GIT 101: THE BASICS

Lab 02 – CS 411 @ Boston University

WHAT IS GIT/GITHUB?

Decentralized Version Control System

- Each developer has a copy of the code base
- Concurrent work on code by many developers is possible
- Local versions (called *branches*) are embraced

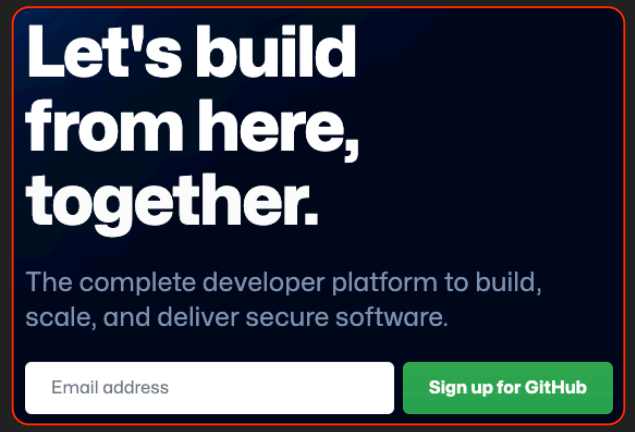
Advantages

- No locking of files (think about a *centralized* version control system?)
- Concurrency
- Branching is relatively straightforward

Disadvantages

- Local branches must be merged
- History can get complex
- Dealing with merge conflicts too

SETTING UP GIT/GITHUB



- Head to *https://github.com/* and create an account if you don't already have one
  - ***Use your BU email for free access to GitHub Pro***

https://github.com/git-guides/install-git

Use *git version* to verify if git is/was installed

$ git --version

https://docs.github.com/en/authentication/connecting-to-github-with-ssh

Setup an SSH Key

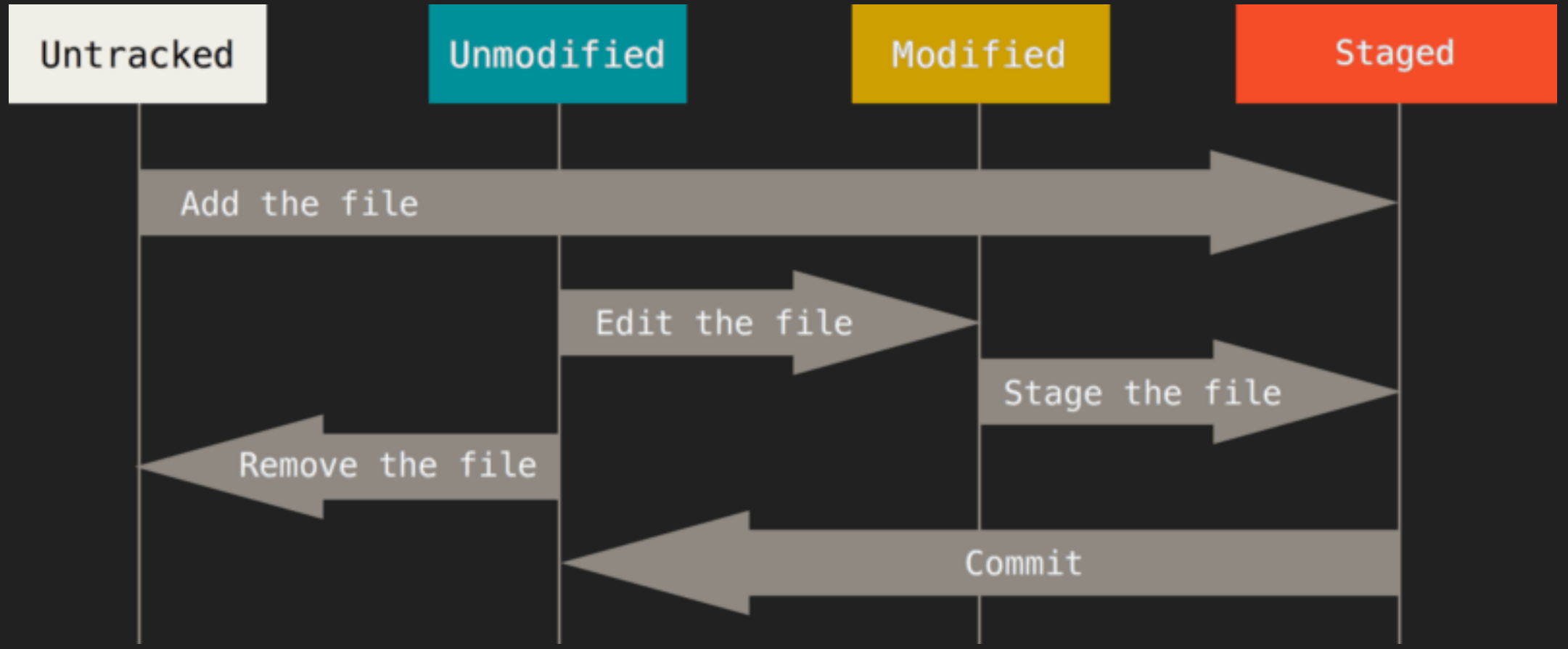
git commit -m “added new commands”

