What is Git?

Git is a distributed version control system, which means it keeps a copy of every stage of your development process
Some could say it’s backups on steroids

It’s really useful when you have many programmers on one code base
Lots and lots and lots of features

What is GitHub?

Think of GitHub like Google Drive or OneDrive.
Now think of Git like an external hard drive you back your photos to.
GitHub is an online server for storing git repositories… but it’s waaaay more than that.

GitHub allows you to manage repositories, doing things like Pull Requests, Issues and more, which are super useful when you’re working with several people

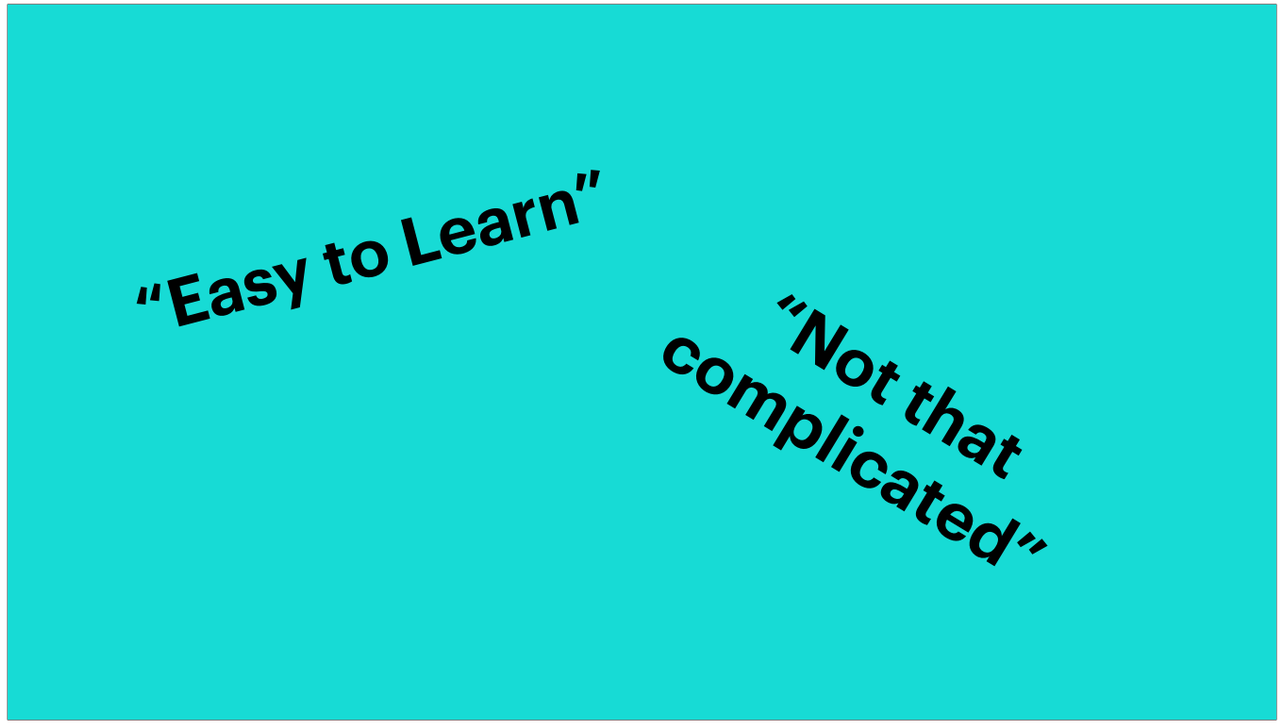
GitHub doesn’t own git, it just uses git. Same for similar services like GitLab or companies internal git servers.

Now some may say Git is “Easy to learn” or “Not that complicated”

That’s not necessarily true
But that’s okay!

I’m going to break it down for you so you’ll understand the basics by the end of the workshop

“Easy to Learn”


“Easy to Learn”
“Not that complicated”

And that’s okay!

