**What is Git?**

Git is a free and open-source distributed version control system or protocol. it’s a system that allows you to record changes to files over time and allows you to view changes and specific versions of those files later on. Git is largely helpful in projects which involve a group of developers working on a single project. Git helps in managing the code changes and errors for projects with a large codebase. It makes it easy to stage changes and revert back to a specific version easily. It has is now being used to automate the process of deployment for an application.

**What is GitHub?**

GitHub is a startup which was recently bought by Microsoft. It is a user-friendly client application for git. It can be used to view and edit code from the website, raise issues for any non-functional code, etc. Github is the leading git client amongst the many such clients such as GitLab, Bitbucket, etc. GitHub now supports various cloud provides like Azure, AWS, GCP to automate the deployment pipelines.

**TLDR**

Installing git  
Signing up to GitHub  
Setup on a machine  
Creating a repo on GitHub  
creating the files  
Initializing an empty git repository locally  
Adding a remote  
Checking status of files  
Creating a .gitignore  
Adding files to the local git repository  
Committing a change  
Creating a new branch  
Pushing the code  
Merging to the master branch  
Cloning a repository  
Forking a repository

**Let’s get started…**

**Working with Git and GitHub**

**Installing git**

Click on [this](https://git-scm.com/book/en/v2/Getting-Started-Installing-Git)URL to download and install Git for Mac and Linux  
Click on [this](https://git-scm.com/download/win)URL to download and install Git for Windows.

**Signing up to GitHub**

Now that you have git install on your computer. Let’s create an account on Github.

Click on [this](http://github.com/)URL to sign up to GitHub.

**Setup on a machine**

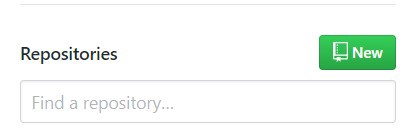
Now that we have installed Git and created an account on GitHub. Lets set up our machine. Open up a terminal or command prompt and type the following command.

git config --global user.name "GitHub User Name"git config --global user.email "email"

**Creating a repo on GitHub**

Go to [this](https://github.com/)URL if you are logged in you will be presented with a dashboard. To the left, under the repositories section, you will notice an icon “New”. Click on it to create a new repository or repo in short. Repos are where all your code is saved.

