20210913 Git

Saturday, September 11, 2021 3:07 PM

20210911 Git

Git

Boston University CS 506 - Lance Galletti

GitHub vs Git

GitHub --> [browser]

GitHub is a **website** that allows you to upload your git repositories online. It allows you to have a **backup of your files online**, has a visual interface to navigate your repos, and it allows other people to be able to view, copy, or contribute to your repos.

Git --> [terminal]

Git is a **version control system**. It allows you to manage the history of your git repositories.

Motivation

For each codebase (repository) I own, I want to write code where:

1. Progress loss is minimized
2. Iterating on different versions of the code is easy
3. Collaboration is productive

Minimal Progress Loss

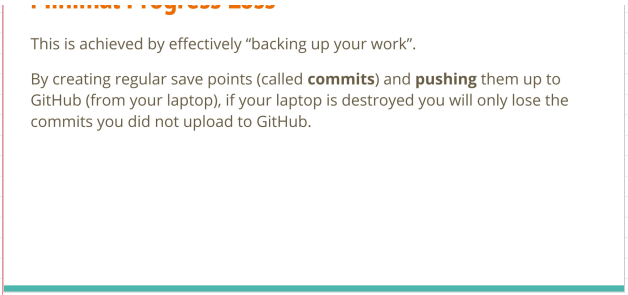
This is achieved by effectively "backing up your work".

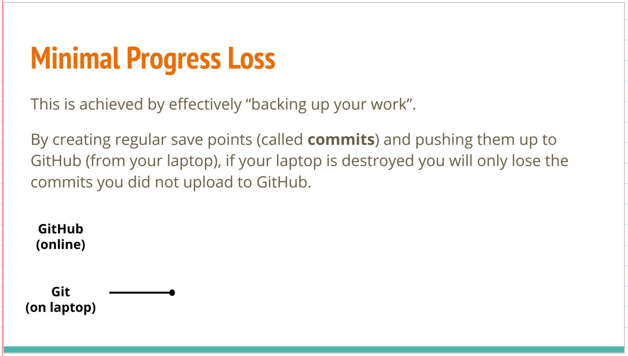
By creating regular save points (called **commits**) and **pushing** them up to GitHub (from your laptop), if your laptop is destroyed you will only lose the

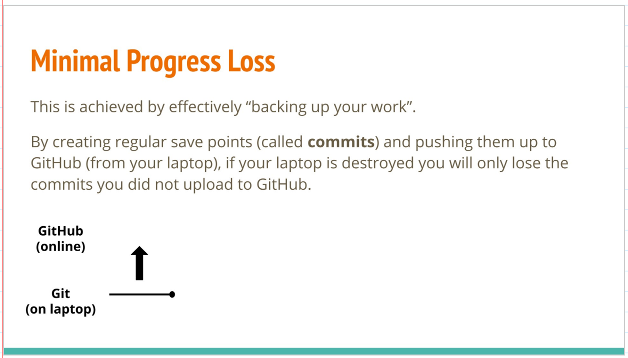
- Changed Tas
- Office hours will be posted soon
- labs start this week
  - First 3 labs are posted
  - Labs intended to set baseline for the course
  - Can ask questions of the TA as you work through the excercises
- Git fundamental HW is due on the Sept 20th
  - Will get you used to how we submit HW in this course
  - Low stakes HW, everyone should do it
- Think of individual projects
- Spark project descriptions will hopefully be out tomorrow
  - You also get to attend pitch day
  - You can submit an individual proposal, and then switch to Spark, but it's not easy to do the reverse
- No notes uploaded
  - Can submit as images
  - Extra credit
  - If people aren't feeling well, this will help them (in addition to professor's annotations)
- Will do a number of demos in this class, follow along if you have your laptop
- You don't have to have the same workflow as him, but he's been doing this a long time

