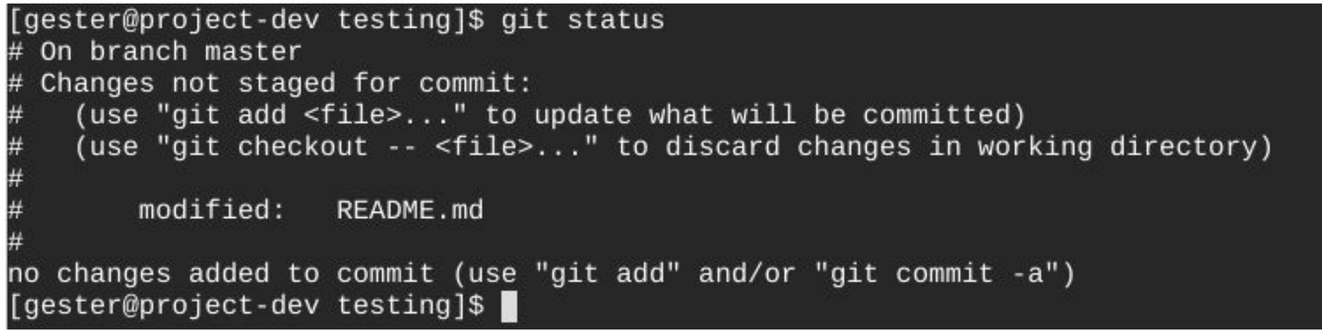
```
[gester@project-dev testing]$ echo >> README.md
[gester@project-dev testing]$ echo "This is an update" >> README.md
```

We can see our updates with `cat`

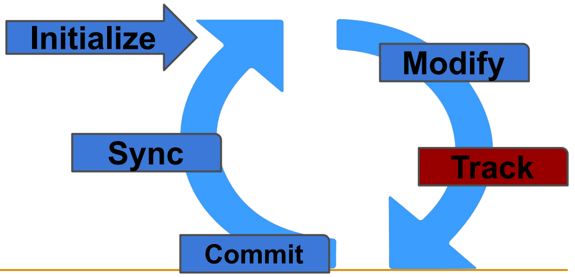


```
[gester@project-dev testing]$ cat README.md
# testing
This is an update
```

We can now see our `git status` command has changed



```
[gester@project-dev testing]$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   README.md
#
no changes added to commit (use "git add" and/or "git commit -a")
[gester@project-dev testing]$ █
```



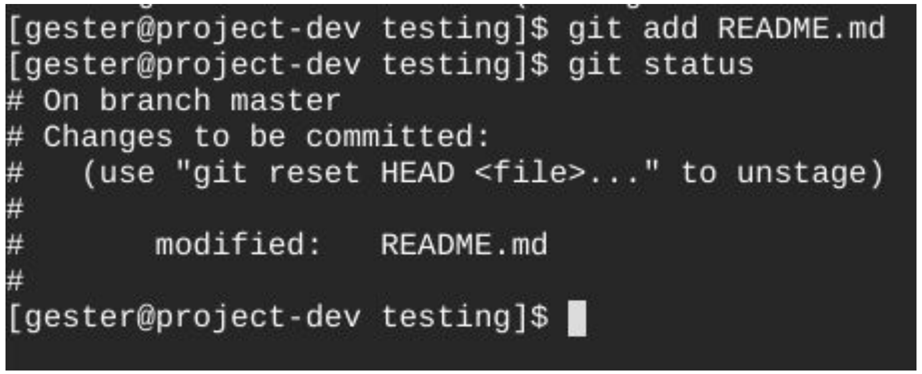
Tracking a change

Now we need to let git know we want to keep this change.

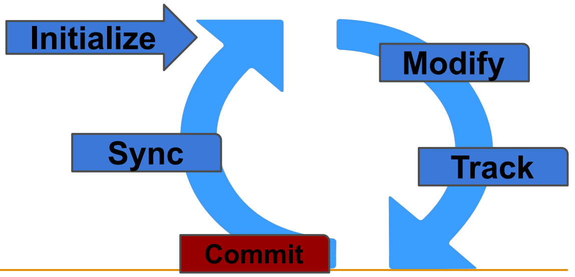
First we add it to the list of files we want to commit, this is also called staging.

`git add README.md`

This produces no output but our `git status` will change.



```
[gester@project-dev testing]$ git add README.md
[gester@project-dev testing]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README.md
#
[gester@project-dev testing]$ 
```



Commiting the change

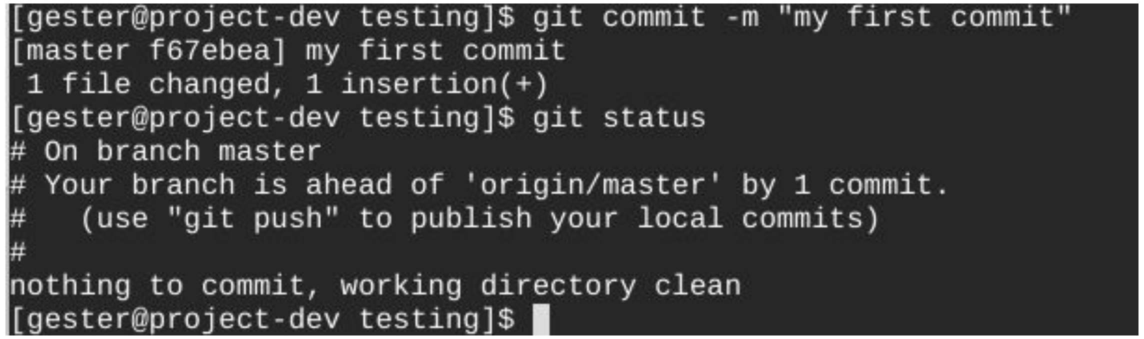
Now we need to tell git to save this change in the file history.

`` `git commit` `` allows us to do this.

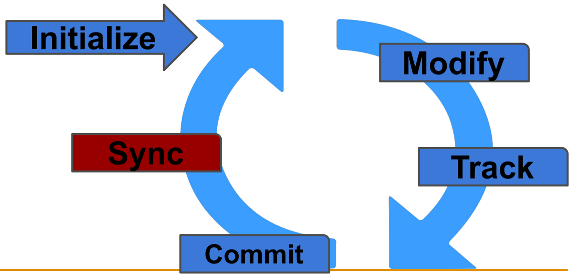
Commits require a commit message, this allows you to give context to your changes. We will do this with the `-m` flag.

`` `git commit -m "my first commit"` ``

As we should expect now this changes `` `git status` `` again.

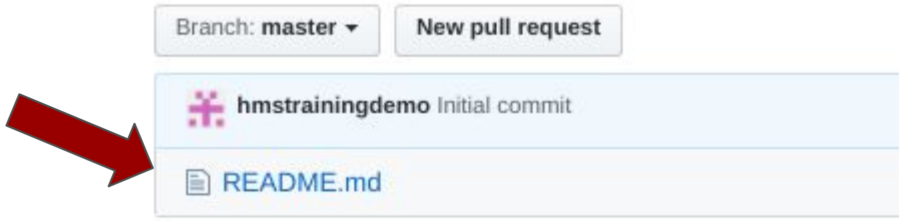


```
[gester@project-dev testing]$ git commit -m "my first commit"
[master f67ebea] my first commit
 1 file changed, 1 insertion(+)
[gester@project-dev testing]$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
nothing to commit, working directory clean
[gester@project-dev testing]$ 
```