BASIC COMMANDS

git status

- Your best friend and savior in git – sanity check
- Typically used before commits/pushes
- Prints a log of:
  - Current branch
  - Staged changes (files added)
  - Unstaged changes (files that have not been added)

Before the rest… A visual overview!



git add [file]

- Stages content (file/folder) to current commit
- Stages files as they are at the time… NOT a reference
- Further changes must be re-added
- E.g., *git add main.py*

git commit –m [“message”]

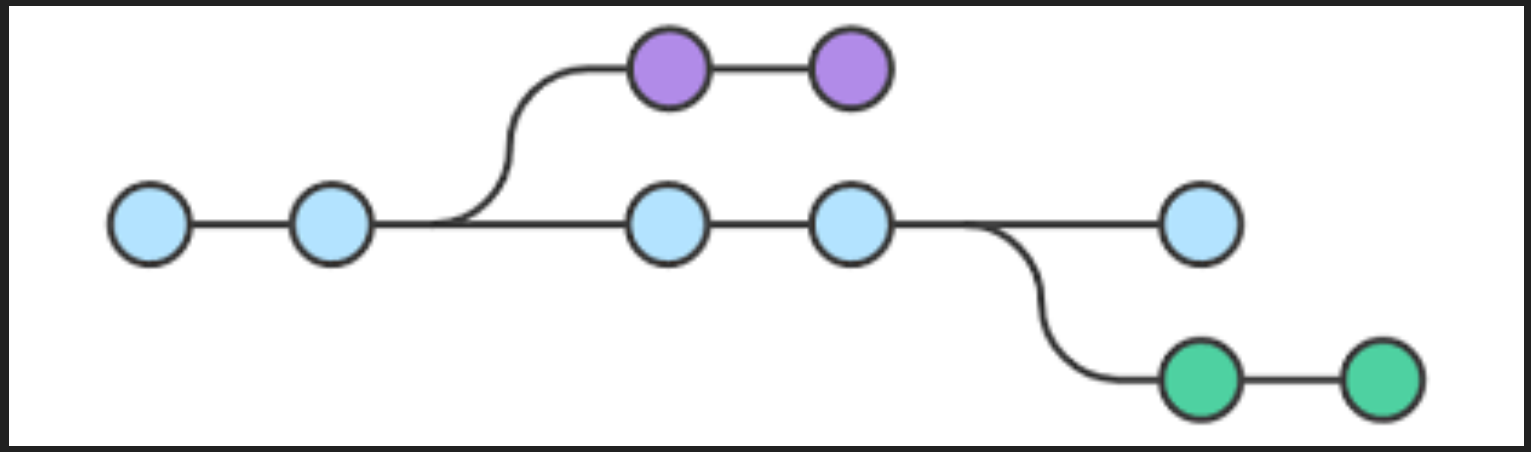
- Git only records changes when explicitly told to
- Saves the changes to the current local branch
- Commit records the changes with a message detailing changes

git push [alias] [branch]

- Pushes local branch changes to a remote branch
- Typically: *git push origin master*
  or *git push origin main*

