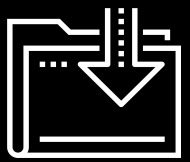
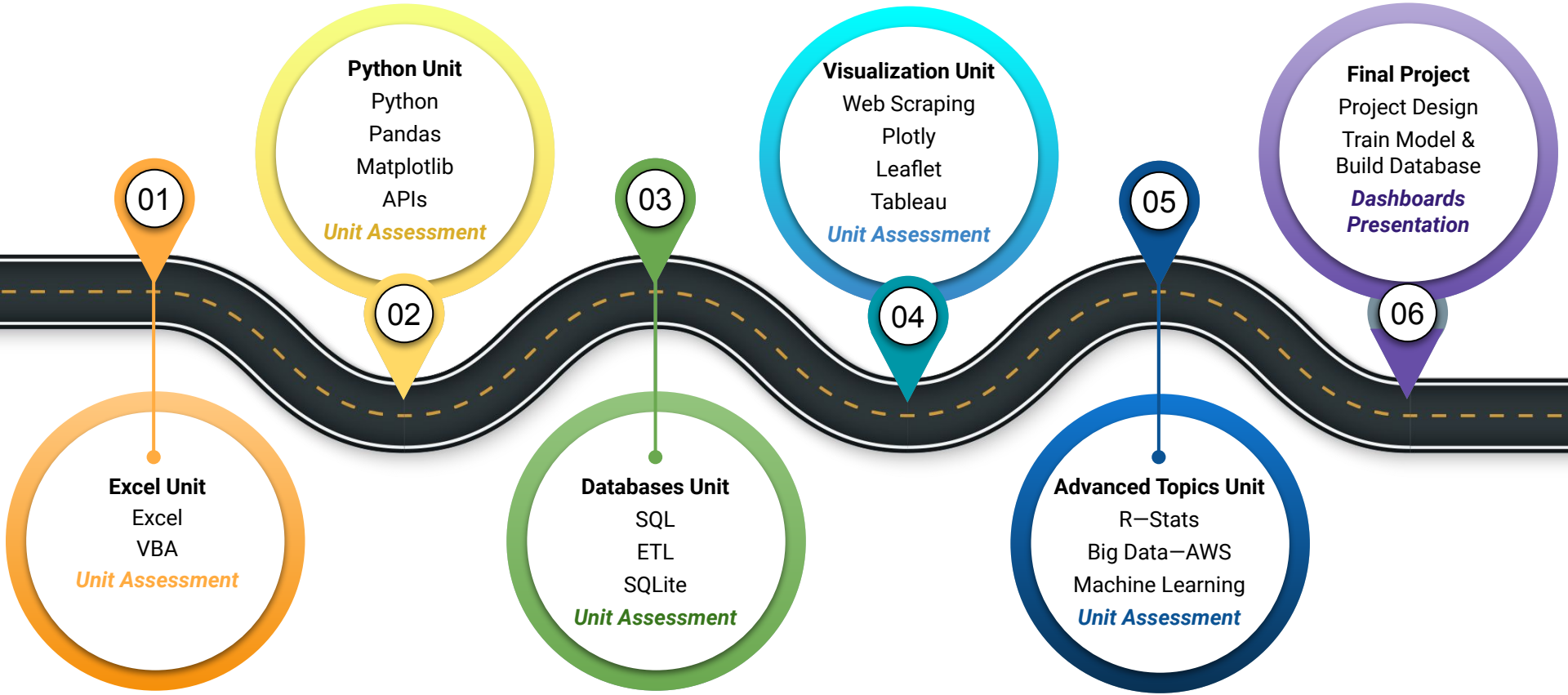
Capstone Project: Segment 1

Data Boot Camp

Lesson 20-1.2



The Big Picture



Today is the beginning of the group project weeks.

Quick Tip for Success:

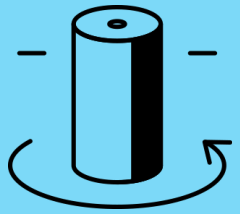
We'll begin working within our project groups this week. Take the time to connect with your group members early and often; communication is what will lead you all to success!

Project Segments

Your project work will be broken down into four segments

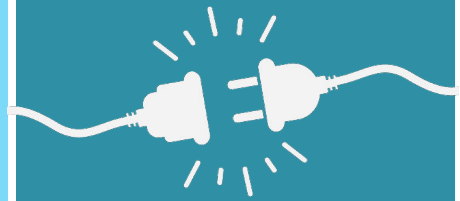
Sketch It Out

Decide on your overall project, select your question, and build a simple model. You'll connect the model to a fabricated database, using comma-separated values (CSV) or JavaScript Object Notation (JSON) files, to prototype your idea.



Build the Pieces

Train your model, and build out the database you'll use for your final presentation.



Plug It In

Connect your final database to your model, continue to train your model, and create your dashboard and presentation.

Put It All Together

Put the final touches on your model, database, and dashboard. Lastly, create and deliver your final presentation to your class.

This Week: Capstone Project

By the end of this week, you will:

- Decide on a topic for your project
- Find preliminary datasets for your project
- Perform rudimentary EDA on the datasets that have been found
- Create a mockup of the database
- Create a mockup of the machine learning model you are going to use

Module 20

Today’s Agenda

Today's Agenda

By completing today's activities, you'll learn the following skills:

01 Creating a GitHub repository for the project with feature branches

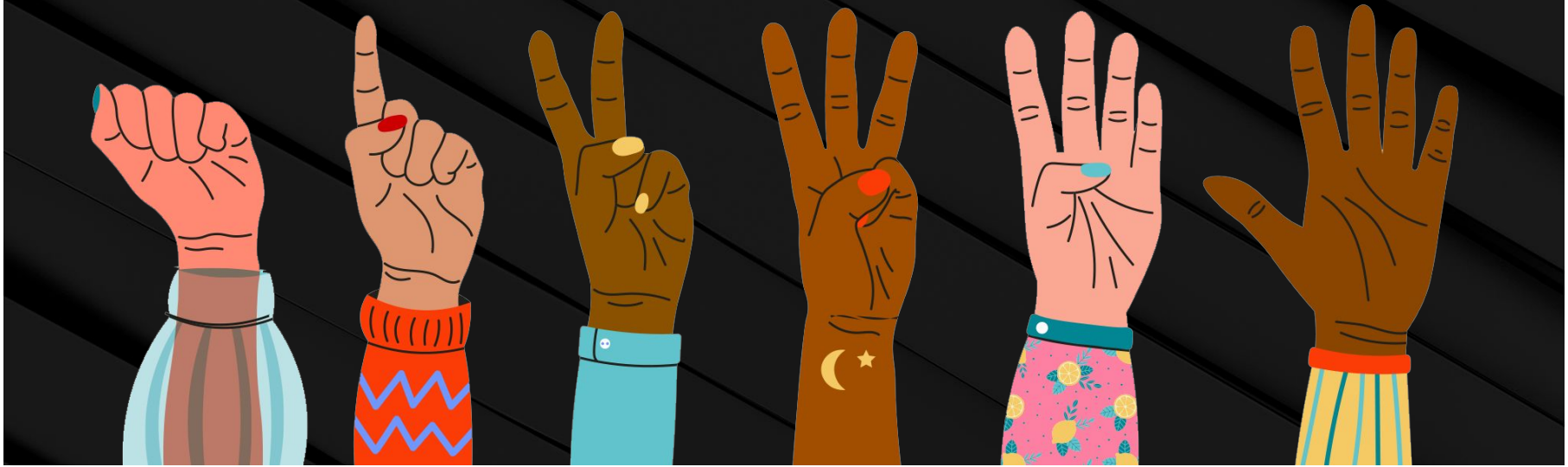
02 Adding a README.md file to the repository to describe the project

03 Working in a group environment

Make sure you've downloaded any relevant class files!

FIST TO FIVE:

Are you having any issues getting started?



GitHub Practice

What Is Git?

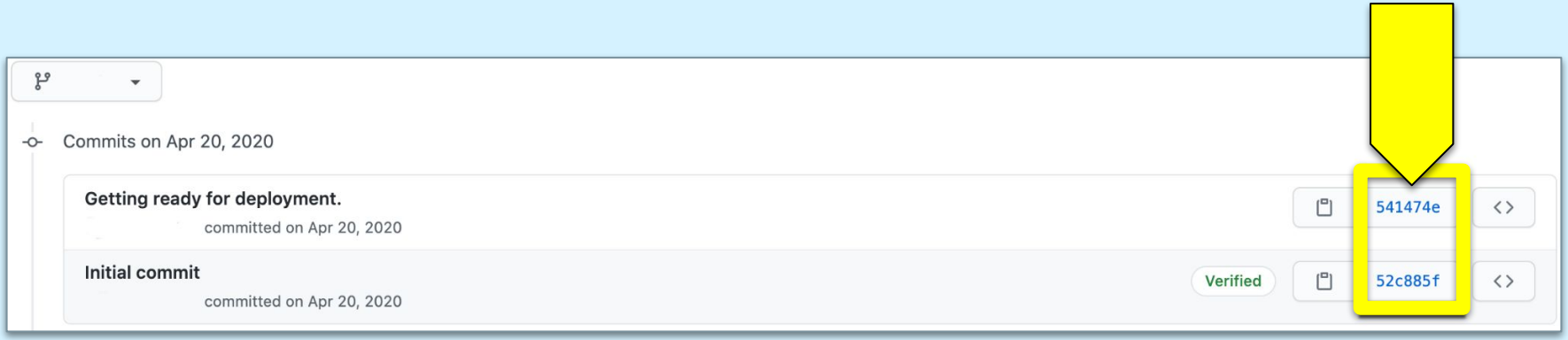
Git is a tool for saving our work as we develop a project.

Git keeps track of our work over time.

Git Commits

If we make a change that breaks something else in the project during the development phase, Git allows us to restore the working code from a previous commit.

Everytime you save using Git, Git remembers these “checkpoints” as a **commit hash**, which is a unique identifier.

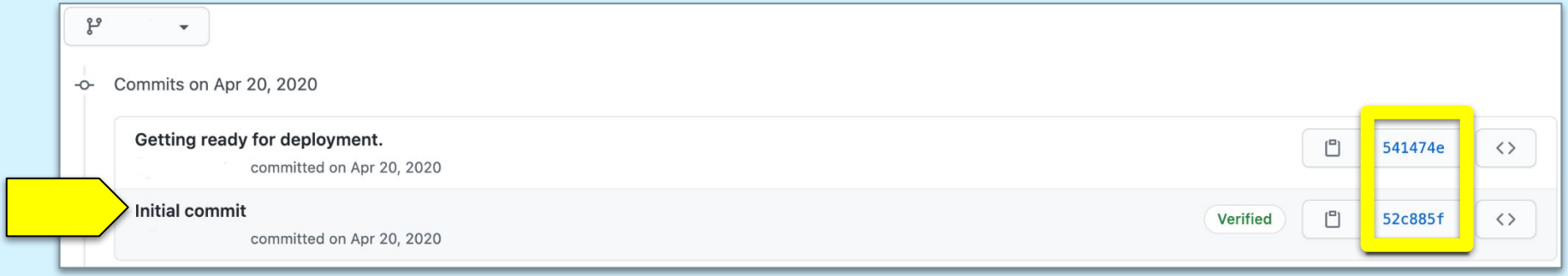


If we made a commit that "broke" something in another part of the code, then we can use this unique identifier to revert the code to a previous commit.

Git Commits

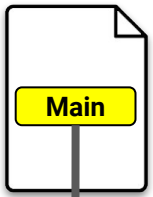
The following Git command reverts the repository to the initial commit shown in the image below.

```
git reset 52c885f
```



Git Branching

A branch is essentially a history of updates or changes.

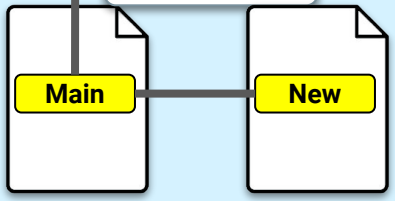


Staging area is where you edit the files that will be part of the next commit.

Staging area

Clone repo

branch



Create a new branch on your local repository.

As other team members make changes to the main code, their versions live on different branches.

Git Branching

You are working on a project where you need to update the main code in order to analyze demographic data for Uber riders. The code you work on will be added in a branch called `demographic_analysis`.

```
# Make sure you are in the main branch.
git status # If you are in the main branch this command will return, "On branch main".

# Pull the latest changes from main.
git pull

# Clean up the main branch on your computer
git clean -xdf

# Create a new branch "demographic_analysis"
git checkout -b demographic_analysis
```

Git Branching

Next, you add the updated file from your computer to the repository and execute the following commands:

```
git status # This will tell what has been added, deleted or modified.

# If you are satisfied with the status of the commit, type.
git add . # To add all the content of your commit.

# Not necessary but a good practice is to check the status again.
git status

# Write a required commit message
git commit -m "adding demographic analysis"

# Push the changes to the branch.
git push --set-upstream origin <branch-name>
```

Git Branching

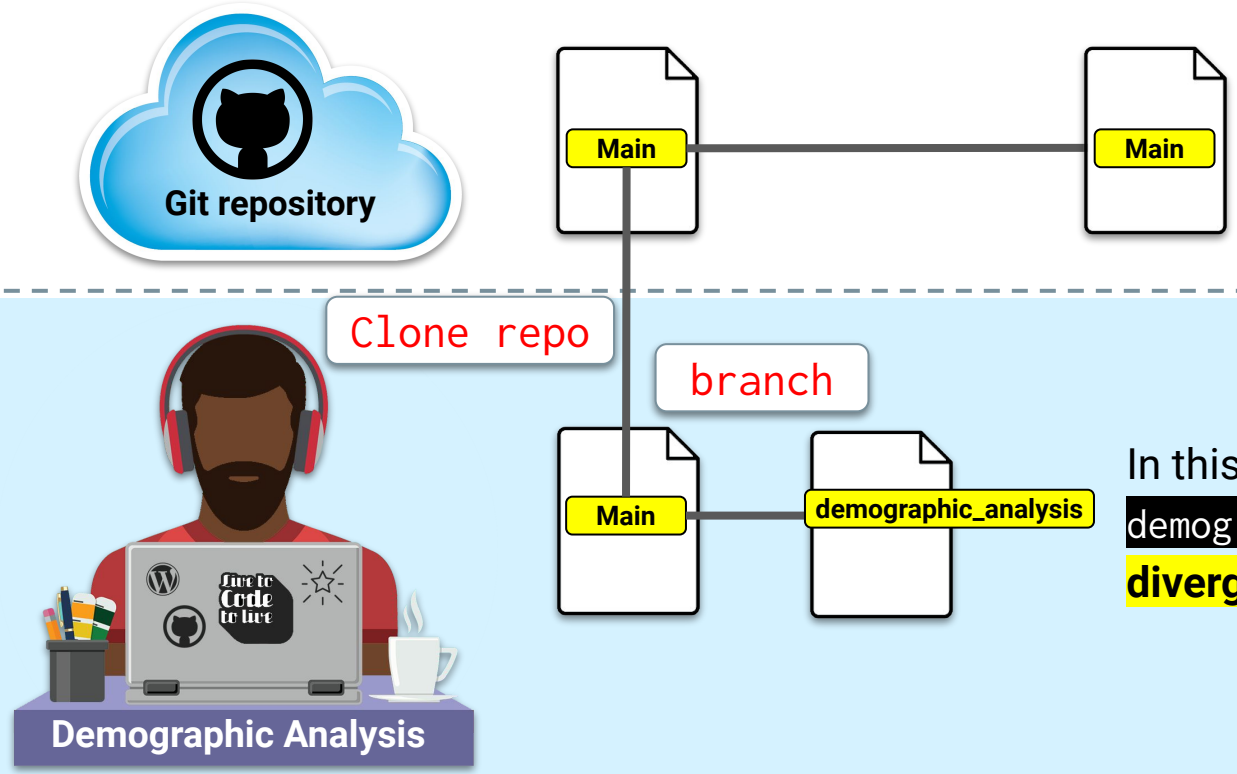
It is a good practice to navigate back to the `main` branch to perform a cleanup of repository files that are unneeded but not tracked by `.gitignore`.

```
# Switch back to the main branch
  git checkout main

  # Clean up the the repository
  git clean -xdf
```

Git Branching

We have a version of the code called `main`, which is the "main" version of our code, and a version called `demographic_analysis`, which contains updates.



In this case, the `demographic_analysis` branch **diverged** from the main branch.

Git Branching

The benefits of having a separate branch for analyzing Uber rider data:

01

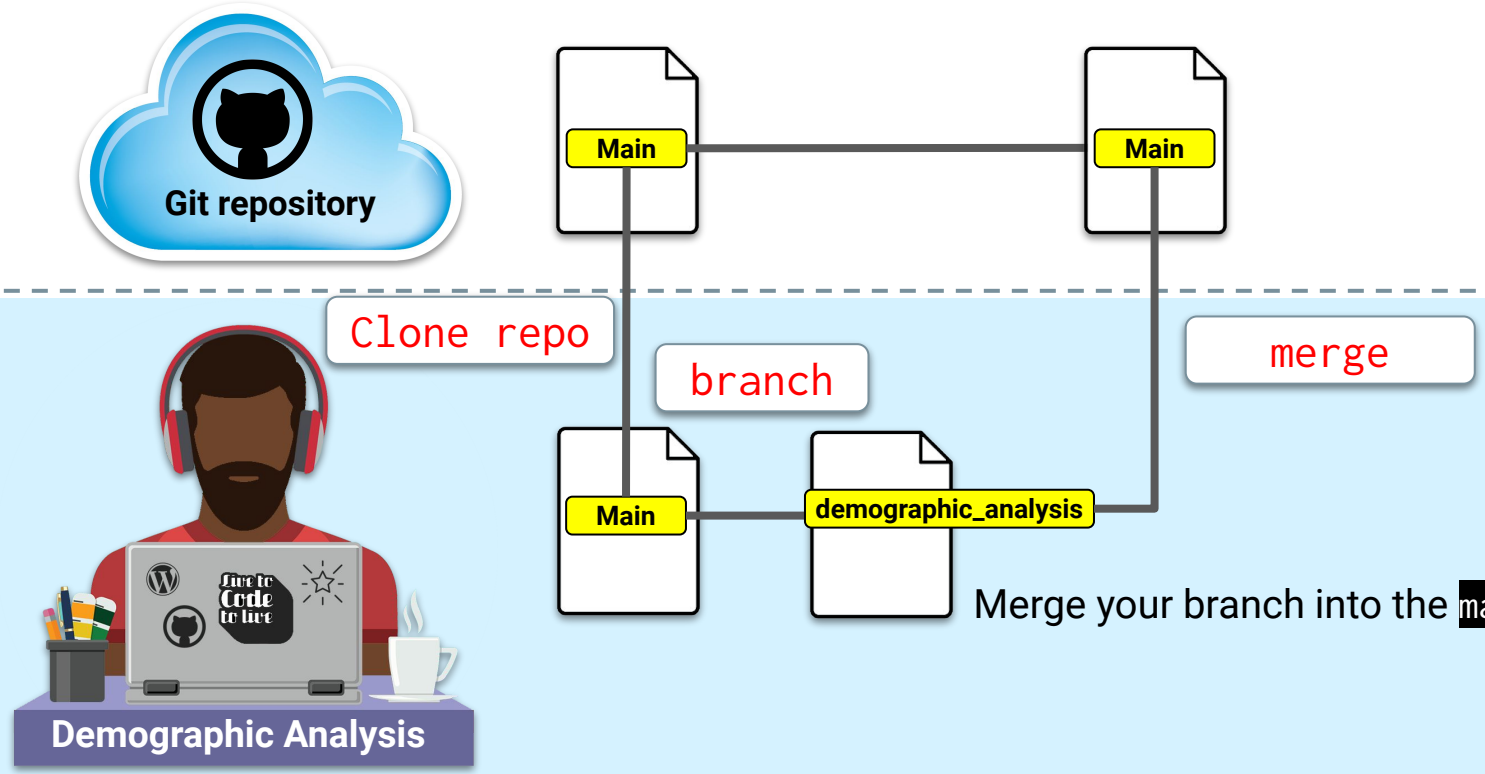
It gives our collaborators a chance to review the branch for errors and offer suggestions.

02

After the proposed changes have been reviewed, we can update the `main` branch to include the changes from the `demographic_analysis` branch by initiating a **merge**.

Git Merging

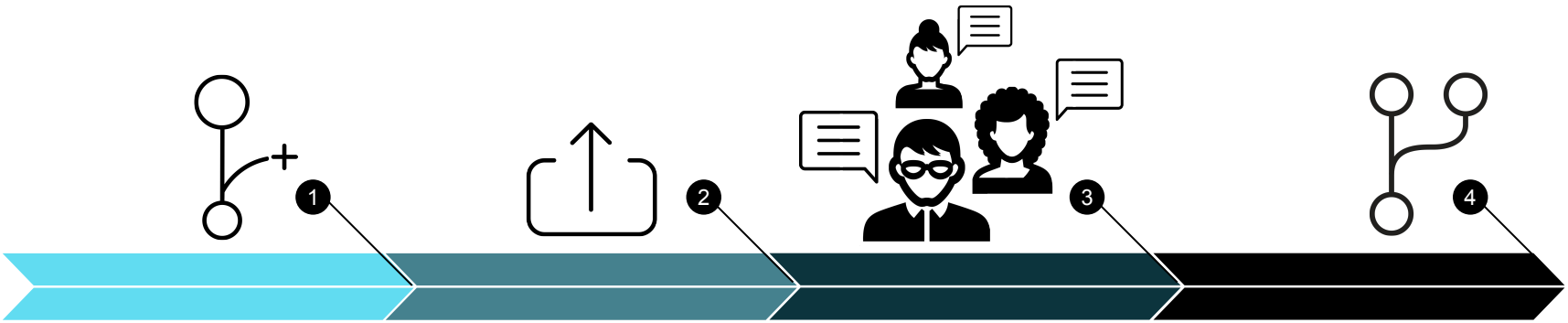
After the changes have been reviewed, we can **merge** the `demographic_analysis` branch into the `main` branch.



Merge your branch into the `main` repository.

Git Merging

Merging two branches adds the content from the changed branch to the branch it is based on.



Create a Branch:
Make some changes, commit the changes, and push the changes to the GitHub repo.

Open a Pull Request:
A pull request is essentially a request to merge code from one branch into another branch on GitHub.

Discuss and Review:
Teammates review the code on the pull request, suggest changes to new code, and eventually approve the changes to be merged.

Merge and Deploy:
After approval, you can merge the changes into the main branch and deploy the code.

Group GitHub Activities

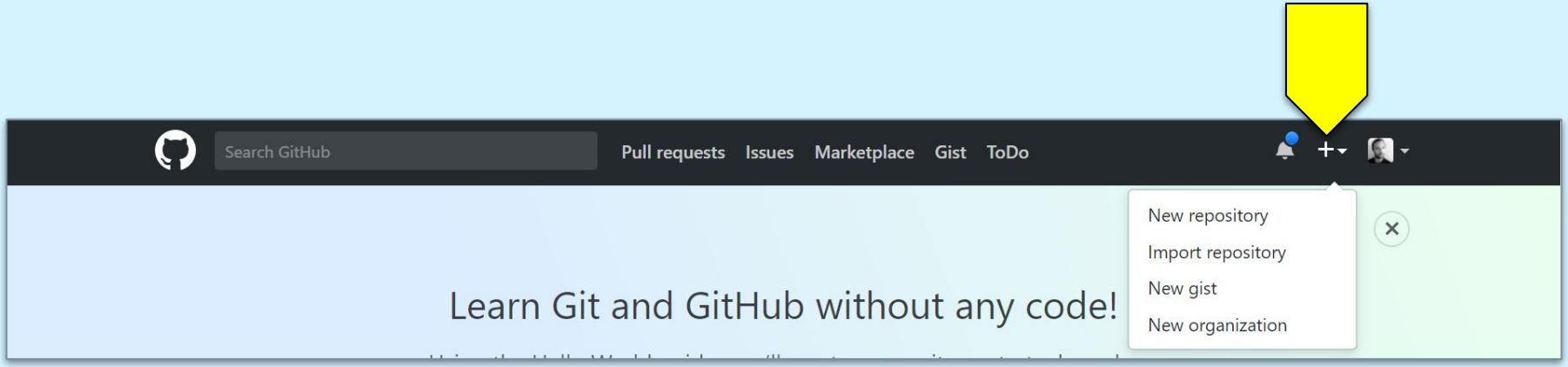
Create a Project Repository

Suggested Time:

10 minutes

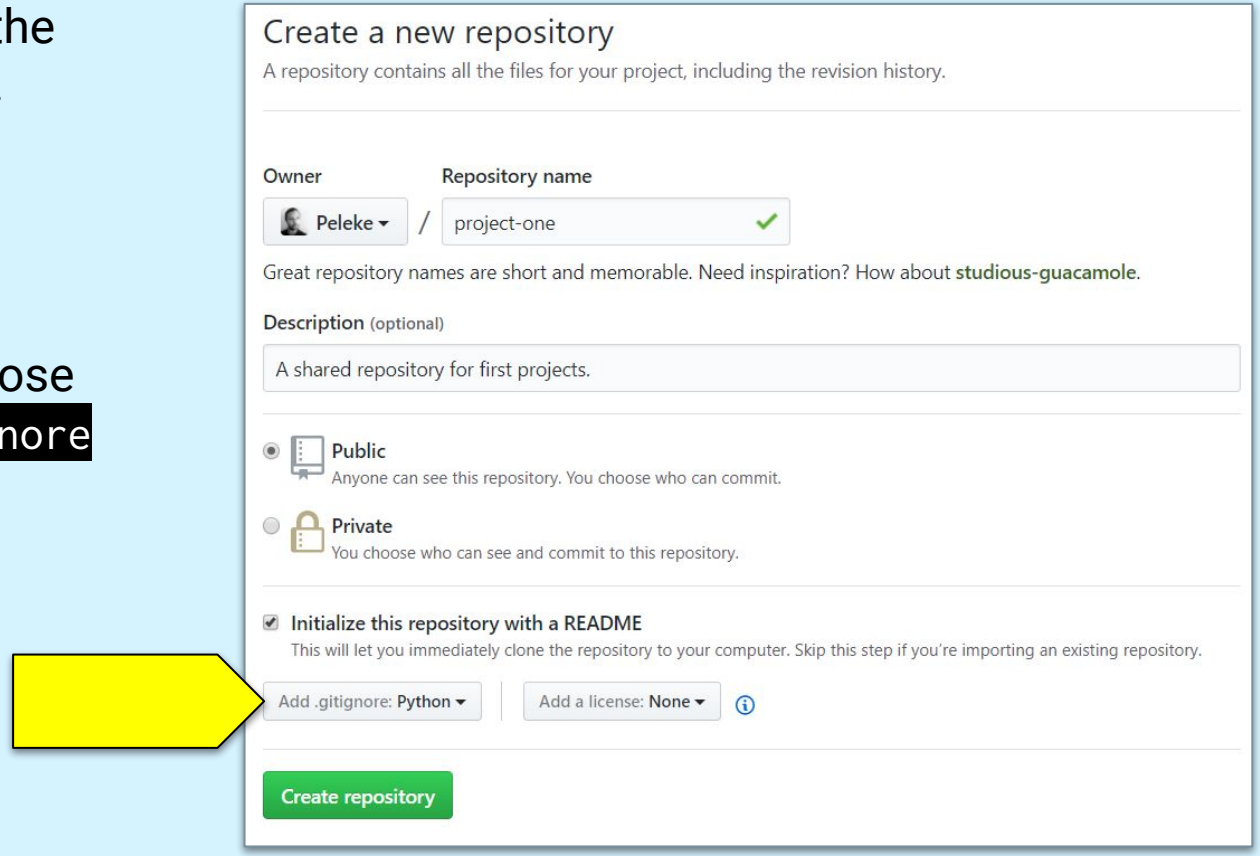
Create a Project Repository

Go to GitHub, then click the **+** in the top-right corner to create a new repository.



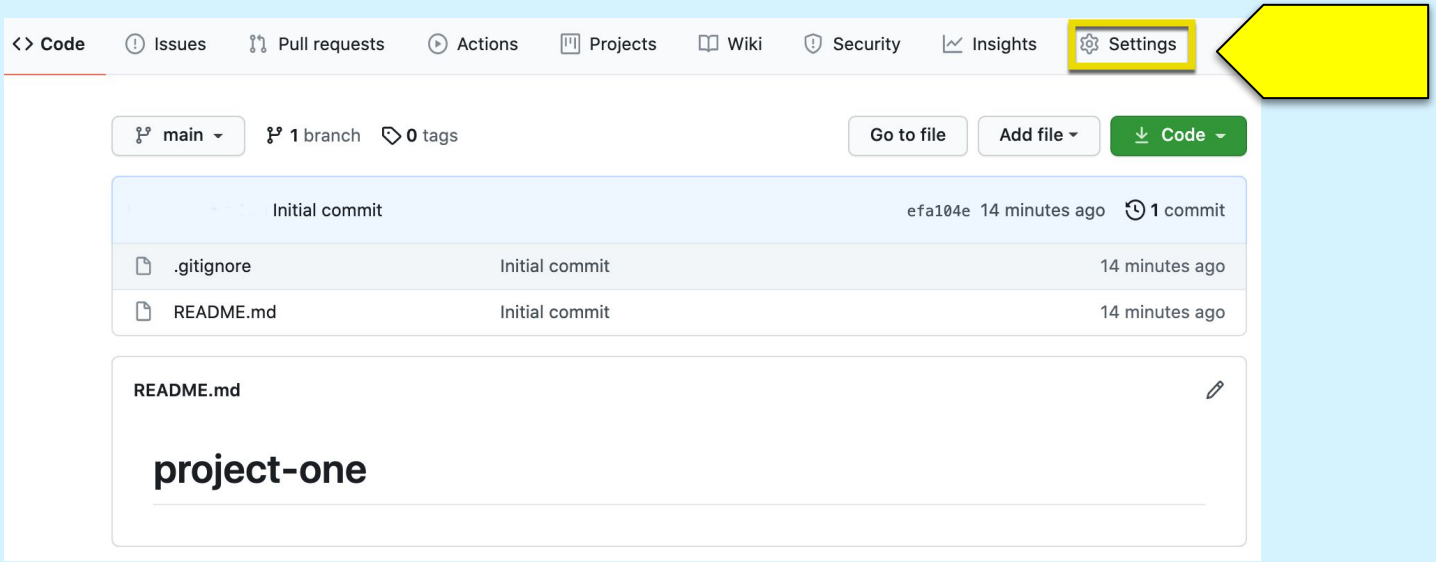
Create a Project Repository

- Fill out the fields on the new repository page.
- Initialize with a `.gitignore`.
- Students should choose Python in the `.gitignore` dropdown.