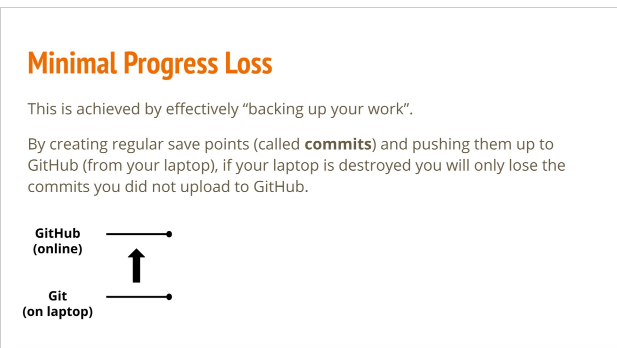
Git demo

- Look at history of repository
  - Today two commits
    - Initial commit
    - How two checkpoints
- git log
  - Allows you to see all commits that have been
  - Unique ID based on content and time
  - Can see that local and website has same content
  - Press "q" to get out of interface
- Make change to code base
  - Each project will have own repository
- git status
  - view changes that have been made
- Visual Studio Code provides nice interface
- git diff
  - To see differences
- Want to know what you commit and why
- Don't just do
  - git add .
    - Will be a lot of hidden files that get committed that are just trash
- git add README.md
- git status
  - See changes are in staging area
- git commit -m ""
  - Want to have a coherent commit message
  - Each commit should be a LOGICAL step
    - Not just every new line
  - Message should tell you what happened
  - Need to be large enough to have meaningful features
  - Small enough that you can roll back if needed
  - git commit -m "fixing a typo in the hyperlink"
  - Your branch is ahead of origin/master by one commit
- git push origin branchName
  - git remote -v
    - Shows all the remote repositories
    - Can fetch changes and push changes
  - git push origin master
    - Conventions are changing
    - main is the protected branch (master is the same thing)
- git log
- Possibility of undoing changes
  - Want to preserve both versions of history
  - Want to overwrite history (which is dangerous)
- Can remove commit as if it never happened
- Let's go over some conflicts
  - Want to rebase off a particular commit
  - There is an interactive rebase command
  - git rebase -i ####################
    - vim editor
    - You can modify your config for which editor git uses
    - Can see two commits
    - Delete "pick .... " to remove commit and rebase
      - Commit is no longer part of history
    - :wq to exit vim
    - Website saw none of this
  - git push origin master
    - This will produce a conflict
    - Updates were rejected because the tip of the current branch is behind its remote counterpart
      - (Read the messages and try to understand what it's saying)
    - We want to tell it we are SURE this is intended
    - git push origin master -f









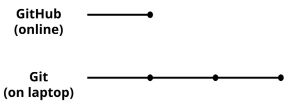
        - Updates were rejected because the tip of the current branch is behind its remote counterpart
            - (Read the messages and try to understand what it's saying)
        - We want to tell it we are SURE this is intended
        - git push origin master -f
            - Force flag forces the overwrite
- He doesn't have two git accounts so he can't show the collab workflow
    - (This is a fixable problem...)
- Commit changes "fixing typo in hyperlink"
    - Pretend someone else commited this
- git log
- git status
    - Doesn't actually FETCH the changes, you need to do it
- git fetch origin
    - It gets the content, but hasn't applied the content yet
- git status
    - Now it can see you are behind by one commit
- git pull
    - NOT good practice
        - Why?
    - Only do when you are confident there are absolutely no conflicts
    - So you git fetch & git rebase
    - How can you selectively apply a rebase?
        - You can do that with git pull
        - Normally you rebase a branch
- git rebase origin/master
- This is the process you should go through to get the changes
- Sometimes you have to rewind to the most recent common commit, and then update it
- There should be away to list the commits for the different branches, and there is a way to pretty print them
    - He has this somewhere

A BUNCH OF MY NOTES DISAPPEARED?

Minimal Progress Loss

This is achieved by effectively “backing up your work”.

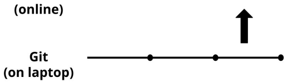
By creating regular save points (called **commits**) and pushing them up to GitHub (from your laptop), if your laptop is destroyed you will only lose the commits you did not upload to GitHub.



Minimal Progress Loss

This is achieved by effectively “backing up your work”.

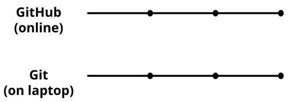
By creating regular save points (called **commits**) and pushing them up to GitHub (from your laptop), if your laptop is destroyed you will only lose the commits you did not upload to GitHub.



Minimal Progress Loss

This is achieved by effectively “backing up your work”.

