


GitHub Guide

Initial Setup For GitHub in the Dev Environment	1
Initial Setup For First Assignment (+ Concurrency)	3
Initial Setup For Any Following Assignments	4

Using Git in your Dev Environment for the First Time!

Wow! Good job setting up your dev-environment for the assignments – now it's time to set up Git. If you have not set up the dev-environment yet, go to:

 [Setup Guide - 2024](#) before returning here.

Steps to set up Git for your docker environment:

(Note: If you already have Git installed locally, you can keep using your local version of Git and the changes will be propagated into your dev-environment as well! If you choose to use your local Git, then you can skip this initial GitHub setup.)

1. Set up your git email by running the command:

```
git config --global user.email "you@example.com"
```

(Note: replace you@example.com with your Github email address)

(Note 2: Make sure you keep the quotation marks around the email)

2. Set up your git name by running the command:

```
git config --global user.name "Your Name"
```

(Note: replace Your Name with... you guessed it: your name)

(Note 2: Make sure you keep the quotation marks around your name)

3. Now when you try to make a push to a GitHub repository, you will encounter this response:

```
Username for 'https://github.com':
```

Type in your GitHub username, and then press enter.

4. Then, you will see this response:

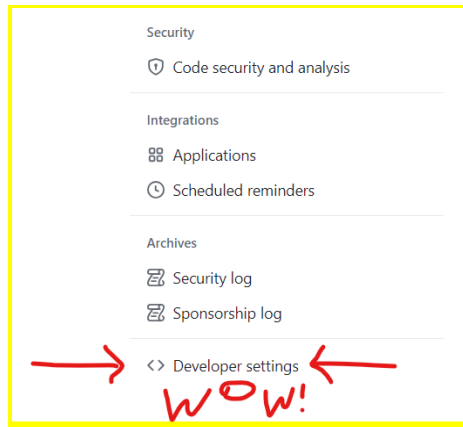
```
Password for 'https://<YOUR_USERNAME>@github.com':
```

Copy and paste over your personal access token into password, and then press enter. If you don't have a personal access token set up, follow the guide below!

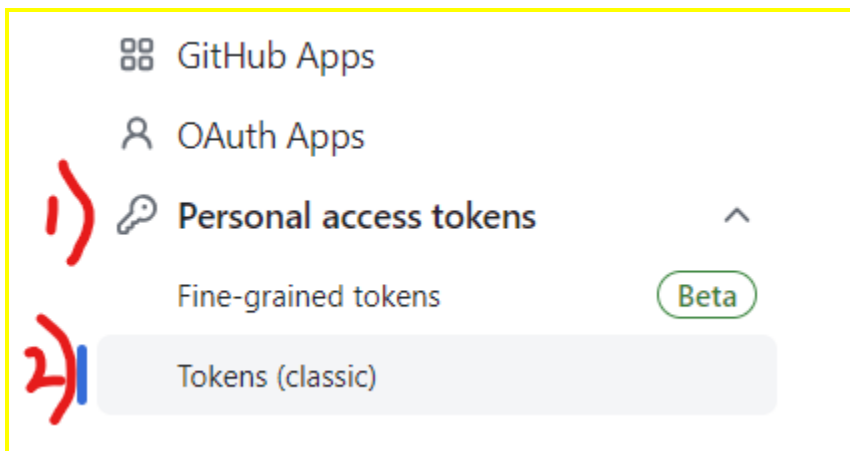
Guide to Setting up Your GitHub Personal Access Token!

Note: If you have not set up a personal access token, follow these steps:

1. Go to your GitHub profile settings
2. Select the **Developer Settings** tab on the bottom left



3. Click **Personal access tokens**, and then **Tokens (classic)**



4. Click **Generate new token** and then **Generate new token (classic)**

Personal access tokens (classic)

[Generate new token](#)[Revoke all](#)

Tokens you have generated that can be used to access the [GitHub API](#).

