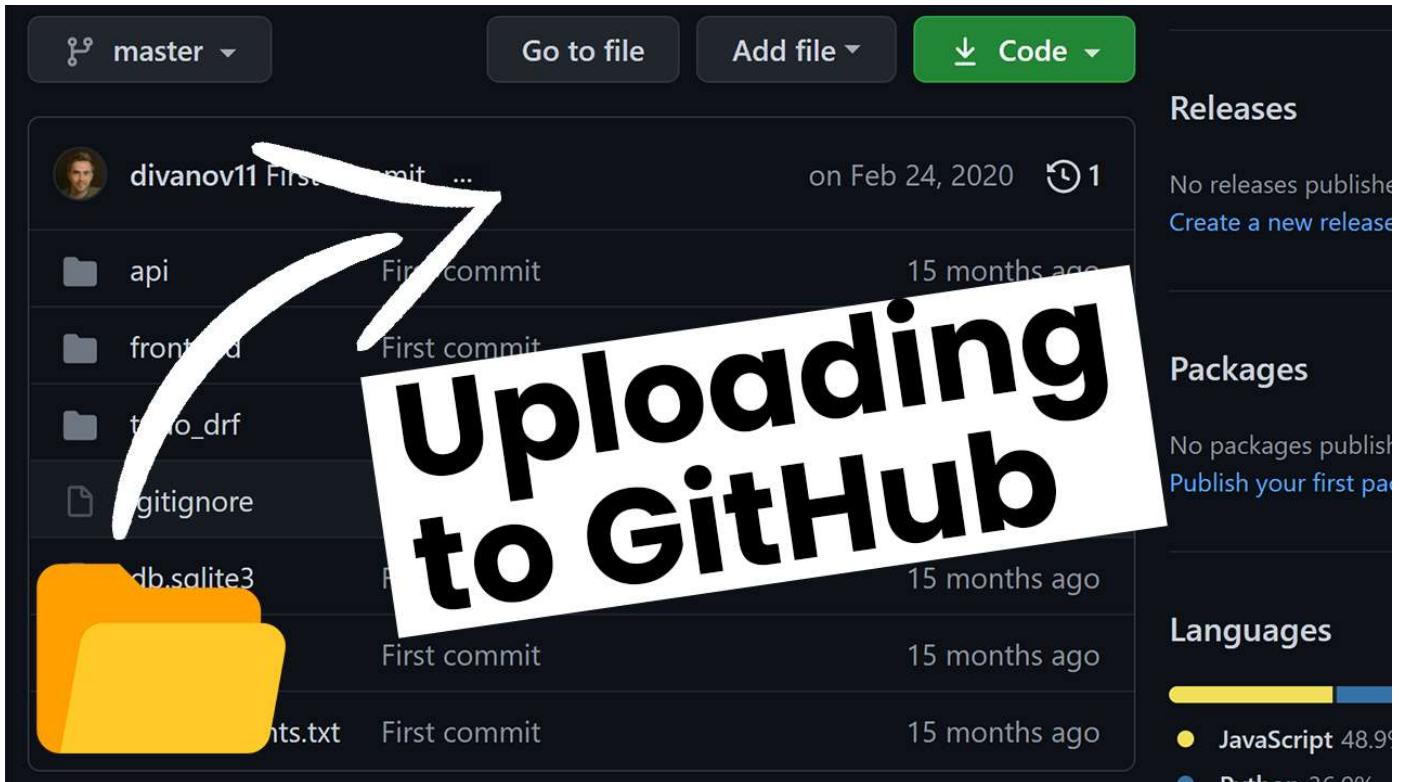


[Blog](#)[Events](#)[Discord](#)

By **Dennis Ivy**

 Posted on 2021-5-31

Upload Files to GitHub Quickstart Guide

So you have a project on your local desktop and now you want to get these files up on github.com, here's how you can do this.

Article summary

Create Remote Repository on GitHub.com

Download Git

1. Download Installer
2. Install git
3. Check Git version

Create Local Repository

```
> git init
> git status
> git add .
> git commit -m "Custom message"
> git remote <remote url>
> git push -u origin <branch name>
```

DON'T drag your files directly into your github repo!

For those of you who saw me do this in my django beginners course, I'm sorry. At the time I didn't realize the issues this would cause. This may work in some cases but I noticed in projects with more files this tends to leave out files in the upload process, why? I have no idea, so even if it works, DON'T DO IT!

