GIT 101: THE BASICS

Lab 02 – CS 411 @ Boston University

WHAT IS GIT/GITHUB?

Decentralized VCS

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

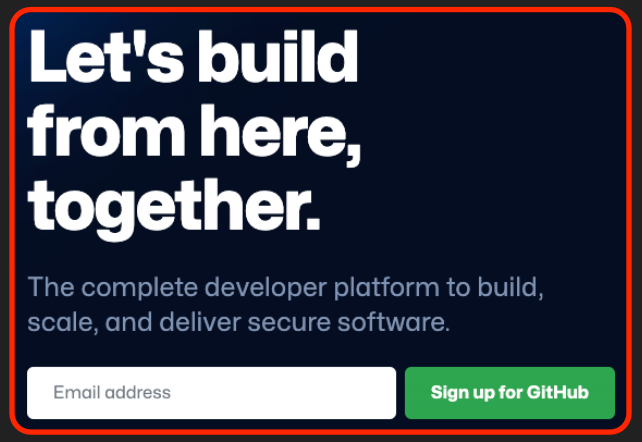
Advantages

- No locking of files
- Concurrency
- Simple branching

Disadvantages

- Local branches must be merged
- History can get complex

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
```

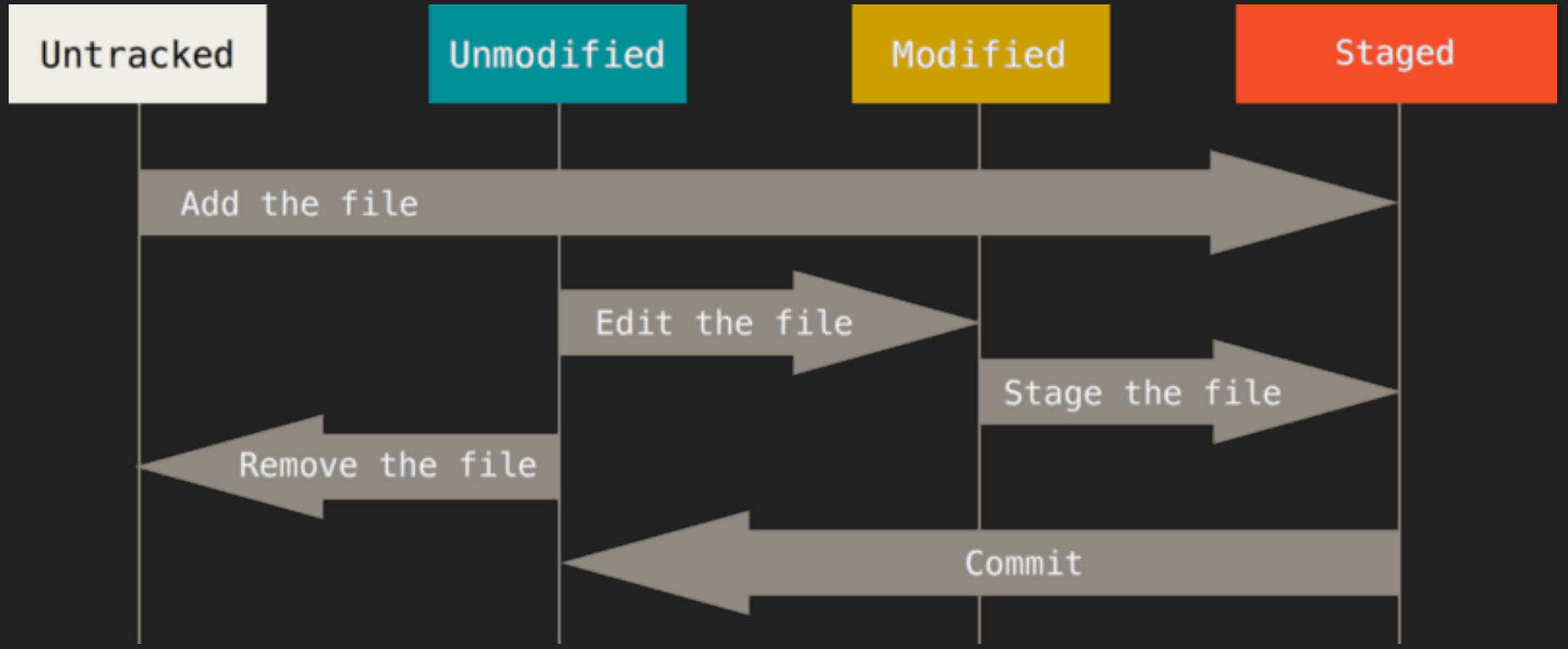
https://docs.github.com/en/authentication/connecting-to-github-with-ssh

Setup an SSH Key

git commit -m “added new commands”

BASIC COMMANDS

BUT FIRST… 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”]

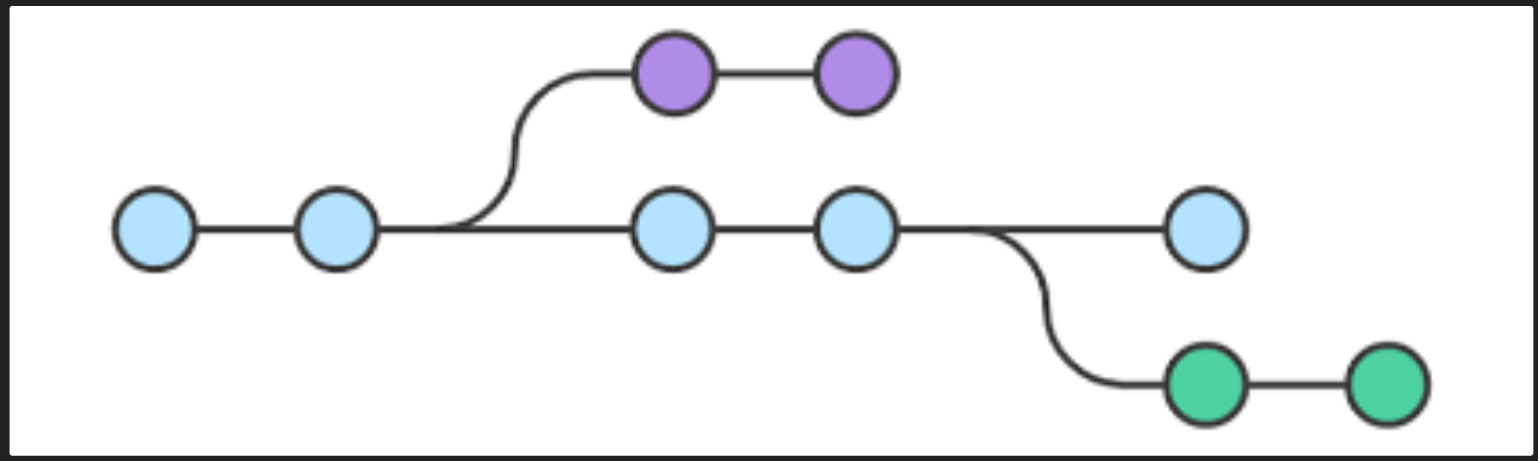
- Git only records changes when explicitly told to
- Saves the changes to the current local branch
- Commit records the changes with a message detailing changes

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

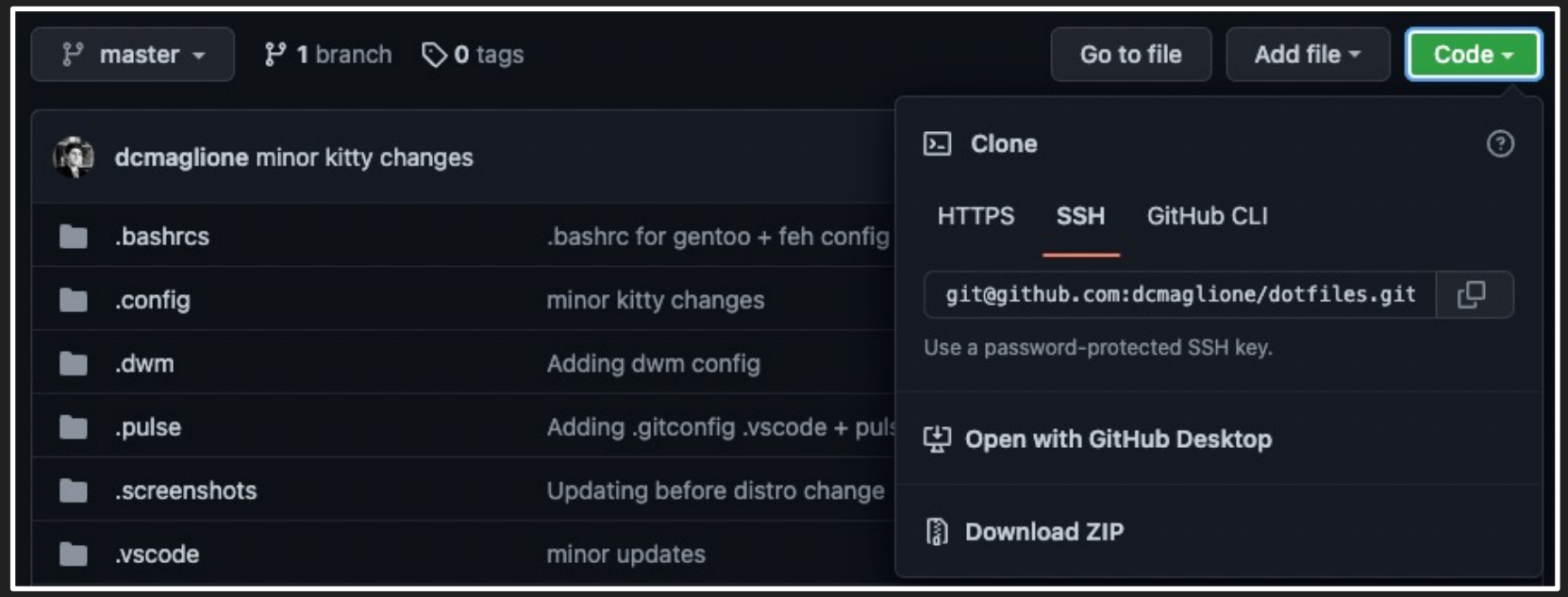


git status

- Your best friend 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)

git clone [url]

