**Git Tutorial**

Git is a distributed version control system that runs locally on your computer. It allows you to track changes to your files over time, enabling collaboration with others and facilitating the management of your project's development history.

In this document, I will provide basic information on various Git commands to get you started initializing repositories, staging changes, committing revisions, and interacting with remote repositories.

**Table of contents**

[Getting started 1](#_Toc169875568)

[git init 1](#_Toc169875569)

[git add 2](#_Toc169875570)

[git commit 2](#_Toc169875571)

[git status 3](#_Toc169875572)

[git log 3](#_Toc169875573)

[git diff 3](#_Toc169875574)

[Creating new repository on GitHub 3](#_Toc169875575)

[Creating a new 3](#_Toc169875576)

[git remote add 5](#_Toc169875577)

[git push 5](#_Toc169875578)

# Getting started

## git init

* This command initializes a Git repository in the current directory.
* Before initializing the repository, ensure you are in the folder where you want the Git repository to be initialized. You can navigate to the desired directory using the **cd** command.

|  |
| --- |
| astridrr@BGS-STU22-058 ~ % cd /Users/astridrr/Desktop  astridrr@BGS-STU22-058 Desktop % |

* Once in the desired folder, you can initialize the Git repository.

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git init  Initialized empty Git repository in /Users/astridrr/Desktop/.git/ |

* **NOTE:** This repository is not visible! If you are not sure if the repository was created, here are two ways to verify:
  + On your keyboard press **Command + Shift + . (period)** on Mac to make the hidden files appear.

A screenshot of a computer

Description automatically generated

* + On the command line, use **ls -a** command (on Unix-like systems such as Linux and macOS) or **dir /a** (on Windows) to list all files including hidden ones.

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % ls -a  .  ..  .git GitHub\_tokens.txt Git\_tutorial.docx ReadME.txt |

## git add

* This command stages changes for the next commit by adding files in their current state to the Git repository.
* To add everything in the current folder to the Git repository, use:

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git add . |

* To add specific files in the current folder to the Git repository, use:

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git add Git\_tutorial.docx ReadME.txt |

* + You can specify multiple files separated by spaces.
* **NOTE:** Whenever you make changes to a file, you **need** to **git add [file name(s)]** or **git add .** **again!** If you don’t, the changes won’t be saved to the Git repository.

## git commit

* This command creates a new commit with the staged changes, committing the changes to the Git repository.
* There are different ways to commit changes. I’ll explain two of the most basic ones using the **-a** and **-m** options.
* To commit changes and add a message describing the commit, you can use the **git commit -m “[descriptive message]”** command. You should always add a message describing your commit; the more descriptive, the better. This will be extremely helpful when navigating through all of the file’s versions.

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git add Git\_tutorial.docx ReadME.txt  astridrr@BGS-STU22-058 Desktop % git commit -m "I’ve added the ‘Git\_tutorial.docx’ and the ‘ReadME.txt’ files." |

* **REMEMBER:** Use **git add** to add changes made to desired files **BEFORE** using **git commit**.
* If you want to commit all changes to tracked files quickly, you can use the **git commit -a** command. This allows you to automatically stage all changes to tracked files and commit them in one step, without explicitly using **git add** to stage each file individually. This might be useful if you made minor changes that don’t merit a full commit. However, it won't add new files (i.e., untracked files) to the commit; you still need to use git add for that purpose.

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git commit -a |

* Additionally, you can combine **-a** and **-m** options if you still want to describe your commit. This is also useful if you want to skip **git add** to stage your files before you commit. However, note that using **-a** may include changes you didn't intend to commit, so it's essential to review your changes carefully before using this option.

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git commit -a -m “I corrected the grammar on the ‘Git\_tutorial.docx’ file” |

## git status

* This command shows the status of the working directory and staging area. It lets you see which changes have been staged, which haven’t, and which files aren’t being tracked by Git.

## git log

* This command displays the commit history for the repository. It shows a list of all the commits in the repository, starting with the most recent.

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git log  commit bead0fd982b7a1e69580819bd3e566d2087e9c2c (HEAD -> main)  Author: AstridRamosRolon <astrid.ramos-rolon@pennmedicine.upenn.edu>  Date: Wed May 1 13:00:42 2024 -0400  I’ve added the ‘Git\_tutorial.docx’ and the ‘ReadME.txt’ files. |

## git diff

# Creating a new repository on GitHub

## Creating a new repository

* Make sure you have a GitHub account. If not you can sign up [here](https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home).
* In your dashboard, you should see an option to create a new repository.

A screenshot of a computer

Description automatically generated

* Alternatively, on your profile, you can navigate to the **Repositories** and select **New**.

A screenshot of a computer

Description automatically generated

* Enter a name for the repository. It does not have to match the name of your local folder.
* Decide if you want this repository to be private or public and optionally write a description of the repository.
* Finally, scroll down and click **Create Repository**.

A screenshot of a black screen

Description automatically generated

* Now, the empty repository on GitHub should have instructions on how to create or push a new repository from the command line. These instructions are important since they give you the URL for the repository.

A screenshot of a computer

Description automatically generated

*This image shows an example of the instructions provided by GitHub. These are important since they give you the URL for the repository.*

## git remote add

* This command creates a new connection to a remote repository (a repository created in GitHub).
* On the command line write **git remote add [alias] [url]**.
  + The **[alias]** is a shorthand or nickname for the remote repository. It allows you to reference the remote repository more easily in future commands. A commonly used alias is **origin**, which is the default name for the main remote repository when you clone a repository. is the address of the remote repository you want to add. The **[url]** is the address of the remote repository you want to add. This will be provided by GitHub when creating a new repository.

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git remote add origin https://github.com/AstridRamosRolon/Git\_Tutorial.git |

## git push

* This command updates the remote repository with the commits made to the associated local repository.
* Use the command **git push [alias] [branch]** to push your changes. The **[alias]** is the name of the remote repository (commonly origin), and **[branch]** is the name of the branch you want to push (commonly **main** or **master**).

|  |
| --- |
| astridrr@BGS-STU22-058 Desktop % git push -u origin |

* You will be prompted to write your GitHub username and password.

|  |
| --- |
| Username for 'https://github.com': AstridRamosRolon  Password for 'https://AstridRamosRolon@github.com': |

* Potential problems: