Protocol

Network protocols are a set of rules, conventions, and data structures that dictate how devices exchange data across networks.

Network protocols control all aspects of network communication from sending and receiving messages, to formatting files for different types of messages.

Similar to the way that speaking the same language simplifies communication between two people, network protocols make it possible for devices to interact with each other because of predetermined rules built into devices’ software and hardware.

A network protocol will specify, for example:

* The format of data packets.
* The addressing system.
* Error-checking procedures use

HTTP Protocol

HTTP stands for Hypertext Transfer Protocol. HTTP is the underlying protocol used by the Web and this protocol defines how messages are formatted and transmitted. It also determines what actions Web servers and browsers should take in response to various commands.

For example, when you enter a URL in your browser, this sends an HTTP command to the Web server asking it to fetch and transmit the requested Web page.

The HTTP request contains different types of methods to pick out the desired action from the server.

The most common methods are:

* **GET**: to retrieve data from the server.
* **POST**: to submit data to the server.
* **PUT**: to update data that’s already on the server.
* **DELETE**: to delete data from the server.

# What Are the Advantages of APIs?

"There is no information exchange without an API. There is no digital transformation without an API. The Open API is the real revolution!"

* **Automation:** With APIs, computers rather than people can manage the work. Using APIs, agencies can update workflows to make them faster and more productive.
* **Application:** APIs can access application components. The delivery of services and information is more flexible.
* **More scope:** Through the use of an API an application layer can be created which can also be used to distribute information and services to new users. These APIs can be customized to create personalized user experiences.
* **Unlimited access to Data:** An API allows all public information that is generated by the government for general use to be accessible to all citizens without any exclusivity.
* **Efficiency:** When access is provided to an API, the generated content can be published automatically and is available for each channel. It makes it easier to share and distribute.
* **Integration:** APIs make it easier to embed content from any site or application. This ensures smoother information flow and an integrated user experience.
* **Personalization:** Thanks to APIs any user or company can personalize the content and services they use the most.
* **Adaptation:** Needs evolve and APIs allow you to anticipate changes. When working with this technology data transfer is easily supported and the information is examined more closely.

Basically, APIs are developer-friendly, easily accessible, and incredibly flexible.

# REST API

REST (Representational State Transfer) is a very popular web API architecture. REST API is an API that must have certain architectural constraints or principles including:

* **Client-server architecture:** The interface is separated from the backend and data storage. This allows for flexibility and for different components to evolve independently.
* **Statelessness:** No client context is stored on the server between requests.
* **Cacheability:** Clients can cache responses, so a REST API response must explicitly state whether it can be cached or not.
* **Layered system:** The API will work whether it is communicating directly with a server or through an intermediary, such as a load balancer.

The REST architecture uses the original instructions of the HTTP protocol.

**Rule # 1:** The URL must operate as a resource identifier.  
**Rule # 2:** HTTP verbs must function as the identifiers of operations.  
**Rule # 3:** HTTP responses must operate as representations of resources.  
**Rule # 4:** Links are like a relationship between resources.  
**Rule # 5:** A parameter, like an authentication token, must exist.

# HTTP Methods

HTTP methods define a set of query methods that indicate the action that we want to perform on the indicated resource. Although they have names, these methods are often called HTTP verbs. Each of them implements a different semantic, but some common functionalities can be shared by different methods.

**GET:** The GET method requests a representation of the indicated resource. GET requests should only be used to retrieve data.  
**HEAD:** The HEAD method requests a response identical to a GET request but the body of the response is omitted (we only have the header).  
**POST:** The POST method is used to send an entity to the indicated resource. This usually results in a change of state or causes side effects on the server.  
**PUT:** The PUT method replaces all current representations of the resource targeted by the content of the request.  
**DELETE:** The DELETE method deletes the specified resource.  
**CONNECT:** The CONNECT method builds a tunnel to the server identified by the target resource.  
**OPTIONS:** The OPTIONS method is used to describe the communication options with the targeted resource.  
**TRACE:** The TRACE method performs a round trip test message by following the path of the targeted resource.  
**PATCH:** The PATCH method is used to apply partial modifications to a resource.

# Definition of a Database

Databases are systematic collections of data.  
Databases support the storage and manipulation of data.  
Databases simplify data management.

Let's take Facebook as an example. A social media platform needs to store, manipulate, and present data to its members. It also needs to provide data to the whole network of friends including their activities, their messages, their likes, and much more. If we stored all this data in files, it would be next to impossible to utilize it.

Fortunately, this is where databases come into play.

Today, companies have to manage large volumes of data while also ensuring data integrity, security, and accuracy.

Spreadsheets are great tools for making calculations and inputting data. But if there’s a requirement to store substantial amounts of data such as customers’ information, employee records, or business inventory, a more efficient data management tool could be the better option.  
In today’s digital age, substituting spreadsheets with databases is a sure-fire way for business growth.

# Types of Databases:

There are many different types of databases. The best database for a specific organization depends on the organization’s purposes, plans, and projects.

These are the database types:

* Relational databases
* Object-oriented databases
* Distributed databases
* Data warehouses
* NoSQL databases
* Graph databases
* OLTP databases

# Database History

The history of databases dates back to the 1960s with the appearance of network databases and hierarchical databases. The next era, in the 1980s, saw the appearance of object-oriented databases. Today, the most used databases are SQL, NoSQL and cloud databases.

In this skill, we’ll talk more about SQL and NoSQL databases.

# What Is SQL?

SQL stands for "Structured Query Language".

It's a language that allows communication with databases to properly manage its data.

### **Is SQL a Programming Language?**

* Yes, SQL is a language. It offers looping, logic directives, variables, and so on. However, it’s not a language in the same sense as Java or C++.
* SQL may not be a language compared to Java or C#, but it is a language nonetheless.
* According to W3Schools: “SQL is a standard language for storing, manipulating and retrieving data in databases.”

# How is SQL used?

While an application might be programmed in a language like Python, PHP, or Ruby, databases are not designed to recognize these languages. Historically, databases recognize only SQL (although this has changed significantly in recent years).  
Similar to programming languages, SQL has its own markup. This makes it a priority for a programmer to learn SQL markup before using it effectively.  
Besides markup, another feature unique to database programming is the concept of tables. A database may be represented as a number of tables. Each table has its own number of columns and rows and represents a set of data.

# Example of SQL Database Management

Imagine a library. We could create a database that stores data about students in the GoMyCode community. In this case, we would need only one table and this table would allow us to store all the information we need.  
There are a few frequently used SQL commands you should be familiar with. When working with databases, a programmer might write commands such as:

* **CREATE DATABASE:** to create a database.
* **CREATE TABLE:** to create tables.
* **SELECT:** to find/extract data from a database.
* **UPDATE:** to make adjustments and edit data.
* **DELETE:** to delete data.

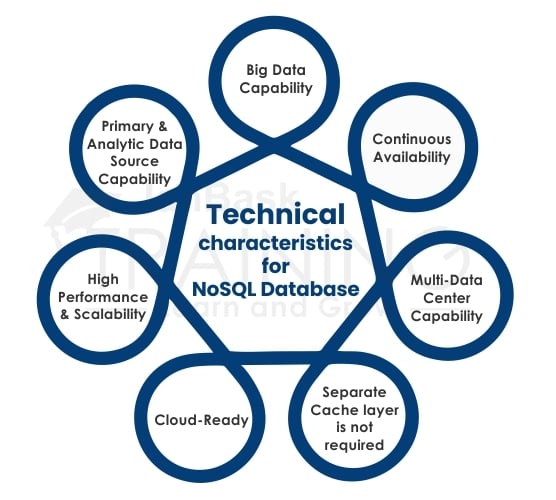
These are just the most common commands. The more complicated the database is, the more commands you as the programmer will need to use.

# What Is NoSQL?

A NoSQL, which stands for “not only SQL,” originally referring to non SQL or non relational, is a database that provides a mechanism for the storage and retrieval of data.

NoSQL is an approach to database design that provides flexible schemas for the storage and retrieval of data beyond the traditional table structures found in relational databases.  
While NoSQL databases have existed for many years, it has recently become more popular in the era of cloud, big data, and high-volume web and mobile applications. They are chosen today for their attributes around scale, performance, and ease of use.

The most common types of NoSQL databases are key-value, document, column, and graph databases.



# Examples of NoSQL Databases

Many NoSQL databases were designed by technology companies like Google, Amazon, Yahoo, and Facebook to provide more effective ways to store content or process data for huge websites. Some of the most popular NoSQL databases include the following:

* **Apache CouchDB:**  
  An open-source, JSON document-based database that uses JavaScript as its query language.
* **Apache Cassandra:**  
  An open-source, wide-column store database designed to manage large amounts of data across multiple servers and clustering that spans multiple data centers.
* **MongoDB:**  
  An open-source document-based database that uses JSON-like documents and schema. It is the database component of the MERN stack.
* **Redis:**  
  A powerful in-memory key-value store used for session caching, message queues, and other specific applications.
* **Elasticsearch:**  
  A document-based database that includes a full-text search engine.

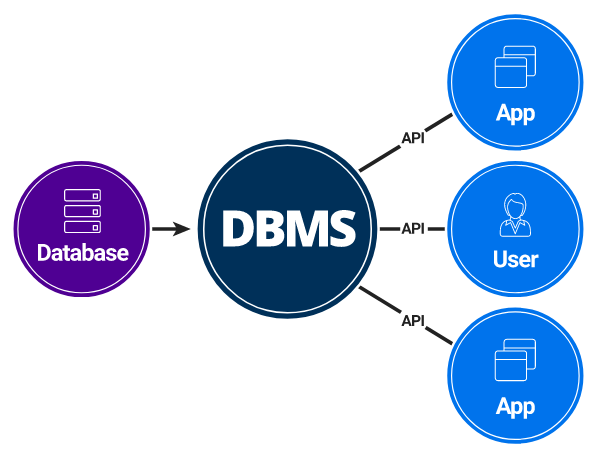
# What’s the Difference Between SQL and NoSQL?

The fundamental difference between SQL and NoSQL is not that complicated. Each has a different philosophy for how data should be stored and retrieved.

1. SQL databases are relational, NoSQL are non-relational.
2. SQL databases use structured query language and have a predefined schema. NoSQL databases have dynamic schemas for unstructured data.
3. SQL databases are vertically scalable, NoSQL databases are horizontally scalable.
4. SQL databases are table based, while NoSQL databases are document, key-value, graph, wide-column stores.
5. SQL databases are better for multi-row transactions, NoSQL are better for unstructured data like documents or JSON.

# Database Management System

A database typically requires a comprehensive database software program known as a Database Management System (DBMS).  
A DBMS serves as an interface between the database and its end users or programs. It allows users to retrieve, update, and manage how the information is organized and optimized. A DBMS also facilitates oversight and control of databases, enabling a variety of administrative operations such as performance monitoring, tuning, backup, and recovery.



## **Using DBMS Allows Us to:**

* Securely store a large volume of data for long periods of time.
* Access data efficiently.
* Control access to data by multiple users at the same time. One user's actions should not affect the other.
* Create new databases as well as specify their schemas.
* Support a data model.
* Query and modify the database’s data.
* Have data redundancy.
* Have more flexibility than the use of files.
* Have data consistency and integrity.

## **The Most Known DBMS:**

* **MySQL:**  
  A free and open source DBMS. It is probably the best known DBMS. We will be using it in this section.
* **PostgreSQL:**  
  A free and open source like MySQL but it has more functionality and it is a little less known.
* **SQLite:**  
  A free and open source. It’s lightweight but extremely limited in functionality.
* **Oracle:**  
  A paid database used by large companies. It is undoubtedly one of the most complete DBMSs.
* **Microsoft SQL Server:**  
  Microsoft's DBMS. Microsoft markets at least a dozen different editions of its Microsoft SQL Server. They’re aimed at different audiences and for workloads ranging from small single-machine applications to large Internet-facing applications that have many concurrent users.

# What Is Web Development?

Web development is the process of building websites and applications for the Internet or for a private network known as an Intranet.

Web development is not only concerned with the design of a website. It’s also about the coding and programming that powers the website’s functionality.

From the simplest, most static web pages to social media platforms and applications and from E-commerce websites to content management systems (CMS), all the tools we use daily via the Internet have been built by web developers.

Web development can be broken down into three layers:

* Client-side coding (front-end)
* Server-side coding (back-end)
* Database technology

# What Defines a Web Developer?

A web developer is by nature an interactive and creative artist. She/he is someone that is driven by a deep desire to create things. A web developer’s canvas is a user’s web browser.

Similar to a curious child who takes pleasure in making toys by joining LEGO blocks and then experiences a similar joy by taking them apart to see how they’re made, a web developer’s job is to use the basic building blocks of the Web (like HTML, CSS, and JavaScript) to create something complex like a web page.

If it seems too complex, no need to sweat it!  
We’ll break down these concepts shortly.

# Types of Web Developers

**Front-end Developer:**  
Client-side scripting, or front-end development, refers to everything that the end user experiences directly. Client-side code is executed on a web browser and directly relates to what people see when they visit a website. Things like layout, fonts, colors, menus, and contact forms are all designed by the front-end.

**Back-end Developer:**  
Server-side scripting, or back-end development, is all about what goes on behind the scenes. The back-end is essentially the part of a website that the user doesn’t see. It is responsible for storing and organizing data and ensuring that everything on the client-side runs smoothly. It does this by communicating with the front-end. Whenever something happens on the client-side, like a user fills out a form, the browser sends a request to the server-side. The server-side “responds” with relevant information in the form of front-end code that the browser can then interpret and display.

**Full Stack Developer:**  
A full stack web developer is a person who can develop both client and server software.  
In addition to mastering HTML and CSS, he/she also knows how to:

* Program a web page (like using JavaScript, jQuery, React, Angular, or Vue)
* Program a server (like using PHP, ASP, Python, or Node)
* Program a database (like using SQL, SQLite, or MongoDB)

# The Skills of a Web Developer

What does it take to become a web developer?

Mainly, it requires the mastery of these three languages: HTML, CSS and JavaScript.

They are the three pillars of web development, which we’ll be learning about over the next few weeks.

Collectively, these three pillars are what make a website function, by defining the content to be displayed (HTML), instructing a browser how to display that content (CSS), and making the content interactive (JavaScript).

A web developer is well-versed in these three technologies. They can read other people’s code and make changes to it. They can find and debug errors that are called “bugs”(shortcomings in existing code).

A web developer might, at times, work on a new project (a new website) from scratch, or may have to work on an existing website and make it better. A typical day in the life of a web developer involves fixing bugs, developing new features (enhancements) and web pages, and working with other developers to discuss ways to solve problems.

Don’t be overwhelmed by all of these details. You’ll soon see that they’re all very connected, and learning one of these languages automatically makes you better at a few others!

# Creating a Website

There are two ways to create a website:

**CMS** (Content Management System) is an easy-to-use system that allows you to control and manage the content within your website without necessarily having technical skills.

You can easily add and delete images, links, presentation links, and other content on your website.

**Hard Coding** requires specific skills and knowledge of programming languages. That is the complex way to create a website and that is what web developers do.  
In hard coding, you have the freedom to change what you want and build a custom website that completely satisfies the needs of the user, the employer, and the client.

# What Is a CMS?

A CMS is software that helps users create, manage, and modify content on a website without the need for technical knowledge.

In simpler terms, a CMS is a tool that helps you build a website without needing to write all the code from scratch or even know how to code at all.

Instead of building your own system for creating web pages, storing images, and other functions, the CMS handles all the basic infrastructure tasks for you so that you can focus on more forward-facing parts of your website.

Besides websites, you can also find CMSs for other functions like document management.

# The Most Known CMS Platform

The most known CMS Platforms are WordPress, Wix, Squarespace, Joomla!, Shopify, Drupal, Blogger, Prestashop, Magento, and Bitrix.

In the following section, we will talk about the two most common ones:

Wordpress has an incredibly clear administration console and an abundance of menus. However, its configuration possibilities can put off some beginner users.

Experienced users can take advantage of all the possibilities of WordPress and transform a blog into a real showcase site, a portfolio or even an e-commerce site. Its versatility is a serious advantage compared to its competitors. There is also a large community of users and developers allowing you to find a multitude of resources (Template, module, plugin) to optimize and personalize your website.

# Prestashop

[Prestashop](https://www.prestashop.com/en) is an open-source CMS that allows you to create an online ecommerce store. The CMS can be modified, by adding modules available on the Prestashop platform, or developed artistically.

* It is a preconceived and modular website. It is made up of two distinct parts:
  + The front office, or the front interface, is the part that’s visible to Internet users on the web.
  + The back office, also called the administration interface, is the part that’s only visible to the site administrator. This allows you to manage the content of the site.

# What Kinds of Websites Can You Build with Content Management Systems?

Nowadays, most content management systems are pretty flexible. While there are a few that focus on a specific use – like Magento and eCommerce – most of the popular content management systems can be used to create essentially any type of website.

For example, you can use WordPress to power:

* Static websites
* Blogs
* E-commerce stores
* Forums
* Social networks
* Online courses
* Membership sites

Portfolios To build a website with a CMS, the proper steps to take are as follows:

1. Pick a Platform: Prior to launching a website, choose the CMS that best suits your web development needs and final expectations.
2. Install the CMS: The next step is to install the platform you’ve chosen. Generally, the installation process is easy, intuitive, and takes only a couple of minutes.
3. Choose and install the plugins: Almost all of the reputable CMSs require plugin installation, so it makes sense to do that right from the start. Plugins add functionality to your project. However, if you don’t plan on launching a full-featured complicated project, you can continue without them.
4. Customize the design: As soon as you are done with plugin installation, it’s a priority to pick a suitable theme and customize your website design. Almost all CMS come with integrated templates, but you can also come across multiple variants on the web.
5. Upload content: When you are done with web design customization, you can start uploading the content. Most templates contain pre-designed content that is customized for its specific purpose. You can replace the content with your own info (images, text, media files etc.) to get a unique, personalized project.
6. Go online:The final step is website publication.

# Create a Website with code

## **Setting Up Your Developer Environment**

For this course, we’ll need three tools that every web developer uses:

1. A text editor to write code.
2. A web browser to preview what we’re building.
3. A version control and Repository Manager (Git & Github).

While you can use any web browser or text editor, we recommend using Google Chrome and VSCode for this course.

* **Google Chrome** can be downloaded from <https://www.google.com/chrome/>
* **Visual Studio Code** can be downloaded from <https://code.visualstudio.com/>.

Like any good craftsperson, it’s important that you become familiar with your tools. In order to do so, we’ll have a look at what is at our disposal.

**Google Chrome :** is a web browser. You may or may not already be using Google Chrome for browsing the web. A web browser allows us to view web pages (similar to the one you’re reading this very text on). We’ll use Google Chrome to view the results after writing and making changes to our code.  
**Visual Studio Code :** is a popular code editor. A code editor is a very aptly named software; it helps us edit text. It is similar to the default text editor that comes preinstalled on Mac or Windows, but has additional features like code highlighting to enhance our coding and debugging experience.

# Code Editor (IDE)

If you’re going to do any kind of development, you’re going to spend a lot of time in front of a text editor. There are many coding [languages](https://levelup.gitconnected.com/web-development-languages-36241b046a81) on the web, but let’s focus on the big three (HTML, CSS & JavaScript) and setup our environment to manipulate them.

Visual Studio Code (or VSCode) is the most used text editor.

VSCode provides a ton of plugins that make a developer’s life as easy as possible. When you’re using it, you have the ability to make it fit your preferences by installing the right plugins.

The plugins that you’re using have a huge impact on your productivity and the way you work.

# VSCode Extensions

Here are some examples of what you can achieve with an Extension API:

## **Prettier**

Prettier is one of the best plugins for developers who need to follow a well-laid set of rules when developing. It’s a robust opinionated code formatter that allows developers to format their codes in a structured way.  
Prettier works together with JavaScript, HTML, CSS, Markdown, and other modern tools and allows you to properly format your code. [Link](https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode)

## **Themes**

Since you’ll be looking at your editor every day, why not make it as pretty as possible? There are tons of customization plugins that change the color scheme and the icons in the sidebar. Some popular themes that are available for free are One Monokai, One Dark Pro, and Material Icon. [link](https://marketplace.visualstudio.com/items?itemName=PKief.material-icon-theme)

## **Bracket Pair Colorizer**

This extension allows matching brackets to be identified with colours. The user can define which characters to match, and which colours to use. [Link](https://marketplace.visualstudio.com/items?itemName=CoenraadS.bracket-pair-colorizer)

Organizing Your Work

One of the fundamentals of web development is organization. In order to achieve that, here is the folder creation step-by-step that we’ll use during our session:

1. Create a folder named “**GoMyCode**” on your **Desktop**.
2. Create a new folder in **GoMyCode** folder and name it whatever you want.
3. Create a new folder inside that one and name it "**res**".
4. Create a new folder inside that one and name it "**css**".
5. Create a new folder inside that one and name it "**js**".
6. Create a new text file, name it "**index**", and change its extension to "**.html**".
7. Inside the "**css**" folder, create a new text file, name it "style", and change its extension to "**.css**".
8. Inside the "**js**" folder, create a new text file, name it "main" and change its extension to "**.js**".

## What you should know by now :

# Recap

So in the previous chapters, we have learned a few basic concepts such as:

* The difference between the Web and the Internet.
* How does the Web work?
* Understanding what protocols, servers, and API are
* Knowing what a database is.
* Learning the meaning of web development.
* The different ways to create a website (with CMS and with code).

Getting Started With Git

We can imagine that web design is like a building and Git is one of the many important pillars that web design is built on.  
Our primary objectives right now are to:

* Understand what Git is
* Learn how to use the basic Git commands
* Discover how to configure Git
* Understand what Github is and how to work with remote repositories
* A version control system is a software tool that helps developer teams track and manage source code changes over time.
* As you edit and add to your code, you tell the version control system to take a snapshot of your files or save a checkpoint of your progress.  
  The version control system saves that snapshot permanently so you can recall it later on if you need it.
* Without a VCS, you’re tempted to keep multiple copies of code on your computer. This is dangerous because it’s easy for a file to get corrupted or deleted meaning you’re susceptible to losing months or even years worth of work.
* Version control systems solve that problem by presenting you with all versions of your code with a clear history of the changes made. This allows you to go back to older versions of the code and try a different approach.

Git is a free and open source VCS designed to handle everything from small to very large projects with speed and efficiency.

In plain English, Git is a tool that allows developers to track versions of their code over time. It does this by creating "snapshots" of the current state of the code base whenever you tell it to.

Git is essential when collaborating with other developers to ensure that there are no code conflicts between them and that previous "snapshots" of the code can be revisited if necessary.

For example, if you are coding and you accidentally break or crash the app, you’ve just lost all your progress and you’re forced to start from scratch. However, it’s easier and safer if you're using Git and you can simply roll back to a previous version of the code.

Access the wanted folder using the terminal (prompt cmd).

To initialize a repository, we only have to run:

git init

Adding Files for Git to Track

At this point, we do not have any files for Git to track.  
We need to add files specifically to Git in order to tell Git to track them. We add files using the “add” command.  
After running git add . Git will add all the repository files to an intermediate area called the **staging area**. We can also add what we want simply by running git add myFileName

# File Status

Git sees every file in your working copy as one of three things:

**Tracked:**  
A file which has been previously staged or committed.  
**Untracked:**  
A file which has not been staged or committed.  
**Ignored:**  
A file which Git has been explicitly told to ignore.

Ignored files are usually built artifacts and machine generated files that can be derived from your repository source or should otherwise not be committed.

**PS**: If we have a file that we do not want to track, we just put it inside the .gitignore file.

# Configuration Settings

The .gitconfig file contains a list of configurations that affects the behavior of git commands. So in order to manipulate it, we use the command git config <configuration>. Therefore, to change the name and email used by Git to identify the user, we run the following command: (PS. change YOUR.NAME and YOUR.EMAIL with your own values.)

# Git Aliases

Oftentimes, you’ll be finding yourself typing git commands over and over again. For example, git add, git init and git status are commands you will be repeatedly using.  
The smart thing to do in this case is use a shortcut, or an alias. Aliases help make your Git experience simpler, easier, and more familiar.  
To create a temporary alias, which will last as long as your terminal session is open, you can type:

git config alias.KEYBOARD\_SHORTCUT COMMAND

So if we wanted to type git st and have it function like git status, we would type git config alias.st status. Now we can type git st and have the same output as if we typed git status.

If you would like your alias to be a part of your global configuration, add the --global command after git config. For example, to alias git i to git init globally, you would type git config --global alias.i init.

# What Is GitHub?

GitHub is a web-based Git repository hosting service.  
Simply put, it is a tool that enables collaboration by hosting shared Git repositories that teams of developers can all contribute to.  
While GitHub uses Git, the functionality it provides is **entirely different** from Git. So try to make it stick in your mind that Git and GitHub are not the same thing.

In short, Git is a Version Control System while GitHub is an online platform for hosting and sharing code, text files, and more complex file formats.

# Why Use GitHub?

GitHub provides a great way for you to store your code in a safe and remote location (in case something happens to your local machine). It's a fantastic way to collaborate with other developers both privately and publicly.

Many large open source projects are hosted on GitHub, which makes it very easy to examine shared code both on GitHub and on local machines.

In the next couple of chapters we will learn how to move code from our local repository to a remote repository on GitHub using the push command, as well as retrieve code from a remote repository on GitHub using the pull command.

We'll also learn about GitHub specific concepts like forking and pull requests.

Want to get a sense for how popular GitHub is in the development community? Here are some projects you may have heard of that are hosted there:

* [Angular](https://github.com/angular/angular)
* [React](https://github.com/facebook/react)
* [Ruby on Rails](https://github.com/rails/rails)
* [Twitter Bootstrap](https://github.com/twbs/bootstrap)
* [Node.js](https://github.com/nodejs/node)
* [JQuery](https://github.com/jquery/jquery)
* [Homebrew](https://github.com/Homebrew/brew)

# Fork

Now that we know how to push code to GitHub, let's explore one of GitHub’s important features: forking.

When collaborating with others you’re unable to push directly to the original repository (Imagine if you could play with one of the world’s largest open source projects and repositories… that would be insane).

So the logical thing to do is make a copy of someone else's remote repository and make sure it is under our username so that we can push code to it. This ensures that our experimenting with the repository doesn’t affect the original version.

To practice forking, head over to any repo on [github](https://github.com/) and on the top right you will see a button with the text Fork. Click on this button and you will have a copy of the repository under your name!

Remember, "forking" is strictly a **GitHub** feature and it is not by any means related to Git. It is simply a way to make your own copy of a repository on your account where you have permission to push your code to GitHub.

# Clone

Once you have applied “forking” on the repository, you need to select it (the remote one you just made) and download the code on your local computer (i.e. make a local repository). Instead of making a folder and going through the whole git init process and adding a remote, you can conveniently use the git clone command, which accepts a link to the repository and downloads it into a folder (with everything already set up!).

In order to do this, click on the button clone and that will provide us with a remote URL. Then just copy the URL.  
In the terminal, just run git clone THE\_COPYED\_ADDRESS.

# Pull Request

Now let's say you are collaborating with an organization on GitHub (where you forked the repository from) and you would like to merge your changes with the original repo that you forked (remember you can't just push to it, because you do not have permission to do so). You can issue a pull request and the person who can grant permission can either merge or reject it.

To do this, click on "New pull request” and then click on "Create pull request". You should then be able to go to the original repository and see your pull request or "PR".

Git and github

1. Create a folder called learn\_git.
2. Cd (change directory) into the learn\_git folder.
3. Create a file called third.txt.
4. Initialize an empty git repository.
5. Add third.txt to the staging area.
6. Commit with the message "adding third.txt".
7. Check out your commit with git log.
8. Create another file called fourth.txt.
9. Add fourth.txt to the staging area.
10. Commit with the message "adding fourth.txt"
11. Remove the third.txt file.
12. Add this change to the staging area. (Using the command "git add . "
13. Commit with the message "removing third.txt".
14. Check out your commits using git log.
15. Change your global settings to core.pager=cat - you can read more about that [here](https://git-scm.com/book/en/v2/Customizing-Git-Git-Configuration).
16. Write the appropriate command to list all the global configurations for git on your machine.
17. You can type git config --global to find out how to do this.