

Git and GitHub

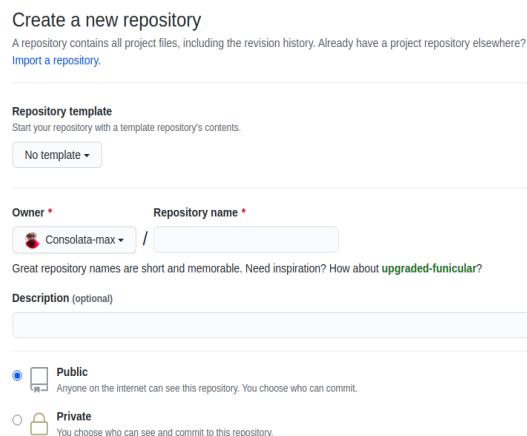
Git is a cloud-based software as a service popularly known as SAAS on which **GitHub**: The Graphical user interface runs.

To use Git and GitHub you have to:

- Install Git in your machine according to your operating system specification
- Create a GitHub account

After creating a GitHub account you'll need to create a **Repository** to store your project

Here are some highlights when creating a git repo:

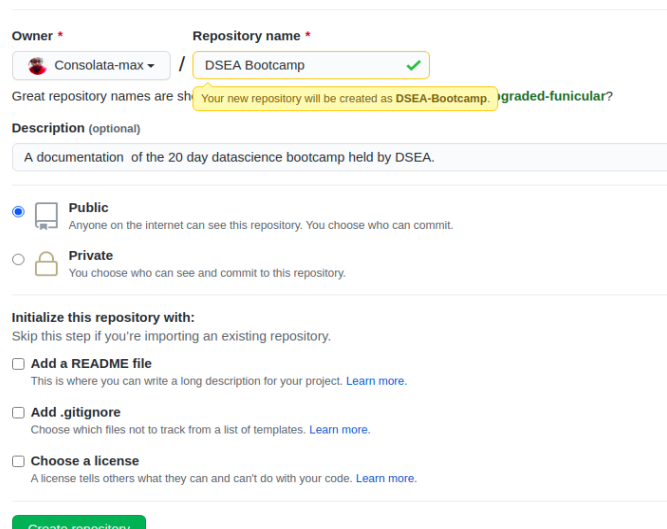


The screenshot shows the 'Create a new repository' page on GitHub. It includes a title 'Create a new repository', a subtitle 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.', a 'Repository template' section with a 'No template' dropdown, an 'Owner' dropdown set to 'Consolata-max', a 'Repository name' input field, a 'Description (optional)' text area, and radio buttons for 'Public' (selected) and 'Private' repository visibility.

One can either import a repo from another repo, create a repo from a repository template, or create a completely new repo.

After giving your repository a name you could choose whether to give it a description or not.

QUICK Tip: when naming a repo give it a name that relates to the content inside the repo.



This screenshot shows the same GitHub form as above, but with additional details. The 'Repository name' field is filled with 'DSEA Bootcamp' and has a green checkmark. A yellow tooltip box points to the name, stating 'Your new repository will be created as DSEA-Bootcamp.' Below the name field, a suggestion 'graded-funicular?' is visible. The 'Description (optional)' field is filled with 'A documentation of the 20 day datascience bootcamp held by DSEA.' The 'Public' option is selected. At the bottom, there are checkboxes for 'Add a README file', 'Add .gitignore', and 'Choose a license', each with a brief explanation and a 'Learn more' link. A green 'Create repository' button is at the very bottom.

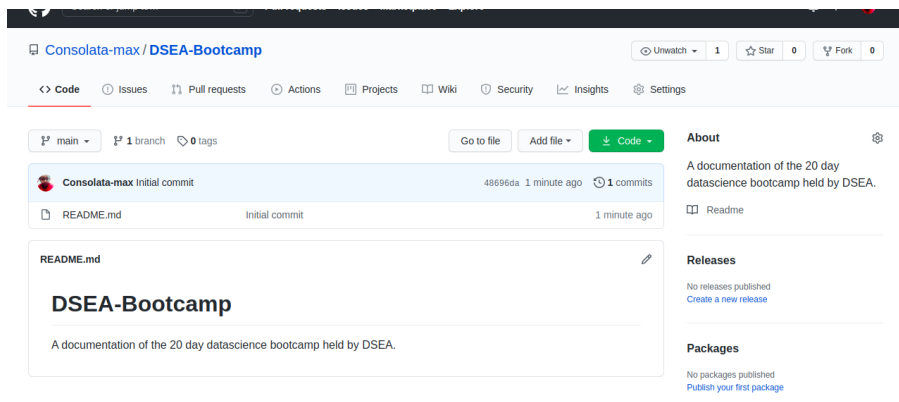
We have two types of repositories:

- **Public:** anyone on the internet can view contents of the project; you choose who can commit
- **Private :** You choose who can see or commit to the project.

You could also choose to either

- **Add a Readme file:** Here you can give a long description of your project
- **Add.gitignore :** add files which you want to ignore when committing a project.
- **Choose a license** from the list of Licenses provided.

Inside a Repository,



- In the **<>code** is where you'll view the committed code
- **Issues** are used to track todos, bugs, feature requests, and more. As issues are created, they'll appear here in a searchable and filterable list.
- **Pull Requests** help you collaborate on code with other people, one can pull your code work on it then push it but before committing it to the main project you'll be asked to review changes before merging.
- **Actions:** GitHub Actions help people Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. One can either create their own workflow or work with a workflow template.
- **Projects:** A project board on GitHub helps you to streamline and automate your workflow.
- **Wikis** provide a place in your repository to lay out the roadmap of your project, show the current status, and document software better, together.
- **Security:** Here one can setup Security policies, advisories, Dependabot alerts and code scanning alerts.
- **Insights:** they give you a visualization of how you've been using GitHub including number of commits, GitHub pull requests and issues.

- **Settings** : Here you can change the settings of the project you could also host your project if it's a site on GitHub pages using the GitHub pages setting.

In the code area they are **branches**

A Branch is an independent line of development which has a unique name .

The master branch now called main branch is the default branch in a repository and is generated on creation of the project.

To have a local version of the repository locally one could clone the repo either using ssh,https, Github CLI

SSH keys are unique keys used to identify a machine .

Commonly used Git commands and their use:

git config

Usage: git config --global user.name "[name]"

Usage: git config --global user.email "[email address]"

This command sets the author name and email address respectively to be used with your commits.

git init

Usage: git init [repository name]

This command is used to start a new repository.

git clone

Usage: git clone [url]

This command is used to obtain a repository from an existing URL.

git add

Usage: git add [file]

This command adds a file to the staging area.

Usage: git add *

This command adds one or more to the staging area.

git commit

Usage: git commit -m "[Type in the commit message]"

This command records or snapshots the file permanently in the version history.

Usage: git commit -a

This command commits any files you've added with the git add command and also commits any files you've changed since then.

git diff

Usage: git diff

This command shows the file differences which are not yet staged.

Usage: git diff --staged

This command shows the differences between the files in the staging area and the latest version present.

Usage: git diff [first branch] [second branch]

This command shows the differences between the two branches mentioned.

git reset

Usage: git reset [file]

This command unstages the file, but it preserves the file contents.

Usage: git reset [commit]

This command undoes all the commits after the specified commit and preserves the changes locally.

Usage: git reset --hard [commit] This command discards all history and goes back to the specified commit.

git status

Usage: git status

This command lists all the files that have to be committed.

git rm

Usage: git rm [file]

This command deletes the file from your working directory and stages the deletion.

git log

Usage: git log

This command is used to list the version history for the current branch.

Usage: git log --follow[file]

This command lists version history for a file, including the renaming of files also.

git show

Usage: git show [commit]

This command shows the metadata and content changes of the specified commit.

git tag

Usage: git tag [commitID]

This command is used to give tags to the specified commit.

git branch

Usage: git branch

This command lists all the local branches in the current repository.

Usage: git branch [branch name]

This command creates a new branch.

Usage: git branch -d [branch name]

This command deletes the feature branch.

git checkout

Usage: git checkout [branch name]

This command is used to switch from one branch to another.

Usage: git checkout -b [branch name]

This command creates a new branch and also switches to it.

git merge

Usage: git merge [branch name]

This command merges the specified branch's history into the current branch.

git remote

Usage: git remote add [variable name] [Remote Server Link]

This command is used to connect your local repository to the remote server.

git push

Usage: git push [variable name] master

This command sends the committed changes of master branch to your remote repository.

Usage: git push [variable name] [branch]

This command sends the branch commits to your remote repository.

Usage: git push -all [variable name]

This command pushes all branches to your remote repository.

Usage: git push [variable name] :[branch name]

This command deletes a branch on your remote repository.

git pull

Usage: git pull [Repository Link]

This command fetches and merges changes on the remote server to your working directory.

git stash

Usage: git stash save

This command temporarily stores all the modified tracked files.

Usage: git stash pop

This command restores the most recently stashed files.

Usage: git stash list

This command lists all stashed changesets.

Usage: git stash drop

This command discards the most recently stashed changeset.