What is GitHub? When was it created? Why? By who? What similar platforms exist? Why would you use such a platform?

Answer. GitHub is a website and service. To understand GitHub, you must first have an understanding of Git. Git is an open-source version control system that was started by Linus Trovalds—the same person who created Linux. Git is a command-line tool, but the center around which all things involving Git revolve is the hub—GitHub.com—where developers store their projects and network with like-minded people. It is created on February 8, 2008. It is created by Tom Preston-Werner. There are some alternative to github are: GitLab, Bitbucket, FogBugz Kiln, Beanstalk, SourceForge, Apache Allura, AWS CodeCommit, GitKraken. Why to use github have many reasons. Because it’s a social network that has completely changed the way we work. Having started as a developer’s collaborative platform, GitHub is now the largest online storage space of collaborative works that exists in the world. Whether you’re interested in participating in this global mind meld or in researching this massive file dump of human knowledge, you need to be here. Before GitHub existed, major companies created their knowledge mainly in private. But when you access their GitHub accounts, you’re free to download, study, and build upon anything they add to the network. Thank famed software developer [Linus Torvalds](http://en.wikipedia.org/wiki/Linus_Torvalds) for Git, the software that runs at the heart of GitHub. (And while you’re at it, go ahead thank him for the Linux operating system, too.) Git is version control software, which means it manages changes to a project without overwriting any part of that project. And it’s not going away anytime soon, particularly since Torvalds and his fellow kernel developers employ Git to help develop the core kernel for Linux.

Why use something like Git? Say you and a coworker are both updating pages on the same website. You make your changes, save them, and upload them back to the website. So far, so good. The problem comes when your coworker is working on the same page as you at the same time. One of you is about to have your work overwritten and erased.

A [version control](http://git-scm.com/video/what-is-version-control) application like Git keeps that from happening. You and your coworker can each upload your revisions to the same page, and Git will save two copies. Later, you can merge your changes together without losing any work along the way. You can even revert to an earlier version at any time, because Git keeps a “snapshot” of every change ever made.

The problem with Git is that it’s so ancient that we have to use the command line—or Terminal if you’re a Mac user—in order to access it, typing in snippets of code like ‘90s hackers. This can be a difficult proposition for modern computer users. That’s where GitHub comes in.

GitHub makes Git easier to use in two ways. First, if you [download the GitHub software](http://github.com/) to your computer, it provides a visual interface to help you manage your version-controlled projects locally. Second, creating an account on GitHub.com brings your version-controlled projects to the Web, and ties in social network features for good measure.

Define the following terms in context of Git:

1. Repository:

The purpose of **Git** is to manage a project, or a set of files, as they change over time.**Git** stores this information in a data structure called a **repository**. A **git repository**contains, among other things, the following: A set of commit objects. A set of references to commit objects, called heads.

1. Commit:

**git commit** "records changes to the repository" while **git** push "updates remote refs along with associated objects". So the first one is used in connection with your local repository, while the latter one is used to interact with a remote repository.

1. Push:

basically **git** commit puts your changes into your local repo, while **git push** sends your changes to the remote location. Since **git** is a distributed version control system, the difference is that commit will commit changes to your local repository, whereas **push** will **push** changes up to a remote repo.

1. Branch:

A **branch** in **Git** is simply a lightweight movable pointer to one of these commits. The default **branch**name in **Git** is master. As you initially make commits, you're given a master **branch** that points to the last commit you made

1. Fork:

A **fork** is a copy of a repository. **Forking** a repository allows you to freely experiment with changes without affecting the original project. Most commonly, **forks** are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea.

1. Merge:

**Set** the commit message to be used for the **merge** commit. Commits, usually other branch heads, to **merge** into our branch.

1. Clone:

**git clone** is a **Git** command line utility which is used to target an existing repository and create a **clone**, or copy of the target repository. ... **Cloning** a local or remote repository. **Cloning** a bare repository.

1. Pull:

**git pull** runs **git fetch** with the given parameters and calls **git** merge to merge the retrieved branch heads into the current branch. ... <repository> should be the name of a remote repository as passed to **git**-**fetch**[1].

1. Pull request:

**Pull** requests let you tell others about changes you've pushed to a **GitHub** repository. Once a **pull request** is sent, interested parties can review the **set** of changes, discuss potential modifications, and even push follow-up commits if necessary.