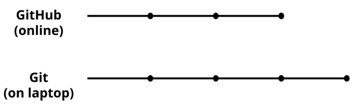
By creating regular save points (called **commits**) and pushing them up to GitHub (from your laptop), if your laptop is destroyed you will only lose the commits you did not upload to GitHub.



Minimal Progress Loss

This is achieved by effectively “backing up your work”.

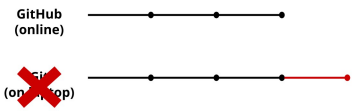
By creating regular save points (called **commits**) and pushing them up to GitHub (from your laptop), if your laptop is destroyed you will only lose the commits you did not upload to GitHub.



Minimal Progress Loss

This is achieved by effectively “backing up your work”.

By creating regular save points (called **commits**) and pushing them up to GitHub (from your laptop), if your laptop is destroyed you will only lose the commits you did not upload to GitHub.



Demo

Iterating on Different Versions

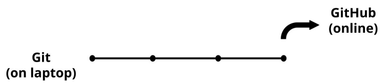
The ease or difficulty of adding a new feature to the code base may depend on the state / version of the codebase.

It may be easiest to add this feature at a specific commit.

Iterating on Different Versions

The ease or difficulty of adding a new feature to the code base may depend on the state / version of the codebase.

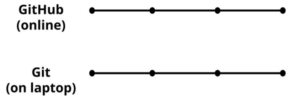
It may be easiest to add this feature at a specific commit.



Iterating on Different Versions

The ease or difficulty of adding a new feature to the code base may depend on the state / version of the codebase.

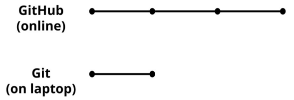
It may be easiest to add this feature at a specific commit.



Iterating on Different Versions

The ease or difficulty of adding a new feature to the code base may depend on the state / version of the codebase.

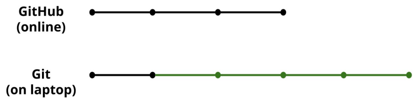
It may be easiest to add this feature at a specific commit.



Iterating on Different Versions

The ease or difficulty of adding a new feature to the code base may depend on the state / version of the codebase.

It may be easiest to add this feature at a specific commit.



Iterating on Different Versions

What happens now?

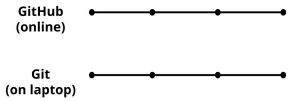
Iterating on Different Versions

Looks like we need:

1. A way to preserve both versions of history
2. A way to overwrite history if we choose (this is dangerous as we will lose that history)

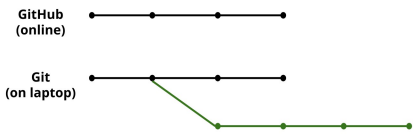
Iterating on Different Versions

Let's try that again!



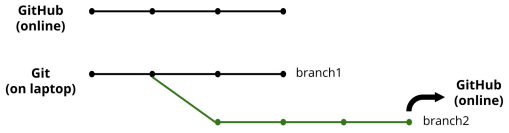
Iterating on Different Versions

We will **branch** off of that particular commit



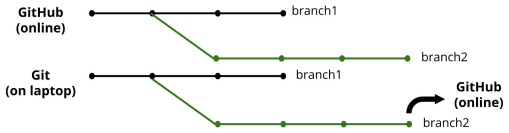
Iterating on Different Versions

We can push **commits** per **branch**



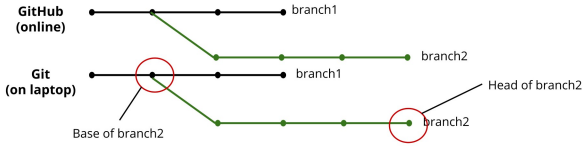
Iterating on Different Versions

We can push **commits** per **branch**



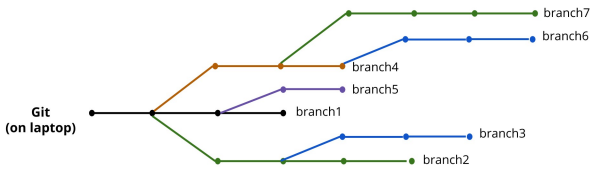
Iterating on Different Versions

We can push **commits** per **branch**



Iterating on Different Versions

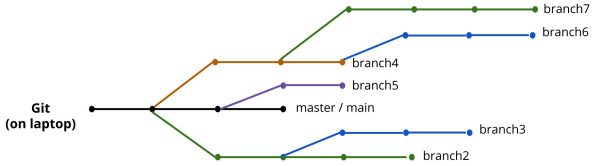
We can create lots of **branches**



Iterating on Different Versions

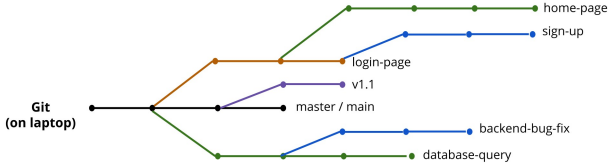
But one branch needs to chosen as the primary, stable branch

