## vscode -> git

* git status //See what’s in git.
* git add [filename] //Stage one untracked file.
* git add . //Stage all untracked files.
* git commit -a -m "Your commit message" //Commit all files.
* git commit -m "Add [filename] to the project" // Commit one file.
* git push (or git push origin main) //Push local git to remote (main branch).

## vscode -> google scripts app

* clasp push //Upload changes.
* clasp open //Open the Google Apps Script editor.

## google scripts app -> vscode

* clasp pull //Get changes.
* clasp status //Check differences between local files and Google Apps Script files.

## git -> vscode

* git pull origin main --rebase
* git push origin main

how can i tie my google apps script app, vscode, and github, so they're integrated?

**1. Project Setup**

* **Create a GitHub Repository:**
  + Go to GitHub and create a new repository (e.g., "flip-audit").
* **Initialize Local Project:**
  + Clone the repository to your local machine using Git:

Bash

git clone <https://github.com/PaulMStathis/flip-audit>

* **Install clasp:**
  + Install the clasp command-line tool globally using npm:

Bash

npm install -g @google/clasp

* **Initialize clasp:**
  + Navigate to your project directory in the terminal.
  + Initialize clasp:

Bash

clasp login

clasp project init

* + - **Project Name:** Choose a descriptive name.
    - **Project Type:** Select "Standalone" (for scripts not bound to a specific sheet).
  + This will create a clasp.json file in your project directory, which stores project-specific configuration.

**2. Develop in VS Code**

* **Install Extensions:**
  + Install the following VS Code extensions:
    - **Google Apps Script:** Provides syntax highlighting, code completion, and debugging support.
    - **ESLint:** (Optional) For linting and code quality checks.
    - **Prettier:** (Optional) For automatic code formatting.
* **Write Code:**
  + Open the project in VS Code.
  + Create or edit your script files (typically in Code.gs).
  + Utilize VS Code's features for efficient development:
    - Debugging: Set breakpoints, step through code, inspect variables. 1

[Debugging configurations for Python apps in Visual Studio Code](https://code.visualstudio.com/docs/python/debugging" \l ":~:text=For%20general%20debugging%20features%20such,dependent%2C%20review%20VS%20Code%20debugging." \t "_blank)

* + - Code Completion: Autocomplete functions, properties, and more.
    - Linting/Formatting: Maintain code quality and consistency.

**3. Integrate with GitHub**

* **Commit Changes:**
  + Stage and commit your changes regularly using Git commands:

Bash

git add .

git commit -m "Your commit message"

* **Push to GitHub:**
  + Push your commits to the remote repository:

Bash

git push origin main

* **Deploy to Apps Script:**
  + Use clasp to deploy your code to the Google Apps Script project:

Bash

clasp push

**4. Workflow**

* **Develop Locally:** Write and test your code in VS Code.
* **Commit and Push:** Commit changes to your local Git repository and push them to GitHub.
* **Deploy:** Use clasp to deploy your code to the Apps Script project.
* **Test and Iterate:** Test your script in the Google Apps Script environment and iterate on your development process.

**Key Considerations:**

* **.claspignore:** Create a .claspignore file to exclude files and directories from being deployed to Apps Script (e.g., test files, local configuration).
* **Version Control:** Use Git effectively for version control, branching, and collaboration.
* **Testing:** Implement unit tests to ensure code quality and prevent regressions.
* **Continuous Integration/Continuous Deployment (CI/CD):** For more advanced projects, consider setting up a CI/CD pipeline to automate the build, test, and deployment process.

By following these steps and leveraging the power of VS Code, Git, and GitHub, you can streamline your Google Apps Script development workflow, improve code quality, and collaborate more effectively.

how do i open my project in vscode?

**1. Using the File Menu**

* **Open VS Code.**
* Go to **File > Open Folder...** (or **File > Open...** in some versions).
* **Navigate** to the location of your project folder on your computer.
* **Select** the project folder and click **"Open"**.

**2. Using the Command Palette**

* **Open VS Code.**
* Press **Ctrl+Shift+P** (Windows/Linux) or **Cmd+Shift+P** (macOS) to open the Command Palette.
* Type **"Open Folder"** and select the **"Open Folder"** command.
* **Navigate** to your project folder and click **"Open"**.

**3. Using the Terminal**

* **Open your terminal or command prompt.**
* **Navigate** to the directory containing your project folder using the cd command (e.g., cd path/to/your/project).
* Type **code .** (or **code .** on macOS) and press Enter. This will open the current directory (your project folder) in VS Code.

**4. Drag and Drop**

* **Open VS Code.**
* **Drag and drop** the project folder from your file explorer directly into the VS Code window.

**Note:**

* If you have already opened a project in VS Code, you can open a new project in a new window by holding **Shift** while clicking **"File > Open Folder..."**.

By following these methods, you can easily open your project in VS Code and start working on your code.

I named my github project and my google apps script project "flip-audit". I created a new file "globals.gs" in my local flip-audit folder using vscode. How do I push it to github and google app script?

I'd be glad to help you push your "globals.gs" file to both GitHub and Google Apps Script for your "flip-audit" project. Here's a breakdown of the steps involved:

Pushing to GitHub:

1. Stage the File:
   * In VS Code, make sure "globals.gs" is visible in the Source Control view (usually on the left side).
   * Click the checkbox next to "globals.gs" to stage it for commit.
2. Commit the Changes:
   * Click on the "+" icon in the Source Control view or type git commit in the integrated terminal (usually at the bottom).
   * Enter a descriptive commit message (e.g., "Added globals.gs file") and click "Commit" or press Ctrl+Enter (Windows/Linux) or Cmd+Enter (macOS).
3. Push to Remote Repository:
   * Click on the "..." icon in the Source Control view or type git push origin main in the integrated terminal.
   * If you haven't set up remote origin yet, you'll need to do so using git remote add origin <your\_github\_repo\_url> (replace <your\_github\_repo\_url> with the actual URL of your remote repository).
   * Enter your GitHub credentials when prompted.

Pushing to Google Apps Script (using clasp):

1. Ensure clasp is Installed:
   * Open your terminal and run clasp --version to check if clasp is installed globally.
   * If not installed, install it using npm install -g @google/clasp.
2. Initialize clasp (if not already done):
   * Navigate to your local "flip-audit" project directory in the terminal.
   * Run clasp login to authenticate with your Google account.
   * Run clasp project init to initialize clasp for your project. Choose "Standalone" script type and a suitable project name.
3. Deploy the Script:
   * Make sure "globals.gs" is saved in your local project directory.
   * Run clasp push in the terminal. This will deploy your script to your Google Apps Script project.

Additional Considerations:

* clasp.json: The clasp project init command creates a clasp.json file in your project directory. This file stores configuration details for clasp. You can edit it to manage deployments, scripts, and dependencies.
* Conflicting File Names: If you have a file named "globals.gs" in your Google Apps Script project already, you'll need to rename your local file or the existing one in your script editor before deploying.
* Testing: After deploying, it's recommended to test your script in the Google Apps Script editor to ensure it functions as expected.

By following these steps, you'll successfully push your "globals.gs" file to both GitHub for version control and Google Apps Script for execution within your project.