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



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

Project Workflow w/ GitHub

- There are two main ways to set up a repo on GitHub
  - Push existing local files to a new repo
  - Set up a new repo on GitHub and clone it
- We'll look at the second method, which is the simpler of the two
- For the first method, a good tutorial is at

https://www.digitalocean.com/community/tutorials/how-to-use-git-effectively

Project Workflow: Setup for LEAD

- All members create GitHub account if it doesn’t exist
- One team member (LEAD) creates a repo
- LEADS add MEMBERs to collaborators list (add collabs in the Settings page (gear icon))
- MEMBERs accept email to invite to be collaborator
- MEMBER navigates to GitHub repo, click on green ‘Clone on 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

- MEMBER creates a personal branch

  (i.e. dcmag would do `git checkout -b dcmag`)
- MEMBER pushes personal branch to set up tracking

  (`git push -set-upstream origin dcmag`)

Project Workflow: Doing Work

- MEMBER: move to project directory on your machine
- Switch to your personal branch

  `git checkout dcmag`
- Update with any changes made since last time you were working

  `git pull origin master`
- Create a new top/feature branch to do work on a specific item

  `git checkout -b oauth`

Project Workflow: Doing Work

- After completing work on the topic branch, merge it into your personal branch
  - `git checkout dcmag`
  - `git merge oauth`
- Push your personal branch to the project's GitHub repo
  - `git push`
- Notify LEAD that your changes are ready to merge into the release branch with a pull request
  - Log onto GitHub navigate to project repo, click on New Pull Request
  - base: master <- compare: <your personal branch>
- LEAD evaluates request, request comments, merges into masterSave files on local branch to remote repo

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

```
git add lab02.txt
git commit -m “complete”
git push origin master
```

Lab 02 - CS 411 @ Boston University

(dcmag@bu.edu)