Module 0: Git and GitHub

Arturo Torres Ortiz

2026-01-13

Learning objectives

By the end of this module, you will be able to:

1. Configure your local Git environment.
2. Clone your class repository.
3. Use the core Git commands (`add`, `commit`, `push`) to submit an assignment.

1. Why version control?

Git is a version control system that tracks changes in your files over time. It allows you to:

1. **Revert** to previous versions of your code if you break something.
2. **Collaborate** with others without overwriting their work.
3. **Document** *why* changes were made using commit messages.

GitHub is a cloud-hosting service for Git repositories that includes powerful tools for project management and collaboration.

2. Setting up the TRGN-515 classroom

Step 1: Join the TRGN-515 classroom

Before starting, you need to accept the assignment via GitHub Classroom using the following link:

`https://classroom.github.com/a/nMEEuL7F`

If you already have a github account, you can link it.

If you don't have a github account, you need to create one. Clicking the link will direct you to log in or create an account. Once you create your account, verify your account using the code sent to your email.

Once you accept the link, authorize GitHub classroom.

Finally, you have to select your username from the roster. Accept the assignment. Your repository will be automatically created.

You now have a repository on our github classroom. You can access it with the following link:

`https://github.com/trgn-515/USERNAME`.

To change the name of this repository: 1) click on your repository; 2) Settings > Repository name; 3) Change and click "Rename" twice.

> Important
>
> For security reasons, many of the github operations we will do require you to create a token. A token is like a temporary password.
> To create a token, go to your github account, click user (upper-right icon) > Settings > Developer Settings (all the way down on the left column) > Personal access tokens > Tokens (Classic) > Generate new token > Generate new token (classic)
> Write a name for the token, change the expiration to "No expiration" for convenience and select "repo" scopes. Click Generate token at the bottom of the page. For all GitHub operations, this token is your password. Copy this token. You can create as many tokens as you want.

Step 2: Connect git repository to the cluster

You only need do this once. It downloads the repository from GitHub to your local machine or, if the directory already exists in your local machine, it links it to the github remote repository.

Option 1: Clone (Do not do this one)

Use this option if you are starting fresh and you don't have an existing local repository. You already have a folder in the USC server so we don't need to do this.

```
# Navigate to where you want to keep your coursework
cd /scratch1/username

# Clone your specific repository (replace URL with your classroom repo link)
git clone https://github.com/trgn-515/USERNAME.git
```

```
# Enter the directory
cd USERNAME
```

Option 2: Initiate existing folder

Since you already have a folder for your user on the cluster, we don't need to "clone" one from github. However, you need to turn your normal folder into a git repository.

```
# 1. Enter your folder
cd /scratch1/username

# 2. Initialize Git
git init
git branch -M main

# 3. Link to GitHub (Replace URL with your repo link)
git remote add origin https://github.com/trgn-515/USERNAME.git

# 4. Pull any existing files (like the README) from GitHub
git pull origin main --allow-unrelated-histories
```

3. Setting up Git

Before you can use Git, you need to tell it who you are. This information will be attached to every "save" (commit) you make.

Open your terminal and run the following commands, replacing the name and email with your own:

```
# Set your name (this appears in the project history)
git config --global user.name "USERNAME"

# Set your email (must match your GitHub email)
git config --global user.email "your.email@usc.edu"

# Set the default branch name to 'main' (standard practice)
git config --global init.defaultBranch main
```

To verify your setup, run:

```
git config --list
```

4. The git workflow: clone, edit, commit, push

To submit an assignment in this class, you need to follow these four steps. These are the basic steps you will use in any project you work on in the future.

Step 1: Pull (Sync)

To pull in git means to syncronize with the code from a remote repository (usually the latest version of the code).

Always start your work session with this. It ensures your local computer has the latest changes (e.g: you fixed a typo on the GitHub website or a collaborator merged a change).

```
git pull origin main
```

Step 2: Edit code

Create files, write code, and save your work as you normally would. For this class, you will create a new folder for each assignment within your user repository.

Step 3: Stage and commit (Save)

Git has a two-step saving process.

- Add (Stage): Select which files you want to save.
- Commit: Actually save them with a message describing what you did.

```
# Check which files have changed
git status

# Add everything in the current folder
git add .

# or add a specific file
git add my_script.py

# Save the snapshot with a message
```

```
git commit -m "Completed exercise 1"
```

Step 4: Push (upload)

Your commits currently live only on your computer. To submit your assignment, you must upload them to GitHub.

```
git push origin main
```

5. Using .gitignore to avoid uploading some files

In bioinformatics, we work with massive files (.bam, .fastq, .vcf) that are too large for GitHub. If you accidentally try to commit a 2GB BAM file, your push will fail and your repository may break.

To prevent this, we use a file called `.gitignore`. Any filename matching a pattern in this file will be invisible to Git. Remember, a file with a '.' in front of its name is a hidden file. To list all files, including hidden ones, use `ls -a`.

Create a file named .gitignore in the root of your repository and add the following lines:

```
# Ignore large genomic data files
*.bam
*.sam
*.cram
*.fastq
*.fq
*.fastq.gz
*.fq.gz
*.vcf
*.vcf.gz

# Ignore system files
.DS_Store
```

Now, even if you type git add ., Git will safely skip your massive data files and only upload your scripts and markdown.

6. Advanced workflow: Branching & Merging

In professional software development, you rarely work directly on the main branch. Instead, you create a "branch" to experiment with changes safely.

- **Branch**: A parallel version of your repository. It lets you work on a new idea or code changes without messing up the working code in main.
- **Merge**: The process of taking the changes from your branch and combining them back into main.

Although you are not required to use branches for our assignments, understanding this concept is critical for collaboration.

How to use git branches:

```
# 1. Create a new branch and switch to it. Here we will create a new branch
 ↪  called "my-new-feature".
git checkout -b my-new-feature

# 2. Make edits, add, and commit as usual.
git add .
git commit -m "Testing a new idea"

# 3. Switch back to main when you are done
git checkout main

# 4. Merge the changes from your feature branch into main
git merge my-new-feature

# 5. Delete the branch if you no longer need it
git branch -d my-new-feature
```

Setting Up Assignment 1

For your first task, you will practice this workflow by setting up the directory structure for Assignment 1.

Task 1: Create the Directory Structure

Note

Inside your local user repository (eg: /scratch1/username/trg515, or whatever you called it), create a folder named 1_sequencing_techs.

```
mkdir -p 1_sequencing_techs
```

Task 2: Copy the Template

Note

Download the assignment_template.md file from:
`https://github.com/trgn-515/trgn-515.github.io/blob/main/modules/0_intro_git/assignment_templat`
Save it inside `1_sequencing_techs`
Rename the file to 1_sequencing_techs.md.

Task 3: Render and Submit

Note

Open 1_sequencing_techs.md in RStudio or VS Code.
Change the author field to your name.
Render the file to PDF.
Use Git to push both the qmd file and the generated .pdf to GitHub.

```
# 1. Check that Git sees your new files
git status

# 2. Add the new folder and files
git add modules/1_sequencing_techs/

# 3. Commit the changes
git commit -m "Setup assignment 1 structure"

# 4. Submit
git push origin main
```

Once you have pushed, go to your GitHub repository URL in your browser. If you see the modules folder and your PDF, you have successfully submitted Assignment 0!