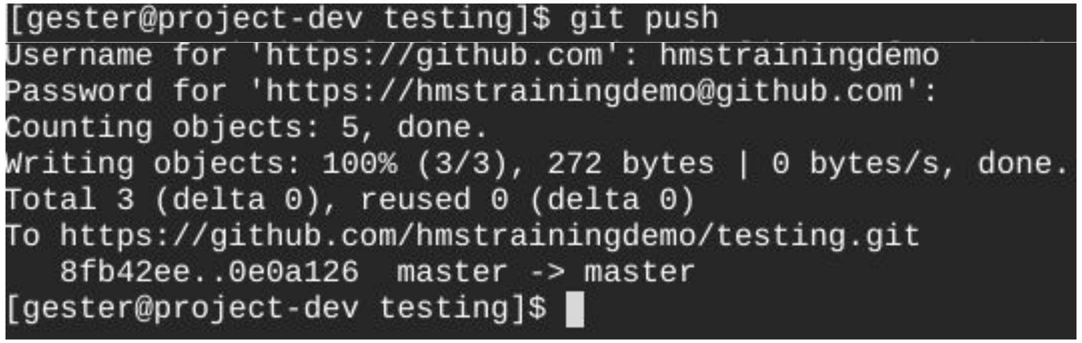


Git push

Switching to Github we can click on our file but we don't see our changes



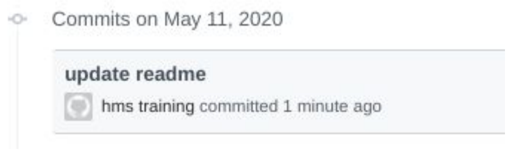
We have to send our changes to to the remote repo with `git push`



```
[gester@project-dev testing]$ git push
Username for 'https://github.com': hmstrainingdemo
Password for 'https://hmstrainingdemo@github.com':
Counting objects: 5, done.
Writing objects: 100% (3/3), 272 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/hmstrainingdemo/testing.git
   8fb42ee..0e0a126  master -> master
[gester@project-dev testing]$ 
```

Now if we check the history we will see our new commit

And if we click on the file we will see our changes.



Commits on May 11, 2020

update readme

hms training committed 1 minute ago

Review so far

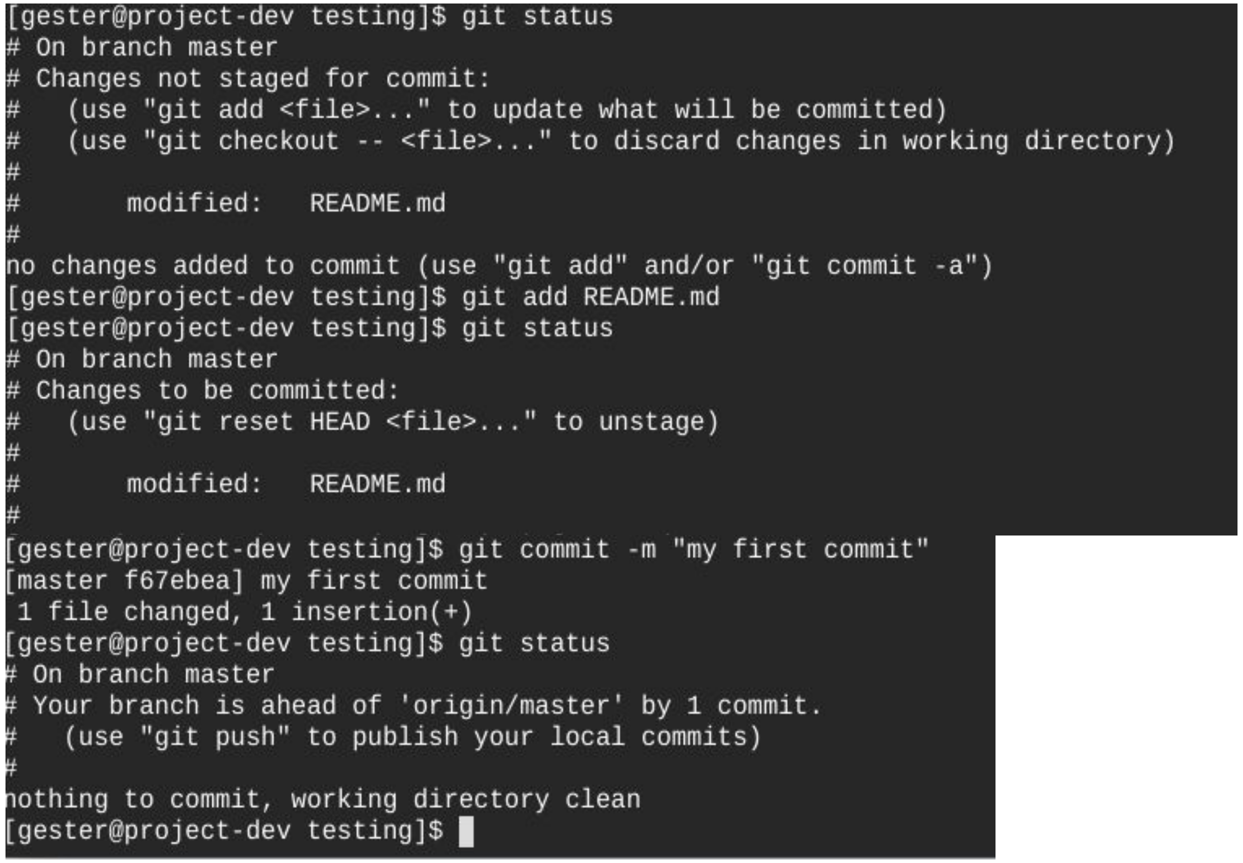
So what just happened here?

- We changed a file
- Told git to track the change
- Saved that change to the history

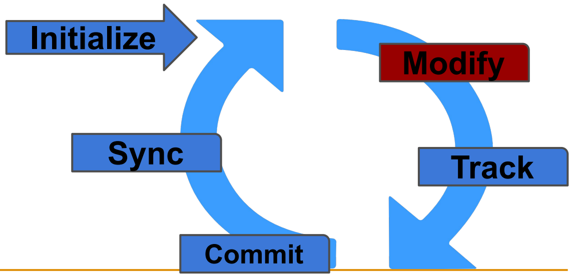
Things to note:

`[master f67ebea]` Is showing a hash.

A git hash is a unique identifier for a given commit. This is just the beginning of the hash.



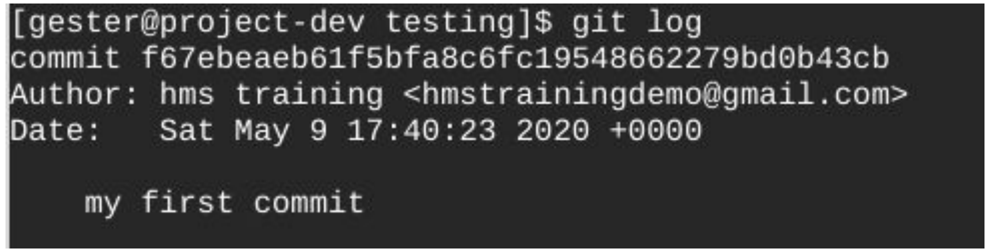
```
[gester@project-dev testing]$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   README.md
#
no changes added to commit (use "git add" and/or "git commit -a")
[gester@project-dev testing]$ git add README.md
[gester@project-dev testing]$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README.md
#
[gester@project-dev testing]$ git commit -m "my first commit"
[master f67ebea] my first commit
 1 file changed, 1 insertion(+)
[gester@project-dev testing]$ git status
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#   (use "git push" to publish your local commits)
#
nothing to commit, working directory clean
[gester@project-dev testing]$
```



git time travel

`git log` - show a list of all previous commits to the current branch

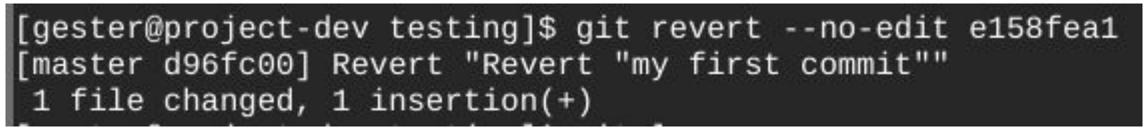
Notice that the first 8 characters of the commit hash match what was returned on when you committed the change



```
[gester@project-dev testing]$ git log
commit f67ebeaeb61f5bfa8c6fc19548662279bd0b43cb
Author: hms training <hmstrainingdemo@gmail.com>
Date:   Sat May 9 17:40:23 2020 +0000

    my first commit
```