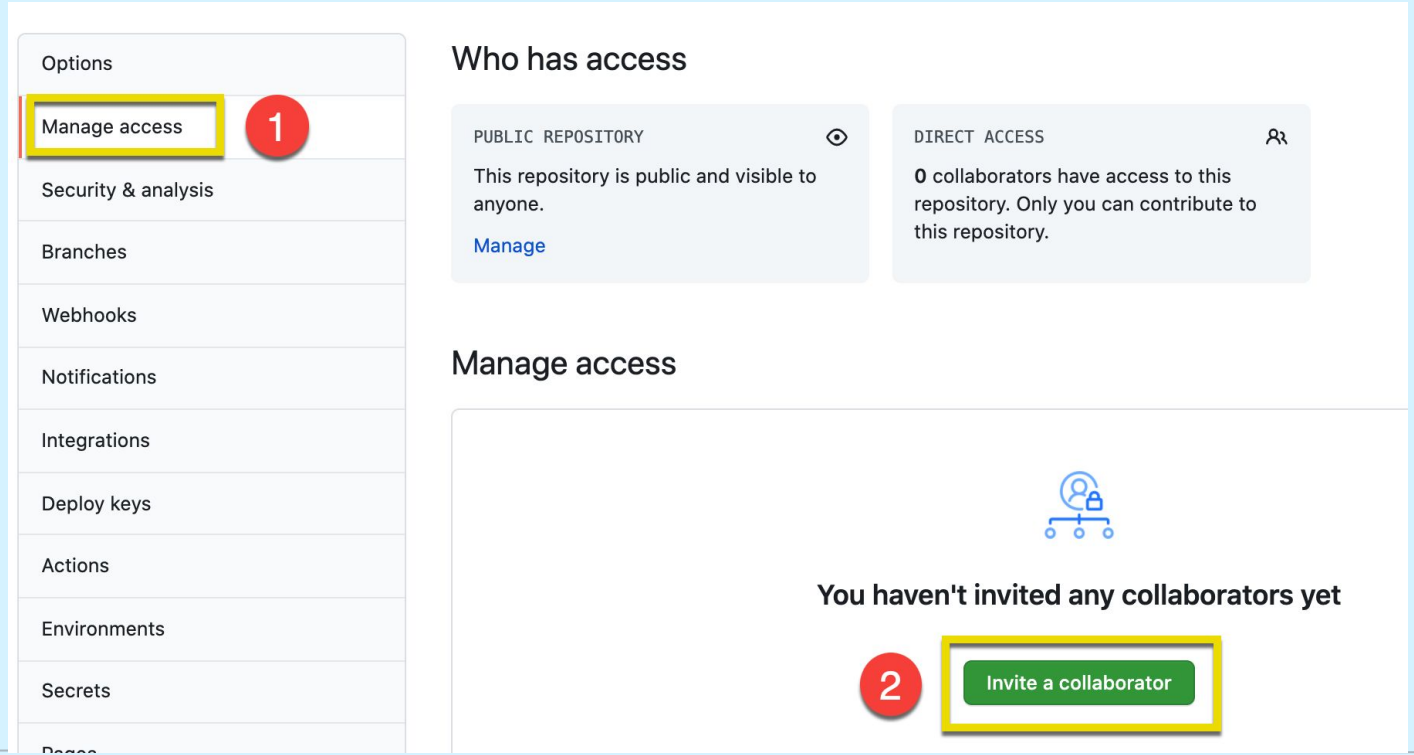
Adding Collaborators

Navigate to the repository settings.



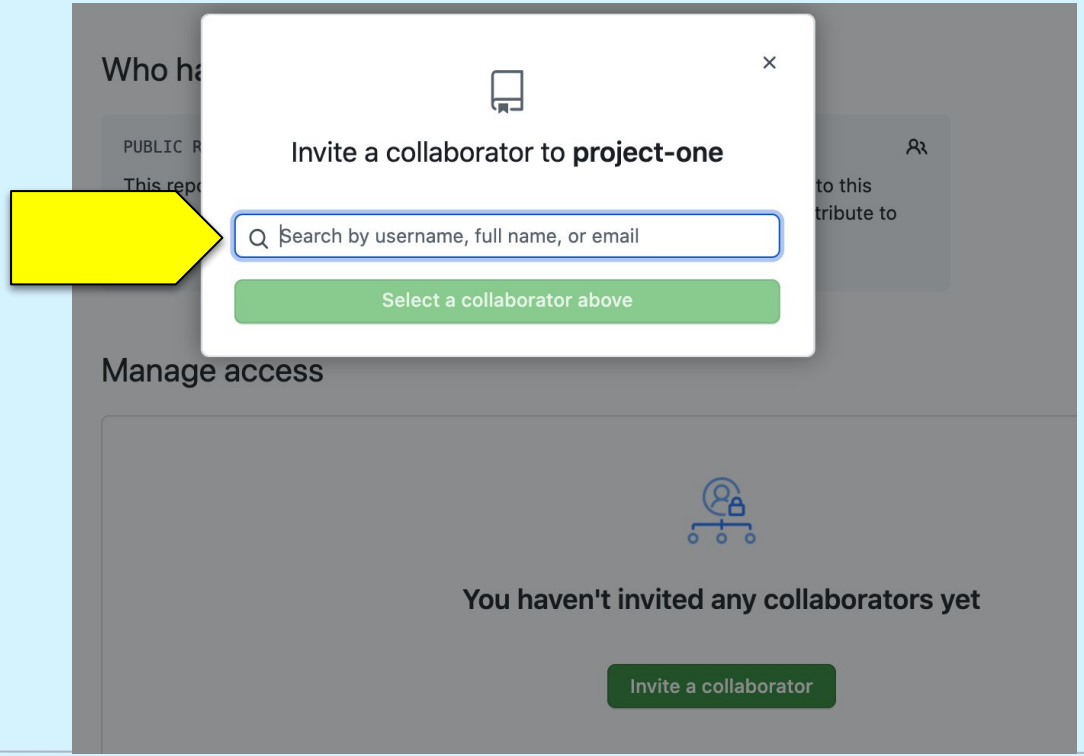
Adding Collaborators

On the "Settings" page, click "Manage access" on the left, enter your GitHub password when prompted, then click "Invite a collaborator."



Adding Collaborators

A popup window will appear for the owner of the repository to search for their teammates by GitHub username.



Questions?

Time to Code

Creating Branches

Suggested Time:

10 minutes

Clone from GitHub

If someone has already shared a repository on GitHub, you can **clone** it to your local machine with `git`.

```
# Clone an existing repository.
git clone <repo_url>
# Navigate into newly created repository directory
cd <repo_name>
```

Adding Files

In the examples below, we use `git status` before every `git commit`. This is a best practice that helps ensure a deliberate commit history.

```
# Create a file, called clean_data.py
touch clean_data.py

# Add and commit clean_data.py...
git add clean_data.py
git status
git commit -m "First commit."

# Add cleanup code to clean_data.py...
git add clean_data.py
git status
git commit -m "Clean up provided data."

# Add code to export clean data...Note that `add .` adds
# everything in the current folder
git add .
git status
git commit -m "Export clean data as CSV."
```

Create Branches

To create a new, isolated development history, we must **create branches**.

```
# Create new branch and switch to it
# Long form: `git checkout --branch data_analytics`
git checkout -b data_analytics
```

Create Branches

Once we've created a new branch, we can develop as normal and then add those changes to the branch.

```
# Create file to contain data analysis
git add analysis.ipynb
git status
git commit -m "Add Jupyter Notebook for data analysis."

# Add notebook cells summarizing data
git add analysis.ipynb
git status
git commit -m "Adding summary tables to Jupyter Notebook."

# Export analyzed data and/or plots
git add .
git commit -m "Exporting analysis results and save plots as PNG files."

# Push the changes to GitHub so others can access your work
git push --set-upstream origin data_analytics
```

Questions?

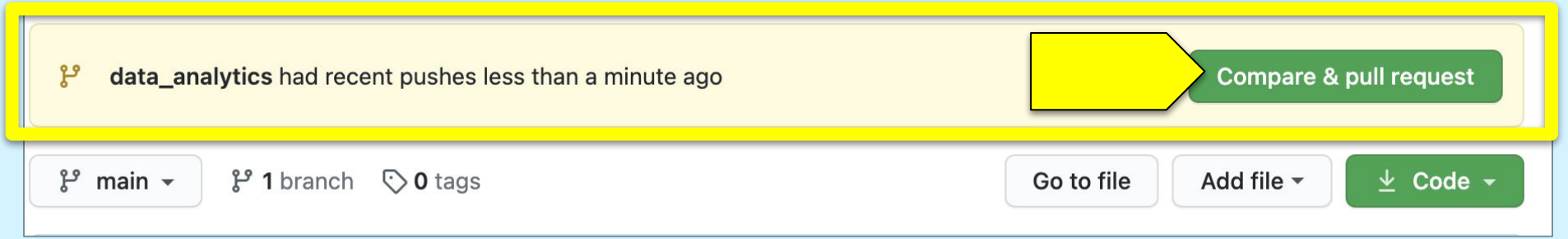
Creating a Pull Request, Reviewing Code, and Merging

Suggested Time:

10 minutes

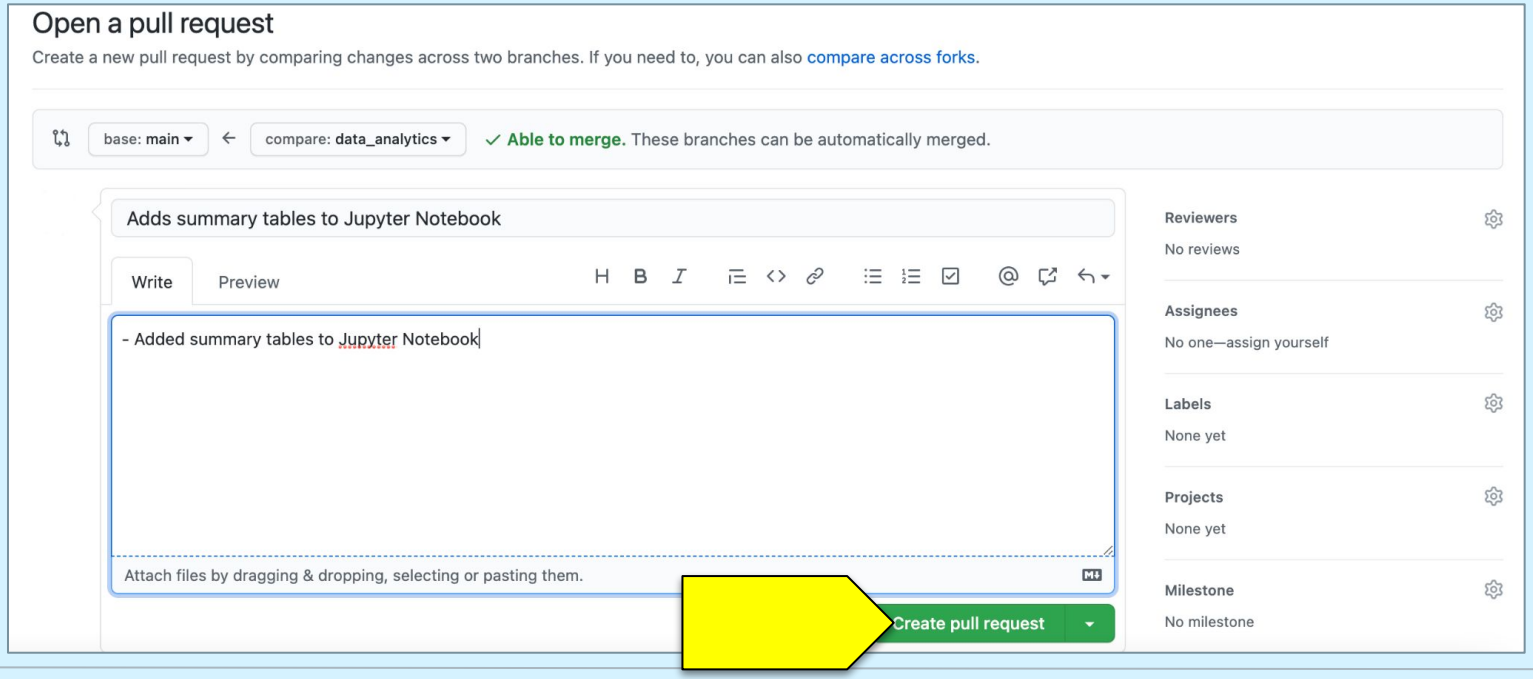
Creating a Pull Request, Reviewing Code, and Merging

Once you have pushed the changes to your branch, GitHub will create a message for you to compare the changes you made against the `main` branch.