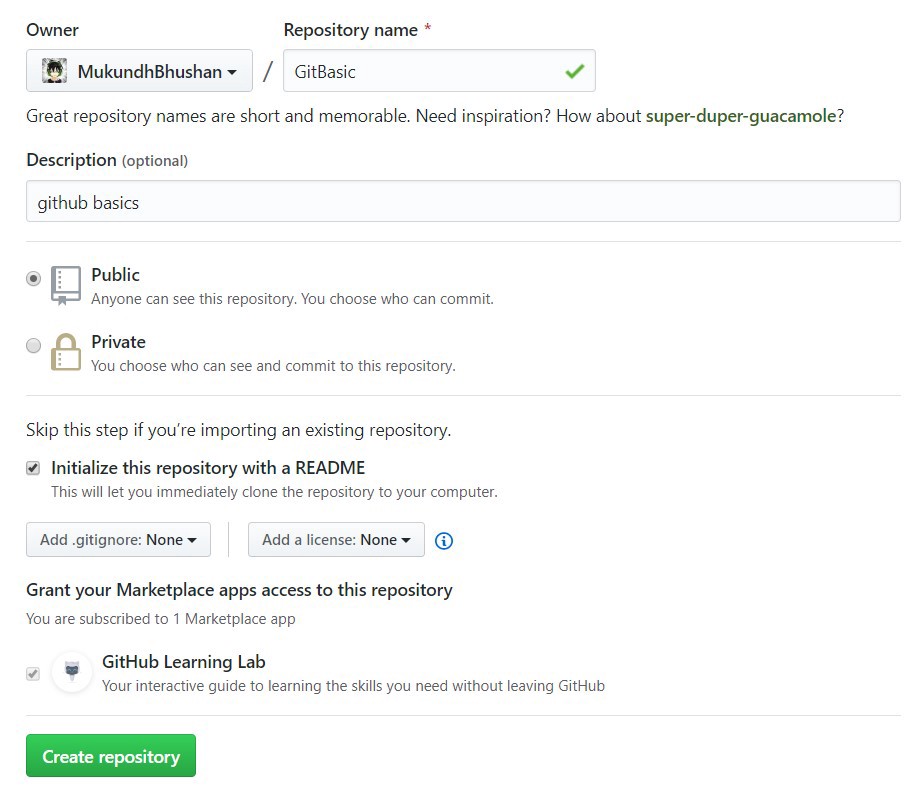
Image for post



**Fill in the form with the following details**  
**Repository name:** Name of the repository.  
**Public:**A repository which can be viewed and edited by all.  
**Private:** Only you and the contributors can see and edit it.  
**Initialize this repository with a README:** Initializes and README.md file with the project name as its heading and the description you have given previously.

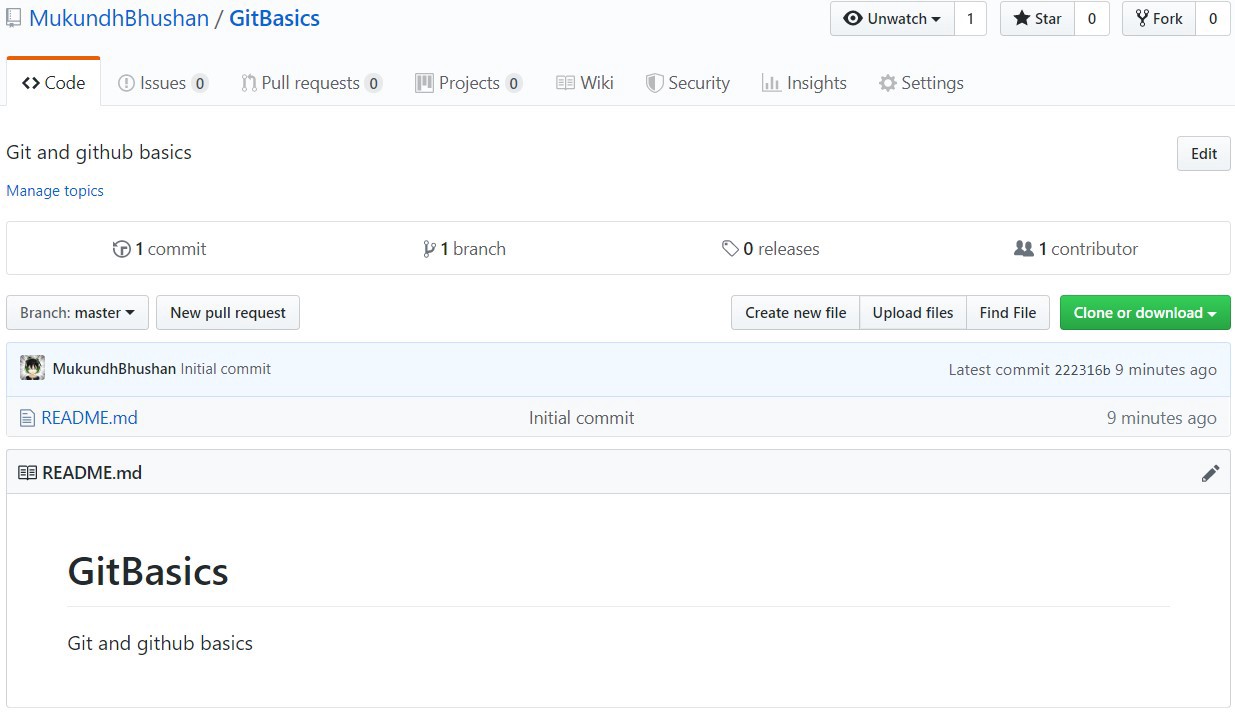
Your form must look something like this

Image for post



Once the repository is created it would look something like this

Image for post



**Creating the files**

Create a new folder in your project directory and cd into it.

mkdir <folder name>  
cd <folder name>

I will be using a simple nodejs application as the code files for this article. Add any file you like into this

**Initializing an empty git repository locally**

Now we need to initialize an empty git repository on our local machine. Git adds all the file and their corresponding changes to a local repository which is later pushed in a client such as GitHub. Once you are in your projects root folder, type the following command.

git init

This will initiate an empty git repository and creates a hidden “.git” file.