5. Type in a note as a description of the token, and select the **repo** box

The screenshot shows the 'New personal access token (classic)' page on GitHub. The left sidebar has 'Personal access tokens' selected, with 'Tokens (classic)' highlighted. A red arrow points from the text 'OPTIONAL!!' to the 'Fine-grained tokens' link. The main form has a 'Note' field with the placeholder text 'ENTER WHATEVER YOU WANT HERE LOL', circled in red with a red arrow pointing to it from the circled number '1'. Below the note is an 'Expiration' dropdown set to 'No expiration'. A yellow warning box states: 'GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)'. Under 'Select scopes', the 'repo' scope is checked, circled in red with a red arrow pointing to it from the circled number '2'. Other scopes include 'repo:status', 'repo_deployment', 'public_repo', 'repo:invite', 'security_events', 'workflow', and 'write:packages'.

6. Click **generate token** at the bottom of the page, and now **WRITE DOWN/COPY DOWN YOUR TOKEN!!!!** You will never be able to see it again after this 🤪🤪

Congratulations, you have successfully created your own personal access token to use as a password for git!

Initial Setup For Project 1 (Go)

We use GitHub Classroom to distribute the assignments throughout the semester. You will be working with just one repository for most projects, which will contain a directory for each project.

To begin, accept the [GitHub Classroom assignment](#) for Assignment 1: Go, and then clone the repo into the **home** directory of your dev-environment:

1. First accept the google classroom assignment.
2. Clone the repo. Do this by going to the GitHub Repo link created by classrooms, and clicking clone. Copy the url in the box, navigate into the home directory of your dev-environment in a terminal, and then use the following command:

```
$ git clone <google classroom url>
```

Note: Make sure to use the correct url for ssh or https

Once you cloned the repository, if you open it you should see the directories `cmd/dinodb`, `pkg`, and `test`, as well as some other supporting files. The only directories you will have to touch in this course is `pkg` and `test`.

The implementation work you will do for this course will be in the `pkg` directory, and the `test` directory is where you will write any of your own `tests` for your projects. If you open `pkg` you will see the directories `list` and `rep1`. These contain the files you will need to edit to implement the first project, `Go`. Throughout the course, we will add more directories to `pkg` for future projects you will complete.

Setup For Projects 4-8

10/28: Setup for Project 7: Concurrency

We are creating a new GitHub Classroom repository for Concurrency to try to avoid creating a bunch of nasty merge conflicts for this assignment. The GitHub Classroom stencil will use the solution code for the previous assignments, and we recommend you keep it like this to avoid some potential nasty bugs that can be hard to catch.

Link to GitHub Classroom: <https://classroom.github.com/a/ibv6Ca3Y>

Projects 4-6:

For some projects (2, 3, 8, 9), you'll be repeating the above steps with a new GitHub Classroom link because those assignments will be run on a completely new stencil. But for projects (4, 5, 6, 7, 8), you will be building up your database implementation in the codebase stencil you pull for Assignment 3.

Update:

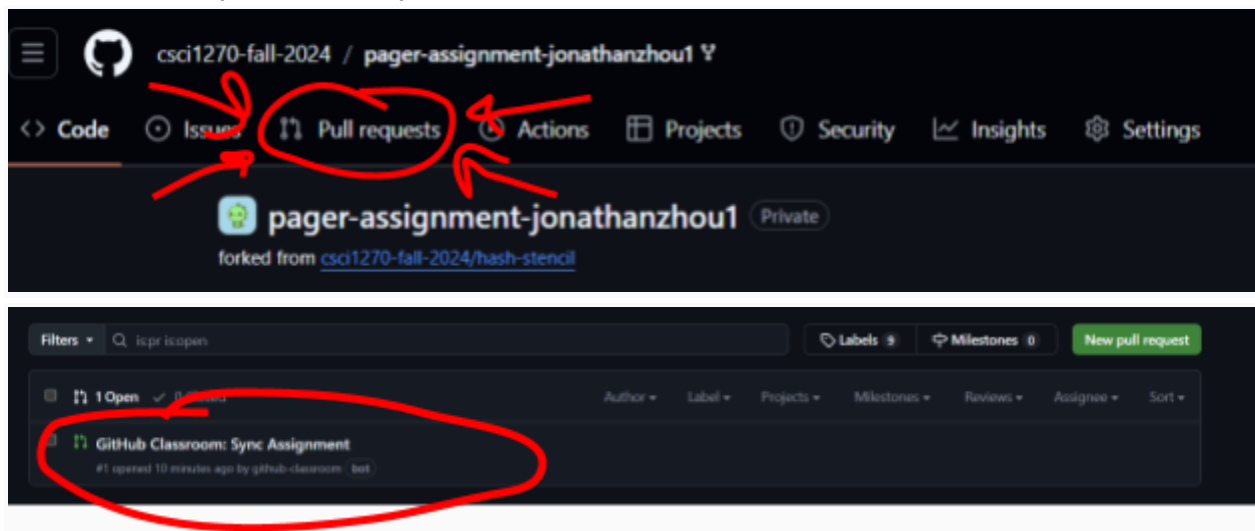
If GitHub classroom does not automatically create a pull request in your stencil, please follow the instructions listed in the [Fallback Github Guide](#).