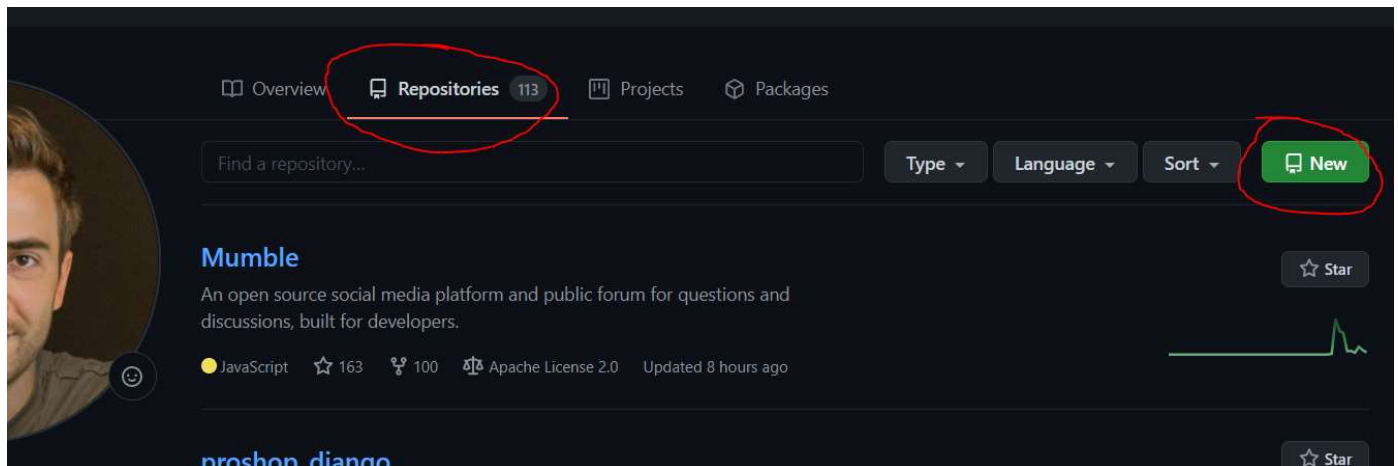
This won't be a full Git & Github tutorial, I'm gonna assume you get the point and just need to get your files up online. For a full overview I recommend you checkout this [tutorial](#) by my friend Brad Traversy.

So, Let's get started...

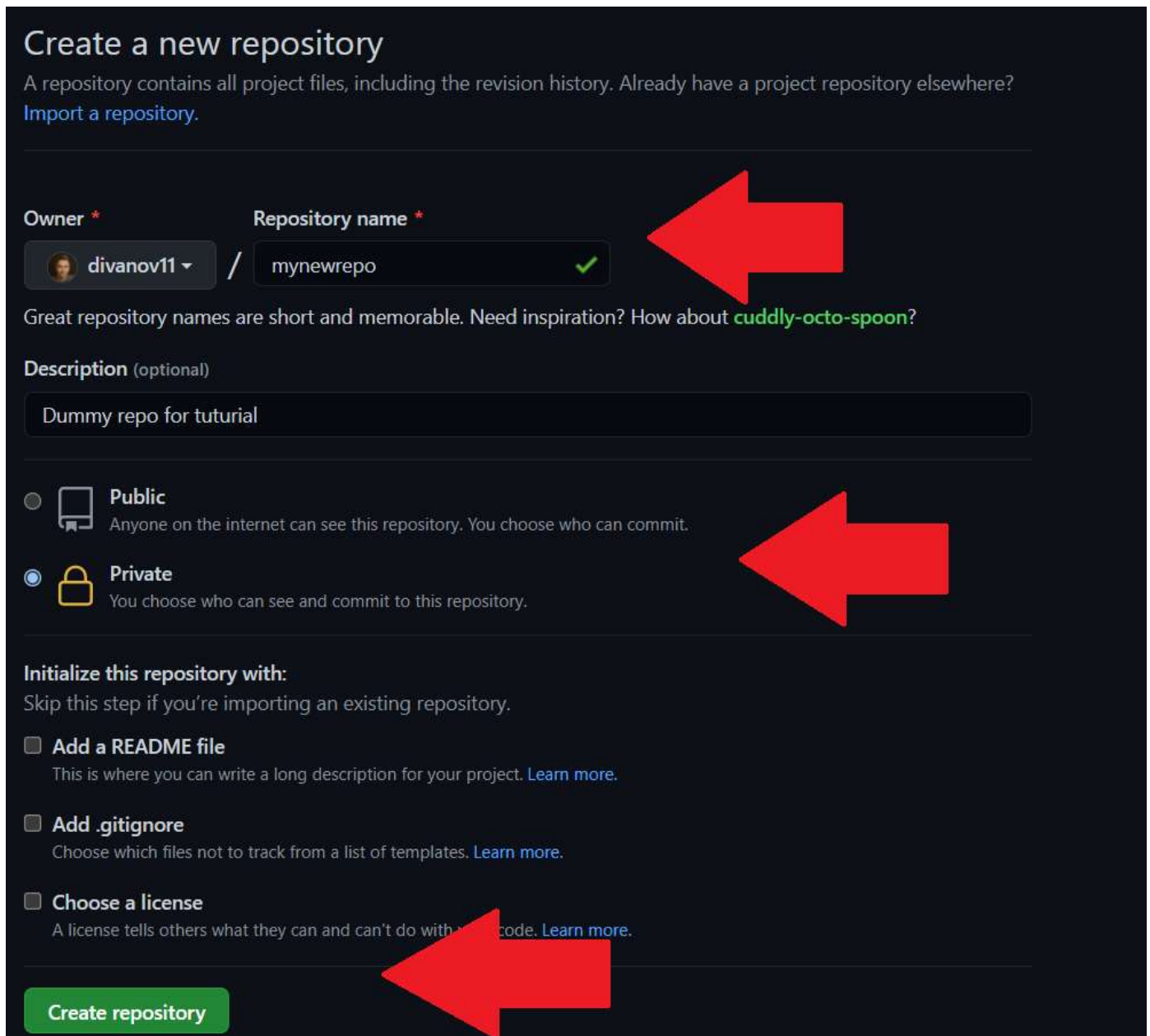
Create Remote Repository on GitHub.com

If you don't already have an account go ahead to github.com, signup, and create a repository, this is where our files will be added.

You can create your repository by going to “repositories” and by clicking “new”




In this example I will set my project to “private” and will not be creating a readme file. If you plan on sharing your project with the world make sure you leave it as “public”



Create a new repository

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

Owner * **Repository name ***

 divanov11 / mynewrepo ✓

Great repository names are short and memorable. Need inspiration? How about [cuddly-octo-spoon?](#)

Description (optional)

Dummy repo for tutorial

☐ **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.

