**Git and GitHub Tutorial for Beginners**

Git and GitHub are two technologies that are very useful for bioinformatics analysis pipelines and all bioinformaticians should learn, irrespective of their field.

This tutorial will help you understand what Git and version control are, the basic Git commands you need to know, how you can use its features to boost your work efficiency, and how to extend these features using GitHub.

This guide is beginner friendly, as the examples will be very easy to understand. It will also be a generalized tutorial so anyone can follow along no matter what your favorite programming language is.

For our project, we'll create a snakemake workflow project written in a text (txt) file. You will see how we can use the features of Git to work on and create a final version of the snakemake workflow file.

**Prerequisites**

* In order to complete this tutorial, you'll need the following:
* A command line interface.
* A text editor of your choice (I will be using VS Code).
* A GitHub account.

**What is Git?**

Git is a version control system which lets you track changes you make to your files over time. With Git, you can revert to various states of your files (like a time traveling machine). You can also make a copy of your file, make changes to that copy, and then merge these changes to the original copy.

For example, you could be working on a workflow file and discover that you do not like the logic of the workflow. But at the same time, you might not want to start altering its components because it might get worse.

With Git, you can create an identical copy of that file and play around with the logic. When you are satisfied with your changes, you can merge the copy to the original file.

You are not limited to using Git just for source code files – you can also use it to keep track of text files or even images. This means that Git is not just for developers – anyone can find it helpful.

**How to install Git**

In order to use Git, you have to install it on your computer. [official website](https://git-scm.com/downloads). Follow the instruction to install for your operating system.

**How to configure Git**

I will assume that at this point you have installed Git. To verify this, you can run this command: “git –version”. This shows you the current version installed.

Text

Description automatically generated

The next thing you'll need to do is to set your username and email address. Git will use this information to identify who made specific changes to files.

To set your username, execute these commands:

git config --global user.name "<<YOUR\_USERNAME>>"

git config --global user.email "<<YOUR\_EMAIL>>"

**How to Create and Initialize a Project in Git**

Created a folder in your home folder called

mkdir Git GitHub

Using the command line, navigate to your new project's location. For me, I would run the following commands:

cd ~/

cd Git

*If you are new to the command line and are still learning how to use it to navigate around your PC, then I would suggest using Microsoft's Visual Studio Code. It is a code editor which has an inbuilt terminal for executing commands. You can download it [here](https://code.visualstudio.com/download).*

*After installing VS Code, open your project in the editor and open a new terminal for your project. This automatically points the terminal/command line to your project's path.*

To initialize your project, execute the command

git init

This will tell Git to get ready to start watching your files in the current folder for every change that occurs. It looks like this:

Text

Description automatically generated

The output is telling you that my repository (repo) has been initialized. It is considered empty because we have not told Git what files to track.

A repository is just another way to define a project being watched/tracked by Git.

**Git project files**

Created one file called first\_snkemake.smk. This is what the file looks like:

nano first\_snkemake.smk containing

rule summarize:

input:

#TODO

output:

#TODO

shell:

"wc -c {input} > {output}"

rule extract:

input:

#TODO

output:

#TODO

shell:

"sed 's/^[ ]\*//' {input} | cut -f1 -d ' ' > {output}"

**What is GitHub?**

GitHub is an online hosting service for Git repositories. With your project hosted on GitHub, you can access and download that project with a command on any computer connected to the internet. Then you can make changes and push the latest version back to GitHub.

GitHub facilities:

* Store your repo on their platform.
* Collaborate with other users with access to the repo from any location.

**How to push a repository to GitHub**

The following steps will help you understand the process more clearly.

**Step 1 – Create a GitHub account**

To be able to use GitHub, you will have to create your user account. Follow the instruction on the [website](https://github.com/) to create an account. <https://github.com/>

**Step 2 – Create a repository**

Click on the + symbol on the top right corner of the page then choose "New repository". Give your repo a name then scroll down and click on "Create repository".

Graphical user interface, website

Description automatically generated

**Step 3 – Add and commit file(s)**

Before we "add" and "commit" our files, you need to understand the stages of a file being tracked by Git.

**Committed state**

A file is in the committed state when all the changes made to the file have been saved in the local repo. Files in the committed stage are files ready to be pushed to the remote repo (on GitHub).

**Modified state**

A file in the modified state has some changes made to it but it's not yet saved. This means that the state of the file has been altered from its previous state in the committed state.

**Staged state**

A file in the staged state when it is ready to be committed. In this state, all necessary changes have been made so the next step is to move the file to the commit state.

**How to add files in Git**

When we first initialized our project, the file was not being tracked by Git. To start versioning control of a file, execute the command

git add [.|<<file name>>]

The dot means add all the files that exist in the repository. If you want to add a specific file, specify the file name

To check the state file file is in, execute the command

git status

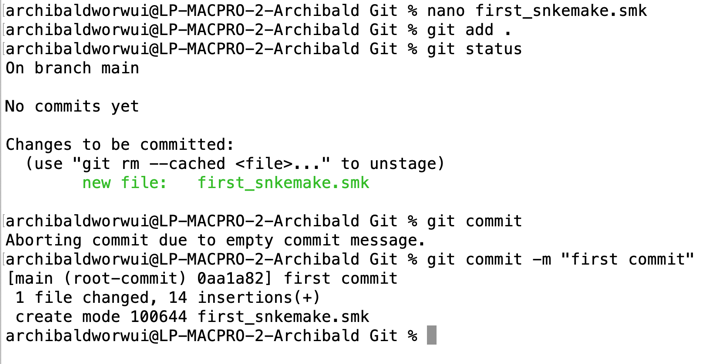
**How to commit files in Git**

The next state for a file after the staged state is the committed state. To commit file, execute the command

git commit -m "first commit"

The first part of the command git commit tells Git that all the files staged are ready to be committed so it is time to take a snapshot. The second part -m "first commit" is the commit message. -m is shorthand for message while the text inside the parenthesis is the commit message.

