# Creating Repositories and Collaborating with Git

// FLATIRON SCHOOL

# Learning Objectives
## (You Will Be Able To)

- **Initialize** a new repository
    - Connect that new repo to GitHub
- Create and switch between git **branches**
- Implement a **branching workflow**
- Avoid **merge conflicts**

# Where We Left Off

**Covered in ‘Introducing Git and GitHub’**

- Starting from an existing repository
- Forking workflow
- Pushing changes

**What haven’t we covered?**

- Creating your own repo from scratch
- Branching workflow
- Merging changes

# Initialize a New Repository

**Initializing** turns any local directory/folder into a local git repository.

```
git init
```

This creates a repository using ONLY Git, without involving GitHub at all.

**Do not create a Git repository inside of another Git repository!**

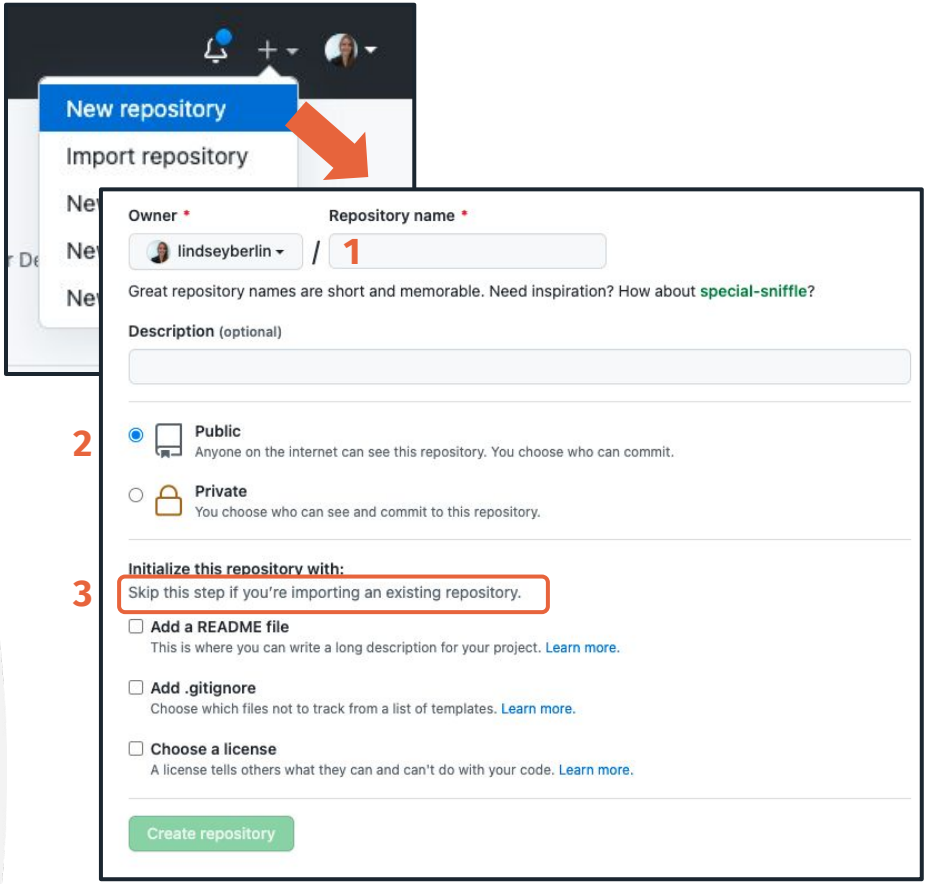
- Use `git status` to check!

# Connect a Local Repository to GitHub

Easy to create a new repository on GitHub.

Then:

1. Name your new repository (will become the URL)
2. Set permissions
3. **READ CAREFULLY!** If you already have an existing repository on your local computer (already ran `git init` in a folder with contents in it), make sure you DO NOT click any of the options!