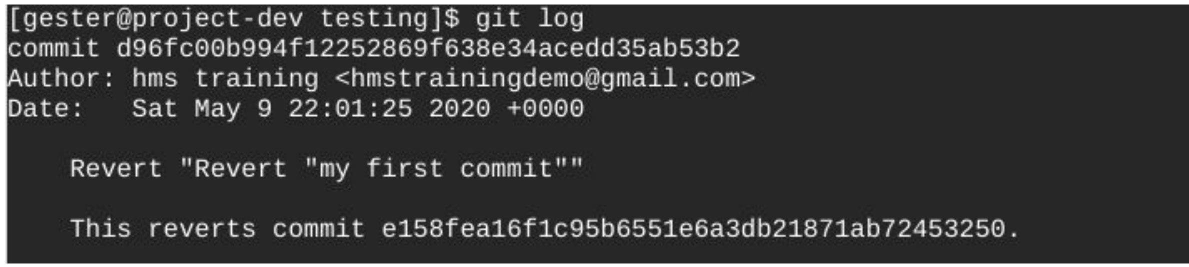
`git revert` - go back in time

Use `--no-edit` for the demo.
revert needs a hash to revert to. Notice you don't need the entire hash, just enough to be unique (usually the first 8 chars)



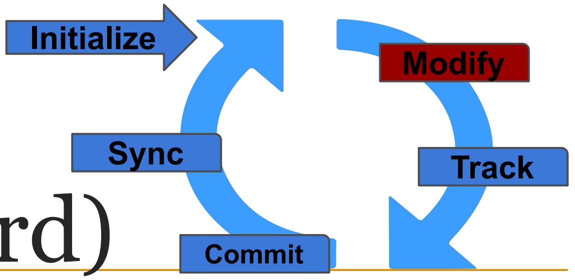
```
[gester@project-dev testing]$ git revert --no-edit e158fea1
[master d96fc00] Revert "Revert "my first commit""
 1 file changed, 1 insertion(+)
```

`git log` now shows a new entry labeled revert

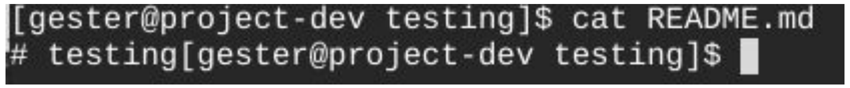


```
[gester@project-dev testing]$ git log
commit d96fc00b994f12252869f638e34acedd35ab53b2
Author: hms training <hmstrainingdemo@gmail.com>
Date:   Sat May 9 22:01:25 2020 +0000

    Revert "Revert "my first commit""

    This reverts commit e158fea16f1c95b6551e6a3db21871ab72453250.
```

Git time travel (cont, time travel is hard)



If we look at our `README.md` file now we will see that the last thing we added is gone.

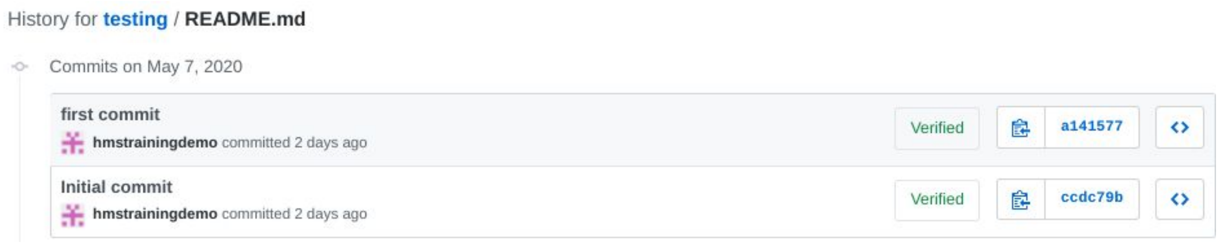


```
[gester@project-dev testing]$ cat README.md
# testing[gester@project-dev testing]$ 
```

Wibbily Wobbaly Timey Wimey

`` `Git checkout [hash]` `` - travel to any point in the history of a git repo
You can use the log to find any point in your repo and instantly move to that point in time.
There you can either inspect all the files at that state, run the code, or make a new branch.
Time travel is confusing and can be dangerous, be careful!

History is also available in the github web interface.

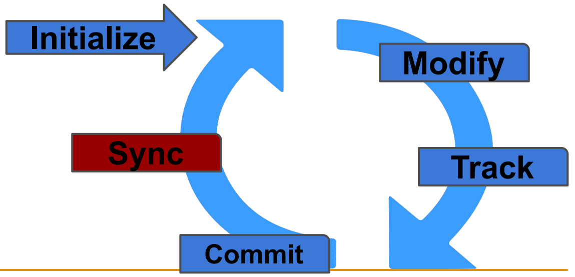


History for testing / README.md

Commits on May 7, 2020

first commit
hmstrainingdemo committed 2 days ago — Verified — a141577

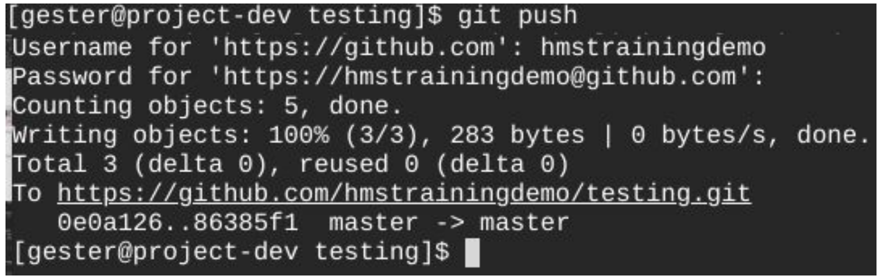
Initial commit
hmstrainingdemo committed 2 days ago — Verified — ccdc79b



Push the revert to remote

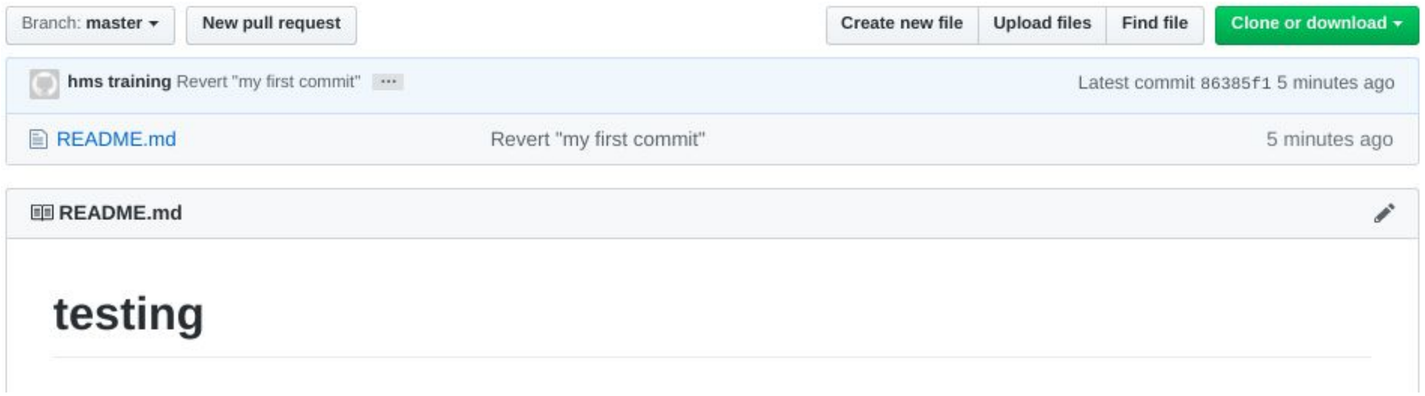
Note: `` `git revert` `` gets you the track and commit steps for free

To make sure the remote has the change removed as well we need to run `` `git push` `` again



```
[gester@project-dev testing]$ git push
Username for 'https://github.com': hmstrainingdemo
Password for 'https://hmstrainingdemo@github.com':
Counting objects: 5, done.
Writing objects: 100% (3/3), 283 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/hmstrainingdemo/testing.git
   0e0a126..86385f1  master -> master
[gester@project-dev testing]$
```

Now we will see those changes reflected on github

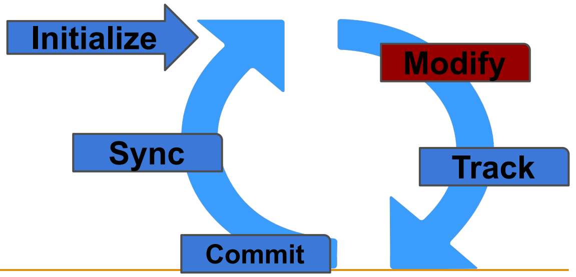


hms training Revert "my first commit" ... Latest commit 86385f1 5 minutes ago

README.md Revert "my first commit" 5 minutes ago

README.md

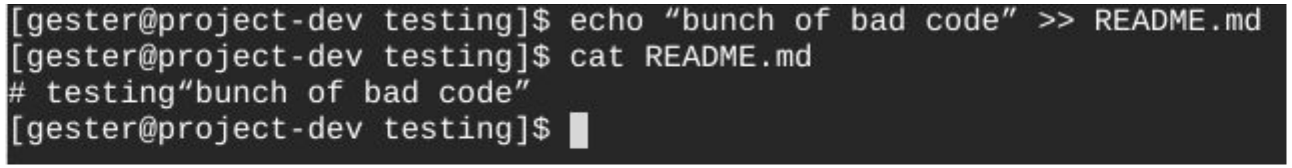
testing



Undoing a change

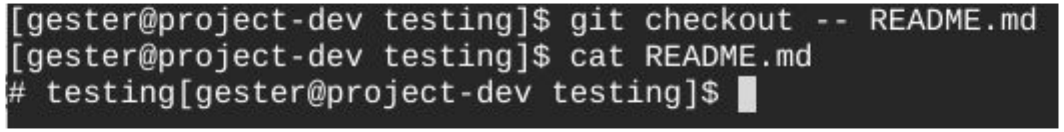
How to quickly revert a file before committing it.

Let's make a quick change to our file
`` `echo "bunch of bad code" >> README.md` ``



```
[gester@project-dev testing]$ echo "bunch of bad code" >> README.md
[gester@project-dev testing]$ cat README.md
# testing"bunch of bad code"
[gester@project-dev testing]$
```

We can then use `` `git checkout -- README.md` `` to roll back to the last committed state of a file



```
[gester@project-dev testing]$ git checkout -- README.md
[gester@project-dev testing]$ cat README.md
# testing[gester@project-dev testing]$
```

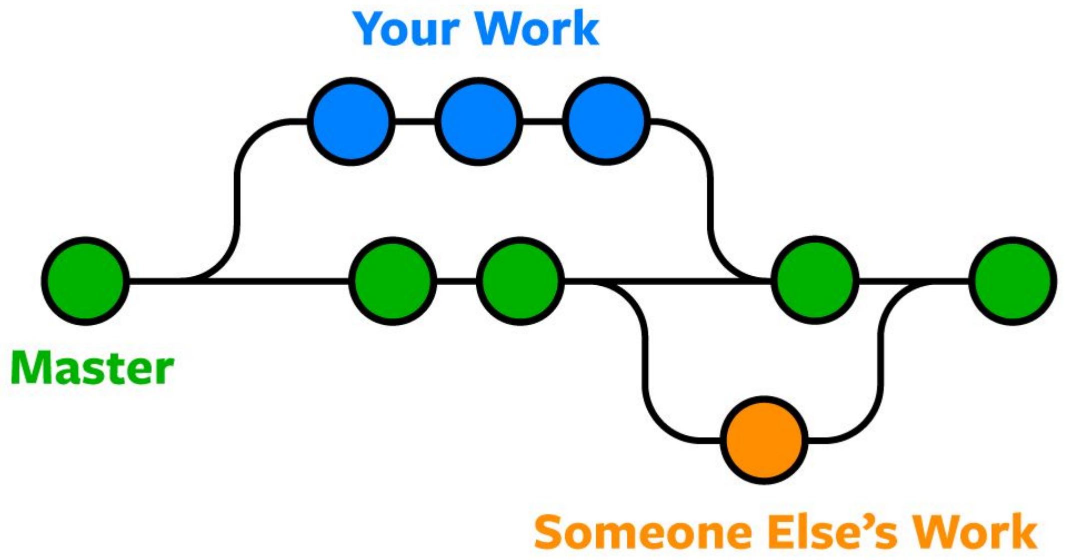
`` `git checkout --` `` without specifying a file will undo all committed changes in the current branch

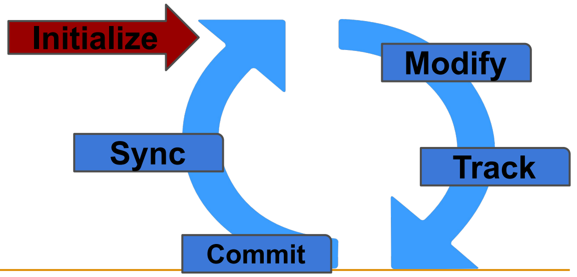
Collaboration

Collaboration in source control is the ability for multiple people to work on the same code at the once.

For this class we will use the git branch and merge model.

Rough depiction of this workflow.

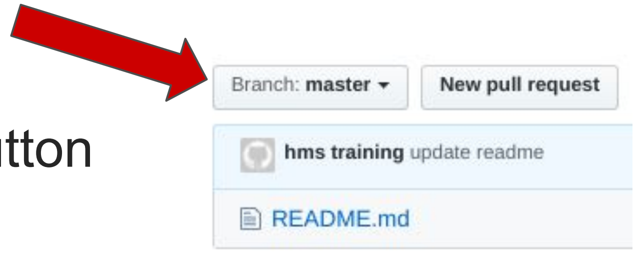




Creating a new branch

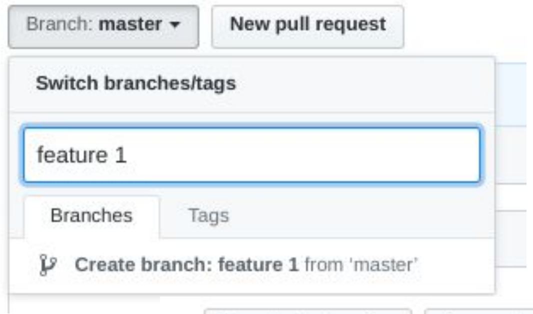
On github.com:

Click the “Branch: Master” button



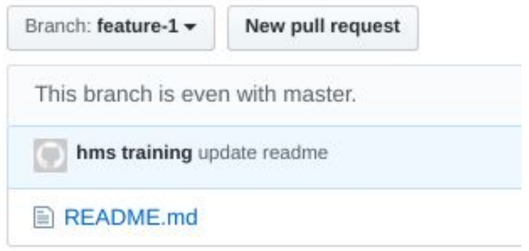
hms training update readme
README.md

Enter a new branch name in the box and hit enter



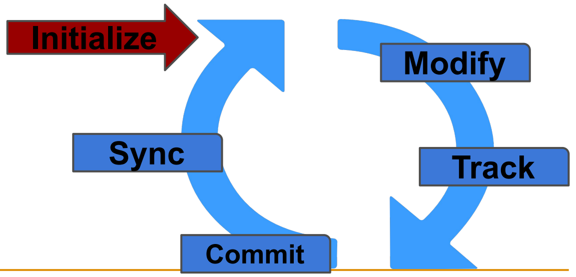
feature 1
Create branch: feature 1 from 'master'

We are now on the new branch



This branch is even with master.
hms training update readme
README.md

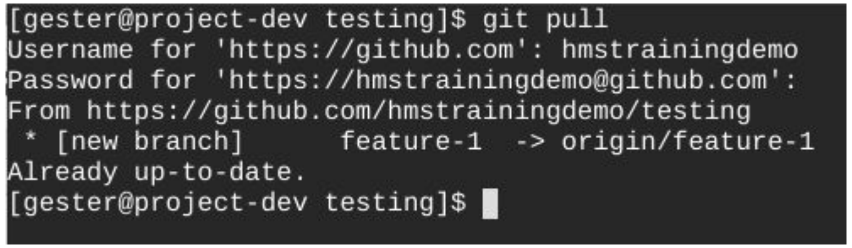
Note: the addition of a “-” in our branch name. Git doesn’t like blank spaces.



git pull and git checkout

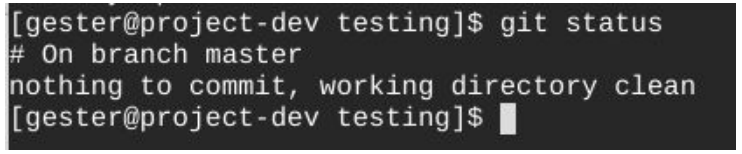
Back on the command line:

We will run a `git pull` to get the latest data from our github repo



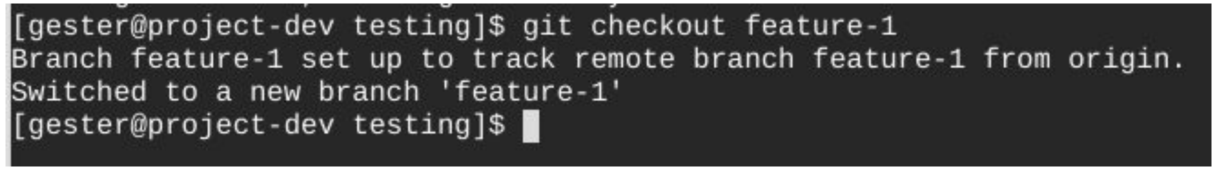
```
[gester@project-dev testing]$ git pull
Username for 'https://github.com': hmstrainingdemo
Password for 'https://hmstrainingdemo@github.com':
From https://github.com/hmstrainingdemo/testing
 * [new branch]      feature-1  -> origin/feature-1
Already up-to-date.
[gester@project-dev testing]$ 
```

Git helpfully tells us there is a new branch but `git status` still shows us on the master



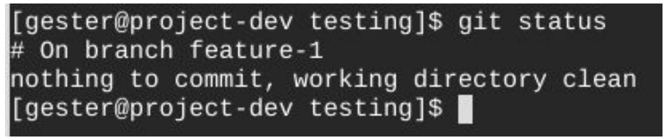
```
[gester@project-dev testing]$ git status
# On branch master
nothing to commit, working directory clean
[gester@project-dev testing]$ 
```

We can change to our new branch with `git checkout`

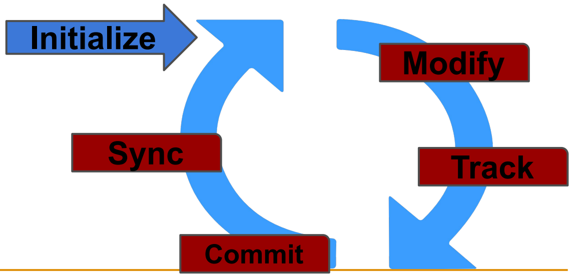


```
[gester@project-dev testing]$ git checkout feature-1
Branch feature-1 set up to track remote branch feature-1 from origin.
Switched to a new branch 'feature-1'
[gester@project-dev testing]$ 
```

Now `git status` shows what we want



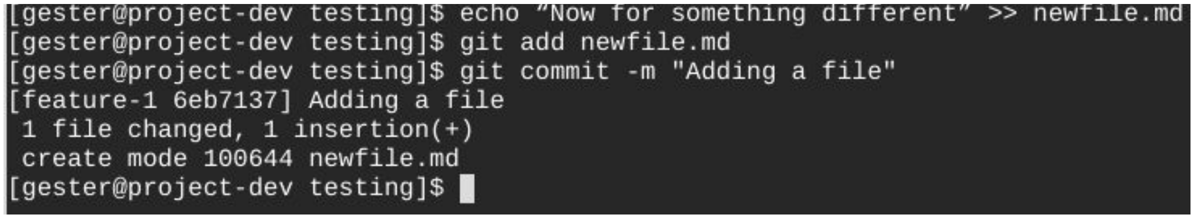
```
[gester@project-dev testing]$ git status
# On branch feature-1
nothing to commit, working directory clean
[gester@project-dev testing]$ 
```



Updating a branch

A branch is your working copy of the state of the code at the time the branch was made.

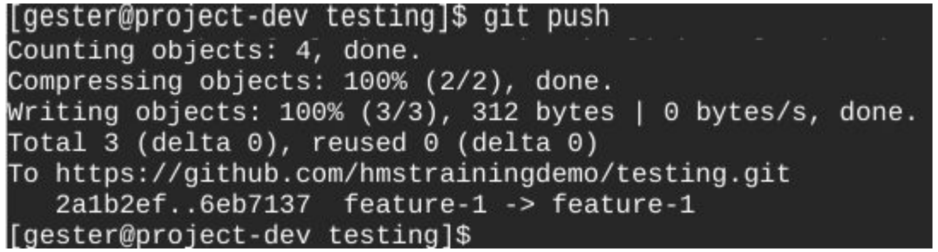
Let's make an update, we'll create a new file with `` `echo "Now for something different" >> newfile.md` ``



```
[gester@project-dev testing]$ echo "Now for something different" >> newfile.md
[gester@project-dev testing]$ git add newfile.md
[gester@project-dev testing]$ git commit -m "Adding a file"
[feature-1 6eb7137] Adding a file
 1 file changed, 1 insertion(+)
 create mode 100644 newfile.md
[gester@project-dev testing]$
```

We can also add and commit this file as well

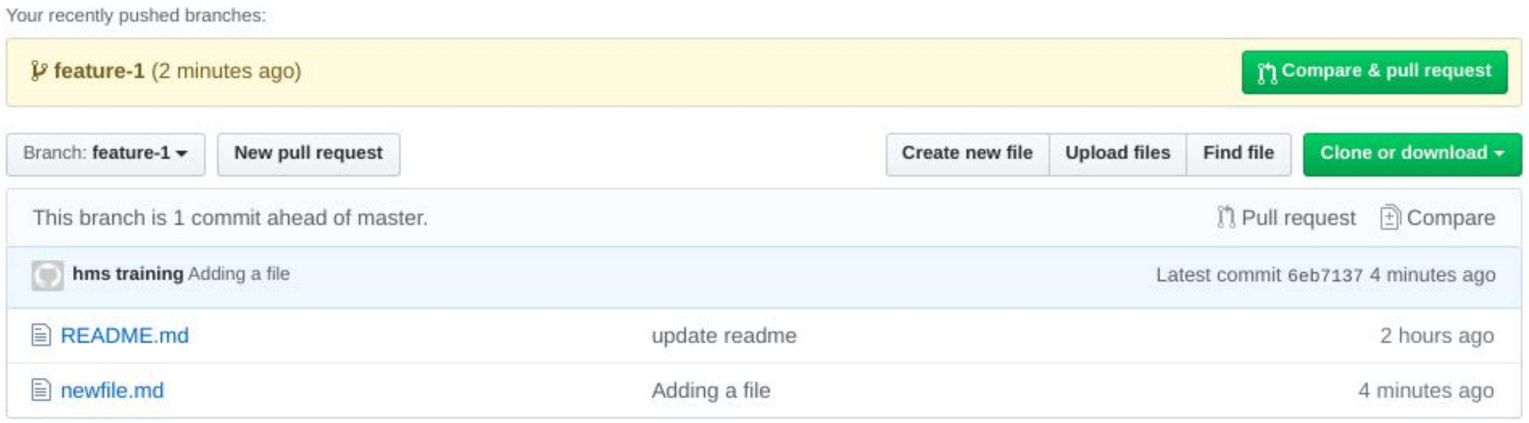
And push this up to github with `` `git push` ``



```
[gester@project-dev testing]$ git push
Counting objects: 4, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 312 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/hmstrainingdemo/testing.git
   2a1b2ef..6eb7137  feature-1 -> feature-1
[gester@project-dev testing]$
```

Updating a branch (cont)

We should now see our new file on github



Your recently pushed branches:

feature-1 (2 minutes ago) — Compare & pull request

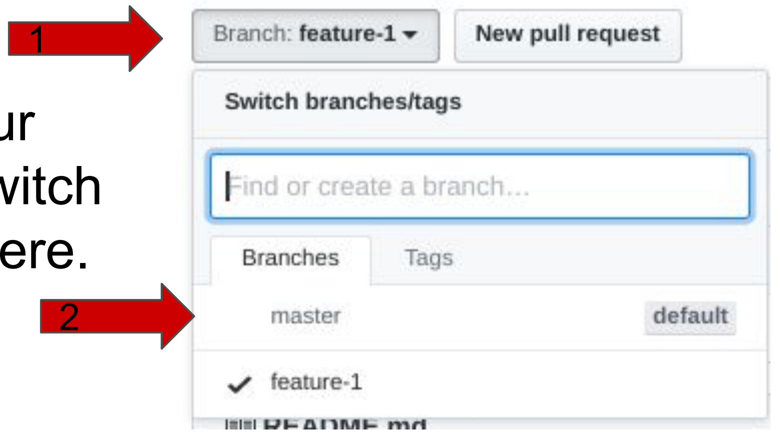
Branch: feature-1 | New pull request | Create new file | Upload files | Find file | Clone or download

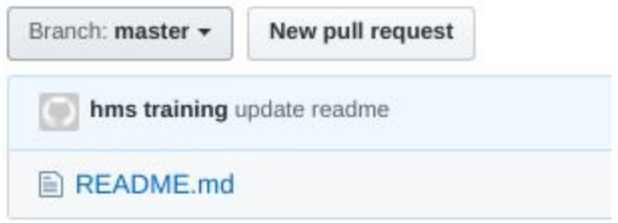
This branch is 1 commit ahead of master. — Pull request | Compare

hms training Adding a file — Latest commit 6eb7137 4 minutes ago

README.md	update readme	2 hours ago
newfile.md	Adding a file	4 minutes ago

This file only exists on our branch however. If we switch back to master it's not there.

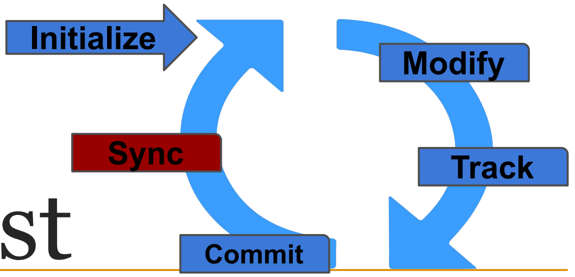




Branch: master | New pull request

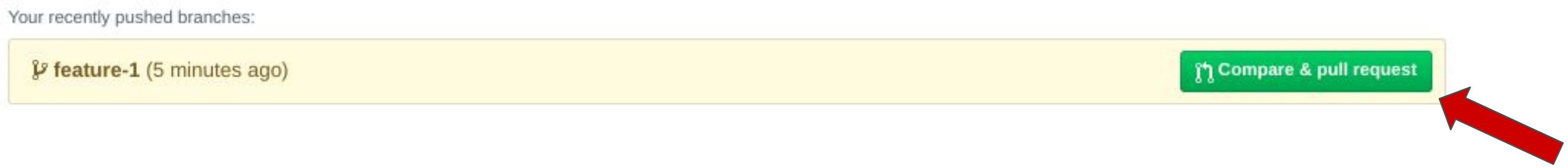
hms training update readme

README.md



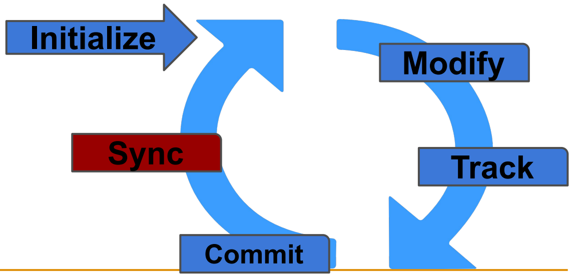
Merging branches with a pull request

To get our file onto the master branch we will create a pull request. Github already gives you a hand with this.


Your recently pushed branches:

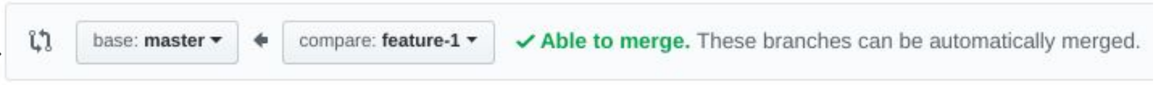
feature-1 (5 minutes ago)

This will let you open a new pull request. There is a lot here so well take a detailed look.



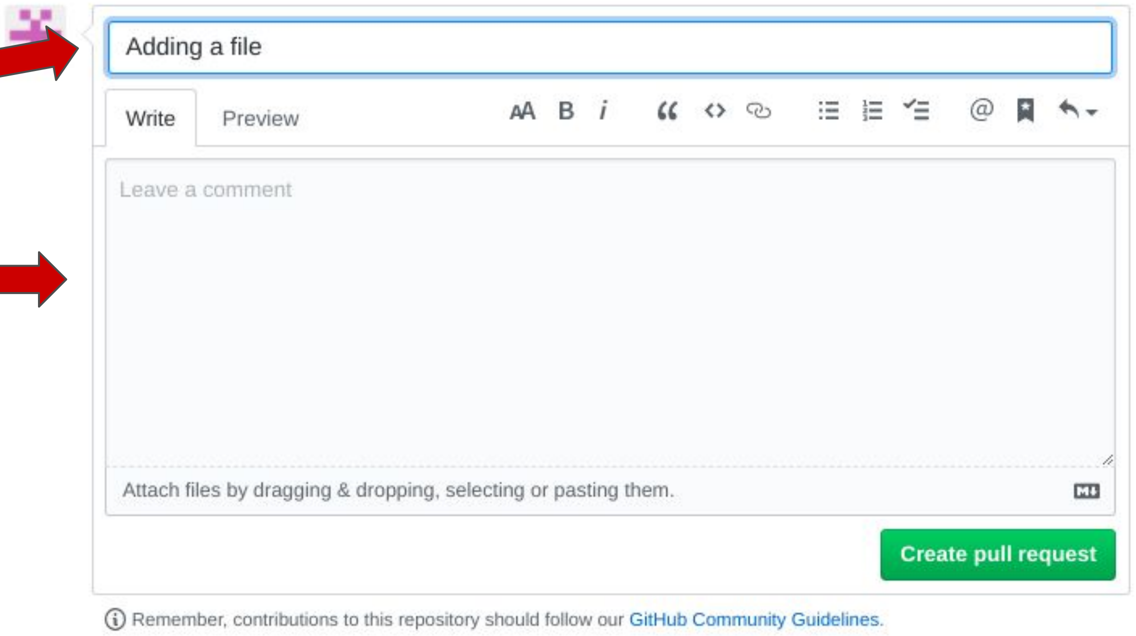
The pull request screen.

This is telling us that we are merging “feature-1” into “master”


base: master ← compare: feature-1 ✓ Able to merge. These branches can be automatically merged.

A title for your pull request

You can add a comment about what your merge is to provide context to your teammates. This can include any information you need.


Adding a file

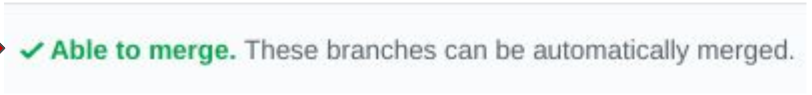
Leave a comment

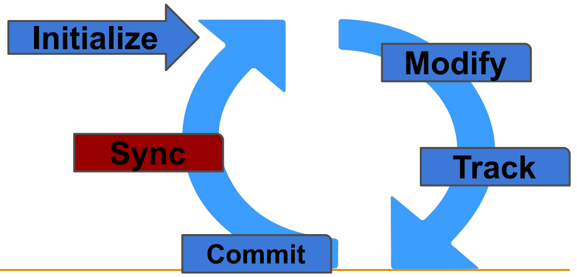
Attach files by dragging & dropping, selecting or pasting them.

