RStudio and VS Code to GitHub.

Guide covering the prerequisites, authentication setup, IDE connections, and project linking:

**Phase 1: Prerequisites (Everyone)**

1. **Create a GitHub Account:**
   - Go to github.com.
   - Click "Sign up" and follow the instructions to create a free account. Choose a sensible username.
2. **Install Git:**
   - Neither RStudio nor VS Code *is* Git; they are interfaces *to* Git. Students need Git installed on their computers.
   - Go to git-scm.com/downloads.
   - Download the installer for their operating system (Windows, macOS, Linux).
   - Run the installer. For beginners, the default installation options are usually fine.
   - **Verification:** Open a terminal or command prompt (Terminal on Mac/Linux, Git Bash which comes with the Windows install, or the built-in terminal in VS Code/RStudio) and type git --version. If it returns a version number, Git is installed correctly.

**Phase 2: Authentication - Generating a GitHub Personal Access Token (PAT)**

This token acts like a password specifically for applications like RStudio and VS Code to interact with GitHub on the student's behalf.

1. **Log in to GitHub:** Go to github.com and log in.
2. **Navigate to Settings:** Click on your profile picture in the top-right corner, then select "Settings".
3. **Go to Developer Settings:** In the left-hand sidebar, scroll down and click "Developer settings".
4. **Select Personal Access Tokens:** In the left-hand sidebar under Developer settings, click "Personal access tokens". Choose "Tokens (classic)". (Fine-grained tokens offer more control but are more complex for beginners).
5. **Generate New Token:** Click the "Generate new token" button (you might need to re-enter your GitHub password). Select "Generate new token (classic)".
6. **Configure the Token:**
   - **Note:** Give the token a descriptive name, e.g., "RStudio-VSCode-Access" or "[YourComputerName]-Git-Access".

    - **Expiration:** Choose an expiration duration. For student use, 90 days or even a custom range covering the course duration might be appropriate. Never setting an expiration is *not* recommended for security.
    - **Select Scopes:** This is crucial. Scopes define what the token is allowed to do. For standard Git operations (cloning, pushing, pulling public and private repos), select the entire **repo** scope. This will automatically check all sub-options under repo. You might also want to select workflow if dealing with GitHub Actions, but repo is the minimum for basic interaction.
7. **Generate Token:** Scroll down and click the "Generate token" button.
8. **COPY THE TOKEN IMMEDIATELY:** GitHub will display the token **only once**. Copy it immediately and store it securely (e.g., in a password manager). **Treat this token like a password.** If you lose it, you'll have to generate a new one.

**Phase 3: Connecting RStudio to GitHub**

RStudio integrates well with Git and often uses helper programs to manage credentials, but explicitly configuring the PAT can be reliable.

1. **Ensure Git is Found by RStudio:**
    - Open RStudio.
    - Go to Tools -> Global Options.
    - Select the Git/SVN pane.
    - Make sure "Enable version control interface for RStudio projects" is checked.
    - Check if the "Git executable" path is correctly identified. If not, browse to find the git.exe (Windows) or git (Mac/Linux) file installed in Phase 1.
2. **Authentication Method (Using the usethis package - Recommended for R users):**
    - The usethis package provides helper functions that make Git/GitHub interaction within R easier.
    - Install and load it: In the RStudio console, run install.packages("usethis") (if not already installed) and then library(usethis).
    - **Store the PAT:** The best way is to store the PAT as an environment variable that RStudio can access. Run usethis::edit_r_environ() in the console. This opens the .Renviron file.
    - Add a new line to this file: GITHUB_PAT="YOUR_COPIED_PAT_HERE" (Paste the token you copied in Phase 2 between the quotes).
    - Save the .Renviron file and close it.
    - **IMPORTANT:** Restart RStudio completely for the environment variable to be loaded.
    - **Verify:** After restarting RStudio, run usethis::git_sitrep() in the console. It

should provide a situation report, hopefully indicating your GitHub PAT is found and showing your GitHub username.

3. **Alternative Authentication (Credential Prompt):**
    - Sometimes, when you perform your first push or pull operation to a private GitHub repository (or any HTTPS operation if no credential helper is configured), RStudio (via Git) will pop up a dialog asking for your GitHub username and password.
    - Enter your GitHub **username**.
    - Enter your **Personal Access Token (PAT)** in the **password** field.
    - Your operating system or Git Credential Manager might offer to save these credentials securely so you don't have to enter the PAT every time.

**Phase 4: Connecting VS Code to GitHub**

VS Code has excellent built-in Git support and integrates smoothly with GitHub.

1. **Ensure Git is Installed:** VS Code relies on the Git installation from Phase 1.
2. **Open the Source Control View:** Click the icon that looks like branching lines in the Activity Bar on the left (usually the third icon down). If you have a project open that isn't a Git repository yet, it might prompt you to "Initialize Repository".
3. **Authentication (Usually Automatic via OAuth or Prompt):**
    - **Recommended Method (OAuth):** The first time you try to clone a private repository, or push to any repository requiring authentication using HTTPS, VS Code will typically prompt you to sign in to GitHub. Click "Allow" or "Sign in with your browser". This will open a browser window asking you to authorize VS Code to access your GitHub account. This is generally the easiest and most secure method.
    - **Alternative (PAT Prompt):** If the OAuth flow doesn't trigger or you prefer, VS Code might prompt for your GitHub username and password (similar to RStudio). Enter your GitHub **username** and use your **PAT** as the **password**.
    - **GitHub Pull Requests and Issues Extension (Recommended):**
        - Go to the Extensions view (square icon in the left Activity Bar).
        - Search for "GitHub Pull Requests and Issues" (published by GitHub).
        - Install it.
        - You'll likely see a "Sign In" prompt appear either in the bottom status bar (person icon) or when you try to use the extension's features. Follow the prompts to authorize it with your GitHub account (again, usually via the browser OAuth flow). This extension provides much richer GitHub integration directly within VS Code (managing PRs, issues, etc.).