Image for post

Image for post

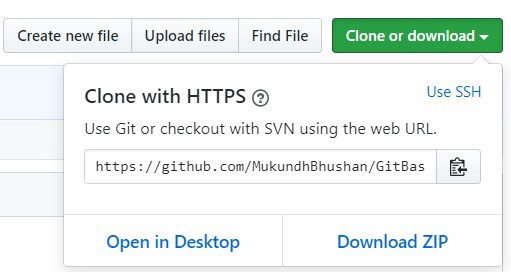
**Adding a remote**

Remotes are used to push data in the local git repository to clients like GitHub, Bitbucket and also cloud providers like Azure, AWS while developing apps.

Now let’s add the remote to our repository

Navigate to your repository you have created on Github and copy the link under “Clone or download” button.

Image for post



Now type the following command

git remote add <alias> <remote URL>

An alias is the shorthand form which refers to the actual URL. I named mine “origin”

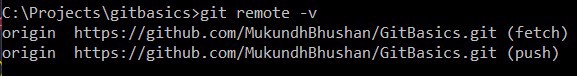
Image for post

Image for post

You can add as many remotes as you feel like but make sure the alias names are different. Here is how you can list all the remotes.

git remote -v

Image for post

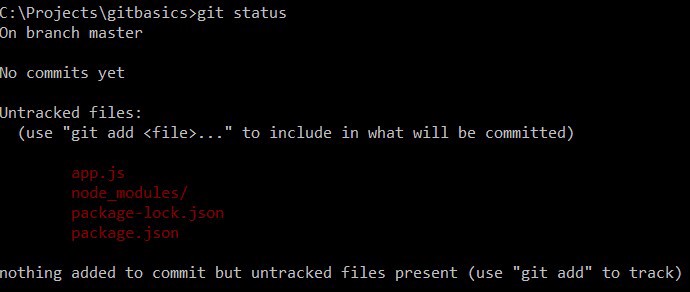


**Checking status of files**

This helps us understands all the files which have been modified, removed, changed location, etc. Type the following command to check the status of your files

git status

Image for post



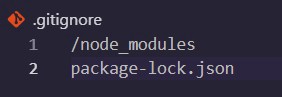
Red indicates unstaged changes in your file system.

**Creating a .gitignore**

If you have run the “git status” command you might have noticed that there are a couple of files or folder which you don’t want to add. This is where the “.gitignore” file comes into the picture. This file helps you to mention all the files and folders you don’t want git to add.

The file format is simple just state the folder names in the “.gitgnore” file.

Image for post



**Adding files to the local git repository**

Now that we have all the necessary files that we want to add and the others not included. Let’s now add the with the following command.

git add .

The “.” tells git to add all the files in the directory

Image for post

Image for post

If we now type “git status” we can notice the files are now green in color.

**Adding a single file**

Let us add a new file to the project. I added a README.md to my project.  
Type the following command.

git add <file name>

Image for post

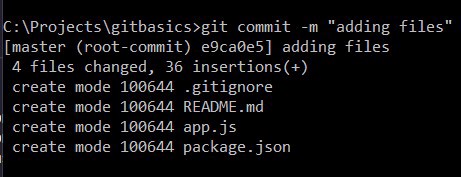
Image for post

**Committing a change**

In git, every change in the existing file system or file needs to be followed by a commit message for git to consider it as a unique change in the file system. Git generates a hash for all each change to compare it with the existing one. This is the command to add a commit.

git commit -m "<message>"

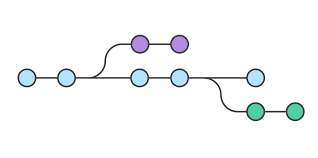
Image for post



**Creating a new branch**

By default, GitHub creates a master branch for your repository. The master branch is where all the final code is pushed and could be used in production. Git does not allow you to push any code to the master branch directly, instead encourages you to push it to another branch and merge it with the master branch later. Branches are widely used while you are working with a team of developers on a project. A branch is a unique set of code changes with a unique name they can have their own commits and pushes. This illustration could help you better understand a branch.

