GITHUB BASICS

A guide to get you started on GitHub
Serrano Lab

1. Terminology

1.1 Fork

- **What it means**: A fork is your own personal copy of someone else's repository. You can freely make changes to your forked copy without affecting the original repository.
- **When to fork**: You fork a repository if you want to develop your own version or if you are *not* a direct collaborator on the main repository but want to propose changes via Pull Requests.

1.2 Clone

- **What it means**: Cloning a repository means creating a local copy of it on your computer so you can view, edit, and manage the code offline.
- **When to clone**: You clone a repository if you have permission to push code directly to it (e.g., you're a collaborator) or if you have forked it and want to work on your own copy locally.

1.3 Commit

- **What it means**: A commit is a snapshot of your changes, recording what changed in the code and (ideally) why.
- **When to commit**: Commit often—whenever you have made a logical unit of change or have reached a stable state. Each commit should be accompanied by a clear message describing what changed.

1.4 Push

- **What it means**: "Push" sends your local commits to a remote repository (on GitHub).
- **When to push**: Push after you have made a series of commits (or at least one) that you want to share or back up on GitHub.

1.5 Pull

- **What it means**: "Pull" fetches any changes from the remote repository and merges them into your local copy.
- **When to pull**: Pull before you start working and frequently while you work, to ensure your local copy is up to date with other collaborators' changes.

1.6 Branch

- **What it means**: A branch is an isolated line of development. The default branch is usually called `main` (or historically `master`).
- **When to branch**: Use branches to develop features, fix bugs, or experiment without disturbing the main codebase.

1.7 Merge & Pull Requests (PRs)

- **Merge**: Combines changes from one branch to another.
- **Pull Request (PR)**: A GitHub-based mechanism to propose changes from one branch (or fork) to the main repository.

2. Setting up your environment

2.1 Install Git

1. If you don't already have Git installed, download and install Git.
2. Configure Git with your username and email (in the **terminal**, e.g., Command Prompt, Bash, or PowerShell):

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

   Make sure the email matches the one you use for GitHub.

2.2 RStudio and Git Integration

1. In RStudio, go to **Tools** > **Global Options** > **Git/SVN**.
2. Check that RStudio detects Git. If not, point RStudio to the path where Git is installed.

2.3 renv basics

- **What it is**: The **{renv}** R package helps you create isolated project environments with specific package versions. This ensures reproducibility across different machines and collaborators.

1. To install **{renv}** in a new R session:

   ```
   install.packages("renv")
   ```

2. (Optional) If you're creating a new project, initialize **renv** within that project:

   ```
   renv::init()
   ```

   This will create a `renv` folder and a `renv.lock` file that tracks packages used in the project.

3. If you are *cloning or forking an existing repository* that already uses **{renv}**, once you open the project in RStudio, run:

   ```
   renv::restore()
   ```

   to install the required packages matching the versions specified in `renv.lock`.

3. Working with a repository as a collaborator

If you are officially added as a collaborator on a GitHub repo (meaning you have permission to push changes directly), follow these steps:

3.1 Clone the repository

1. Open the repository on GitHub in your web browser.

2. Click the green "Code" button and copy the HTTPS URL (e.g., `https://github.com/username/repo.git`).

3. In **RStudio**, go to **File** > **New Project** > **Version Control** > **Git**.

4. Paste the repository URL, select a local directory to place the project, and click "Create Project".

   - *Alternatively*, clone via the **terminal**:

     ```
     git clone https://github.com/username/repo.git
     ```

   - Once cloned, open the `.Rproj` file in RStudio if the project includes one.

3.2 Set up renv (if the repo uses renv)

1. In the RStudio **Console**, run:

   ```
   renv::restore()
   ```

   This installs all the packages needed as specified by the project's `renv.lock` file.

3.3 Pull the latest changes

1. In RStudio, click the **Git** tab (usually in the upper-right pane or as a separate pane).

2. Click **Pull**.

   - *Alternatively*, in the **terminal**:

     ```
     git pull
     ```

     This ensures your local copy is in sync with the remote repository.