# Git Branches

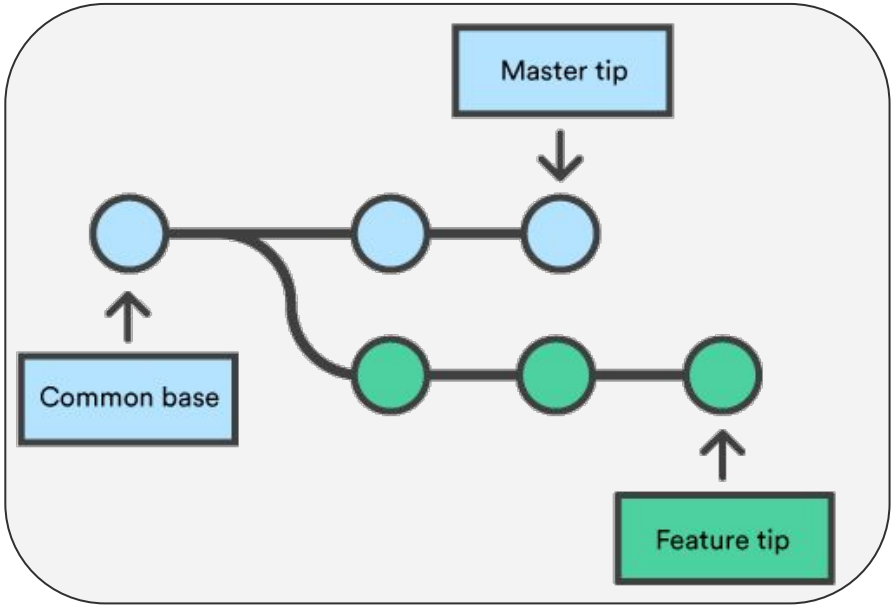
**Branches** allow you to do your own work based off a main ‘trunk’ of code, without disrupting other people working off that ‘trunk’.

Default branch name: `main` (or `master`)

`git branch`
Check your branches

`git branch [BRANCH]`
Create a new branch, named `[BRANCH]` - but won’t move you to that branch!

`git checkout [BRANCH]`
Move to the branch named `[BRANCH]`



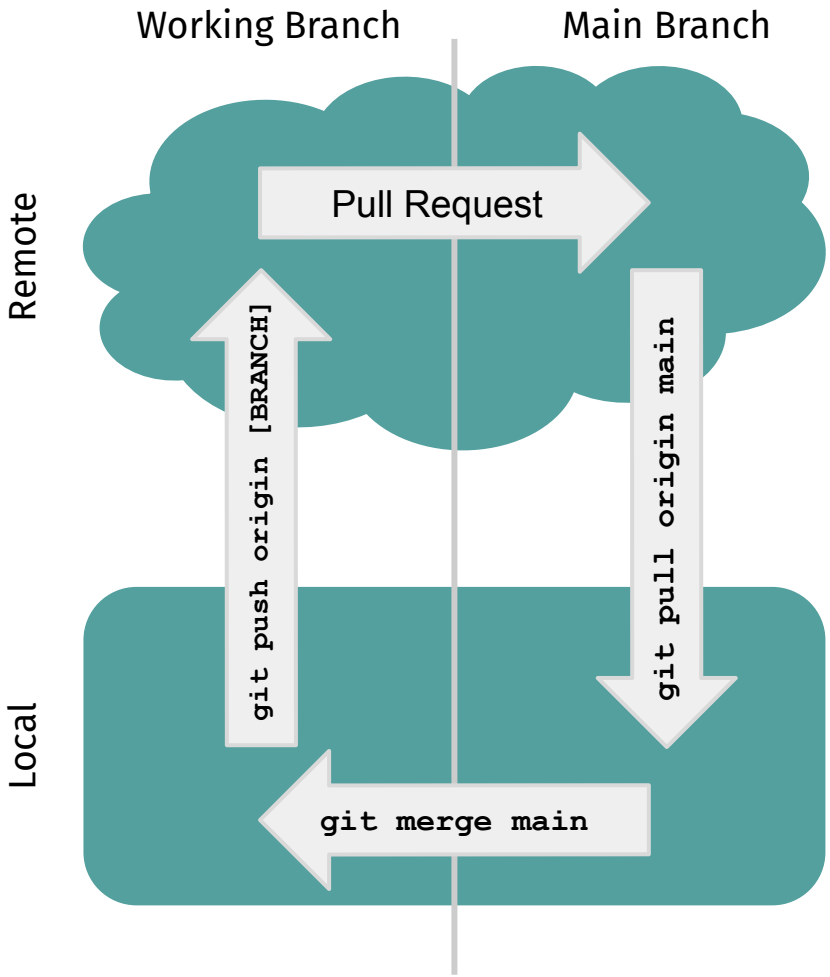
# Branching Workflow

A **branching workflow** is one common way to collaborate with others using GitHub.

One repository - many branches.

**The Idea:** code is written on a feature branch, then merged into the main branch via a **pull request** (a request to pull the content into main).

**You will use this workflow for your projects!**



# Merge Branches

**Merging** allows you to bring changes together into one harmonious project . . .

HOW TO AVOID MERGE CONFLICTS:

- Plan ahead and communicate
- Work on different Jupyter notebooks
- Use your own branch

# Time to Try It!

# Any Questions?