What Is GitHub?

There are a lot of people asking "What is GitHub?" online. In fact, there are over 615 million responses to that query on Google. It's a popular name in the tech space, even if you're not a developer (yet).
person using a computer to learn what github is used for
But GitHub can be confusing if you've never used it. Is it a cloud storage site? A social networking site for programmers? A code-sharing site? The answer to all these questions is yes, and more.

Download Now: 50 Code Templates [Free Snippets]
Today, GitHub is one of the most popular resources for developers to share code and work on projects together. It's free, easy to use, and has become central in the movement toward open-source software.

If this all sounds like something you should know, read on — we'll explain what GitHub is, what it's used for, and how to get started. So keep reading, or click one of the links below to jump to the section you're looking for:

What is GitHub?
How does GitHub work?
What is Git?
What is GitHub used for?
How to Get Started Using GitHub
What is GitHub?
GitHub is an online software development platform. It's used for storing, tracking, and collaborating on software projects.

It makes it easy for developers to share code files and collaborate with fellow developers on open-source projects. GitHub also serves as a social networking site where developers can openly network, collaborate, and pitch their work.

Since its founding in 2008, GitHub has acquired millions of users and established itself as a go-to platform for collaborative software projects. This free service comes with several helpful features for sharing code and working with others in real time.

On top of its code-related functions, GitHub encourages users to build a personal profile and brand for themselves. You can visit anyone's profile and see what projects they own and contribute to. This makes GitHub a type of social network for programmers and fosters a collaborative approach to software and website development.

How does GitHub work?
GitHub users create accounts, upload files, and create coding projects. But the real work of GitHub happens when users begin to collaborate.

While anyone can code independently, teams of people build most development projects. Sometimes these teams are all in one place at once time, but more often they work asynchronously. There are many challenges to creating collaborative projects with distributed teams. GitHub makes this process much simpler in a few different ways.

What is GitHub platform home

First, all the code and documentation are in one place. This limits issues with access for anyone who wants to contribute to a project. Each repository also contains instructions and other details to help outline project goals and rules.

Next, coding is more creative and abstract than most non-technical people think it is. For example, say two devs are working on different pieces of code. These two pieces of code should work together. But sometimes one piece of code can make the other code fail. Or a piece of code can have an unexpected impact on how the other code works.

GitHub solves these problems by showing how both files will change the main branch. It catches these errors before pushing changes, making the coding process more efficient.

GitHub also makes it easier to track changes and go back to previous versions of a project. To explain this, we'll need to understand the technology that GitHub is based on, Git, and talk about version control.

What is Git?
Git is open-source version control software, used for managing and tracking file revisions. You can use Git with any file type, but it's most often used for tracking code files.

What is Git platform home

Git is the most widely used version control system in software development, and GitHub leverages this technology for its service, hence its name.

What is version control?
You've likely worked with some form of version control before. For example, Google Docs (as well as other Google Workspace tools) has a "Version History" tool where you can view changes to the document at different points in time. Microsoft Office has something similar with its "Track Changes" feature. Or, you might prefer saving multiple copies of a file and labeling them "v1", "v2", etc.

Advanced version control is necessary for software projects, especially collaborative ones.

Why is version control helpful for coding?

When building software, developers frequently and simultaneously update the code to add features and fix bugs. It wouldn't make sense to make these changes to the source code directly, since any issues would affect users. Instead, developers work with their own copies of the code, then — after the code has been thoroughly tested — add this code to the main codebase.

GitHub version control example: Bug fixes

That seems all well and good. But, with multiple contributors, things can get messy quickly. That's especially true if there's no way to combine everyone's contributions into one unified codebase or see who contributed what changes. Each change must also be tracked and stored. This is helpful when something breaks and the developers need to backtrack and restore a previous version.

That's what Git is for. When a developer wants to make a change to a piece of software, they:

Download their copy of the source code from its central storage location (called a repository) to their local system
Make modifications safely to their copy
Merge their revised copy back with the source files in the repository
Add comments explaining the changes
Git tracks all of these modifications and stores previous versions as backups. And GitHub makes it easier to see when devs make changes as a group, not just specific individual changes. This is helpful when troubleshooting complex projects.

What is GitHub used for?
GitHub allows software developers and engineers to create remote, public-facing repositories on the cloud for free. A repository, or "repo" for short, is a coding project's files and the revision history for each file.

Once you've set up a repository on GitHub, you can copy it to your device, add and modify files locally, then "push" your changes back to the repository where your changes display for the public.

So, why would you prefer GitHub over developing with a private repository? Here are the key reasons why GitHub is such a big player:

Enhanced Collaboration
The single biggest selling point of GitHub is its set of project collaboration features, including version control and access control.

To illustrate what's possible with GitHub, imagine this scenario. You want to code an online game, and you enlist your friend to help you.

You create a repository on GitHub that stores all the files, including current and past versions, then give your friend collaborator access to this repo as well.

What is GitHub used for example: Repository

Image Source

You decide to work on the main gameplay and screens while your partner tackles the game's menu and settings screens. In order for you both to push your changes at the same time without interfering with the other's work, you create a branch — a separate development area — where your teammate can build out their screens. Meanwhile, you continue work in your branch.

What is GitHub used for example: Branch

Once your friend finishes their work, they can make a pull request asking to combine their work with yours. If you

approve, you can merge your branches, and thus your code.

What is GitHub used for example: Pull request

Now, say another developer sees your game's repository and has an idea to add multiplayer mode. They can fork — or make their own copy of — your repository, then add their new features.

What is GitHub used for example: Fork

Once finished, they can submit a pull request to you. If you approve, this forked repository merges with yours, and your game is now multiplayer.

Note: Anyone can fork a public repository, but it's up to the repository owners to accept or reject pull requests.

No software is perfect on the first go, which is why your GitHub repo also has an Issues section for listing to-dos and reporting problems with your game, discussing them, and marking them resolved.

What is GitHub used for example: Issues

To address these issues, you might look back at a file's changelog to see when and where things went wrong.

To sum up, GitHub provides a centralized space where several, dozens, or even thousands of developers can seamlessly contribute to a project, without worrying about overriding anyone's work or losing track of changes.

Easy File Management
Using GitHub means you're not limited to one device or environment.

GitHub adds a sleek graphical user interface (GUI) layer on top of Git. On its own, Git operates through the command line (a computer's text-based interface). Developers know how to use the command line, but for many, it's not always the most efficient way to interact with files.

GitHub's interface provides a clean and user-friendly way to perform Git actions as well as view file history. This is more convenient for developers and more accessible for beginners getting the hang of Git.

Another benefit that makes GitHub more accessible is its cloud-based infrastructure. A GitHub user may access their repository from any location and any device, download the repository, and push their changes.

Social Networking
Any GitHub user knows the platform is more than just a place to work on code. All GitHub users have profiles to display their projects, contributions, and activity on the site, and can see anyone's public-facing profile and repositories.

What is GitHub used for example: Social networking

GitHub's social network is critical to its success, as it encourages developers to explore and contribute to open-source projects of all kinds. Previously, aspiring collaborators would have to personally contact project owners asking for permission to contribute.

With GitHub, it's as easy as forking a project and then making a pull request. A project owner can then vet someone's profile and past contributions before accepting their request.

a user profile on github

Image Source

GitHub also serves as a way to showcase projects for employers, acting as a portfolio of sorts. For example, recruiters often use GitHub to scout talent, since prospects' code is available for anyone to review.

Open-Source Projects
Thanks to the benefits we've learned about, GitHub has fueled a surge of open-source collaboration, leading to the creation of many widely used software technologies. From CSS frameworks to data visualization libraries to a game you might spend too much time playing, a lot of impressive feats wouldn't be around without open GitHub repositories.

What is GitHub used for example: Open-source software

As has been a trend with Web 2.0, GitHub has also opened up software development to anyone who wants to learn programming, fostering an engaged, innovative, and productive community.

Private Repositories
Sure, that's all great, but how does GitHub make money if everything is free and open-source?

The answer is that GitHub provides paid services as well, including private repositories. On a paid plan, teams can collaborate on GitHub while keeping their code behind closed virtual doors. GitHub also offers enterprise solutions that equip organizations with internal collaboration tools.

If you want to learn more about what GitHub is used for or how it works, this video is great for beginners:

How to Get Started Using GitHub
Sold on GitHub? Here's how to get started. A quick note: You should get comfortable using the command line before working with GitHub, as Git uses the terminal as its interface.

1. Install GIT.
Install the latest version of Git on your device. You'll need Git installed to work with your GitHub repository. There are various ways to do this, so follow the recommendations on the Git website. The Git software is free.

2. Sign up for GitHub.
After installing Git, go to GitHub's website and create an account with your email address.

3. Start a repository.
Once your GitHub account is set up, you'll be taken to your dashboard. To start your first repository, click Create repository. This lets you keep all of your code for your new GitHub project in one place.

4. Name your project.
On the Create a new repository screen, enter your repository name and an optional description (you can change both later).

5. Add project details.
On the same screen, add a README file (a text file that describes your project, and a best practice in development), a .gitignore (which removes irrelevant files like .DS_Store), and a license for your project.

These details make it easier for collaborators to understand your project and any guidelines you'd like them to follow.

the repository setup option screen in github

6. Create your repository.
Click Create repository. You'll be taken to your main repository page, which lists your files.

7. Create a local copy of your repository.
You'll now create a local copy of your GitHub repository (or in GitHub terms, "clone" your repository) where you'll edit your files and push your changes. On your main repository page, click the green Code button, then copy the HTTPS URL of your repository.

You already have a copy of your project on your hard drive, so why do you need to clone your repository? This is a best practice because it makes it easy to see when people added or removed files. It also makes it easier to fix merge conflicts.

a repository screen in github with the clone repository option open

8. Choose a directory.
Open your terminal and navigate to the directory you want to place your repository copy.

9. Paste your repository URL.
In the terminal, enter git clone. After this paste in the repository URL that you previously copied. Your command should look like this:

Git clone syntax example

10. Clone and check your copied repository.
Press Enter to clone the repository. You'll see a new file added to your local filesystem with your repository's name. If you open this file, you'll see it contains the files in your GitHub repository. These are copied versions of your repository's files that you can edit and then push back to your repository.

11. Create a new file in your new repository.
Let's end by creating a new file in your cloned repository, then pushing it to GitHub. In your local clone, create a new text file called hello.txt. In it, paste the text Hello, world! and save the file.