3.4 Create a branch or work on main

- **Create a new branch** (recommended for new features or fixes):
    1. In RStudio, under the **Git** tab, click on **Branches** > **New Branch**, and give it a name.
        - *Alternatively*, from the terminal:

```
git checkout -b my-feature-branch
```

- **Or** work on the `main` branch (less recommended if you're working in a team, but sometimes used for direct small fixes).

3.5 Make changes and commit

1. Modify your files in RStudio.
2. Check the **Git** pane in RStudio to see changed files.
3. Stage the changes by checking the boxes next to the files you want to commit.
4. Click **Commit**.
5. Enter a meaningful commit message.
6. Click **Commit** again to finalize.

3.6 Push to GitHub

1. Click the **Push** button in RStudio's Git pane.
    - *Alternatively*, from the terminal:

```
git push origin my-feature-branch
```

2. This will update the remote repository on GitHub with your commits.

3.7 Create a Pull Request (if your branch is ready to merge)

1. Go to the repository page on GitHub.
2. You'll see a banner prompting you to "Compare & pull request" for your recently pushed branch. Click it.
3. Fill in the PR details (title, description) and submit.
4. Your collaborators can review and merge your changes.

4. Working with a repository as a fork

If you are *not* an official collaborator or you prefer to keep your own copy for separate development, you can **fork** the repository:

4.1 Fork the repository on GitHub

1. Open the repository you want to fork in your web browser.
2. Click the **Fork** button (top-right corner of the page).
3. Choose your GitHub account as the destination.
4. GitHub creates a copy (fork) of the repository in your account.

4.2 Clone your fork

1. Go to **your** forked repository (e.g., `https://github.com/your-username/repo.git`).
2. Click the green "Code" button and copy the URL.
3. **Clone** in the same way as above (either through RStudio's "File > New Project > Version Control > Git" or via the terminal):

   ```
   git clone https://github.com/your-username/repo.git
   ```

4.3 Set up renv (if applicable)

- Inside your forked repository in RStudio, run:

  ```
  renv::restore()
  ```

  to synchronize the packages.

4.4 Syncing with the original repository

If the original repository (often called "upstream") has updates, you can pull those updates into your fork. First, set the upstream remote link:

```
# From inside your local forked repo
git remote add upstream https://github.com/original-owner/repo.git
```

Then, whenever the original repo changes, you can do:

```
git pull upstream main
```

(This fetches and merges the latest changes from the original repo into your local fork.)

4.5 Commit and push changes to your fork

- Commit and push works similarly:

  ```
  git add .
  git commit -m "Your commit message"
  ```

```
git push origin main
```

- Or through RStudio's Git pane with **Commit** and **Push**.

4.6 Open a Pull Request from your fork to the original repo (if you want to contribute back)

1. Go to your fork on GitHub.
2. Click "Contribute" > "Open pull request" or "Compare & pull request".
3. Choose your fork's branch as "head" and the original repo's `main` branch as "base".
4. Submit your Pull Request.

5. Good Practices

1. **Pull often**: Before starting new work, make sure you have the latest changes from your collaborators.
2. **Meaningful commit messages**: Write short but descriptive messages, e.g., *"Fix bug in data loading function."*
3. **Use branches**: Keep the main branch stable. Use feature branches for new work.
4. **RStudio Projects**: Always open the `.Rproj` file so that your environment is correctly set up.
5. **renv**: Keep `renv.lock` updated if you install or update packages:

   ```
   renv::snapshot()
   ```

   This ensures other collaborators know about and can install the new package versions.

Summary

- **Fork** if you want your own copy and don't have collaborator push rights.
- **Clone** either the main repo (if you're a collaborator) or your fork.
- **Commit** local changes frequently with good messages.
- **Push** them to GitHub to back up or share your work.
- **Pull** updates from GitHub to stay in sync with others' changes.
- **Use branches** for separate lines of development.

- **Use renv** to manage R package dependencies and ensure reproducibility.

With these steps in place, you'll be well prepared to contribute to a GitHub repository collaboratively or maintain your own forked project with best practices for version control and environment management.

Reference

Text prompt: "Please write stepwise clear instructions about how to start working on a GitHub repo..."

Response by ChatGPT: (2025, January 2). ChatGPT response to the prompt "Please write stepwise clear instructions about how to start working on a GitHub repo..." [Large language model output]. OpenAI. https://chat.openai.com/