**What is GitHub?**

Hi, my name is Jasmin, and I'll be your instructor.

We'll be learning about GitHub, how to use it for projects and version control, and everyday tasks on the platform, including how to generate personal access tokens.

**3. Chapter overview**

First, we'll review what GitHub is. We'll learn about the benefits of using the platform for different projects and see how GitHub differs from Git.

**4. GitHub**

GitHub is a cloud-based hosting service for users to upload and track their work. Usually, this work is code-based. This tracking is often referred to as version control. Cloud-based means that GitHub provides on-demand resources to its users over the internet, or the cloud. For example, it offers storage space, so we don't have to keep large files on our computers. Many platforms offer a similar service, such as GitLab and BitBucket, but GitHub is the most popular.

1. 1 iStock image credit:vectorwin

**5. GitHub uses**

The primary uses for GitHub are storing and keeping track of projects and files and collaborating with others. It also acts like a social network, allowing us to connect with other users. There are also several open-source projects (projects that are open to the public) that anyone can learn from, practice with, or edit.

1. 1 iStock illustration ID:981887940

**6. GitHub vs. Git**

Before we go any further, let's distinguish how GitHub differs from Git. Git is a version control software and can be used independently from GitHub or another hosting platform. GitHub is a platform that enhances Git to make it easier to manage projects and collaborate. It is entirely dependent on Git. We cannot use GitHub without Git!

1. 1 Git Logo by Jason Long is licensed under the Creative Commons Attribution 3.0 Unported License.

**7. Collaborating on GitHub**

Now, we'll dig into how it can make collaboration easier. Recall the version control workflow. Version control is the concept of tracking a file through its different states and allowing several people to work on the same file. This workflow can all happen on GitHub rather than using Git on a Command Line Interface. Since GitHub stores a project in the cloud, anyone can access it, making collaboration super easy. GitHub also benefits solo projects or projects we do independently without collaborating with others, as it provides a complete history of every project stage.

1. 1 iStock image credit: IkonStudio

**8. GitHub repo**

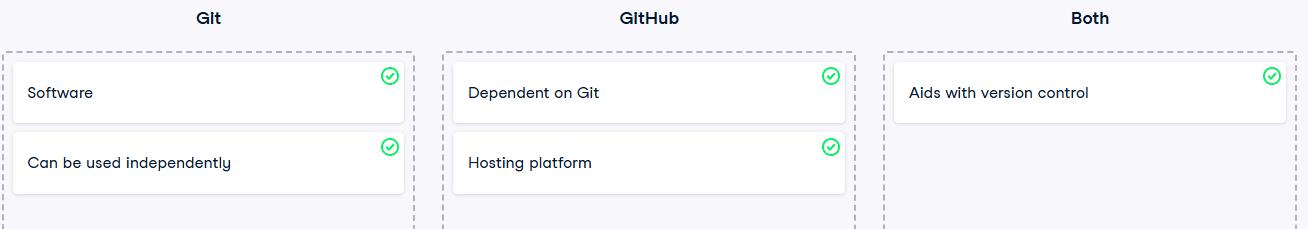
A project usually has many files, and when working with Git, there is a dot-git file that stores the history of a file. Together, these create a repository or repo. GitHub is built on top of Git, hence the name. The GitHub repo will contain all the files related to a particular project and a record of past versions of the files. Since this GitHub repo is stored on the internet, it is also known as a remote repo. This is because Git defines a local repo as saved on our local computer and a remote repo as saved on the internet, or cloud.

**9. GitHub website**

We've spoken a lot about Github; let's now see what it looks like. Here is an example of a profile overview. We can navigate to our Repositories by clicking

**10. GitHub website**

here. We may see a tab called Projects. This is a GitHub spreadsheet feature to manage the activity on our repos. We won't be looking at it in this course.



**Setting up a repo**

Now that we've solidified our GitHub knowledge, we'll learn how to set up a GitHub repo.

**2. Create a new repository**

We have two options that both lead us to the first step. In the profile view, we can either click the plus sign in the top-right to see the dropdown menu and select New repository,

or, in our repositories view, we can click the green New button. Both take us to the Create a new repository page. Here, we can import a repo or use a template from a previous repo. As we are starting from scratch, we will take the third option to create a new repo.

The first step is to name our repo. Our project will use data on soccer results from top-level men's teams in the UK, Italy, and Spain to look at goal trends. We'll name it soccer analysis. We can also include a short description of our project, which is helpful if we expect to collaborate with others.

**Public repository**

We have a few more decisions to make before we can create our repo. This will be an open project, so we will select the Public option. This means our project will be visible to anyone on the internet, but we can still control who can make changes to our projects.

**Repository files**

Nearly there! We need to initialize the repo with some files: a README file and a dot-gitignore file.

It is standard practice for all repo's to include a README file, these act as a guide for the project and are helpful for collaboration as they provide details on what the project is about and how it can be used.

We'll also add a dot-gitignore file. This is like the blocked phone numbers list on the phone. If we don't want someone to call us, we add their number to the block list! The gitignore file blocks or ignores specific files from being committed or saved into our repo. These files usually contain confidential information or are system files. GitHub offers several templates. We'll use Python.

**Repository files**

We'll also select None for the license, which means default copyright laws apply.

We're ready to create our first public repo by hitting the big green button!

**Navigating a repo**

Here is a view of our new repo! We can see the README and dot-gitignore files are part of the project already.

We are in the Code section of the repo. This is where all the folders and files are visible.

Notice the About section on the right. This is the description we added when creating the repo.

We can also edit it by clicking on the icon that looks like a gear.

The following section is called Issues. Issues are where we track tasks or problems and also communicate with others. We don't have any issues at the moment.

The next section is called Pull Requests. Pull requests or PR for short, is a request to make a change to the project. Think of it like a suggestion box. A PR will show the suggested changes and compare them to the project's current version. We can view the suggestion and decide if we want to accept it. We don't have any PRs at the moment, either!

Before practicing, let's look at the Settings. Here we have various options to make changes to our repo, including changing the name and access permissions. We'll explore access options later on. For now, we only need to know where to find these settings.

**Creating a README**

Fantastic work. We’ve created a repo! We’re now going to focus on the README file, reviewing how to edit it and what it should include.

**Edit the README**

We are in the default view of a repo, which is the Code section.

Regardless of the number of files we have, the README will always show after the list of files. It is a markdown file, as seen with the dot-md extension.

As we scroll down, we see a pencil icon in the top-right corner of the README rendering. Clicking on this will open the file for editing.

**Markdown**

The view of the README file will change to show the markdown syntax.

**Edit vs. Preview**

We can toggle between the Edit and Preview views to see if we have used the markdown syntax correctly. Let’s look at some fundamentals of markdown syntax we’ll likely need to use regularly.

**Headings**

First, we’ll look at headings. We can add headings by including one or several hashtags together. One hashtag is the biggest title heading available,

down to six hashtags stringed together, which is the smallest available in markdown.

Note that specific headers also add a line to the file.

**Text formatting**

We can make our text italic or bold by adding one or two asterisks on either side of our text.

**Hyperlinks**

We can include hyperlinks with the use of square brackets and parentheses. The square brackets will contain the text we want to be visible and clickable, while the parentheses will contain the desired URL.

This renders the link clickable in the preview tab and the file.

**Images**

Images use a similar format to the hyperlinks but with an added exclamation mark at the front. The text in the square brackets is alternative text, which is the text that appears in place of an image if it fails to load and is used by screen readers. It is important for accessibility. We can drag and drop an image, and GitHub will automatically render the code.

**Writing a README**

Now that we know how to style our README, let’s look at what it should include. README files need to be descriptive to allow anyone to understand our project, the contents of our repository, and how to use it. It is the first file a person will see, and as a data professional, it is a crucial skill to communicate a project to others. Consider it to be the instruction manual for our repository.

**README fundamentals**

Some essential items are the project title, a description of the technology used and why, a description of the process used to answer the question and why, and a table of contents.

**README extras**

Other characteristics of a good README are if it includes details of how the project came about and the motivation behind it, any limitations or challenges encountered, a recap of the problem it intends to solve, and what its intended use is. If we are sourcing information from elsewhere, it’s also essential to include the necessary credits.