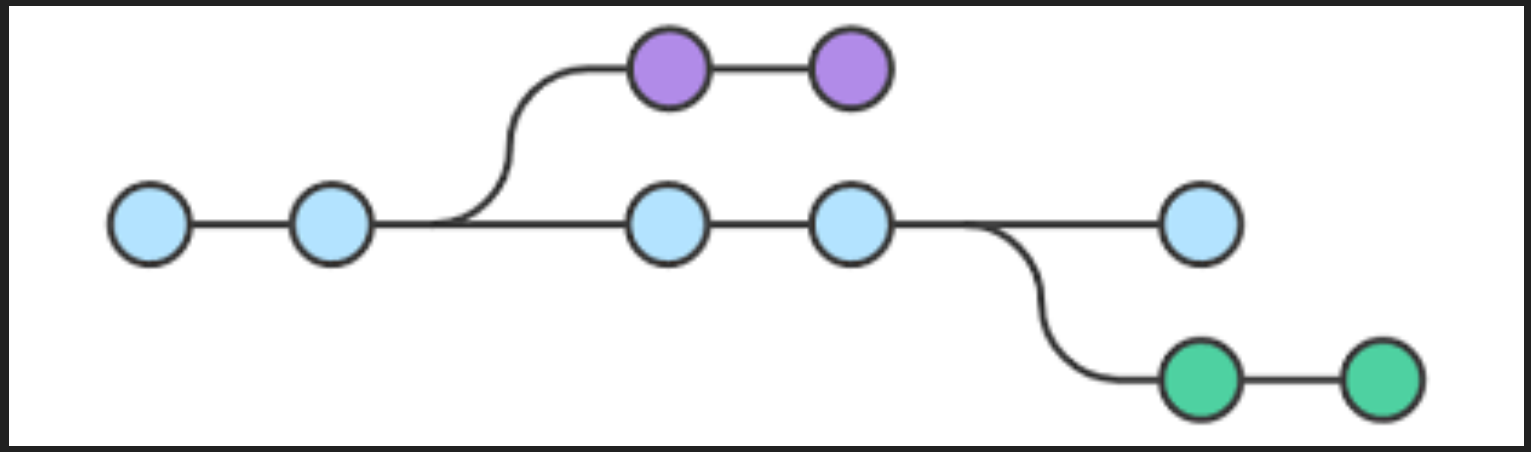
git checkout

- Moves you from the branch you're currently on to the one specified
- Make sure your changes are staged before checking out



git checkout -b [branch_name]

