What is GIT (Good answer with example)

Imagine you're making a sandwich. Let's also imagine you have a fancy refridgerator called Git that can make special copies of each step of your sandwich making places on an infinite sumber of shelves.

1. You start with two slices of bread and add it in Git. Git now contains two slices of bread on one of it's shelves.
2. You add a slice of cheese to your sandwich and also add it to Git. Git now contains a slice of cheese on the shelf just above the last shelf.
3. You add a slice of ham lettuce and add it to Git. Ham and lettuce sits on a shelf.
4. You add ketchup. ketchup sits on a shelf.

But wait, putting ketchup was a mistake so you go into Git and get the sandwich as you built it up to step 3. So now you're back to the sandwich with no ketchup and now you can put the correct condiment, mustard. Rolling back this way is much better than scraping the ketchup off because you might miss some of it.

Git also has the ability to keep the sandwich as is but also let you experiment on another copy (Forking). It'll store this fork on a new set of shelves beside your 'main' shelf.

(This is about as far as I can take the ELI5 as I don't fork too much - but you get the idea I think)

When you make things like programs, games, and web sites on your computer, it's a lot of work. And sometimes you mess up. But one of the great things about computers is that they can help you fix those mess-ups by remembering how it was before you screwed up. To do this, computer programmers often use something called "source control."

Source control is great! One of the most popular ones is called Git, and it can make your life a whole lot easier. Here are some of the benefits:

* When you make changes to a web page or program or something, it's really easy to see what has changed.
* You have a really good history of what happened in your project, and can always look at things from any point in that history.
* If you want to work with other people on the same project, Git makes it really easy to be sure that you're not going to screw up what those people have done.
* If you and the people you're working with DO happen to change the same file, it's pretty easy to fix any errors between the two.
* Git's set of tools makes it super easy to maintain and send your files from your computer to the one where your website or project is stored -- which we usually call a "server".

And it's free! You'll still have to do some other setup stuff, just like you normally would. Setting up things like databases would still have to be done as usual.

level 2

**ELI5 GIT**

GIT stores the state of your application at points in time known as **commits**. It does this in a clever way that means you don't have to store the whole state of your application, it just records a series of changes.

**More details**

In GIT you have a **master** branch which (usually) stores the working, production state of your application. You can also create other **branches** for working on specific things such as **features** or **bug fixes**. This allows **you** and **other developers** to continue development without affecting the master branch.

Once testing is complete, you can **merge** these branches into your master/production branch. If you are developing a website using GIT, you might have the **repository** stored on a server somewhere. You can set up a **hook** in GIT that **pushes** your **master commits** to this repository, but no others.

One advantage this has over FTP is that GIT will just push the **changes** and not the files themselves, making uploading new versions ***a lot*** quicker.

GIT, however, has a few other **fantastic** features. Finding the source of **bugs** in GIT is ***extremely*** easy. It has built in tools for finding where a piece of code stopped working.

One of the other great features of GIT is allowing people to **collaborate on code *very* easily**. If you have one person, **person A**, working on implementing **widget x** and one person, **person B**, working on **bug fix y**, they can each make a branch for working on their particular project. Let's say **person B** finishes his bugfix after two hours and merges his branch with the master branch, but **person A** is still working on his **widget x** using the previous code. In GIT this doesn't matter, **person A** can either **pull** in the changes **person B** has made or he can just **merge** the changes he's made with the master branch and GIT will handle everything. If there are conflicts, you can use GIT to intelligently manage these differences. This is by far and away the most useful thing about GIT, **working on code with more than one person is made 100 times easier**.

As an example, I was working on a site recently where I wanted to update the site to use Twitter bootstrap 3 (from v2) while someone else was working on implementing a feature of adding charts to the admin system of the website. This would have been very hard, had we not been using GIT, as we would have constantly had to share what we'd changed or what one developer would have had to do so that it would be in line with the other's developments. As it was, GIT's **merge** facility made this extremely easy.

With all that said, if you are just a **solo web developer**, there's not too much of an advantage to working with it except for keeping a track of site versions and for tracking down bugs. GIT is primarily aimed at application development, especially in terms of libraries and code that a lot of people will want to use.

On top of all this, you have **GitHub**. GitHub is a great place to keep track of your code. It allows you to develop with others, keep track of bugs, encourages people to report issues, it also allows people to **fork** (make a copy of for their own development) your code and make improvements, which you can choose to implement or not.

I've highlighted some terms used in GIT like **pull**, **push** and **merge**. I would also like to note that some very popular software is developed using Git and GitHub, for example, the Linux kernal, jQuery, Twitter bootstrap, Zend Framework, and many, many more.

Alongside that, you can use it to track versioning of, not just software, but *anything*.

If you would like some resources for learning about GIT then just let me know. One piece of advice I can give is **use it from the command line**. Their GUI application, frankly, sucks donkey dick, and doesn't capture the essence of *using* GIT for development. Some good beginning tutorials:

[How the Heck Do I Use GitHub? by LifeHacker](http://lifehacker.com/5983680/how-the-heck-do-i-use-github) [Pro Git](http://git-scm.com/book) - this is what I used to get started with Git. It covers everything, pretty much. If anything, I would recommend reading just the first chapter to know what Git is and what it can do.

Happy Git-ing!

Edit: P.S once you go Git you never go back! In case you couldn't tell, I love it.