This branch is typically called the “master” or “main” branch



Iterating on Different Versions

Other branches are usually named after either the feature that is being developed on or the major or minor version of the software / product

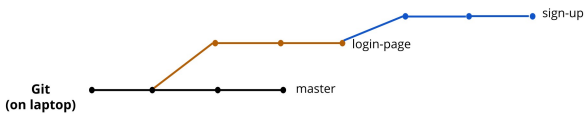


Iterating on Different Versions

At some point we will want to clean up certain branches by **merging** them with the master / main branch or with each other.

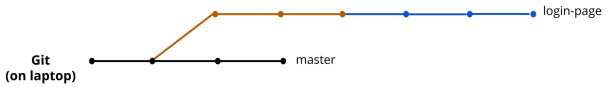
Iterating on Different Versions

At some point we will want to clean up certain branches by **merging** them with the master / main branch or with each other.



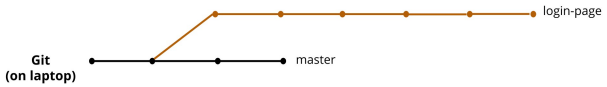
Iterating on Different Versions

Merging is trivial if the **base** of one branch is the **head** of the other - the changes are “simply” appended.



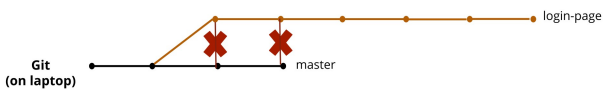
Iterating on Different Versions

When this is not the case, commits can conflict with each other



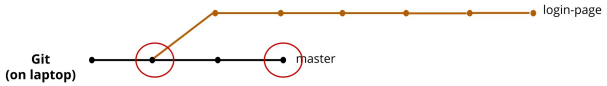
Iterating on Different Versions

When this is not the case, commits can conflict with each other



Iterating on Different Versions

We need to change the **base** of the login-page branch (**rebase**) to be at the **head** of the master branch



Iterating on Different Versions

We need to change the base of the login-page branch (**rebase**) to be at the head of the master branch

Iterating on Different Versions

This is not a simple operation! It will often require **manual intervention** to resolve the conflicts.

Iterating on Different Versions

This is not a simple operation! It will often require **manual intervention** to resolve the conflicts.

Demo

Collaboration

Possible to collaborate on a single repository by having each collaborator create new branches for development and having a process for merging their branch into master.

This request to merge code is done by a **Pull Request** (PR).

- If you had multiple people editing a google doc at once gets very messy and then you have to agree who is on what page
- Can make a copy of the true copy
  - Make changes
  - Then make a request to pull in changes

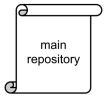
Collaboration

Typically (in particular with open source repositories), repository owners prefer not having to manage collaborator permissions on the repository.

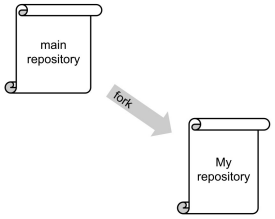
In order to contribute code, collaborators must:

1. Make a copy of (**fork**) the main repository
2. Make all the changes they want to this copy
3. Request that part of their copy be merged into the main repository via a **Pull Request** (PR)

Collaboration

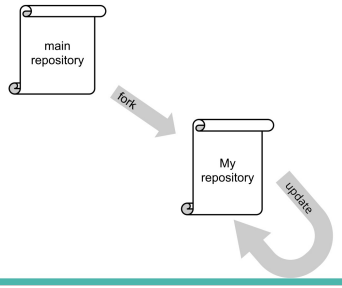


Collaboration

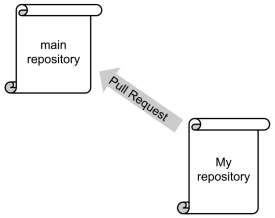


- You could have a single repository, and in settings you can add collaborators with different permissions per person
- Use branches
- pull and merge request are the same, just from different interfaces
- Maybe you want 100s of true copies
  - Not sustainable
- Github does not keep fork repository and main repository are up to date
- Maybe only the main branch needs to be kept in sync
- If you have branches that you are editing
  - You may have a stranded branch which is not fun
- Never edit master branch directly
  - Every time you do another lab, etc. are in their own branch

Collaboration



Collaboration



Collaboration

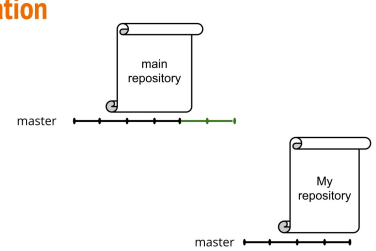
In the time it took you to implement your new feature, the main repository has changed since the time you **forked** the repository.

How do you keep your copy up to date?

You could delete your copy and re-copy the latest... But you would lose all the work you committed to your copy!

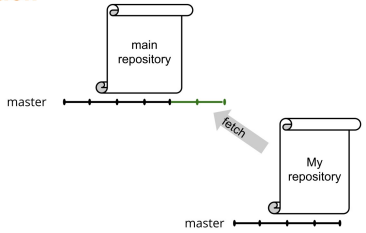
Luckily, we're only interested in keeping the master branch in sync! Why?

Collaboration

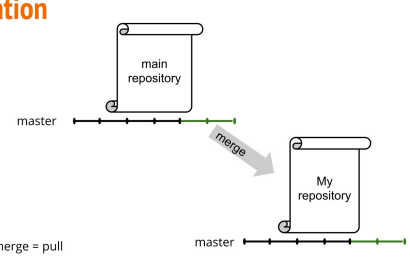


- Two commits on the right are green
- Need to bring them into my repository so they are in sync
- Can fetch and merge those changes into the master branch

Collaboration



Collaboration

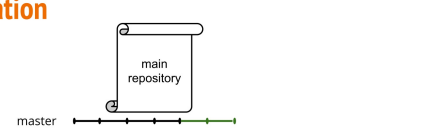


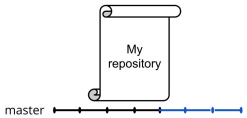
Note: fetch + merge = pull

Collaboration

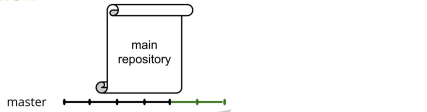
This is trivial when the **base** of one branch matches the **head** of the other.

Collaboration

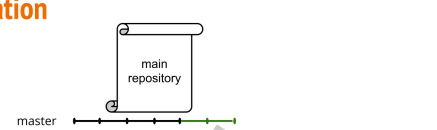




Collaboration



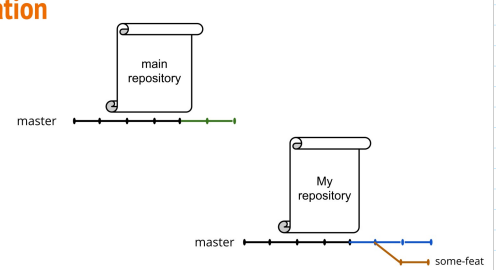
Collaboration



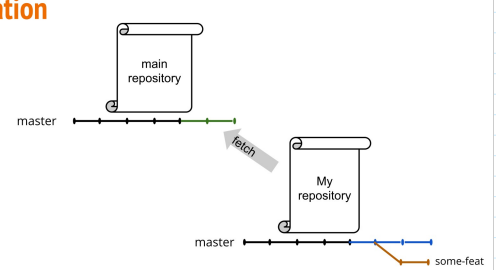
Collaboration

What happens to other branches?

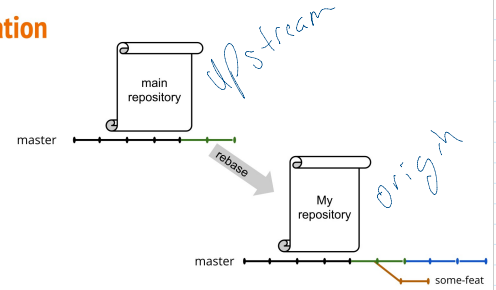
Collaboration



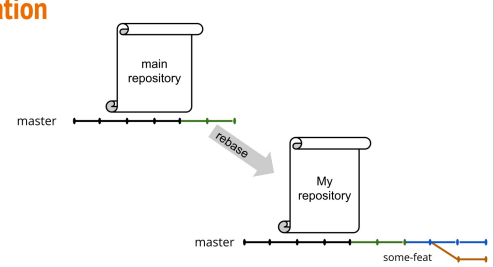
Collaboration



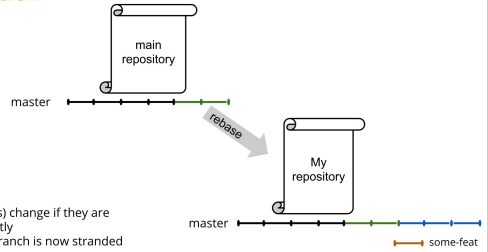
Collaboration



Collaboration



Collaboration



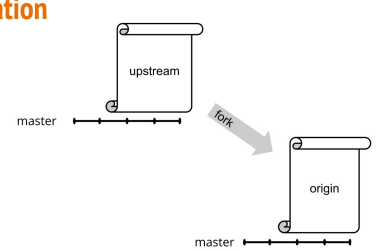
commit IDs (shas) change if they are ordered differently
The some-feat branch is now stranded

Collaboration

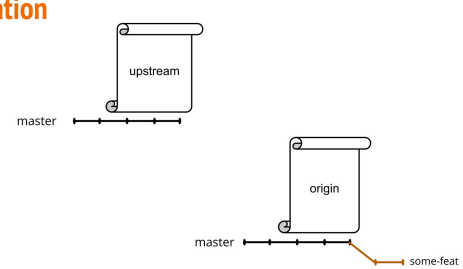
If you never commit anything to your master branch, keeping your master branch in sync with the main repository's is easy! And keeping all branches attached comes for free!

As a rule, always create a new branch when developing - **never commit directly to the master branch**