I don’t understand the jargon

Now Git has a looooot of jargon involved with it…..

While we’re starting out, it can be pretty overwhelming, especially if you find one of those guides that’s 1000+ pages long and has words you’ve never heard of before….

We’re going to pretend that doesn’t exist because you’ll unlikely to need most of that for your every day git needs. In fact, you only need to understand the basics to for 99% of the things you’ll do in git.

Key Terms

- Repository
- Commit
- Remote
- Push
- Pull
- Branch
- Merge
- Pull Request

Let’s begin with some key words. Some of these can seem a bit odd but it will all make sense shortly. First of all we have….

Repository

Repository
A repository is simply just a clever directory or folder. It contains a folder called .git that does all the magic git stuff for us. Ignoring all that, it’s just the folder all your code for your project goes in.

Key Terms

- ~~Repository~~
- Commit
- Push
- Remote
- Pull
- Branch
- Merge
- Pull Request

Next on our list is commit!

Commit

A commit is a collection of content, a message about how you got there, and the commits that came before it.

Now this is a tricky one to get your head around. I'd recommend trying this one in practice, which helps see how it's used. I'll also be going over this one in a bit too!

Key Terms

- ~~Repository~~
- ~~Commit~~
- Push
- Remote
- Pull
- Branch
- Merge
- Pull Request

Now we have push

Push

Pushing is used for sending our commits to the server storing another copy of the repository, in this case GitHub. Think of this as the sync button on Google Drive or OneDrive that makes sure your local folder matches the one in the cloud.

Key Terms

- ~~Repository~~
- ~~Commit~~
- ~~Push~~
- Remote
- Pull
- Branch
- Merge
- Pull Request

Now how does pushing your code know where to go? That’s where a remote comes in!

Remote

A remote is simply just a URL that tells your repository or folder of code where it should put it on the web. GitHub will give you this and you just tell the repository that this is the place you want to send the code.

Key Terms

- ~~Repository~~
- ~~Commit~~
- ~~Push~~
- ~~Remote~~
- Pull
- Branch
- Merge
- Pull Request

Now we've talked about pushing our code to the server…. What if we want to get the newest version of the code from the server?

That's where pulling comes in!

Pull

Pulling is where you ask the server for the latest changes. It will then “sync” these with your local repository so everything is up to date.

Key Terms

- ~~Repository~~
- ~~Commit~~
- ~~Push~~
- ~~Remote~~
- ~~Pull~~
- Branch
- Merge
- Pull Request

Now we’re getting into the cool parts of git. This is where git stands out for collaboration! Branches are very useful when you have multiple people working on the same project…. But why?

Branch

Branches are kinda confusing but think of it like this….

When you make your repository, there’s 1 branch by default, we call this master. Lets say we wanted to try and make a new feature but didn’t want it to break the current code? Well that’s where this comes in nicely!

A branch basically takes the entire code base and makes a copy of it.

Now you can modify the code base, without breaking any of the code that might be used in production.

This comes in particularly useful when working with several people, as you can all have your own copy of the code without getting into each others hair!

Key Terms

- ~~Repository~~
- ~~Commit~~
- ~~Push~~
- ~~Remote~~
- ~~Pull~~
- ~~Branch~~
- Merge
- Pull Request

Now what if we wanted to integrate these new changes into the existing code?