- Creates a new branch

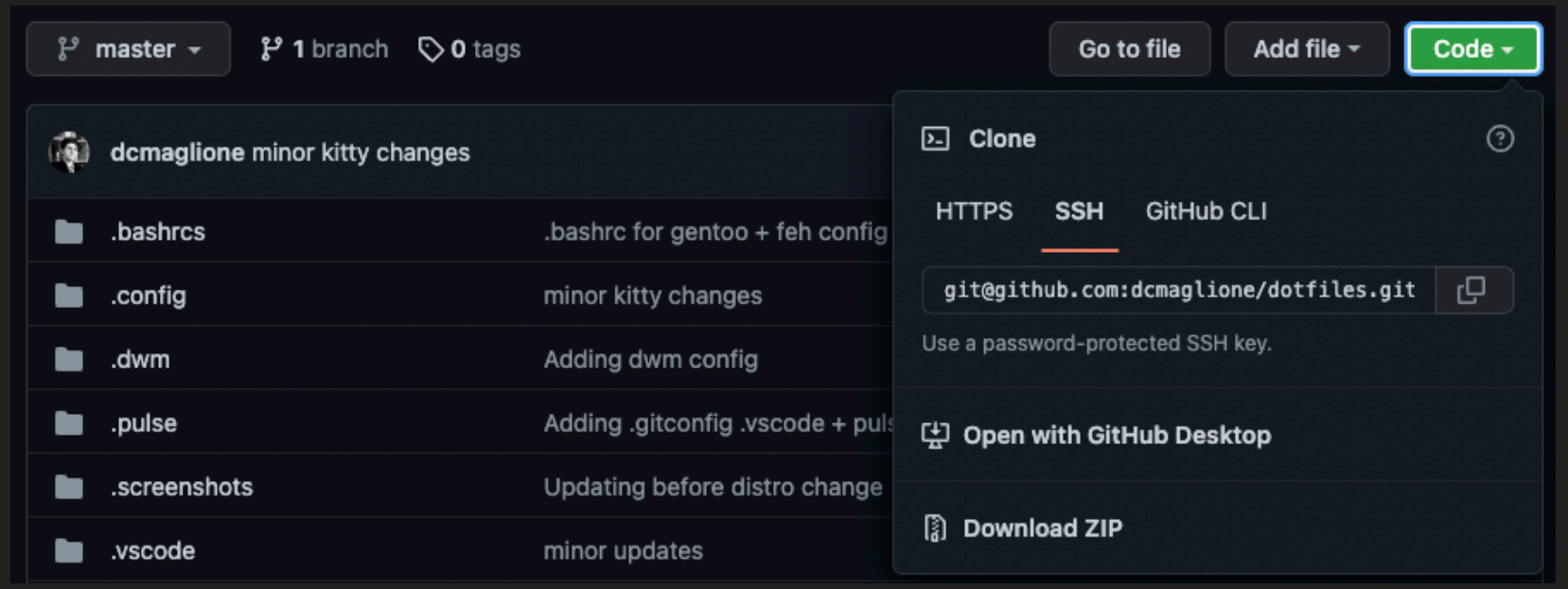


git pull

- Adds changes from a remote repository into the current local branch
- Should be used throughout the development process for incorporating teammate’s code changes together
- Kind of like cloning the repository but only adds files/lines of code that have been changed

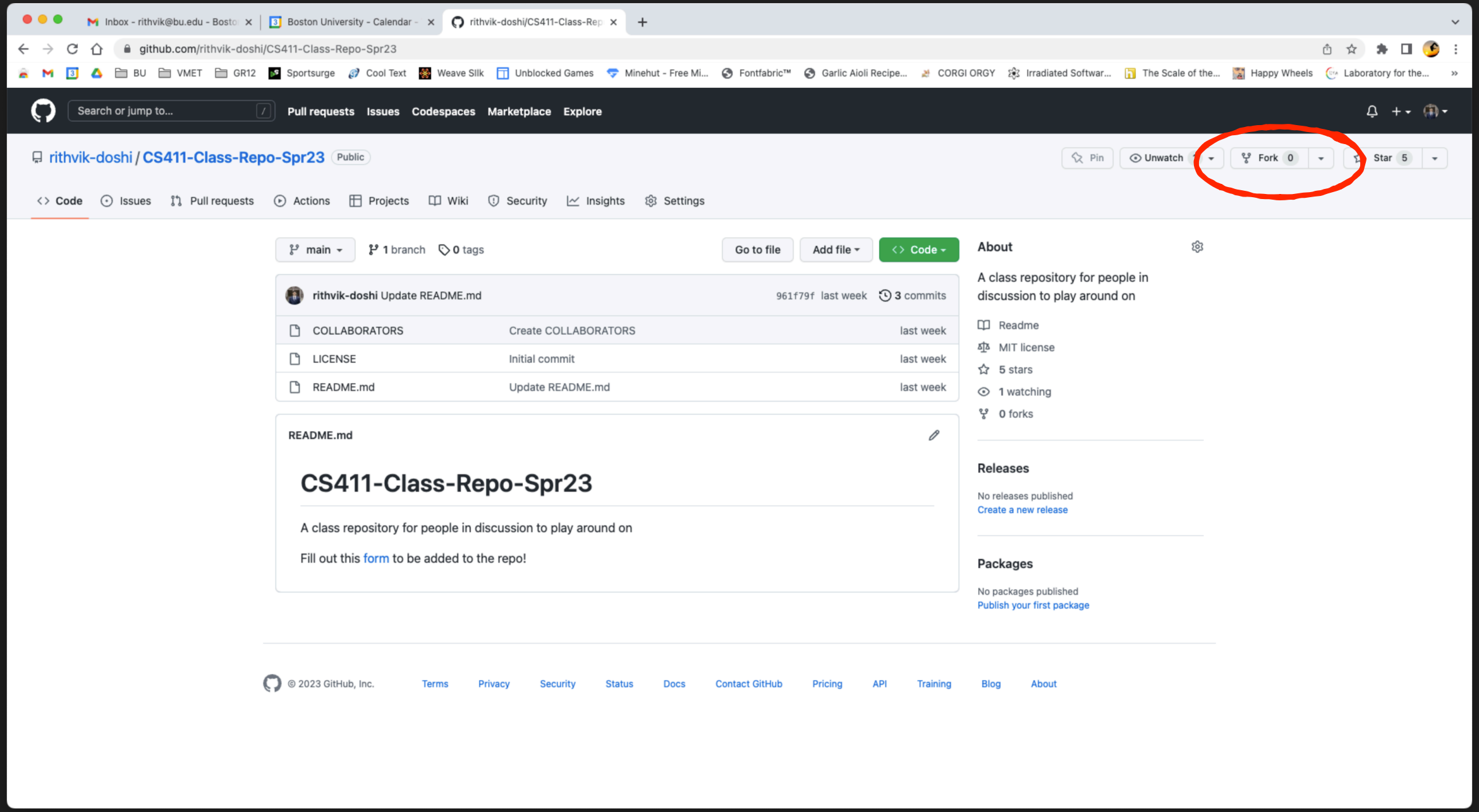
git clone [url]