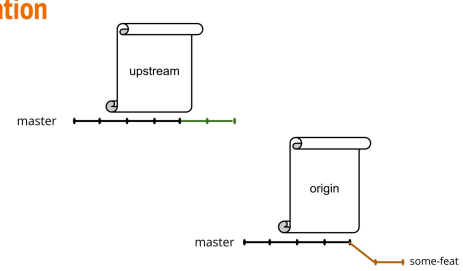
Collaboration



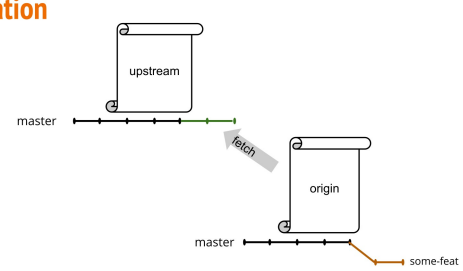
Collaboration



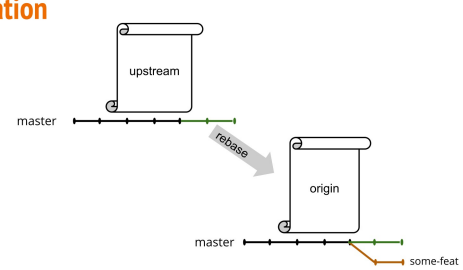
Collaboration



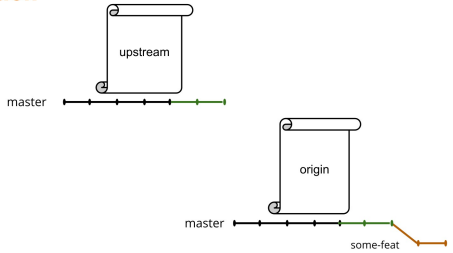
Collaboration



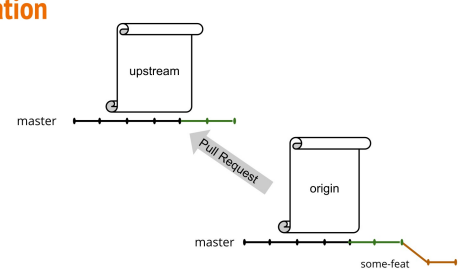
Collaboration



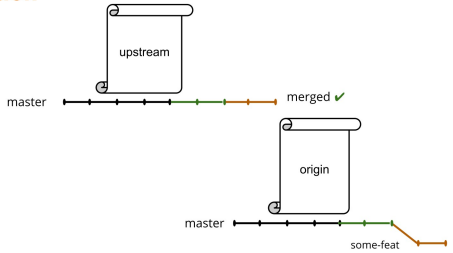
Collaboration



Collaboration



Collaboration



- Will be especially useful in groups

Demo

Another git demo:

- fork repo
- Have a copy of the the repo
  - If already forked and behind by 30, need to update
- git clone
- git log
- git remote -v
  - Copy of repo
- Need a remote that's pointing to the forked repo, so you can get the updated changes
- Copy reference to remote
- git remote add upsteam LINK
- Now you have upstream and origin
- git status
- (clear to clear command line)
- git rebase upstream/master
- git status
- Now can see you are ahead of origin by 33
- git push origin master
  - Now can see it's even
- You should do this on a regular bases
  - This workflow needs to be done on a regular bases
- He will be updated the CS506 repo on a regular basis
- git checkout -b docs
  - Creates branch and checks it out automatically
  - Head of this branch is "docs"
  - Aligned with upstream master and upstream master
    - All 3 are now at the same spot
- Open editor
  - Like Visual Studio Code
- Don't keep changing tools, improve the workflow you are currently on
  - Have to have a good reason to try something new for it to be worth your time
- It's easy to contribute to open source repo, and it may help when applying for jobs
- git diff to see changes
- status
- git add .
- git status
- git commit -m "minor doc update"
  - sure to look at contribution guidelines to repo
    - How you should structure commits, etc.
- git log
  - Now Docs is one commit ahead
- git status
- git push origin docs
  - Pushes changes to docs branch

- Now can see branch on github
  - Can push button and open a pull request
  - Compare with docs branch
  - Can write a little comment
  - Will merge into their master and not ours
  - This repo, has a review required before it can be merged
- tensorflow repo
  - No reason you can't contribute….
  - Check out "good first issue"
  - Can do this with any repo
- git branch docs
- git status
- Remove commit again
- docs is one ahead of master
- git push origin docs
- Create pull request on website
- Git status
- Git log
- If you are used to github desktop app, can use it. Try to use terminal if you can, but there is a steep learning curve
  - Focus on learning one tool at a time until it's limiting your ability to be productive
- oh my zsh
  - powershell themes
  - Customizable
  - He's in fedora? (linux)
  - He especially appreciates contribution to windows content

20210913

Monday, September 13, 2021 3:16 PM

Debugging code

- Software get replaced not when they wear out but when they crumble in their own weight because they have become too complex
- You are always exposed to the symptom and don't know how to fix
- Why does this occur?
  - Programming is often treated as a means to an end
  - But how do you actually build the code properly?
  - Coding is a craft
  - Time it takes you to code is part of your algorithm time too
    - 3 days to code to run in 30seconds when two minutes of coding takes 1 minute to run, you wasted time
- Gems of Clean Code
  - Structure
    - How is someone going to use this code I'm writing?
    - Don't write a file everytime a function adds two numbers…
    - Minimize side effects
    - Each function should do one thing
    - Have very clear blocks and components to your code
      - This also helps with debugging

        - Minimize debugging space
    - Method
        - Top down approach
            - Pretend you have access to all the helper functions to write the main, and THEN implement all the helper function
            - Ideally it runs
            - Then you only write the code you need
            - This works well if you have a type checker
                - Harder with non-typed languages
        - Bottom up approach
            - Try to foresee all the components you need, then try to implement them
            - At every step you have runnable code
            - Might implement something you don't need
        - Solve the problem first - then improve/refine
            - Don't try to optimize at the same time that you initially implement
                - At least you will have something working
        - Boring code is good code: easy to read
        - Late binding: start vague and refine (don't commit to specificity too soon)
            - Length of list vs length of list of integers

    - What is the superset of the problem I'm solving
- Many functions with small bodies is typically better than one function with large body

Next time: demo of live coding, can reach out with questions
Office hours on Wednesday