1h25m

Bootcamp
Git Branching

https://github.com/HackerYou/bootcamp-notes/blob/master/git-and-command-line/git-branching.md

Git Branching

Key Objectives:

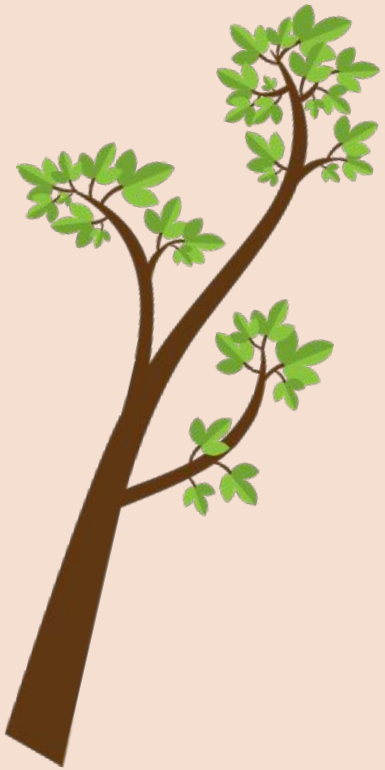
1. Main repo with files on GitHub
2. Keep a clean copy locally
3. Tinker with some new feature
4. Implement a wild idea
5. Collaborate with others (later!)
6. Implement new small tasks
7. Avoid conflicts!
8. Usually: main, development, test, production

Git BRANCHES!

Git Branching

BRANCH:

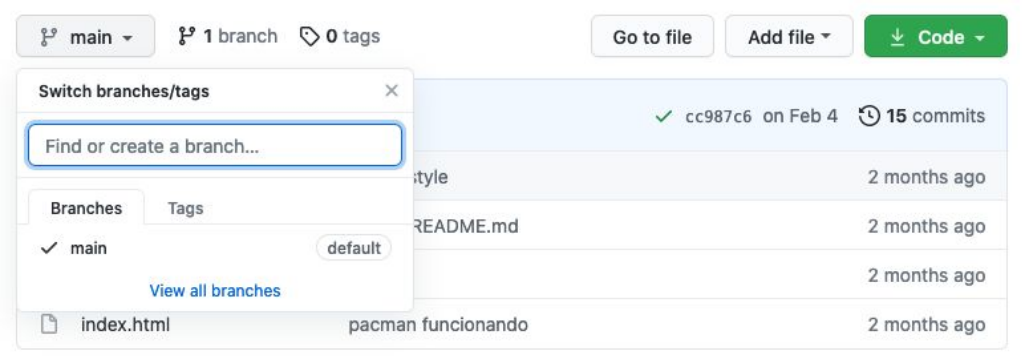
- It is an **isolated** copy of your current work.
- When a branch is created, it is an exact replica of your last **committed** code.

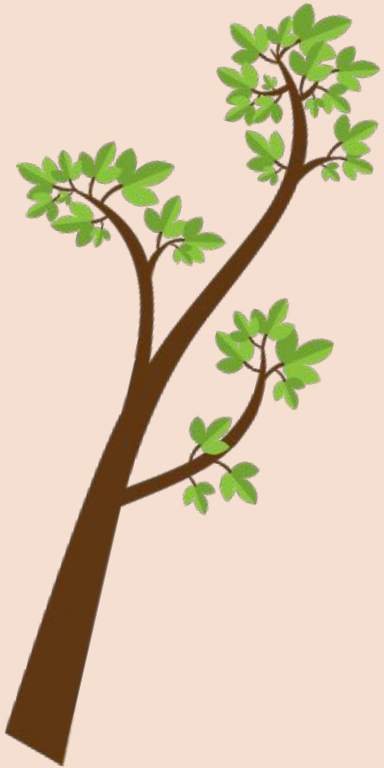


Git Branching

~~MASTER~~ MAIN BRANCH:

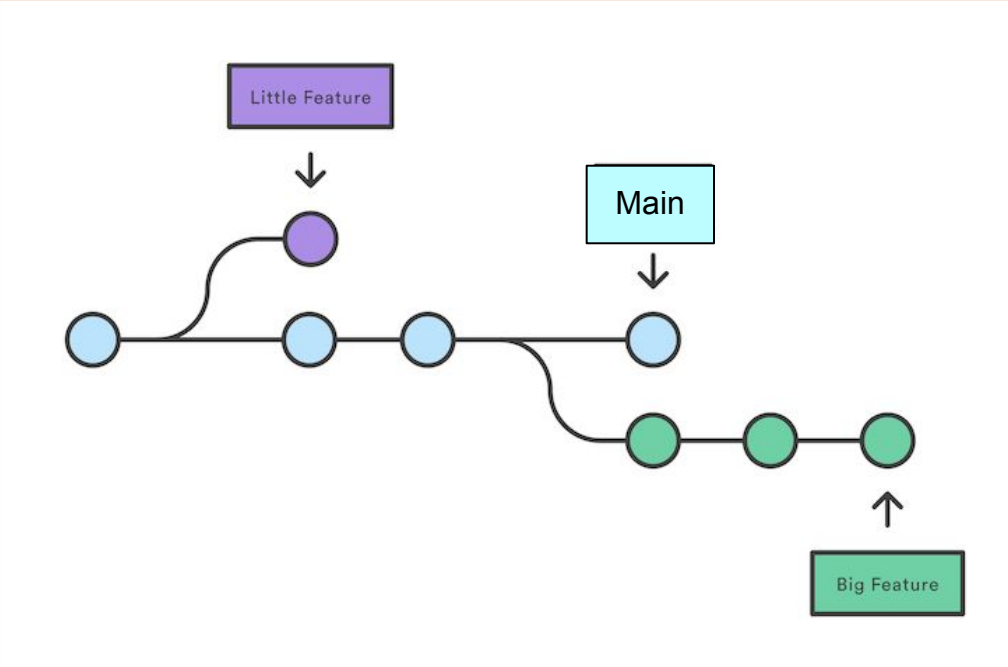
- When you initialize a new repo with a project, you only have one branch - your ~~master~~ main branch!





https://github.com/github/renaming

Git Branching:: Workflow



The ~~master~~ **main branch** is where the most current, tested version of your code lives.

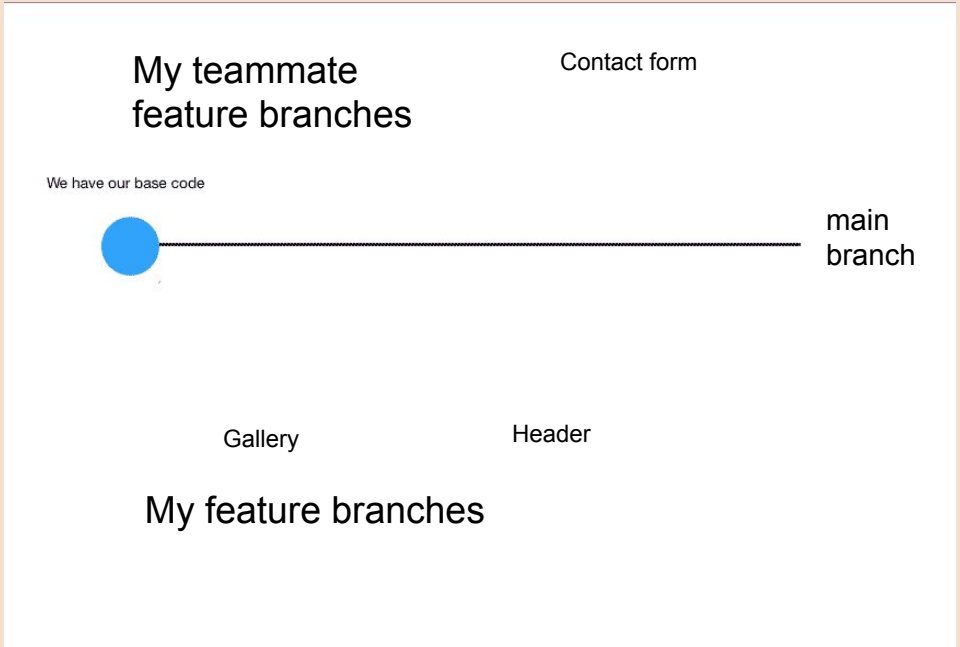
The **production branch** is the official finished version of the repo, and probably the one that is live on the internet.

Other possible options include all the **development branches** and **testing branches.**

Development and testing branching are a great way of quarantining your code so you can build parts of the project without fear of polluting your ~~**master**~~ **main branch**.

Feature branches are specific types of **development branches** where individual developers are writing code and figuring out new features or updates for the website.

Git Branching:: Workflow



The ~~master~~ **main branch** is where the most current, tested version of your code lives.

The **production branch** is the official finished version of the repo, and probably the one that is live on the internet.

Other possible options include all the **development branches** and **testing branches.**

Development and testing branching are a great way of quarantining your code so you can build parts of the project without fear of polluting your ~~**master**~~ **main branch**.

Feature branches are specific types of **development branches** where individual developers are writing code and figuring out new features or updates for the website.