After executing this command, you should get a response similar to this:



Now our file is in the committed state.

**Step 4 – Push the repository to GitHub**

After you create the remote repo on GitHub.com, you are redirected to a page with instruction on how to create a repo locally or push an existing one.

We have a project already and a locall repo, so we will use the push commands. These are the commands:

git remote add origin <https://github.com/<<your> github remote repo>>

git branch -M main

git push -u origin main

The first command git remote add origin [https://github.com/<<your](https://github.com/%3c%3cyour) github remote repo>>  creates a connection between your local repo and the remote repo on Github. You won't usually get a response after executing this command but make sure you have an internet connection.

The second command git branch -M main changes your main branch's name to "main". The default branch might be created as "master", but "main" is the standard name for this repo now. There is usually no response here.

The last command git push -u origin main pushes your repo from your local device to GitHub.

We can at this stage make some changes to your snakemake file and push to your remote repo GitHub.

Recall that our file is now in the committed state. Let's make changes to the file and take note of the states.

After adding the new rule, run the git status command.

**git status**

After making changes to the file, it moved to the modified state – but it's not yet staged for commit, so you can't push it to GitHub yet. Git has not taken a final snapshot of this current state as it's only comparing the changes we have made now with the last snapshot.

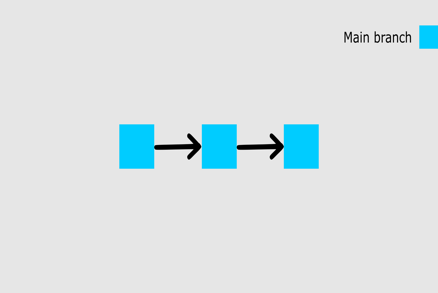
Now we are going to add (stage) this file and then commit and push it. This is the same as in the last section.

We first add the file by using git add . which adds all the files in the folder (one file in our case). Then we commit the file by running git commit -m "added new rule" followed by git push -u origin main

Those are the three steps to pushing your modified files to GitHub. You add, commit, and then push.

**How to Use Branches in Git**

With branches, you can create a copy of a file you would like to work on without messing up the original copy. You can either merge these changes to the original copy or just let the branch remain independent.



The image above shows our main branch with the last two commits (the first commit and the added new task commit).

At this point, I want to add more rules to the list but I am not yet sure whether I want them on my main list. So I will create a new branch called analysis2 to see what my list would look like with more tasks included.

To create a new branch, run the command:

git checkout -b analysis2

git branch

We can add more rules to the snakemake file and merge it with the new branch

Add four new rules. To merge the new state to the main branch, you have to first stage and commit this branch.

You should try doing it yourself. As a hint, add the file and then commit with a message (refer to the previous section for details showing you how to do that).

After committing your analysis2 branch, switch back to the main branch by running this command: git checkout main.

To check all the branches that exist in your repo by running the git branch command.

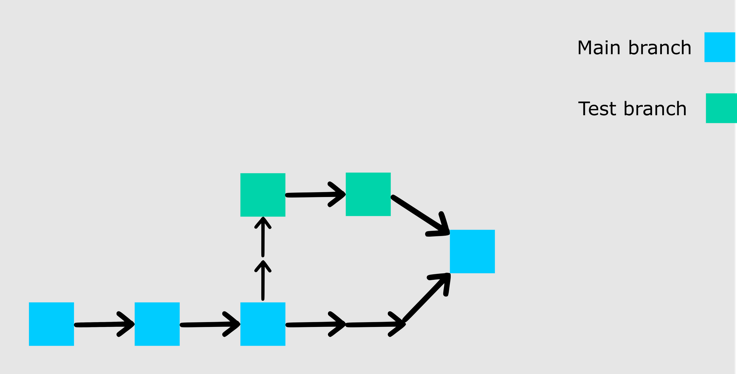
Now we can merge the changes we made in the test branch into the main branch by executing

git merge test

At this point, you will see all the changes made in the analysis2 branch reflected on the main branch.

git merge

Here is a visual representation of our repo:



When push this repo to GitHub, you'll see that the analysis2 branch will not be pushed. It will only remain in your local repo. If you would like to push your analysis2 branch, switch to the branch using

git checkout analysis2

git push -u origin analysis2

**How to Pull a Repository in Git**

To pull in Git means to clone a remote repository's current state into your local computer/repo. This comes in handy when you want to work on your repo from a different computer or when you are contributing to an open source project online.

To test this, don't worry about switching to a new computer. Just run cd ..to leave the current directory and go back one step. Create a new folder, go into the new folder

Go to GitHub, and on your repository's main page you should see a green button that says "Code". When you click on the button, you should see some options in a dropdown menu. Go on and copy the HTTPS URL.

Execute the command

git clone YOUR\_HTTPS\_URL

This command pulls the remote repository into your local computer in a folder

**Conclusion**

This covered the basic commands that'll help get you started using Git. We also started learning how to use GitHub.

**References**

* <https://www.freecodecamp.org/news/what-is-git-learn-git-version-control/>
* <https://www.freecodecamp.org/news/git-cheat-sheet/>
* <https://github.com>
* <https://www.tutorialspoint.com/git/git_clone_operation.htm>
* <https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners>