

## Take A Step Tech Talk - Git Basics

What is **Git/Github**?

- **Git** is a distributed version-control system for tracking changes in the source code during software development.
- Git is designed for coordinating work between programmers and can be used to track changes in any set of files.
- **GitHub** is a Git repository hosting service

Okay, but what does **version-control system** mean?

Version Control System (VCS) is a software that helps software developers to work together and maintain a complete history of their work.

Functions of a VCS –

- Allows developers to work simultaneously.
- Does not allow overwriting each other's changes.
- Maintains a history of every version.

You can use github in three ways -

- Manually through the website UI.
- Through GitHub Desktop.
- Through the command line on the terminal.

Options #1 and #2 are straightforward so in this tutorial, let's focus on option #3 - learning the commands!

First, we must create an account on github - <https://github.com/join>

Second, download and install git - <https://git-scm.com/downloads>

Third, configure your github account on your terminal -

```
$ git config --global user.name <Your name here>
$ git config --global user.email <your_email@example.co>
```

### Let's create our first repository!

What is a **repository**?

A **repository** is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets – anything your project needs.

To create a new and empty git repository -

```
$ git init <new repository name>
```

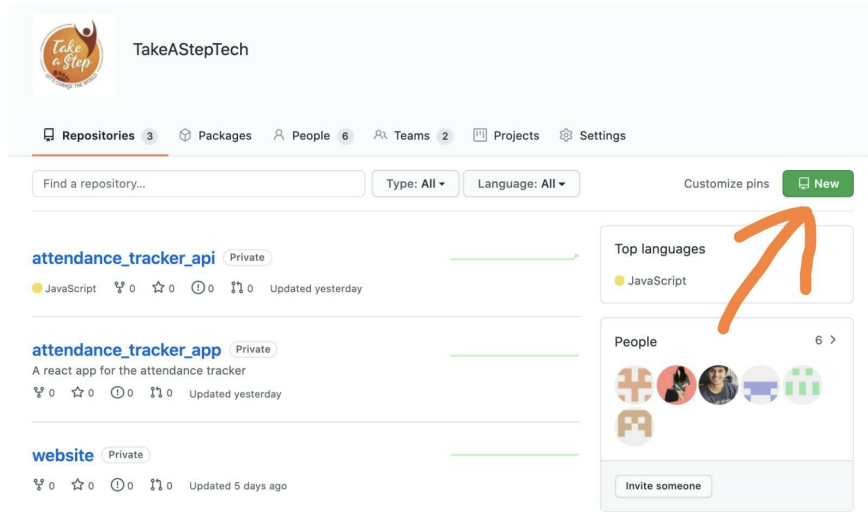
To convert an existing project folder into git repository -

```
$ git init
```

However, github's website UI might be a better place to create repositories - it's easier to manage access!

Here is how you do that -

## 1. Create a new repository



## 2. Enter your repository name, check whether you want this repository to be public or private

Note: Always initialize your repository with a readme otherwise you would be cloning an empty repository and that is buggy.

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

**Owner \*** TakeASStepTech / **Repository name \*** techtalks ✓

Great repository names are short and memorable. Need inspiration? How about **studious-funicular**?

**Description (optional)**

☒ **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

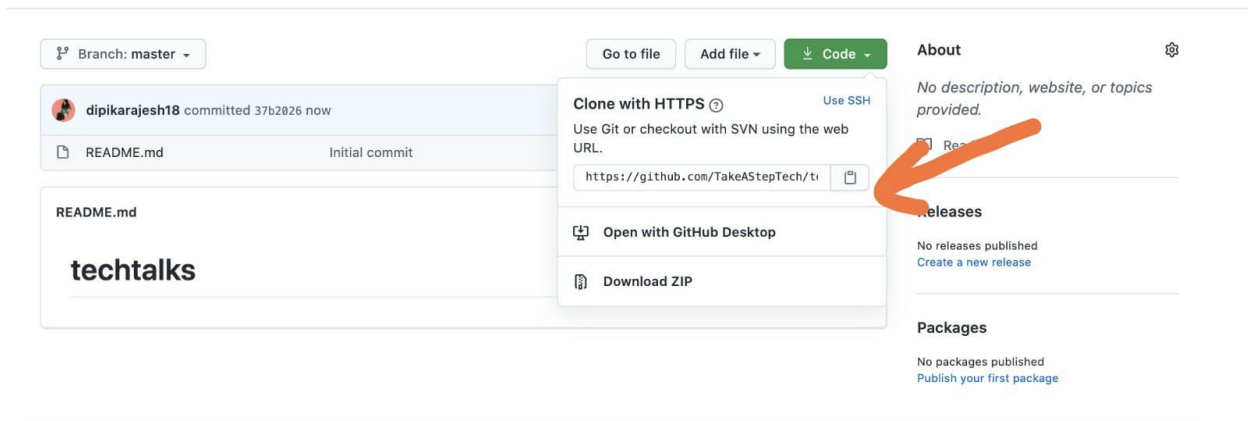
☒ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer.

Add .gitignore: None | Add a license: None ⓘ

**Create repository**

*How do I copy an existing repository into my system?*

You clone it! Find the URL for the repository here -



And go to your command line -

```
$ git clone /path/to/repository
```

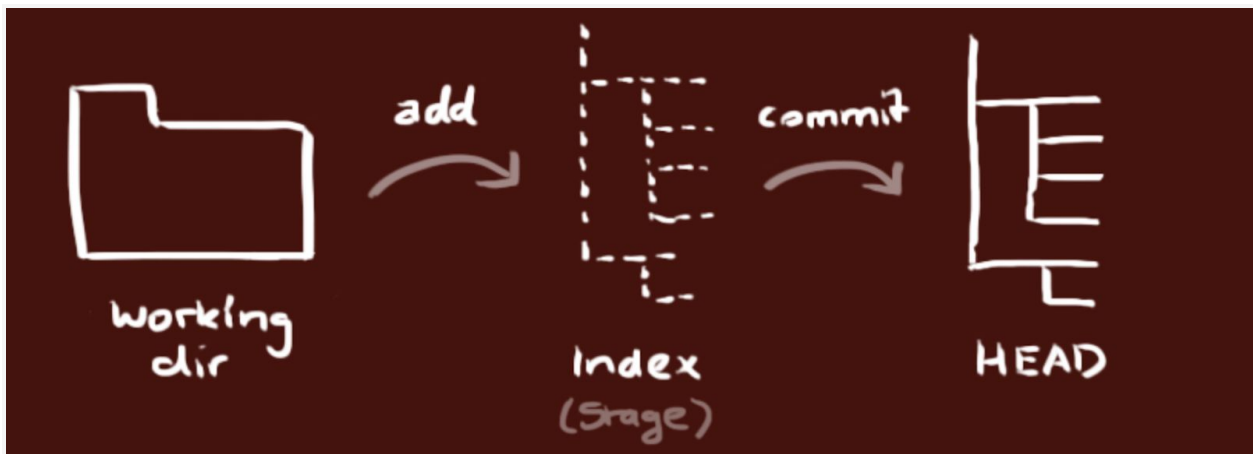
If you are using a remote system -

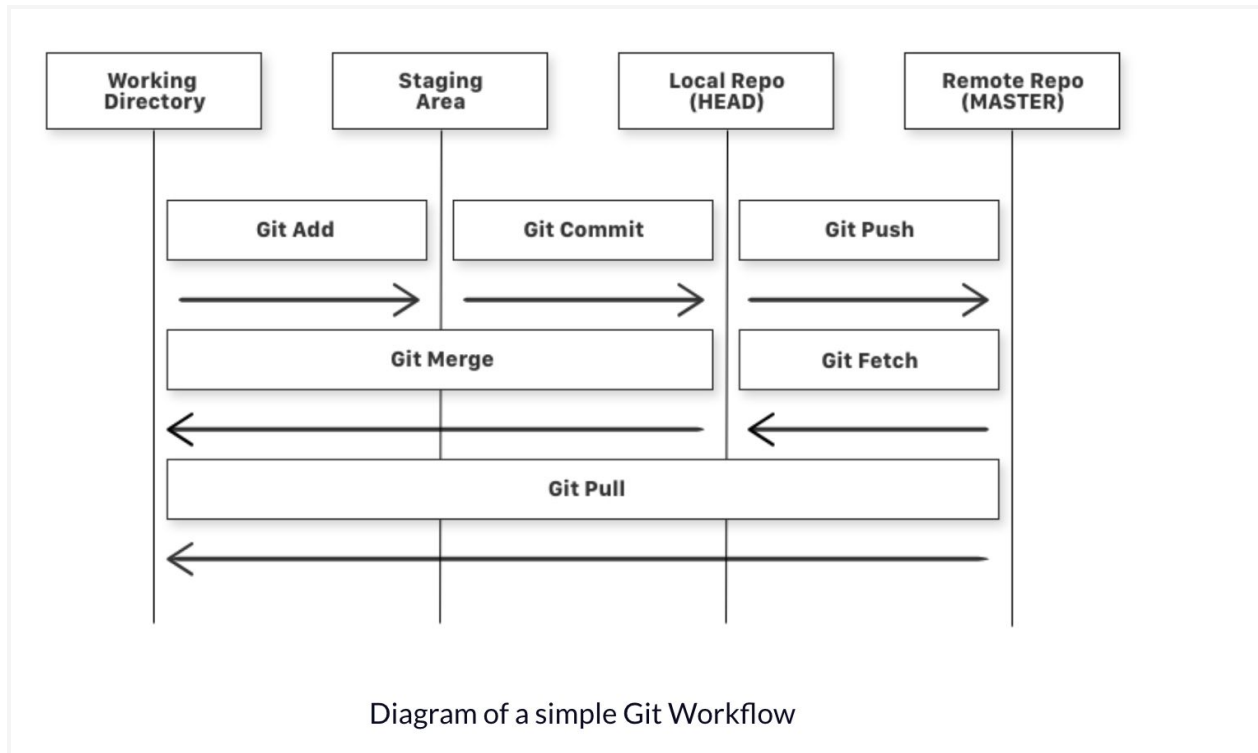
```
$ git clone username@host:/path/to/repository
```

### Git Workflow

Your local repository consists of three “trees” maintained by git -

- Working Directory - contains actual files
- Index - acts as the staging area
- Head - points to the last commit made





Once you've made changes to the code in your working directory, you want to *push* the changes into github for all the contributors to have the latest version.

First, we must add the proposed changes to the Index -

To add a particular file to the index (and not all the new files or changed files) -

```
$ git add <filename>
```

To add all the files in the repository to the Index -

```
$ git add .
```

To actually *commit* these changes,

```
$ git commit -m "<commit message>"
```

Now the file is committed to the head but not in the remote repository on github yet.

To send the changes to the remote repository after pointing the head in the local copy -

To push to the master -

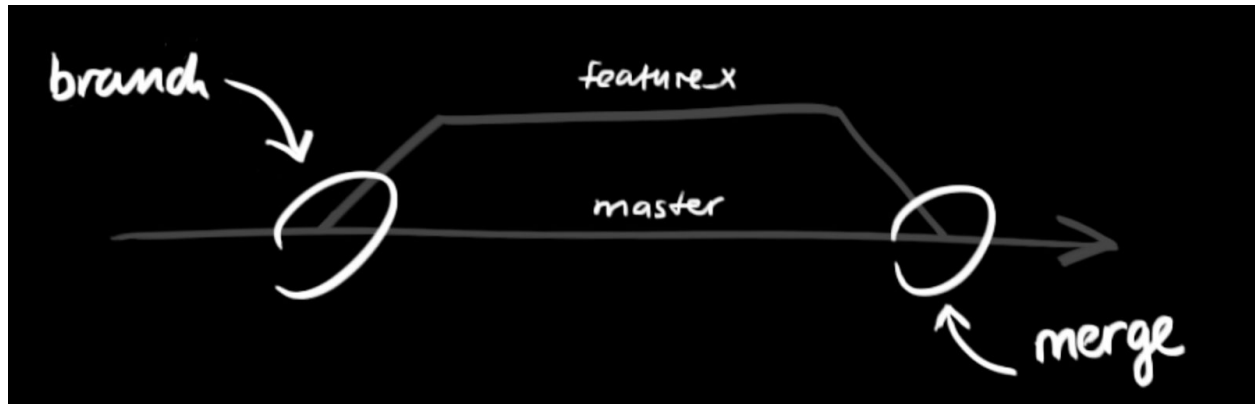
```
$ git push origin master
```

To push to another branch -

```
$ git push origin <branch_name>
```

*Wait, what does another branch mean? What is branching?*

Branches are used to develop features isolated from each other. The *master* branch is the default branch when you create a repository. We use other branches for development and merge them back to the master branch upon completion.



Creating a new branch and switching to it -

```
$ git checkout -b <branch_name>
```

Viewing all the branches in your repository -

```
$ git branch
```

Switching back to the master branch

```
$ git checkout master
```

A branch is not available to others unless the branch is pushed to the remote repository

```
$ git push origin <branch_name>
```

First, an important thing to note is that you *always* want your current branch updated to the latest commit to avoid conflict. To update your repository to the newest commit (assuming you're in the right branch) -

To get the files from the remote repository to the local repository but not into the working directory

```
$ git fetch
```

To get the files from the local repository into the working directory -

```
$ git merge
```

To get the files from the remote repository directly into the working directory -

```
$ git pull
```

*How do I combine the branch to the overall repository?*

Git tries to auto-merge changes. However, this is not always possible and could result in conflicts! You are responsible to merge those conflicts manually by editing the files shown by git! After changing, you need to mark them as merged with -

```
$ git add <filename>
```


Before merging the changes, you can also preview them by using -  
`$ git diff <source_branch> <target_branch>`


*Alert the repository owner that you want to make some changes to their code!*

You can do this using a pull request! It allows the owner of the repository to review the code and make sure it looks good before putting your changes on the master branch!

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



 base: **master** ... compare: **mnelson** ✓ **Able to merge.** These branches can be automatically merged.



Added my name file

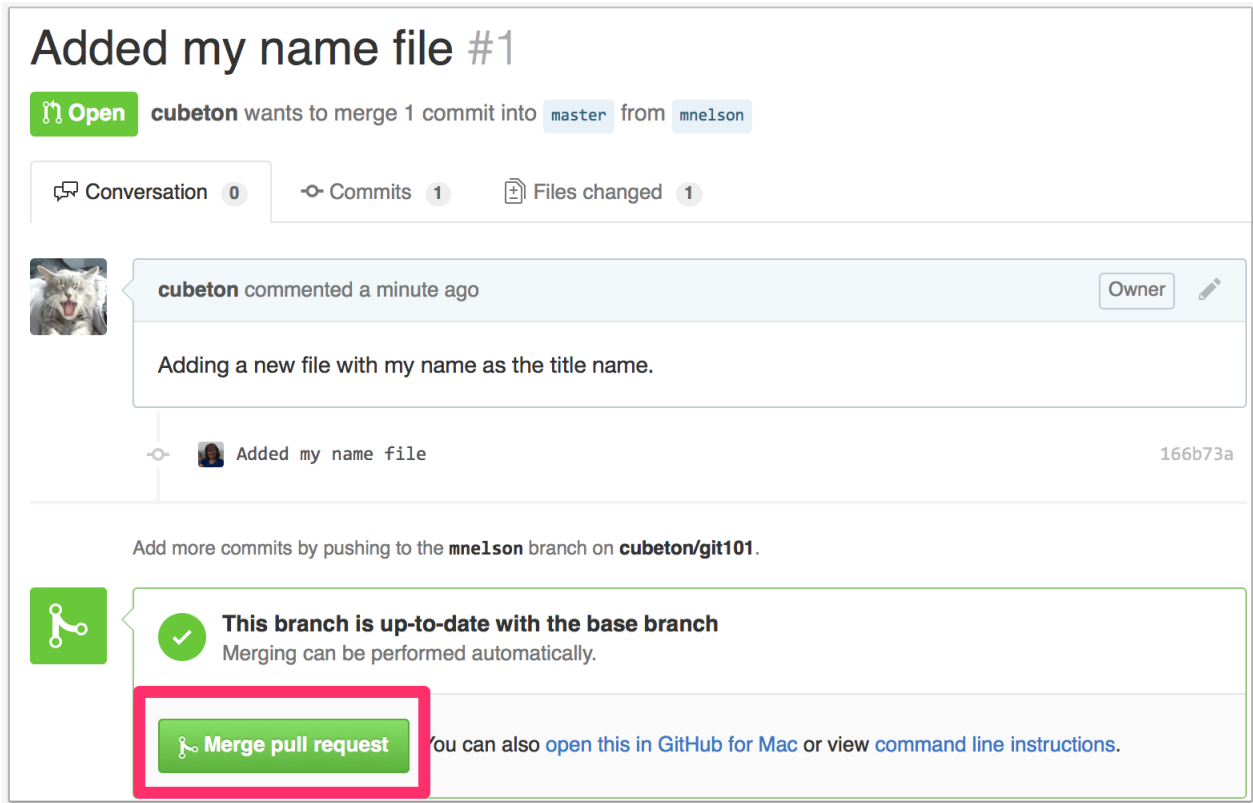
Write

Preview

 Markdown supported  Edit in fullscreen

Leave a comment

Attach images by dragging & dropping or [selecting them](#).



You might see a big green button at the bottom that says 'Merge pull request'. Clicking this means you will merge your changes into the master branch.

Note that this button won't always be green. In some cases, it will be grey - which means you're facing a **merge conflict**. This is when there is a change in one file that conflicts with a change in another file and git cannot figure out which version to use. You'll manually have to go in and tell git which version to use.

Once the pull request is merged, it is recommended to delete the branch!

Additional commands -

To receive basic information about the current status of your Git repository -

```
$ git status
```

To view the whole history of changes made -

```
$ git log
```

To document the given state of your project, i.e. *tagging* different builds, versions, releases -

```
$ git tag <custom_tag>
```

To temporarily stash your work (a temporary version of adding, committing and pushing) -

```
$ git stash
```

To view your stashed work -

```
$ git stash list
```

To reset your files (thereby cancelling all the currently staged changes) -

```
$ git reset
```

To revert some commits to a particular commit ID -

```
$ git revert <commit_id>
```

Links I used to make this tutorial -

<https://rogerdudler.github.io/git-guide/>

<https://www.atlassian.com/git>

<https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/>

<https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners>

<https://areknawo.com/git-basics-the-only-introduction-you-will-ever-need/>