Image for post



To create a branch type the following command

git checkout -b <branch name>

Git will create a branch and automatically switch you to it.

Image for post

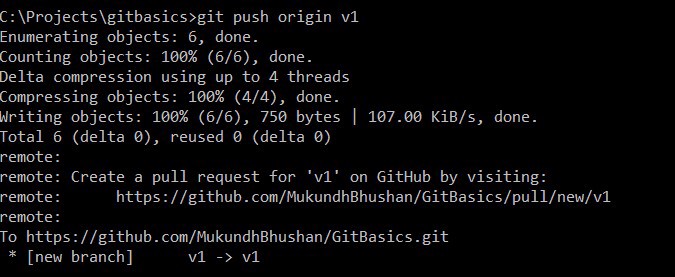
Image for post

**Pushing the code**

Now let’s push the code to GitHub with the following command

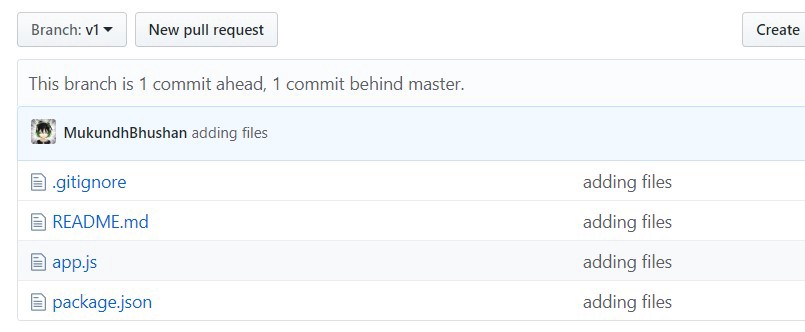
git push <remote alias> <branch name>

Image for post



Reload the GitHub page to view your code.

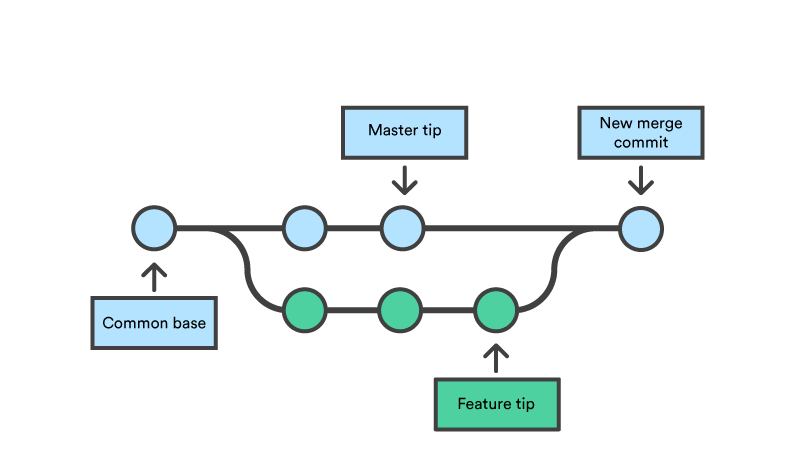
Image for post



**Merging to the master branch**

Merging branches is the way with which code needs to be pushed into the master branch.

Image for post



**Pull request**Pull request is the first step before merging branches. Pull request tells us if our current code is up to date with all the changes in a specific branch. The command for it is.

git pull <remote alias> master  
git pull <remote alias> <branch name>

If you are getting an error of unrelated run the above commands with “ — allow-unrelated-histories”. This forces the pull and merge.

git pull <remote alias> <branch name> --allow-unrelated-histories

It is necessary for us to be up to date with all the braches before we merge the branches.