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

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

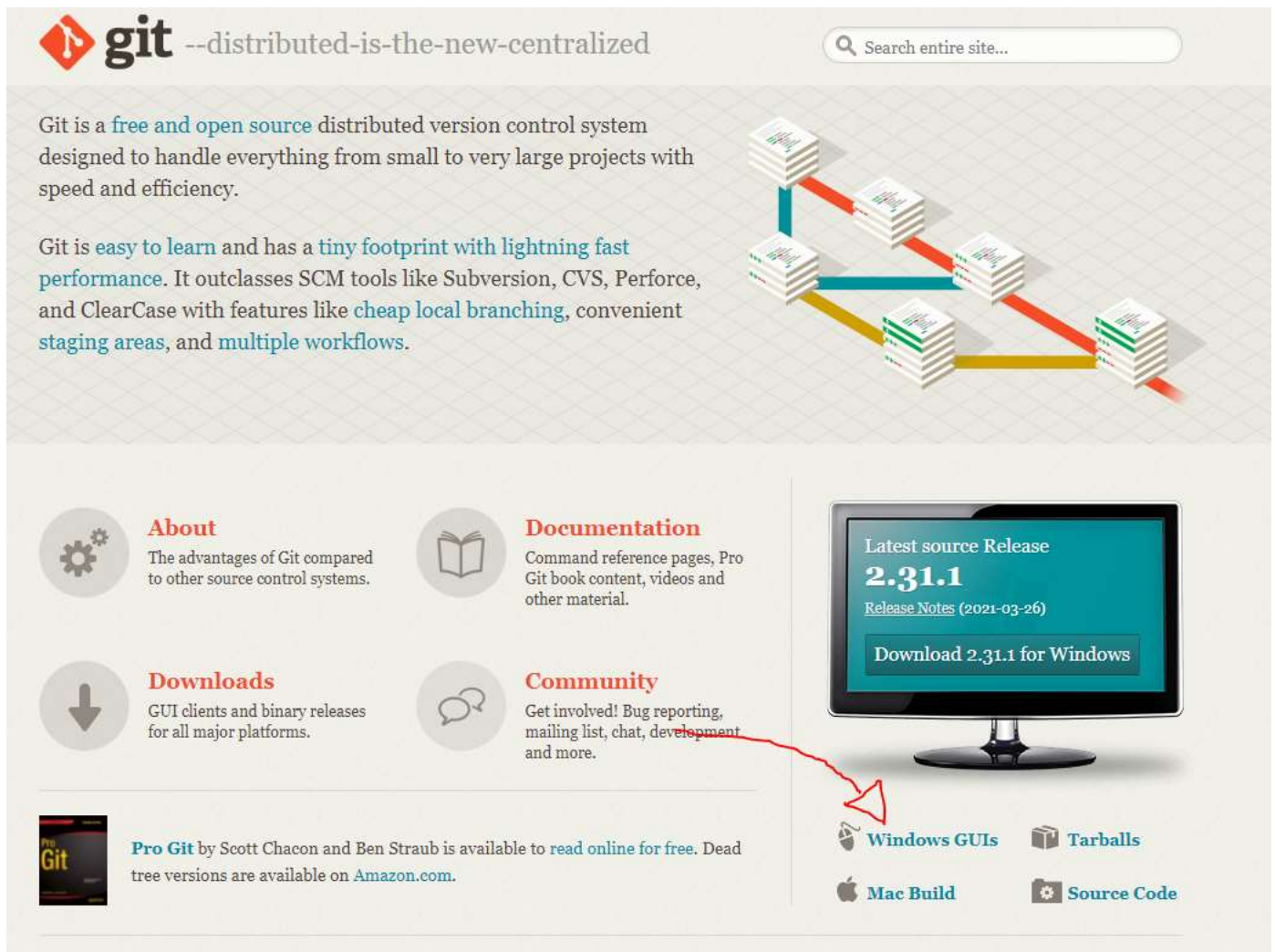
Create repository

Once you create your repo you will see a quick start guide on how to commit frp, your local repository. We will ignore these instructions for now since we need to get set up locally first.

Download Git

Before we can get started we will need to download git locally in order to push files to github.

1 - Download Installer - Got to <https://git-scm.com/> and select the downloader for your machine, I will be using windows



The image shows the Git website homepage. At the top, the Git logo is followed by the tagline "--distributed-is-the-new-centralized". A search bar is in the top right. The main content area describes Git as a "free and open source" distributed version control system. It lists features like "easy to learn", "tiny footprint", "lightning fast performance", "cheap local branching", "convenient staging areas", and "multiple workflows". To the right is a diagram of a distributed version control system with multiple nodes and branches. Below this are four sections: "About" (advantages), "Documentation" (reference pages, book content), "Downloads" (GUI clients, binary releases), and "Community" (bug reporting, mailing list). A red arrow points from the "Community" section to a monitor displaying the "Latest source Release 2.31.1" and a "Download 2.31.1 for Windows" button. Below the monitor are links for "Windows GUIs", "Tarballs", "Mac Build", and "Source Code". At the bottom left, there is a link to "Pro Git" by Scott Chacon and Ben Straub.

git --distributed-is-the-new-centralized

Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

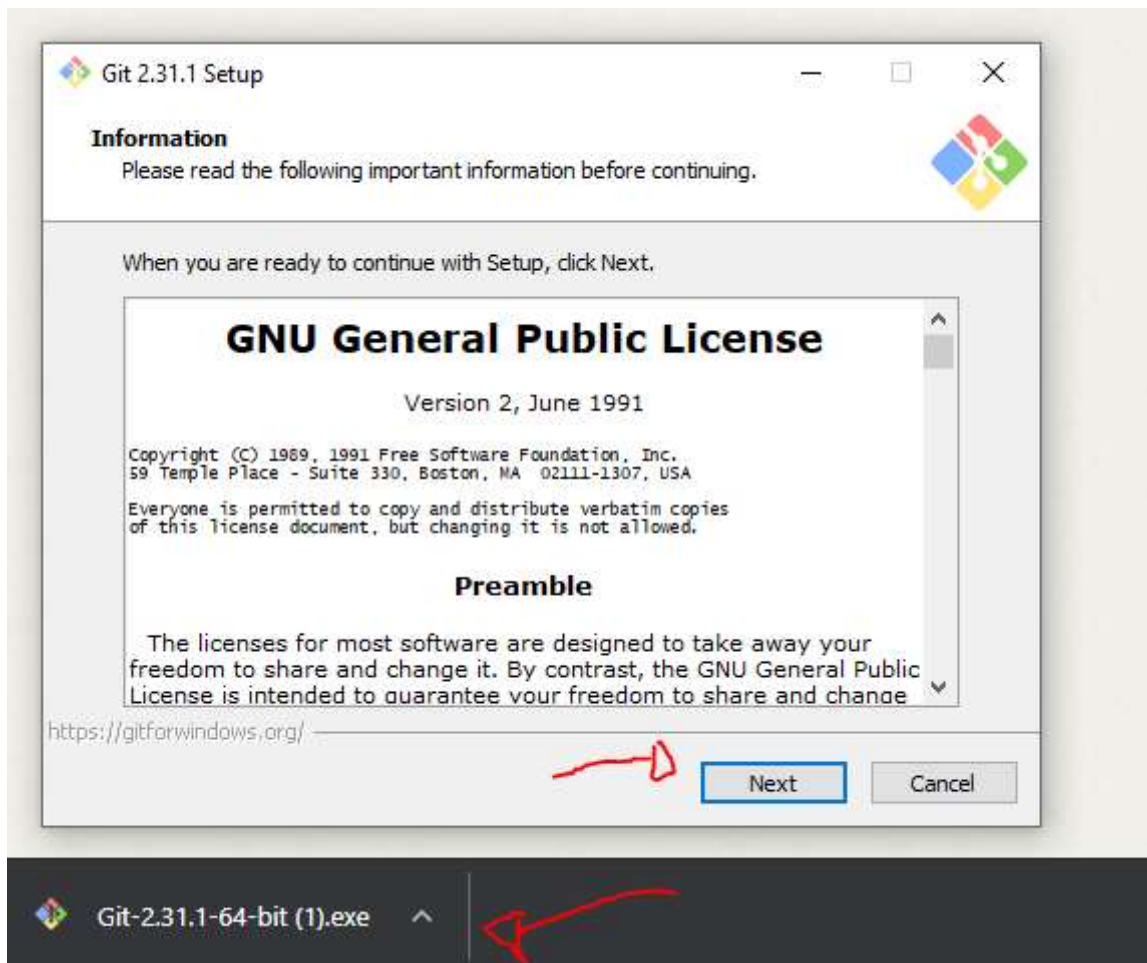
Community
Get involved! Bug reporting, mailing list, chat, development and more.

Pro Git by Scott Chacon and Ben Straub is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Latest source Release
2.31.1
[Release Notes \(2021-03-26\)](#)
[Download 2.31.1 for Windows](#)

[Windows GUIs](#) [Tarballs](#)
[Mac Build](#) [Source Code](#)

2 - Install git - Once the installer is downloaded go ahead and follow the steps to get things set up, I tend to just leave the default settings so if you don't have a preference just let it guide you.



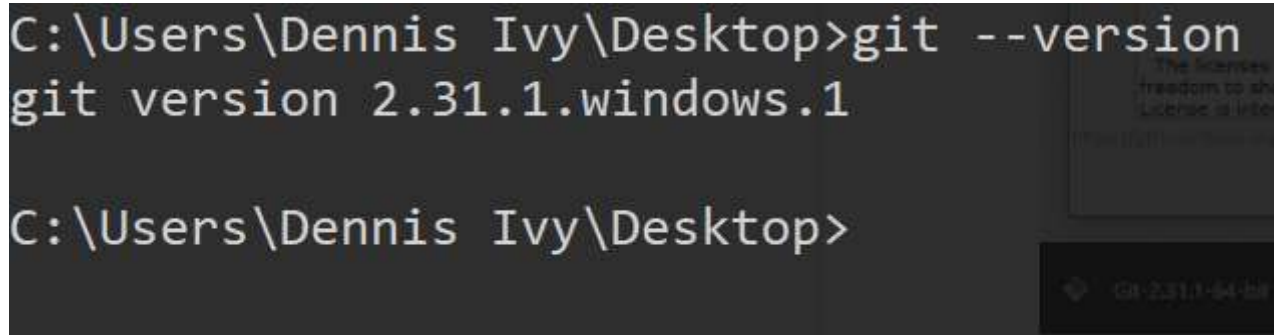
3 - Check git version

```
> git --version
```

Once things are installed do a quick search on your computer for git bash and open it. To check the version of git you have installed go ahead and type `git --version`

```
MINGW64:/c/Users/Dennis Ivy
Dennis Ivy@DESKTOP-HTA907J MINGW64 ~
$ git --version
git version 2.31.1.windows.1
Dennis Ivy@DESKTOP-HTA907J MINGW64 ~
$ |
```


You can also do this from your command prompt, in fact I will be using the command prompt from now on so go ahead and close git bash and open up a new terminal. If you already had your terminal open you may need to close and reopen to get the updates.

A screenshot of a Windows command prompt window. The title bar is not visible. The command prompt shows the path 'C:\Users\Dennis Ivy\Desktop>' followed by the command 'git --version'. The output is 'git version 2.31.1.windows.1'. Below this, the prompt 'C:\Users\Dennis Ivy\Desktop>' is shown again. In the background, a blurred window titled 'The license freedom to share' is visible, showing a license agreement. A small icon in the bottom right corner of the terminal window indicates the Git version 'Git 2.31.1-64-bit'.

Let's look at a few of the commands we will be using in the next steps:

- `git init` - Initialize local a new repository
- `git status` - shows what you have in you staging area
- `git add <file>` - adds files and folders to the staging area
- `git commit` - commits files in staging area to local repository
 - `git push` - Takes a local repository and pushes to a remote - - repository (Github).
 - Create a repo on github
 - set a remote
 - Add your github credentials
- `git pull` - Pull latest changes from remote repository
- `git clone` - Clone repo from github

Create Local Repository

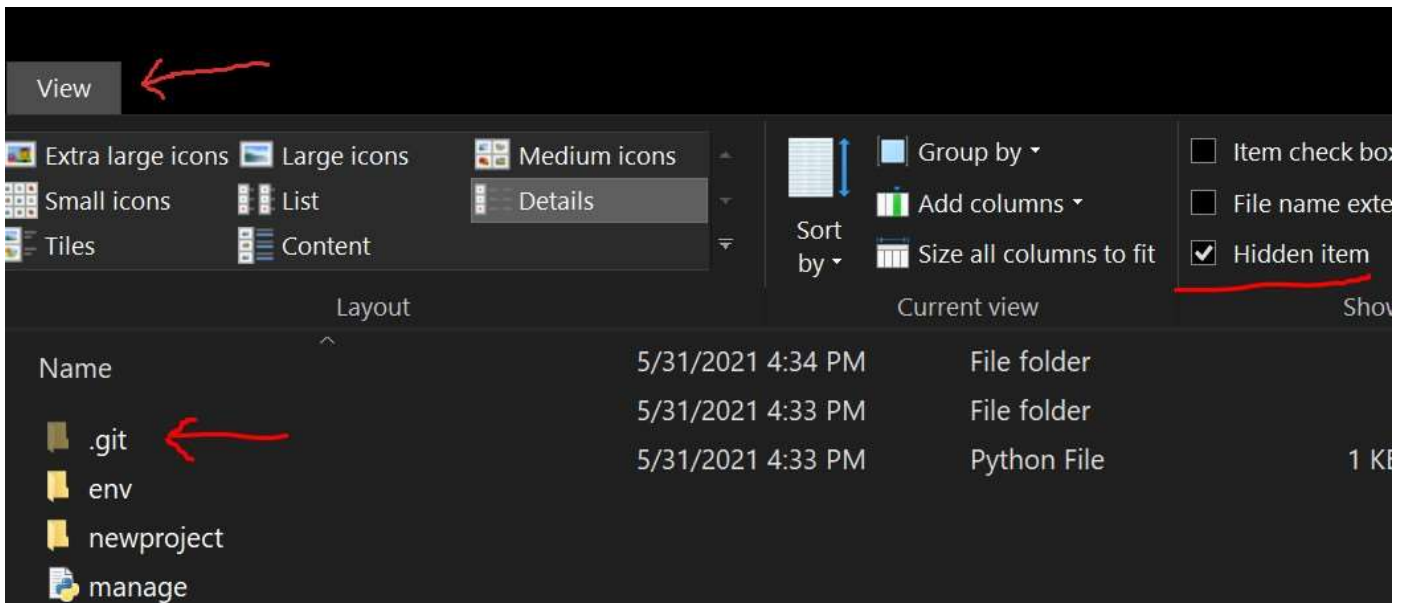
1 - Initialize new git repo

```
git init
```

Now that you have git installed, cd into your project folder and initialize a local repo.

```
cd projectname  
git init
```

When you run `git init` a new folder will be added to your project files, these files visibility are set to “hidden” by default so you will not see them but rest assured they are there. If you want to see these files open up your folder and select “view”



2 - Check your staging area

```
> git status
```

This step is not necessary before we add our files but for the sake of seeing the difference and showing exactly what happens lets run “git status”