**Phase 5: Connecting a Project to a GitHub Repository**

There are three main scenarios:

**Scenario A: Starting a NEW Project that will be on GitHub**

1. **Create Remote Repository on GitHub:**
    - Go to github.com.
    - Click the "+" icon in the top-right corner and select "New repository".
    - Give it a name (e.g., my-r-project, my-vscode-analysis).
    - Add a description (optional but good practice).
    - Choose Public or Private.
    - **Crucially:** *Initialize this repository with a README file*. This makes cloning easier. You can also add a .gitignore (select R or Python/Node depending on the project) and a license.
    - Click "Create repository".
2. **Clone the Repository Locally:**
    - On the new repository's GitHub page, click the green "Code" button.
    - Copy the HTTPS URL (it looks like https://github.com/YourUsername/YourRepositoryName.git).
    - **In RStudio:**
        - Go to File -> New Project.
        - Select Version Control.
        - Select Git.
        - Paste the copied HTTPS URL into the "Repository URL" field.
        - The "Project directory name" will usually fill in automatically.
        - Choose a location on your computer "Create project as subdirectory of:" where you want the project folder to live.
        - Click "Create Project". RStudio will clone the repo, and the project will open, already linked to GitHub. You'll see a "Git" tab in the top-right pane.
    - **In VS Code:**
        - Open the Command Palette (Ctrl+Shift+P or Cmd+Shift+P).
        - Type Git: Clone and select the command.
        - Paste the copied HTTPS URL and press Enter.
        - Choose a location on your computer where you want the project folder to be saved.
        - Once cloned, VS Code will ask if you want to open the cloned repository. Click "Open". The project is now linked to GitHub, visible in the Source Control view.

**Scenario B: Putting an EXISTING Local Project onto GitHub**

1. **Create Remote Repository on GitHub:**
    - Go to github.com.
    - Click the "+" icon -> "New repository".
    - Give it the *same name* (or a relevant name) as your local project folder.
    - **Crucially:** *DO NOT* initialize it with a README, .gitignore, or license. You want an *empty* repository because your local project already has files.
    - Click "Create repository".
2. **Link Local Project to Remote Repository:**
    - On the new *empty* repository's GitHub page, copy the HTTPS URL provided under "...or push an existing repository from the command line".
    - **Open a Terminal/Shell:**
        - *In RStudio:* Go to Tools -> Shell... or use the "Terminal" tab next to the Console.
        - *In VS Code:* Go to Terminal -> New Terminal.
        - *Standalone:* Open Git Bash (Windows) or Terminal (Mac/Linux).
    - **Navigate to your project directory:** Use the cd command (e.g., cd path/to/your/project).
    - **Initialize Git (if not already):** If this project isn't already a Git repository, type git init and press Enter. You should see a message like "Initialized empty Git repository...".
    - **Add all files:** Type git add . and press Enter. (The dot means "all files in the current directory and subdirectories").
    - **Make initial commit:** Type git commit -m "Initial commit" and press Enter. Replace "Initial commit" with a meaningful message.
    - **Add the remote:** Type git remote add origin YOUR_COPIED_HTTPS_URL (paste the URL you copied from GitHub) and press Enter. This tells your local Git where "origin" (the remote repository) is.
    - **Push to GitHub:** Type git push -u origin main (or master if your default branch name is master) and press Enter.
        - The -u flag sets the upstream tracking reference, so future pushes/pulls can use git push and git pull simply.
        - You might be prompted for your GitHub username and PAT (as password) here if authentication hasn't been cached.
3. **Refresh:** Refresh the GitHub repository page in your browser. You should see your project files appear. Refresh the Git interface in RStudio/VS Code.

**Scenario C: Cloning an Existing GitHub Repository (e.g., collaborating,**

**downloading sample code)**

This is the same process as Step 2 in **Scenario A**.

**Basic Git Workflow within RStudio/VS Code (After Linking)**

1. **Make Changes:** Edit your code or files.
2. **Stage Changes:**
    - *RStudio:* Go to the "Git" tab. Check the boxes next to the files you want to include in the next commit under the "Staged" column.
    - *VS Code:* Go to the "Source Control" view. Changed files will appear under "Changes". Click the "+" icon next to each file (or the "+" next to "Changes") to stage them.
3. **Commit Changes:**
    - *RStudio:* Click the "Commit" button in the Git tab. A new window will open. Enter a descriptive commit message in the top pane. Click "Commit".
    - *VS Code:* Type a descriptive commit message in the box at the top of the Source Control view. Click the checkmark icon (or press Ctrl+Enter / Cmd+Enter).
4. **Push Changes to GitHub:**
    - *RStudio:* Click the "Push" button (up arrow) in the Git tab/commit window.
    - *VS Code:* Click the "Synchronize Changes" button in the status bar (usually shows arrows and numbers), or click the "..." menu in the Source Control view and select "Push".
5. **Pull Changes from GitHub (if collaborating or working on multiple computers):**
    - *RStudio:* Click the "Pull" button (down arrow) in the Git tab.
    - *VS Code:* Click the "Synchronize Changes" button or use the "..." menu -> "Pull".