- Clones' full repository to your local machine
- Should be used in directory where you want to store repository



Forking a Repo

- Why? Well, what if you want to work on a repo you don’t own/can’t collaborate on?
- By forking a repo, you make a copy to your account, you can make changes on it, and then you can propose merges to the original repo.
- Click on the fork button to make a fork of the repo and save it to your account


Inbox - rithvik@bu.edu - Bosto
Boston University - Calendar -
rithvik-doshi/CS411-Class-Rep
github.com/rithvik-doshi/CS411-Class-Repo-Spr23
Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
rithvik-doshi / CS411-Class-Repo-Spr23
Public
Pin
Unwatch 1
Fork 0
Star 5
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
1 branch
0 tags
Go to file
Add file
Code
rithvik-doshi Update README.md
961f79f last week
3 commits
COLLABORATORS
Create COLLABORATORS
last week
LICENSE
Initial commit
last week
README.md
Update README.md
last week
README.md
CS411-Class-Repo-Spr23
A class repository for people in discussion to play around on
Fill out this form to be added to the repo!
About
A class repository for people in discussion to play around on
Readme
MIT license
5 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
© 2023 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About

Working on a forked repo

- git remote add upstream https://github.com/og_user/og_repo.git
  - Replace og_user and og_repo with the appropriate user and repo name
- git fetch upstream
- git checkout main/master
- git merge upstream/main
- git push
- Open a pull request to the main repo

Sync your fork to upstream

- git checkout main/master
- git pull upstream master
- git reset --hard upstream/master
- git push origin master --force
  - THIS WILL DELETE ALL YOUR CHANGES AND RESET THE REPO TO THE FORK

Fork my repo, add your GitHub username to COLLABORATORS and make a PR!

https://github.com/rithvik-doshi/CS411-Class-Repo-Spr23/

GitHub
GIT CHEAT SHEET

Git is the free and open source distributed version control system that's responsible for everything GitHub related that happens locally on your computer. This cheat sheet features the most important and commonly used Git commands for easy reference.

INSTALLATION & GUIS

With platform specific installers for Git, GitHub also provides the ease of staying up-to-date with the latest releases of the command line tool while providing a graphical user interface for day-to-day interaction, review, and repository synchronization.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

For Linux and Solaris platforms, the latest release is available on the official Git web site.

Git for All Platforms
http://git-scm.com

SETUP

Configuring user information used across all local repositories

`git config --global user.name "[firstname lastname]"`
set a name that is identifiable for credit when review version history

`git config --global user.email "[valid-email]"`
set an email address that will be associated with each history marker

`git config --global color.ui auto`
set automatic command line coloring for Git for easy reviewing

SETUP & INIT

Configuring user information, initializing and cloning repositories

`git init`
initialize an existing directory as a Git repository

`git clone [url]`
retrieve an entire repository from a hosted location via URL

STAGE & SNAPSHOT

Working with snapshots and the Git staging area

`git status`
show modified files in working directory, staged for your next commit