Creating a Pull Request, Reviewing Code, and Merging

After you click "Compare & pull request," GitHub will create a "pull request," or a PR, where you can add a description of the changes you made to the "main" code.



Creating a Pull Request, Reviewing Code, and Merging

Use the `checkout` command on the `main` branch, then navigate to the repository on your computer.

```
# Move back to main
git checkout main
```

Creating a Pull Request, Reviewing Code, and Merging

Type and run `git pull`. This will bring the changes into the `main` branch.

```
# Bring in the latest changes.
git pull
```

Creating a Pull Request, Reviewing Code, and Merging

Then, run `git checkout <branch_name>`, where `<branch_name>` is the name of one of your teammates' branches.

```
# Checkout the branch.
git checkout data_analytics
```

At this point, your teammates can review and test the code. After testing the code, a teammate can approve the changes or request that changes be made.

Creating a Pull Request, Reviewing Code, and Merging

Once the code has been approved, the `data_analytics` branch can be **merged** into the `main` branch.

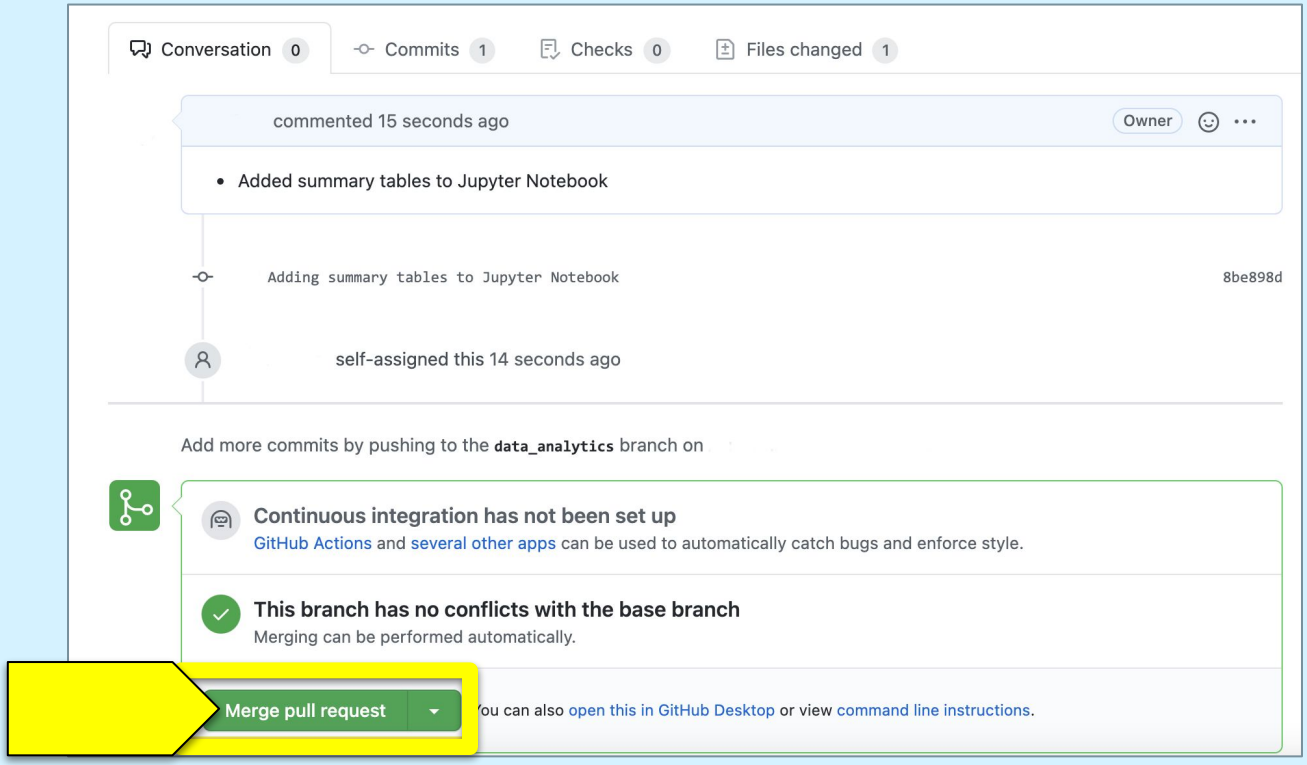
```
# Move back to main
git checkout main

# Merge changes on data_analysis with code on main
git merge data_analytics

# Delete the data_analysis branch
git branch -d data_analytics
```

Creating a Pull Request, Reviewing Code, and Merging

Alternatively, you can `merge` the branch on GitHub.



Deleting the `data_analytics` branch isn't necessary, but it is best practice to prune unneeded branches.

Working in Teams

Working in Teams

Each team member should have one of the following roles:

Square	The team member in the square role will be responsible for the repository.
Triangle	The member in the triangle role will create a mockup of a machine learning model. This can even be a diagram that explains how it will work concurrently with the rest of the project steps.
Circle	The member in the circle role will create a mockup of a database with a set of sample data, or even fabricated data. This will ensure that the database works seamlessly with the rest of the project.
X	The member in the X role will decide which technologies to use for each step of the project. If the team has only three members, then all three members should be contributing to these decisions.

Project Development Consultation

Project Progression—Next Steps

Project Progression—Next Steps

For the next class, you should have completed the following:

Create a mockup of a machine learning model.

Create a mockup or fabricate a database.

Create an outline for the final dashboard with a storyboard of visualizations.

Continue using GitHub to create and merge code on branches.