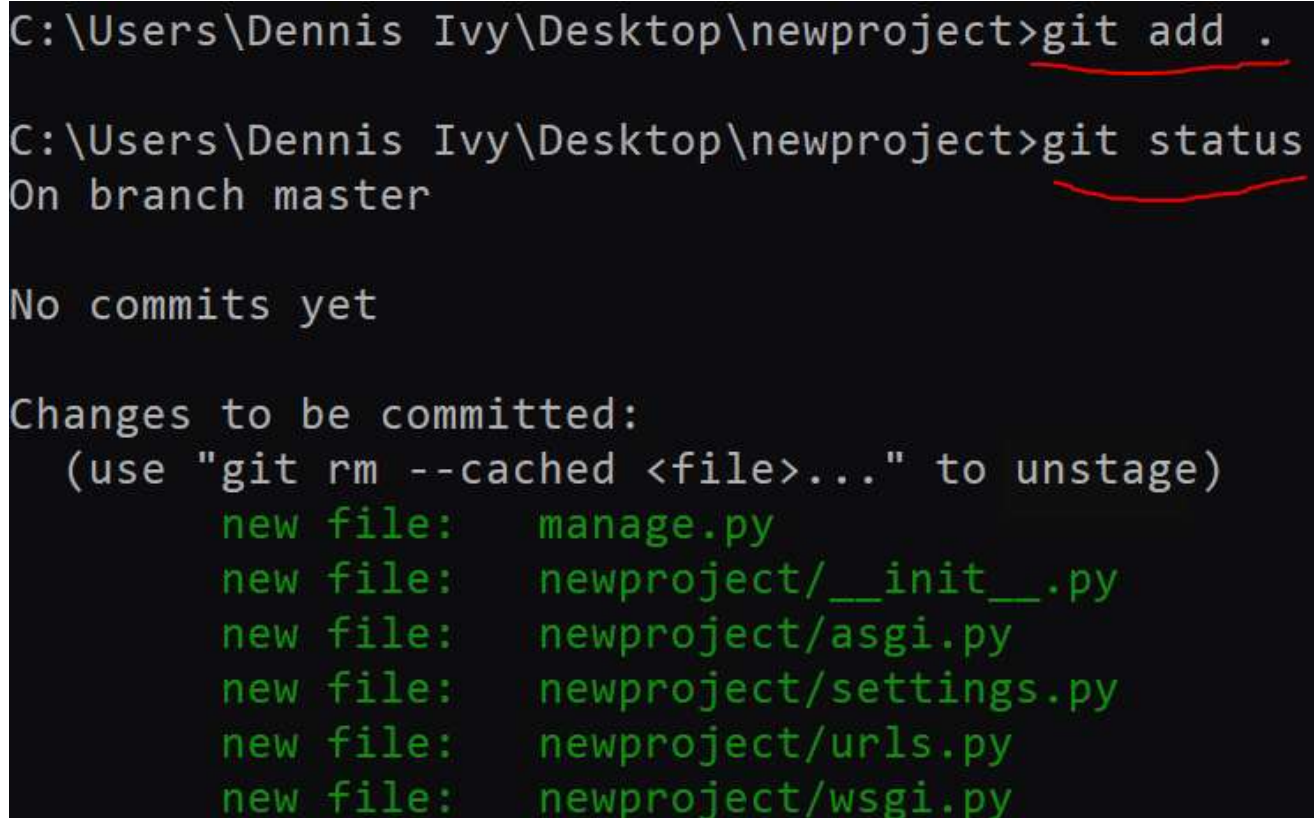
```
C:\Users\Dennis Ivy\Desktop\newproject>git status  
On branch master  
  
No commits yet  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    manage.py  
    newproject/
```


3 - Add files

```
> git add .
```

To add a particular file or folder to your staging area you can run “git add ”. In our case we want to add all the files highlighted in red so lets run `git add .` . In case you can’t see it there is a period after `add` indicating we want ALL the files.

After running `git status` I typically like to run `git status` again to ensure all changes we made.

A terminal window with a black background and white text. The first command is 'C:\Users\Dennis Ivy\Desktop\newproject>git add .' with a red underline under 'git add .'. The second command is 'C:\Users\Dennis Ivy\Desktop\newproject>git status' with a red underline under 'git status'. The output shows 'On branch master' and 'No commits yet'. Then it says 'Changes to be committed:' followed by '(use "git rm --cached <file>..." to unstage)'. Below that, a list of new files is shown in green text: 'manage.py', 'newproject/__init__.py', 'newproject/asgi.py', 'newproject/settings.py', 'newproject/urls.py', and 'newproject/wsgi.py'.

```
C:\Users\Dennis Ivy\Desktop\newproject>git add .  
  
C:\Users\Dennis Ivy\Desktop\newproject>git status  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  (use "git rm --cached <file>..." to unstage)  
    new file:   manage.py  
    new file:   newproject/__init__.py  
    new file:   newproject/asgi.py  
    new file:   newproject/settings.py  
    new file:   newproject/urls.py  
    new file:   newproject/wsgi.py
```

What if we make changes after adding running `git “add .”`?

No worries, just run `git add` again and changes will be updated. You can always run “`git status`” beforehand to see exactly which files have been modified.