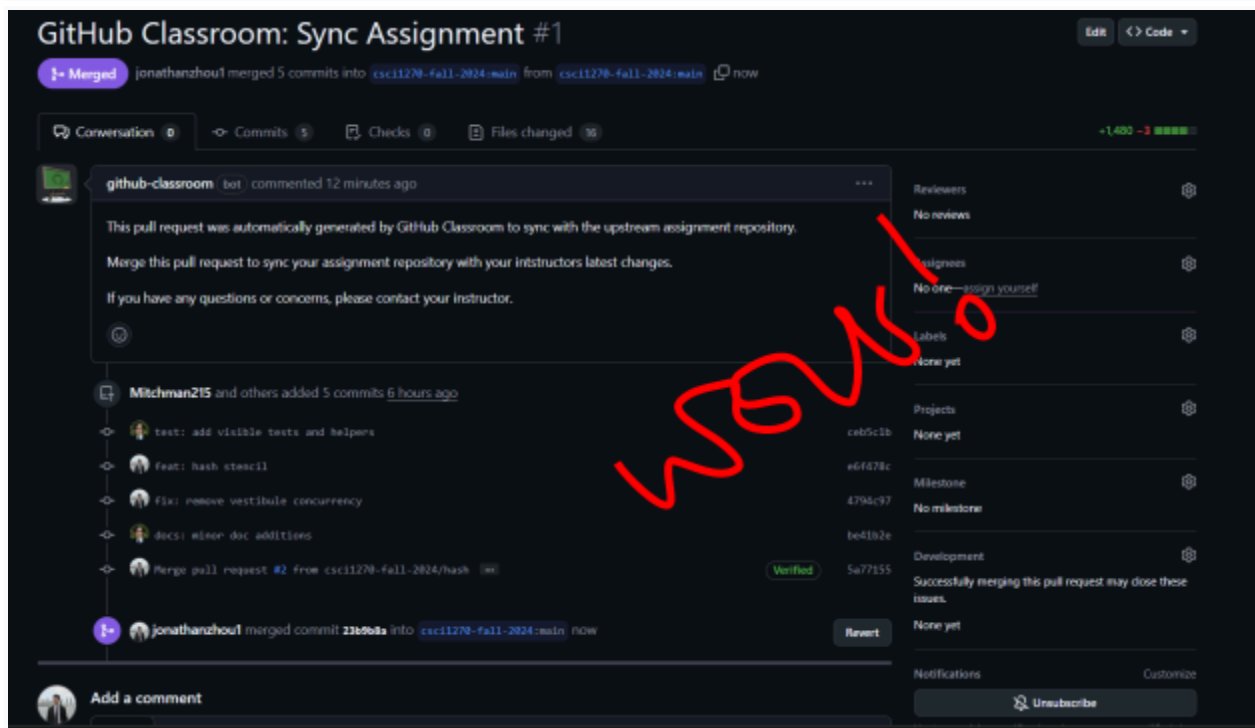
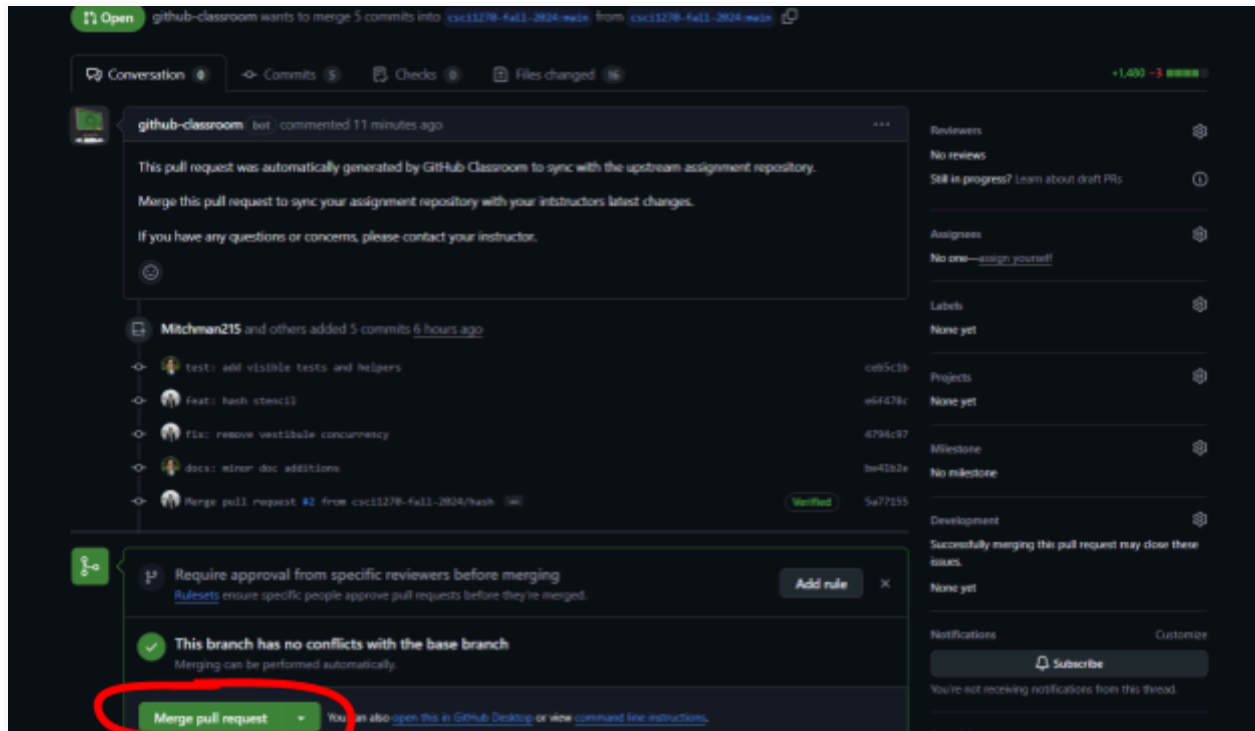
Pulling stencils for new projects

We will update the stencil for each assignment to contain the new files. Then, we will use GitHub Classroom to create a pull request in the repository you used for Assignment 3: Pager. After merging this pull request (and figuring out any merge conflicts as necessary), make sure to call 'git pull' on your local repository to catch up to the most recent version of the repository on GitHub, and then you are ready to start the next assignment!

Pull Requests

We expect you to be somewhat familiar with Git from the intro course and systems course you took as prerequisites for this course, but here is a quick picture guide about how to merge the pull request into your repository.





And you are done merging!!