



# Git + Deployment Guide

**Useful links if you want to skip this course and do another one:**

[Learn Git with Bitbucket Cloud](#) | [Atlassian Git Tutorial](#)

[Git Tutorial \(w3schools.com\)](#)

[An introduction to Git: what it is, and how to use it \(freecodecamp.org\)](#)

[Introduction to Git - Training](#) | [Microsoft Learn](#)

**and for deployment:**

[Get started with Netlify](#) | [Netlify Docs](#) - Dynamic websites

[Creating a GitHub Pages site](#) - [GitHub Docs](#) - Static websites

## GitHub and Git

### What is GitHub?

GitHub is a website that hosts files, whether it be code, pdfs, images or else. It is commonly used to host program files in what is called a 'repository'.

### What is a repository?

You can think of a repository like a directory in the sense that it is a folder that contains files and other folders.

But what is the key difference?

Say you have a folder on your machine, and this folder contains a file. This folder is your directory. Your repository is like an artificial copy of that directory, it is like the logical thing that stores and tracks changes in your files.

How does it do that?

Let's wind back, and let me explain...

## What is Git?

Git is a version control system that helps us track and save changes in our files (among various other things which we will go through) and to also communicate with our remote repository that is on a platform such as GitHub (others include GitLab, Bitbucket, etc).

## So, how do we use GitHub and Git?

Sign up / install required program

GitHub can be used fully online so all you need to do is signup for an account. (although there is a desktop version of it if you'd like)

We need to install Git on our local machine.

## Installing Git

Go to the following link and follow the steps to install git: [Git - Installing Git \(git-scm.com\)](https://git-scm.com).

When you get to the setup, feel free to just use the default setup, unless you know what you want/need to use.

## Use Git and GitHub

### Create a repository on GitHub

<https://www.loom.com/share/1d612e83c41d4dab974037094a78f87e>

## Using Git

Jump to a local directory where you want to put your project. Open a terminal in that directory and type:

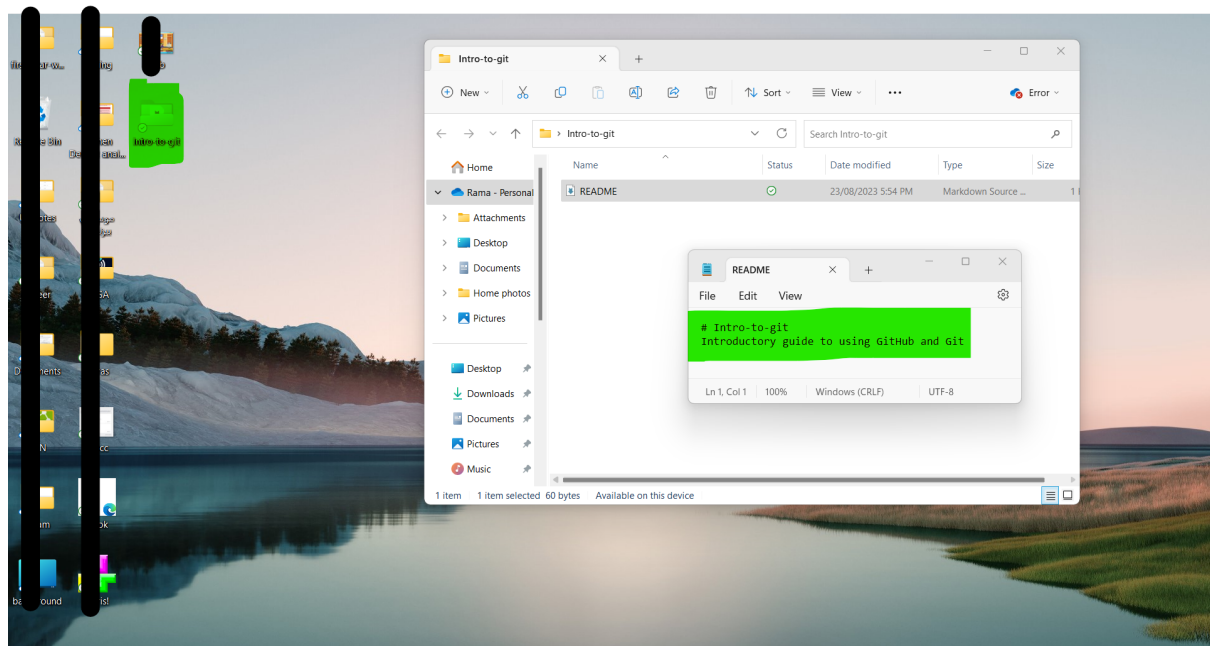
```
>> git --version
git version 2.40.0.windows.1
```

If you get a git version, then git is correctly installed and you can use it.

Next, go to your online repository and get the HTTPS link and use the following command to copy the remote repository and all its files to your local machine:

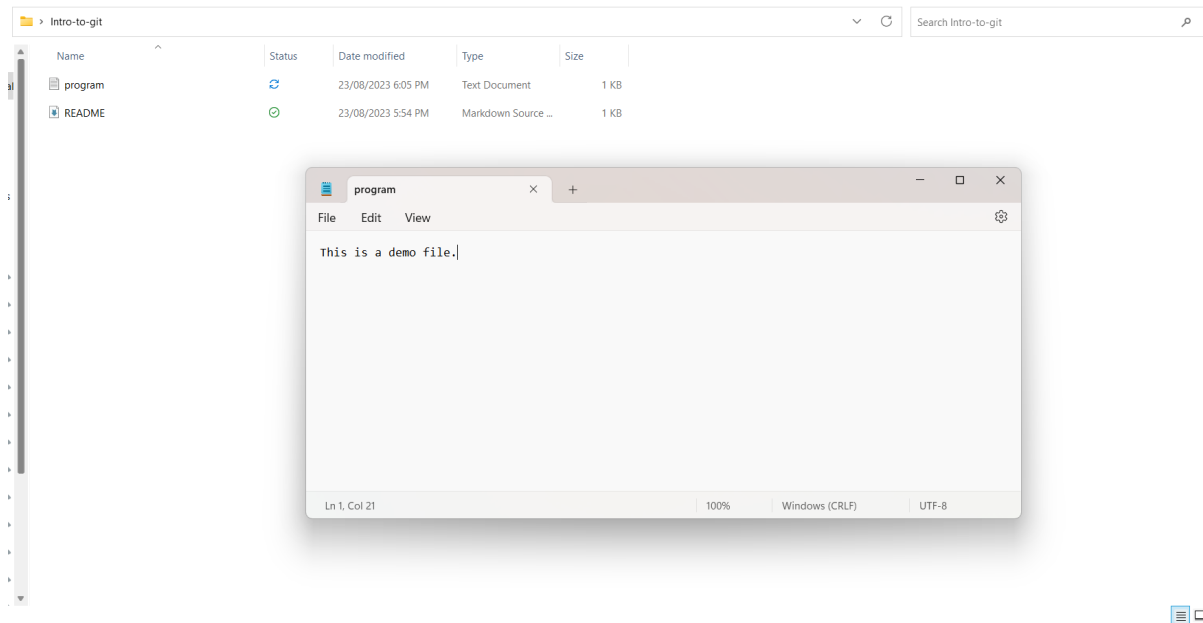
```
>> git clone https://github.com/RamaKaorma/Intro-to-git.git
Cloning into 'Intro-to-git'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

and you should be able to see your repository and its files like so:



## Making changes and uploading

I will create a new file in my directory for the sake of demonstration:



Now that we've added a file, let's check the status of our repo. Checking the status allows us to know which files have been changed and which ones have not. Use the following command:

```
>> git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    program.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Notice that it says **untracked files**. That's because there are two types of files in a repo:

1. Tracked – Git is aware of the file, and the file is in the repository.
2. Untracked – Git is aware of the file, but it is not in the repository, only in the directory.

(Recall the earlier example of a repo being like an artificial directory)

So what do we do? We made a change and we want that change to be saved. We need to stage the change and then commit it.

## Staging

```
>> git add program.txt
>> git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   program.txt
```

We can see that after staging, the file becomes tracked and notice it now says

`Changes to be committed` So let's commit:

## Committing

```
>> git commit -m "I have added a new file to my repo"
[main 7aa57cf] I have added a new file to my repo
 1 file changed, 2 insertions(+)
 create mode 100644 program.txt
```

This means success! Notice it states `2 insertions(+)` that means our changes were adding 2 new lines (and that was the case as I wrote a description in the program.txt file then pressed enter). If, say, I removed 10 lines from two files, the description could be something like:

```
2 files changed, 10 deletions(-)
```

