

# 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!

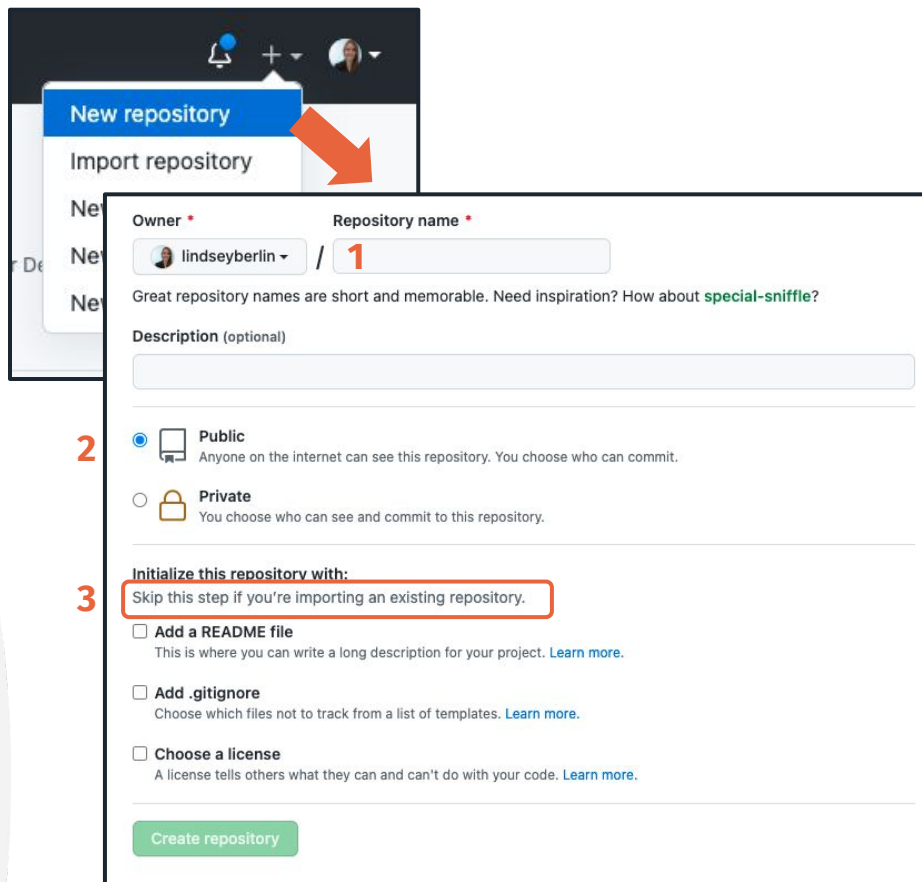


# Connect a Local Repository to GitHub

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!




The screenshot shows the GitHub 'New repository' form. A red arrow points to the 'New repository' button in the top navigation bar. The form itself is divided into sections. A red box with the number '1' highlights the 'Repository name' field, which contains the text '1'. A red box with the number '2' highlights the 'Public' radio button under the 'Visibility' section. A red box with the number '3' highlights the 'Initialize this repository with:' section, which includes the text 'Skip this step if you're importing an existing repository.' and three checkboxes: 'Add a README file', 'Add .gitignore', and 'Choose a license'. The 'Create repository' button is at the bottom.

New repository

Import repository

Owner <sup>\*</sup> / Repository name <sup>\*</sup>

 lindseyberlin / 1

Great repository names are short and memorable. Need inspiration? How about [special-sniffle?](#)

Description (optional)