Merge

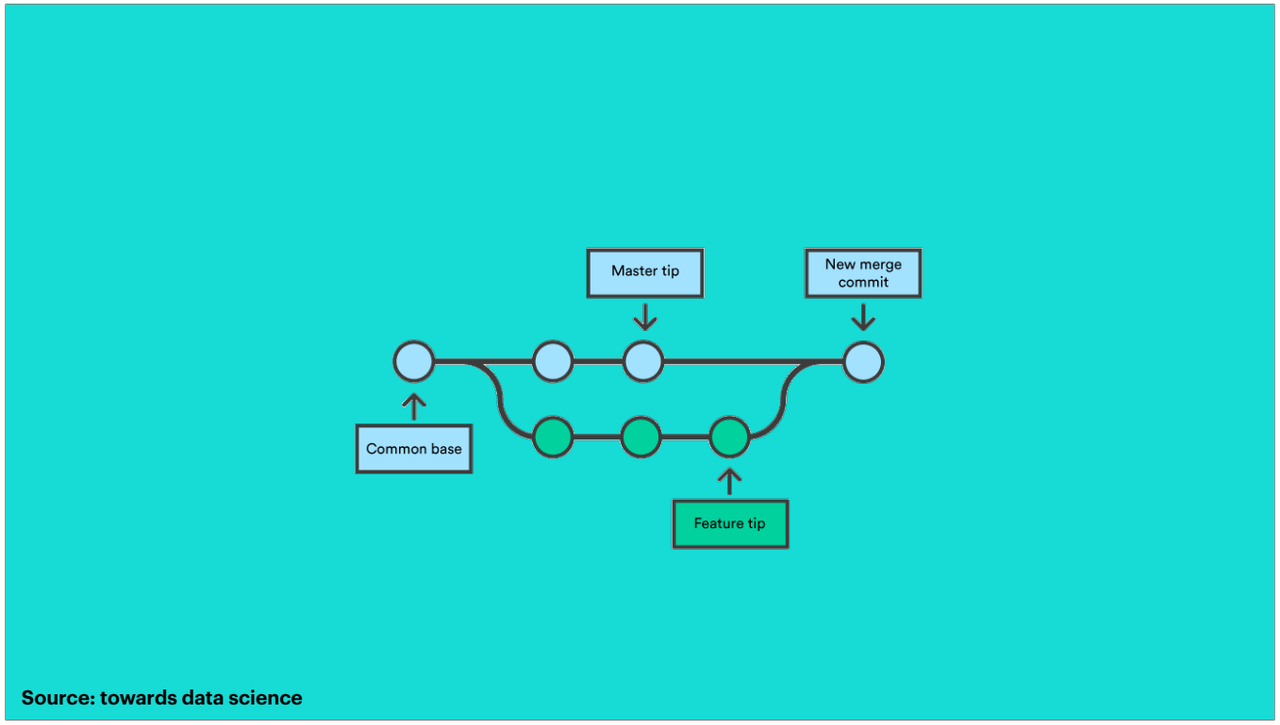
This is where merging comes in! This is where 2 branches “merge” together to become 1. Sometimes this is really nice and everything just “works”

Sometimes everything breaks and and burns and everyone cries…. This is known as a merge conflict…. Something was changed on both branches but git doesn’t know which change you want. This can be a simple fix, or more complicated….

Source: GIPHY

You’ve probably seen this meme before but it’s very relatable for many of those who’ve used Git before

Source: GIPHY



Source: towards data science

Here’s a nice diagram that shows what’s going on.
We start off with the master branch, and then branch off. Master continues getting commits while the new branch does. Eventually the two combine to make the new start of master

Key Terms

- ~~Repository~~
- ~~Commit~~
- ~~Push~~
- ~~Remote~~
- ~~Pull~~
- ~~Branch~~
- ~~Merge~~
- Pull Request

Now we've got that one out of the way…. Let's talk about good ways to merge!

Pull Request

A pull request is where someone asks for someone else to review the changes they've made in their branch, before it gets merged into another branch.

This helps stop things breaking and goes horribly wrong. Last thing you want is broken code going into live production, like onto the main website!

Key Terms

- ~~Repository~~
- ~~Commit~~
- ~~Push~~
- ~~Remote~~
- ~~Pull~~
- ~~Branch~~
- ~~Merge~~
- ~~Pull Request~~

Now we've covered all the key terms, I think it's time to put them into practice…..

Live Demo Time!!!

All the information I’m going to be talking about today will be on workshops.will-russell.com (when GitHub Pages decides to work again…)