Git Branching:: Setting up

- Download these starter files.
- Navigate to the directory:
  > cd git-branching-lesson
- Go to GitHub and create a repo called git-branching-lesson
- Go back to your command line
- Initialize git
  > git init
- Add the files
  > git add .
- Create the first commit:
  > git commit -m "first commit"
- Change the branch name to main:
  > git branch -M main
- Link the repo to the files adding git origin to your directory
  > git remote add origin https://github.com/anarodrigues/git-branching-lesson.git
- Push to the new repo:
  > git push -u origin main
- Reload the GH repo, you should see the index.html
- Open the code on VSCode

Git Branching:: Creating a new feature branch

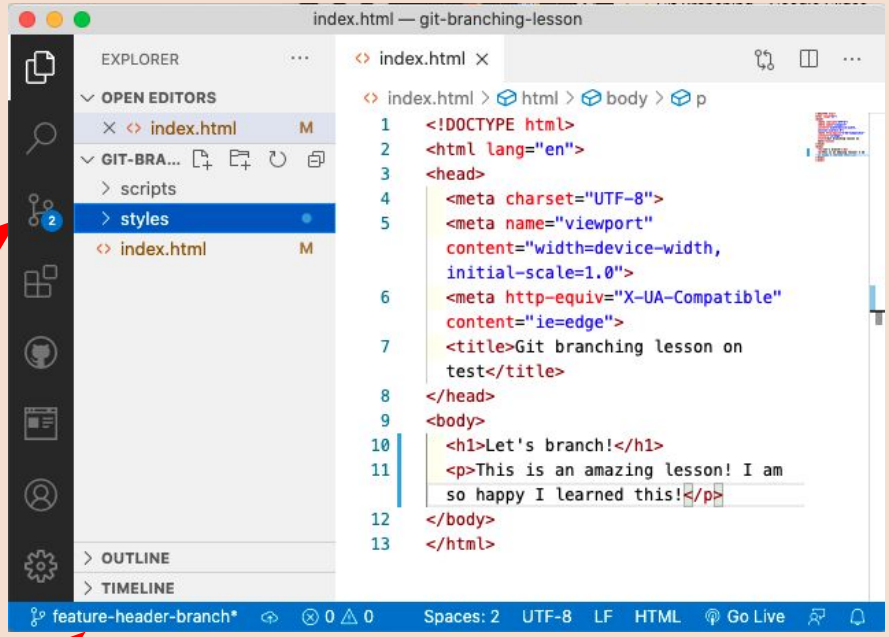
- Create a new branch:
  > git checkout -b feature-header-branch main

  Switched to a new branch 'feature-header-branch'

- Open this branch:
  > code .
- Add an h1
- Write some content

Let's branch!

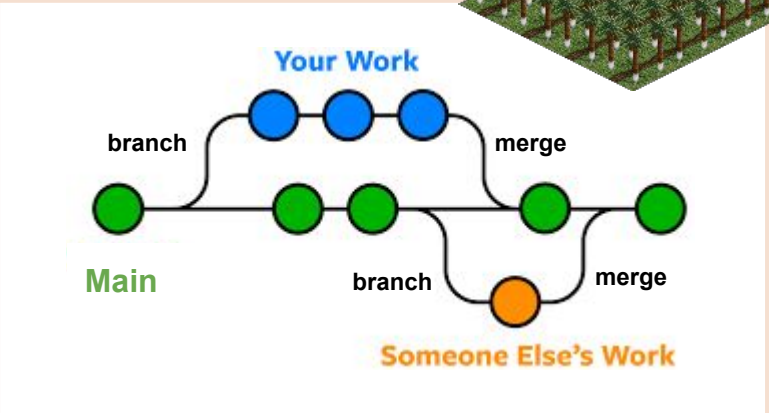
This is an amazing lesson! I am so happy I learned this!



When naming git branches, each workplace will have guidelines.
Recommended: consistent and descriptive. **branch-2** vs. feature-header-nav

Git Branching:: Working every day

- Pull from master first!
  > git pull origin main
- Check which branch you are:
  > git status or > git branch
  On branch feature-header-branch
- Work on your code
- > git add -A, > git commit -m “message” and > git push origin **feature-header-branch** until your code is done. That will **NOT** change the main branch.
- When you are finished with the feature:
  > git checkout **main**
  > git pull origin **main**
  > git merge **feature-header-branch (PRs later)**
  > git push origin **main**



Optional: delete the branches
> git checkout main
> git push origin :feature-header-branch
> git branch -d feature-header-branch (LOCAL)