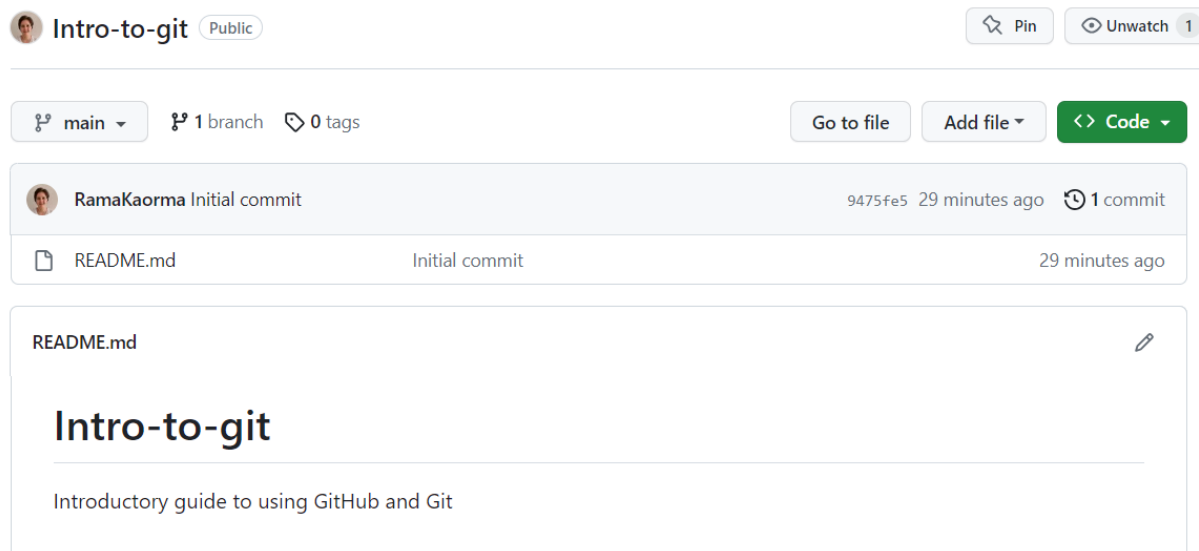
Now that we have committed the changes, we can do `git status` again:

```
>> git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

and the key statement now is `Your branch is ahead of 'origin/main' by 1 commit`. What this basically means is that our remote repository is not up to date anymore

(origin/main is referring to the remote repo). If you check the repo on GitHub, you will not see program.txt



The screenshot shows the GitHub interface for a repository named 'Intro-to-git' which is public. At the top, there are buttons for 'Pin' and 'Unwatch' (with a count of 1). Below this, a navigation bar shows 'main' as the selected branch, '1 branch', and '0 tags'. Action buttons include 'Go to file', 'Add file', and a green 'Code' button. The commit history section shows a single commit by 'RamaKaorma' titled 'Initial commit' with hash '9475fe5' from '29 minutes ago'. Below the commit list, the 'README.md' file is displayed, containing the title 'Intro-to-git' and the subtitle 'Introductory guide to using GitHub and Git'.

No worries, easy fix:

```
>> git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 325 bytes | 325.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/RamaKaorma/Intro-to-git.git
9475fe5..7aa57cf  main -> main
```

and what that does is push the current version of your files to the remote repository, and now this will appear on GitHub:

The screenshot shows the GitHub interface for a repository named 'Intro-to-git' which is public. At the top, there are buttons for 'Pin' and 'Unwatch' (with a count of 1). Below this, a navigation bar shows 'main' branch, '1 branch', and '0 tags'. Action buttons include 'Go to file', 'Add file', and a green 'Code' button. A commit history section shows two commits by user 'RamaKaorma':

- A commit with message 'I have added a new file to my repo' (hash 7aa57cf, 9 minutes ago) with 2 commits.
- A commit for 'README.md' (Initial commit, 33 minutes ago).
- A commit for 'program.txt' (I have added a new file to my repo, 9 minutes ago).

Below the commit history, the 'README.md' file content is displayed, showing the title 'Intro-to-git' and the subtitle 'Introductory guide to using GitHub and Git'.

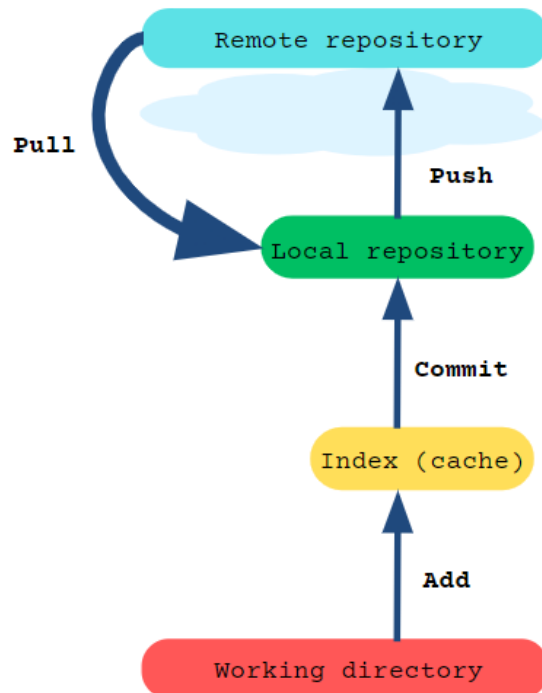
Also notice that you can press on the **commits** button above, and that will show you all the changes that have been made over time:

The screenshot shows the 'Commits' page for the repository. It features a 'main' branch selector and a timeline of commits. A vertical line indicates 'Commits on Aug 23, 2023'. The commit history shows:

- A commit by 'RamaKaorma' with message 'I have added a new file to my repo' (committed 10 minutes ago), hash 7aa57cf, with a 'Code' button.
- An 'Initial commit' by 'RamaKaorma' (committed 34 minutes ago), hash 9475fe5, marked as 'Verified' with a 'Code' button.

And that covers the basics of Git and GitHub.

Here's a visualization of what the different commands we learnt about do:



*A couple of optional (but useful) topics:*

## ▼ Start locally, go remote

What if you already have files offline and you want to push them to GitHub? One way is to create an online empty repo and then clone it onto your pre-filled directory, then add, commit and push. Or you can start locally:

1. Open the terminal and direct to your folder
2. Initialise a local repo like so:

```
>> git init
Initialized empty Git repository in <THE LOCAL ADDRESS OF YOUR DIRECTORY>
```

3. Configure the git settings, meaning let git know who you are:

```
>> git config --global user.name "<YOUR GITHUB USERNAME>"
>> git config --global user.email "<YOUR GITHUB EMAIL>"
```

4. Where do we save the changes though? Create your remote repo on GitHub, and get the HTTPS link, then set the origin/main (remember this from



earlier?)

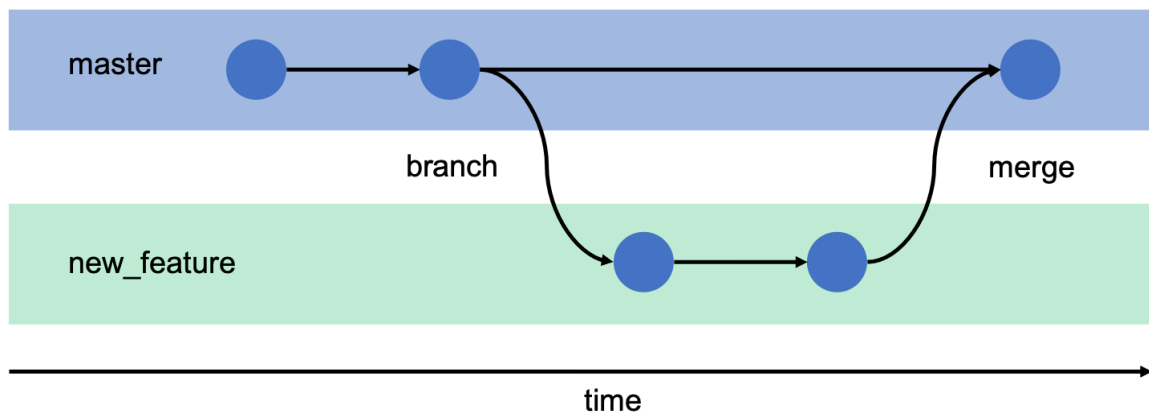
```
>> git remote add origin <THE HTTPS LINK>
```

and you're all set up to make changes, add, commit and push as much as you like.

## ▼ Branches, merges and conflicts

### Branches

Come with me on a journey, you're working on an assignment, and you've done Part 1, works perfectly and you're happy with your code. You go to Part 2, and oh my, you just messed up both parts, you can't find your old code or retrieve it 😞 Sad to say, we've all been there. Well, not anymore with Git. Check out the following diagram:



Yes, like it indicates, you can make a copy of your code, work on that copy, and once you know it works perfectly, you will merge back with your old code, and now you have Part 1 and Part 2 of the assignment done!

Let's go through it one step at a time:

Check what branches you have as of right now:

```
>> git branch
* main
```

This means I have 1 branch called `main` and sometimes it will be called `master`, it depends on your OS (and you can also rename it as you wish).

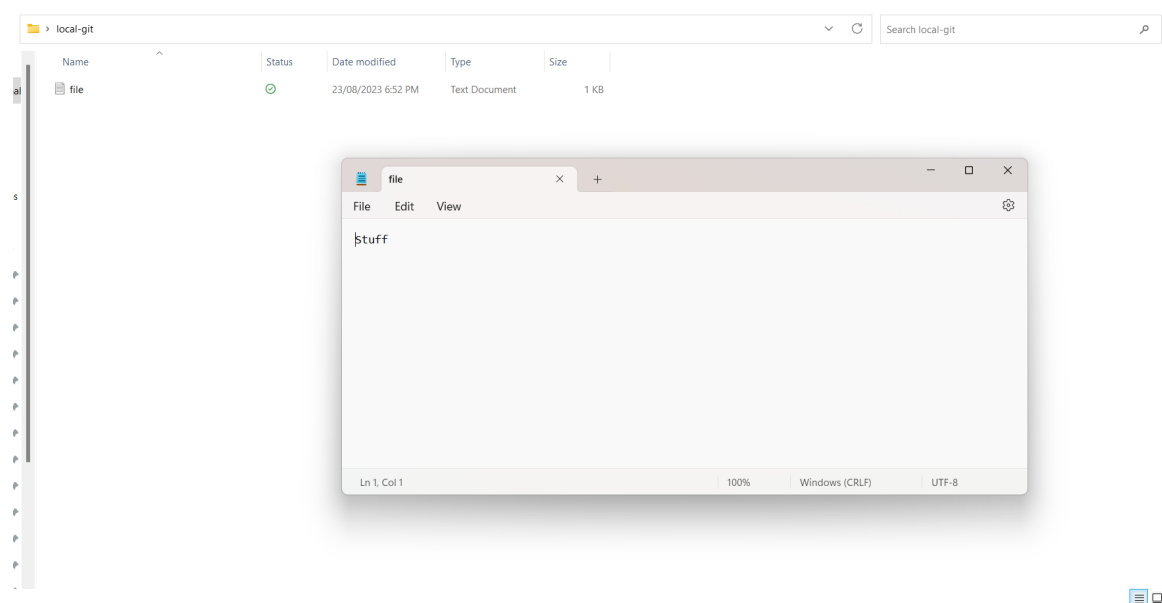
Let's introduce a new feature `Feature` that makes changes to my file

```
>> git branch newFeature
>> git branch
* main
  newFeature
```

Notice how `* main` is in green and has an asterisk next to it. This means that currently, I am on the main branch, and any changes I make happen on that branch. What do I do? I switch to `newFeature` to do my bad work there:

```
>> git checkout newFeature
Switched to branch 'newFeature'
>> git branch
  main
* newFeature
```

Great, now, this is what's in the main branch:



So when I created the branch `newFeature`, I copied this `file` to the new branch, and kept a copy in the `main` branch. Check this out:

<https://www.loom.com/share/6c746d42a3ff4cd0a2f6a5021599e600?sid=b3fc23c6-8cc4-4b57-aa29-074f6c3d85e4>

Now what? I want to merge it with my main branch! I want 1 big program not 10 separate bits and pieces!

I got you, switch back to your main branch (where you can't find any of the changes you made), and use the following git command:

```
>> git merge newFeature
Updating c092bbd..2970423
Fast-forward
 file.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Can you see your changes now? Yup.

## Conflicts

Another scenario, you're working with a teammate (let's call them X) on a project, and this happens:

1. You create an online repo and share it with them.
2. You clone the repo to your local machine, and so does X
3. Now X makes changes to the files, and pushes them to GitHub, good on X.
4. Notice that now you're working on the old versions of the files!!! Not good, not good at all. So what do we do? Before you make changes, **pull** the changes from the repo to your local machine like so

```
>> git pull origin main
```

Now you have the new version, and you can make changes on that.

Consider a worse scenario... The previous steps 1, 2 and 3 are the same, then:

4. You make changes to the files on your machine, then try to push, good on you... or not.

Git is smart, it knows that you have made changes to the original file you got from GitHub, but it also knows that there are changes that you did not make! This causes what's called a 'merge conflict'. In such a case, you need to check the differences and make changes accordingly. Please checkout the following article

about merge conflicts and how to resolve them: [Git merge conflicts | Atlassian Git Tutorial](#)

## Deployment with Netlify

Check out the following video run-through:

<https://www.loom.com/share/71a950f7ea04475f9814e84ec5395ab6?sid=a203790d-a0e4-4c55-b3cc-7c6d5c6e5339>