2 ☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

3 **Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

# 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
```

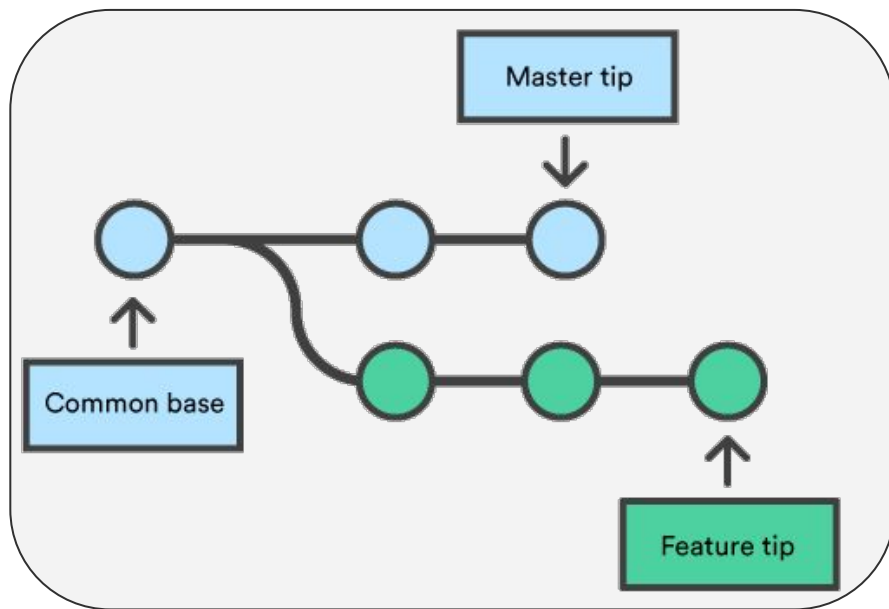
Check your branches

```
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]`



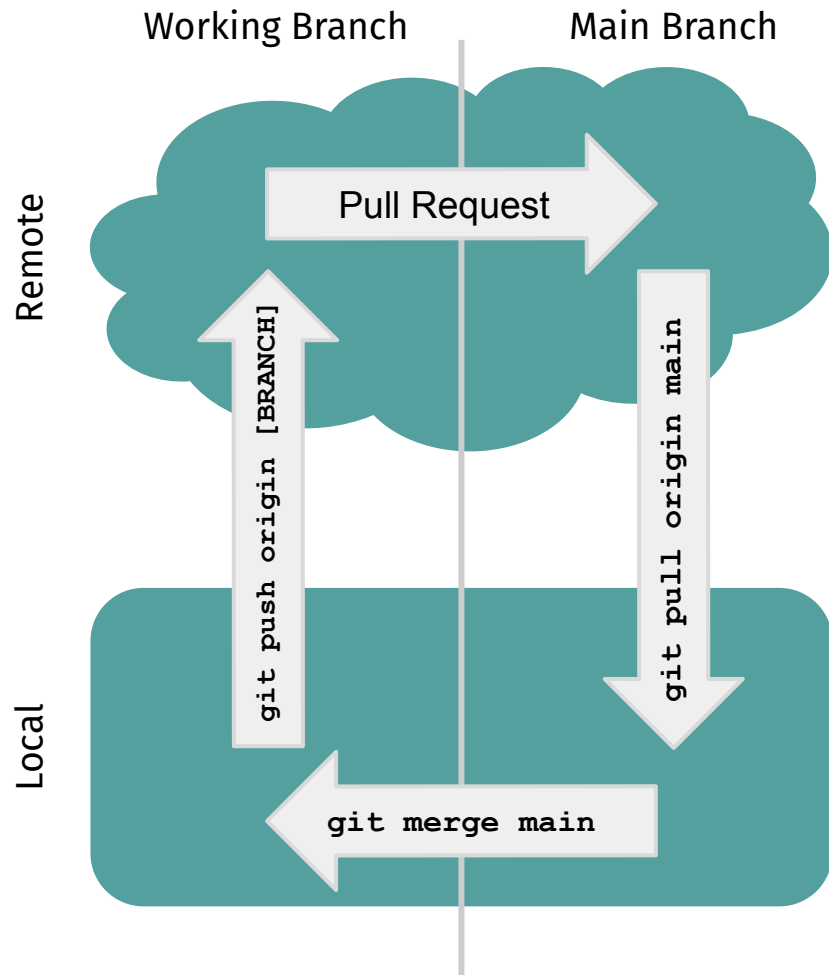
# Branching Workflow

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?*