4 - Commit Changes

```
> git commit -m “Your custom note”
```

Now that we added files to our staging area we need to commit them to our local repository.

```
git commit -m "first commit".
```

A terminal window with a black background and white text. The command 'git commit -m "First commit"' is entered and executed. The output shows the commit hash '4a08248', the message 'First commit', and a summary of 6 files changed with 195 insertions. The files listed are manage.py, __init__.py, asgi.py, settings.py, urls.py, and wsgi.py, all created with mode 100644.

```
C:\Users\Dennis Ivy\Desktop\newproject>git commit -m "First commit"
[master (root-commit) 4a08248] First commit
6 files changed, 195 insertions(+)
create mode 100644 manage.py
create mode 100644 newproject/__init__.py
create mode 100644 newproject/asgi.py
create mode 100644 newproject/settings.py
create mode 100644 newproject/urls.py
create mode 100644 newproject/wsgi.py
```

Files are now committed and ready to be pushed to our remote

5 - Set remote

```
> git remote add origin <repo url>
```

Now that all the files are set locally we are ready to push them to our remote repository. We set the remote so when we run `git push`, git knows which remote repo to send these files to.

Ex:

```
> git remote add origin https://github.com/divanov11/newproject.git
```

6 - Push to remote

```
> git push -u origin <branch name>
```

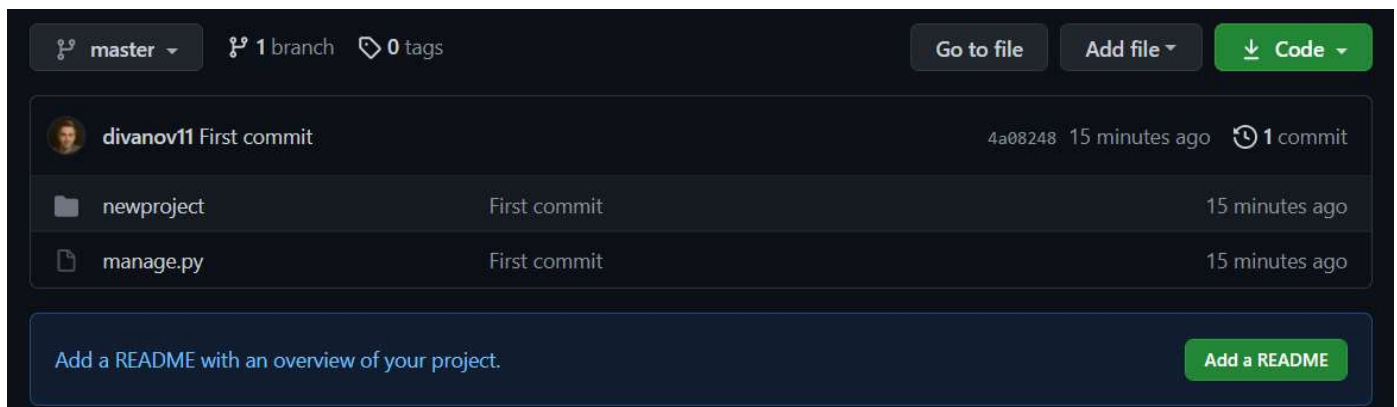
Now that we have our remote set, we can push our local github repo live. The default branch on github is called “master” unless you set your own. When you first create a repository on

github you will see steps to rename this branch to “main”, if you decided to rename the default branch then use that name instead of master.

```
git push -u origin master
```

```
C:\Users\Dennis Ivy\Desktop\newproject>git push -u origin master
info: please complete authentication in your browser...
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (9/9), 2.64 KiB | 2.64 MiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/divanov11/newproject.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Now if you refresh your repository on github.com you should see all your local files here.



The screenshot shows the GitHub interface for a repository named 'newproject' by user 'divanov11'. At the top, there's a navigation bar with 'master' selected, '1 branch', and '0 tags'. To the right are buttons for 'Go to file', 'Add file', and a green 'Code' button. Below this, a commit summary shows 'divanov11 First commit' with a commit hash '4a08248' and a timestamp '15 minutes ago', along with a clock icon and '1 commit'. A table lists the files in the commit: a folder 'newproject' and a file 'manage.py', both marked as 'First commit' and '15 minutes ago'. At the bottom, there's a blue box with the text 'Add a README with an overview of your project.' and a green 'Add a README' button.

Ignore files & cloning repos

Ignoring files: Add a .gitignore file and write in the file or folder name you want to ignore.

Clone a repo: copy the url of the repo you want to clone and simply run `git clone <repo url>`