You will notice the code now is changed if you had an older version of a file. The updated files can be verified with “git status”. We need to add this too. using the previous commands.

git add .  
git commit -m "<message>"  
git push <remote alias> <branch name>

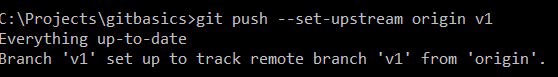
If you are facing a merge conflict type the following command

git checkout <branch name> -- <filename>

**Setting branch as upstream**To push data to master we need to make our current working branch into an upstream branch. Type the following command

git push --set-upstream origin <branch>

Image for post

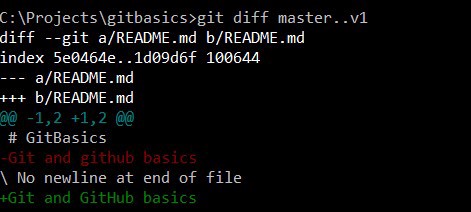


Now if we change the content is a file with the GitHub code editor. I will be changing the contents in the README.md file. Now will pull the code again we notice that the message is different our code has been updated with the latest content.

If you want to know what will change once you have merged the branches use the command

git diff master..<branch>

Image for post



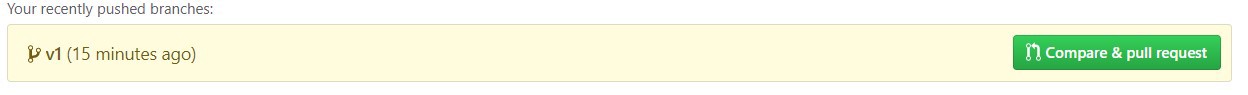
Now that our code is up to date lets merge it with the master. We can do that with the command.

git merge <branch name>

**Now we need to accept the changes in GitHub.**

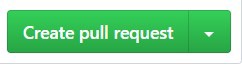
Navigate to your repository on GitHub now click on the button “Compare & pull request”.

Image for post



In the new window click on the “Create pull request” to accept the changes.

Image for post



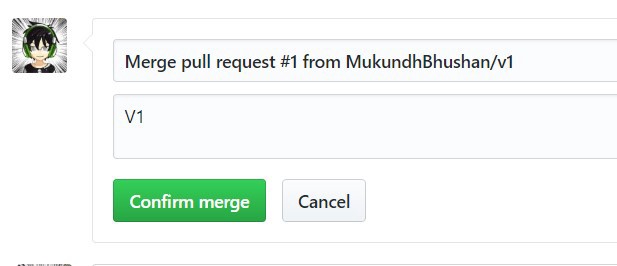
We can merge the branch by clicking on the “Merge pull request”.

Image for post



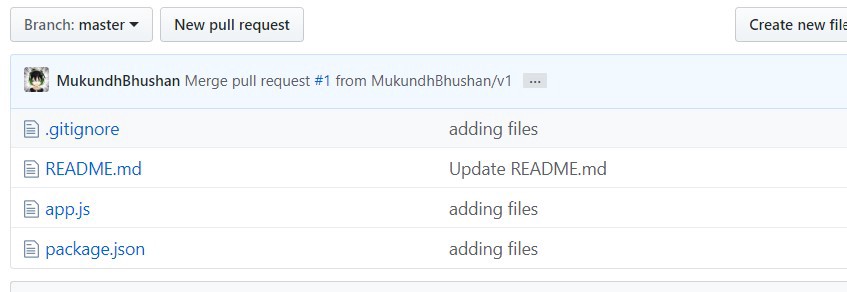
The new window will show you the branch which is being merged and the other basic information. You can add a description if you feel like. Click on “Confirm merge”. The merge is now complete and the master branch is now updated.

Image for post



On navigating to the master branch we can see all the files which were in our other branch are now in the master branch.

Image for post



**Deleting the branch**Once you have merged a branch it needs to be deleted as no changes will be a staged in it any longer. To delete a branch use the command

git push --delete <remote\_name> <branch\_name>

**Cloning a repository**

Cloning a repository is creating a copy of all the project files into your local machine. The command used for this is

git clone <url>

The URL in GitHub can be found in the “Clone or Download” option which we used earlier to add a remote to our repository.

**Forking a repository**

Forking a repository is similar t cloning a repository but unlike cloning a repository where the files are downloaded to your local machine forking creates a new repository in your name and adds all the files to it changes in the original repository will not affect your forked repository.  
GitHub makes it easy to fork a repository all you have to do is click on the fork icon found in the top right corner of the screen.

Image for post

Image for post

**Congratulations YOU DID IT!!!**