Remember, contributions to this repository should follow our GitHub Community Guidelines.

We can this click the Create pull request Button

Note: This message is about merge conflicts. If you see this you are good to go. Resolving merge conflicts is more than we have time for in this class.

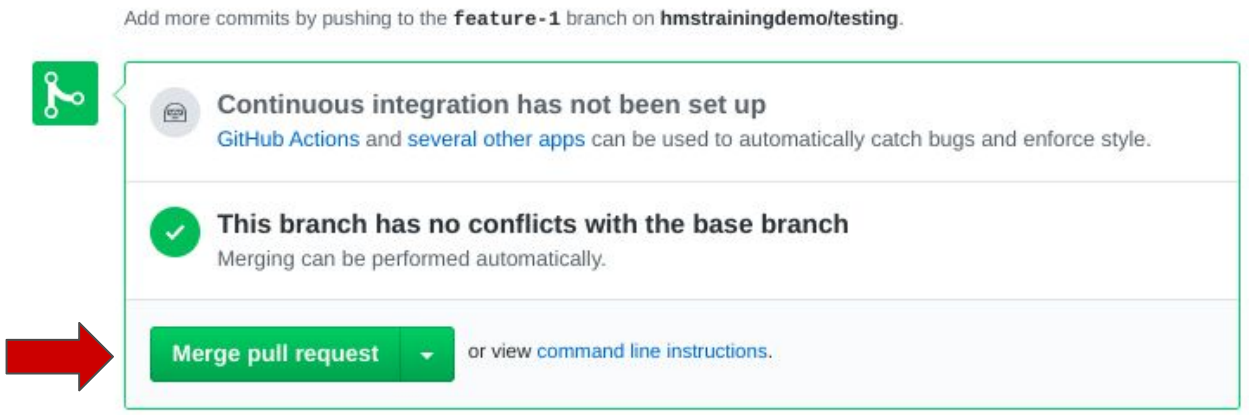

✓ Able to merge. These branches can be automatically merged.



The merge screen

After a few seconds we should get something like this

Clicking this button will bring up a confirm

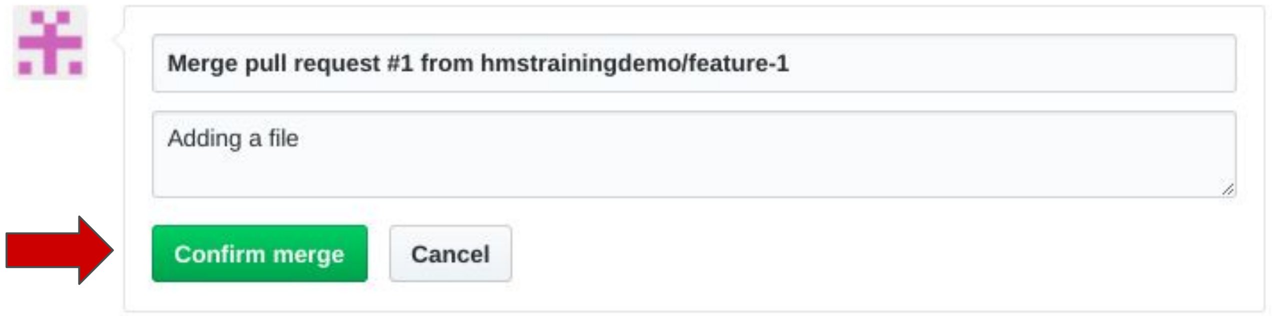

Add more commits by pushing to the **feature-1** branch on **hmstrainingdemo/testing**.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

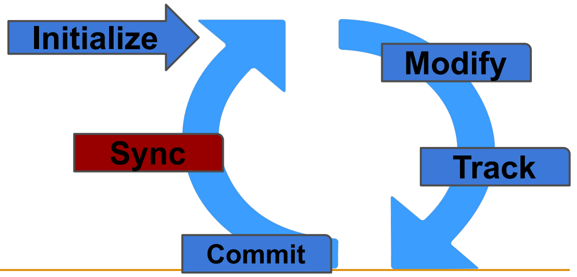
This branch has no conflicts with the base branch
Merging can be performed automatically.

or view command line instructions.

And this will finalize the merge


Merge pull request #1 from hmstrainingdemo/feature-1

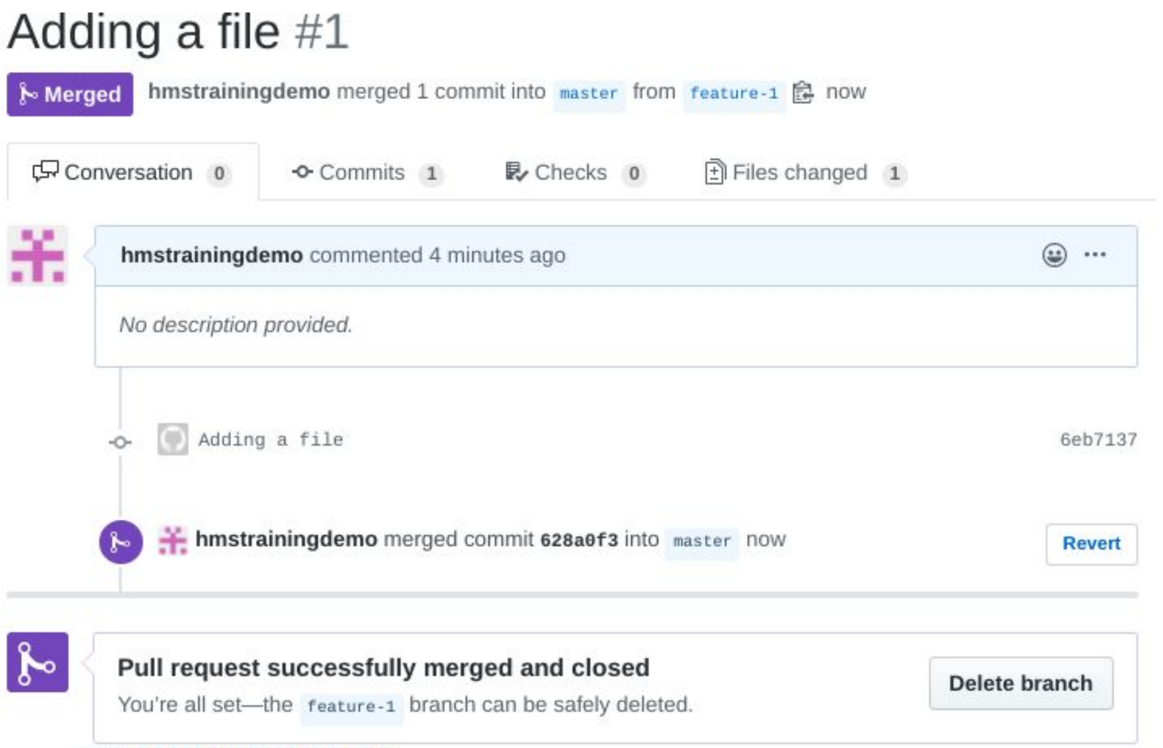
Adding a file



The merge screen (cont)

We have completed the merge.

- There is a description of what was done.
- Revert the merge if we made a mistake.
- The option to delete the branch if we are done with it.


Adding a file #1

Merged hmstrainingdemo merged 1 commit into master from feature-1 now

Conversation 0 Commits 1 Checks 0 Files changed 1

hmstrainingdemo commented 4 minutes ago

No description provided.

Adding a file 6eb7137

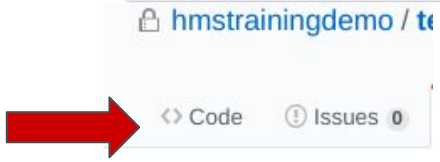
hmstrainingdemo merged commit 628a0f3 into master now Revert

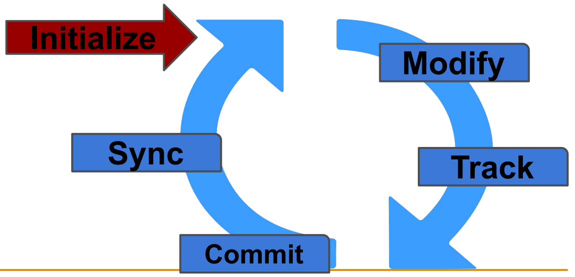
Pull request successfully merged and closed

You're all set—the feature-1 branch can be safely deleted.

Delete branch

clicking on the "code" link will bring us back to the main repo screen where we can see our file on the master branch





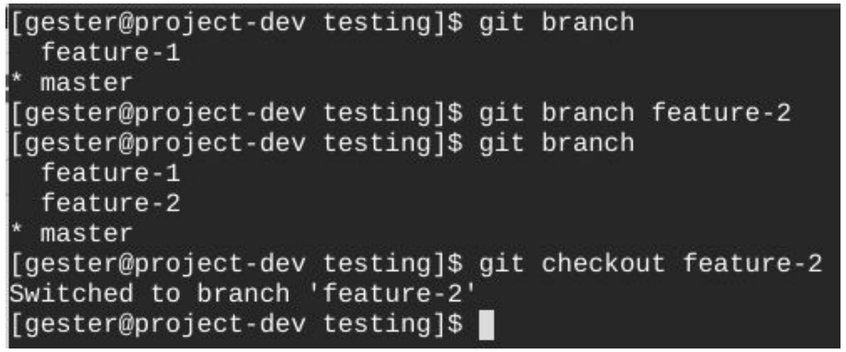
Streamlined branching

Adding a second "feature" via the command line.

`git checkout master` - make sure we are on the master branch

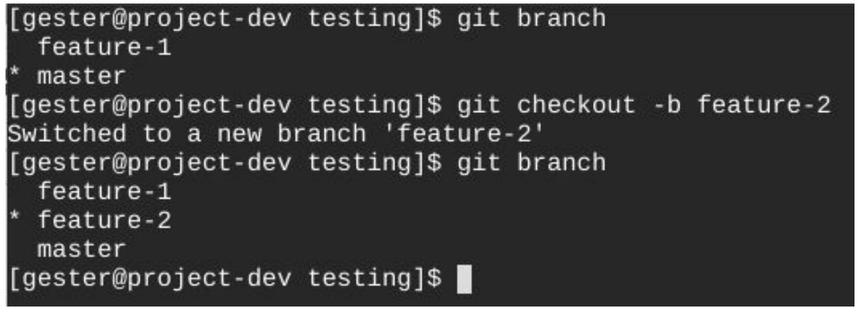
`git branch feature-2` - create a new branch

`git checkout feature-2` - change to the new branch

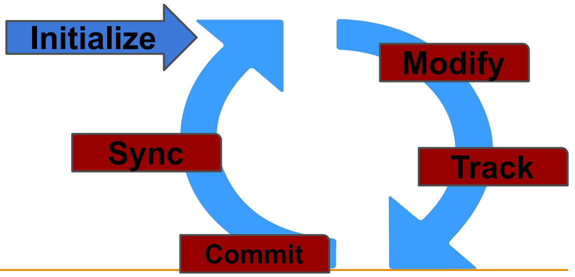


```
[gester@project-dev testing]$ git branch
  feature-1
* master
[gester@project-dev testing]$ git branch feature-2
[gester@project-dev testing]$ git branch
  feature-1
  feature-2
* master
[gester@project-dev testing]$ git checkout feature-2
Switched to branch 'feature-2'
[gester@project-dev testing]$ 
```

We can do this in a single line as well
`git checkout -b feature-2`

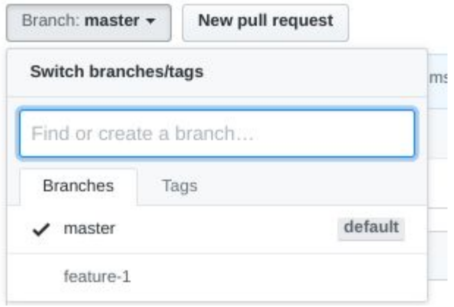


```
[gester@project-dev testing]$ git branch
  feature-1
* master
[gester@project-dev testing]$ git checkout -b feature-2
Switched to a new branch 'feature-2'
[gester@project-dev testing]$ git branch
  feature-1
* feature-2
  master
[gester@project-dev testing]$ 
```



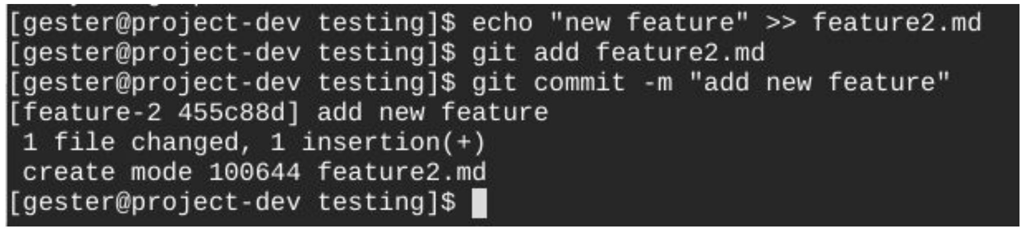
Streamlined branching (cont)

Notice that our new branch does not exist in github yet



We need to push our new branch up first, but we can't push an empty branch. So let's create some new data.

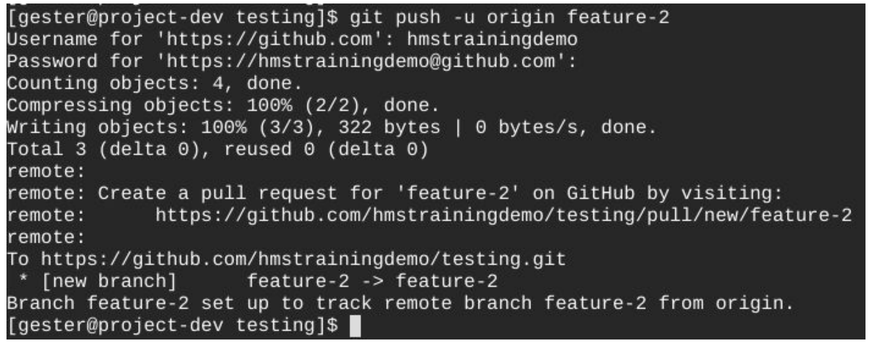
`echo "new feature" >> feature2.md`
`git add feature2.md`
`git commit -m "add new feature"`



```
[gester@project-dev testing]$ echo "new feature" >> feature2.md
[gester@project-dev testing]$ git add feature2.md
[gester@project-dev testing]$ git commit -m "add new feature"
[feature-2 455c88d] add new feature
 1 file changed, 1 insertion(+)
 create mode 100644 feature2.md
[gester@project-dev testing]$
```

Now we can push our new branch and see it on github

`git push -u origin feature-2` – push our new branch data to remote. We need to tell it what to call the new branch and where to put it.



```
[gester@project-dev testing]$ git push -u origin feature-2
Username for 'https://github.com': hmstrainingdemo
Password for 'https://hmstrainingdemo@github.com':
Counting objects: 4, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 322 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: Create a pull request for 'feature-2' on GitHub by visiting:
remote:      https://github.com/hmstrainingdemo/testing/pull/new/feature-2
remote:
To https://github.com/hmstrainingdemo/testing.git
 * [new branch]      feature-2 -> feature-2
Branch feature-2 set up to track remote branch feature-2 from origin.
[gester@project-dev testing]$
```

Questions?

What we have covered:

- repo creation
- cloning a repo
- updating files - with undo
- syncing with a remote
- branching
- merging

Further reading

The git book https://git-scm.com/book/en/v2

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

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

These slides can be found at https://github.com/hmsrc/user-training/intro-to-git-2020.pdf

Jason McDonald jason_mcdonald@hms.harvard.edu
github https://github.com/jfmcdonald