`git add [file]`
add a file as it looks now to your next commit (stage)

`git reset [file]`
unstage a file while retaining the changes in working directory

`git diff`
diff of what is changed but not staged

`git diff --staged`
diff of what is staged but not yet committed

`git commit -m "[descriptive message]"`
commit your staged content as a new commit snapshot

BRANCH & MERGE

Isolating work in branches, changing context, and integrating changes

`git branch`
list your branches. a * will appear next to the currently active branch

`git branch [branch-name]`
create a new branch at the current commit

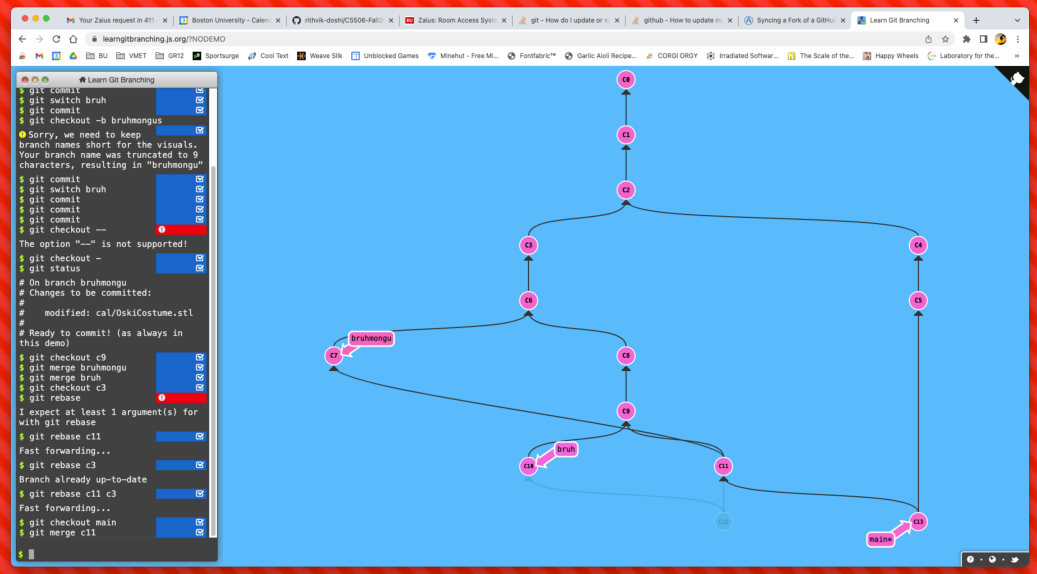
`git checkout`
switch to another branch and check it out into your working directory

`git merge [branch]`
merge the specified branch's history into the current one

`git log`
show all commits in the current branch's history

GIT RESOURCES

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



https://learngitbranching.js.org/

Try this for practice! A great, intuitive tool to help you learn more about git!

Look… I know you wanna use Github Desktop…

- But you're not going to learn anything about the underlying system without getting your hands dirty on the terminal
- Sure, it has its uses, and it's pretty good at helping you not mess things up. But a wise person said "Failure is the best teacher."
- Just my two cents, y'all do whatever you want to!

Group Project Workflow

Things to keep in mind when you start to work on the project together

Project workflow with GitHub

- To set up a repo, you can either
  - Push existing local files to new repo
  - Set up a new repo on GitHub and clone it
- All members create GitHub account
- One team member creates a repo
- That member adds other members to collaborators list (Settings page of repo)
- MEMBERs accept email invite to be collaborator
- MEMBER navigates to github repo, click on green 'Clone or download' button, copy URL displayed
- MEMBER: From a terminal on your local machine, move to the directory you want your local repo to be
- MEMBER: git clone <URL you copied earlier>

Project workflow - setup

- Each member creates a personal branch (i.e. git checkout -b [name-of-branch])
- Each member pushes their personal branch to set up tracking (git push --set-upstream origin [name-of-branch])

Doing work

- On your machine, work in the project directory on your personal branch
- Update it with any changes made with git pull
- Create new topic/feature branches as necessary with git checkout
- After you work on something on another branch, merge it into your personal branch
- Push your branch to the remote repo
- Notify the repo owner that your changes are ready to merge into the release branch by making a pull request