Version Control System

A version control system (VCS) allows you to manage a collection of files and gives access to different versions of these files.

The VCS allows you to capture the content and structure of your files at a certain point in time. You can use the VCS to switching between these versions and you can work on different versions of these files in parallel. The different versions are stored in storage system which is typically called a *repository*

VCS are very good in tracking changes in text files. For example, you may track changes in HTML code or Java source code. If is also possible to use VCS for other file types but VCS are not that efficient to trace changes in binary files.

VCS are very good in tracking changes in text files. For example, you may track changes in HTML code or Java source code. If is also possible to use VCS for other file types but VCS are not that efficient to trace changes in binary files.

In a distributed version control system each user has a complete local copy of a repository on his individual computer. The user can copy an existing repository. This copying process is typically called *cloning* and the resulting repository can be referred to as a *clone*.

Every clone contains the full history of the collection of files and a cloned repository has the same functionality as the original repository.

Every repository can exchange versions of the files with other repositories by transporting these changes. This is typically done via a repository running on a server which is, unlike the local machine of a developer, always online. Typically, there is a central server for keeping a repository but each cloned repository is a full copy of this repository. The decision which of the copies is considered to be the central server repository is pure convention.

Introduction to GIT

Git is the leading distributed version control system.

Git originates from the Linux kernel development and was founded in 2005 by Linus Torvalds. Nowadays it is used by many popular open source projects, e.g., Visual Studio Code from Microsoft, Android from Google or the Eclipse developer teams, as well as many commercial organizations.

The core of Git was originally written in the programming language *C*, but Git has also been re-implemented in other languages, e.g., Java, Ruby and Python.

A Git repository manages a collection of files in a certain directory. A Git repository is file based, i.e., all versions of the managed files are stored on the file system.

A Git repository can be designed to be used on a server or for an user:

- *bare repositories* are supposed to be used on a server for sharing changes coming from different developers. Such repositories do not allow the user to modify locally files and to create new versions for the repository based on these modifications.
- *non-bare repositories* target the user. They allow you to create new changes through modification of files and to create new versions in the repository. This is the default type which is created if you do not specify any parameter during the clone operation.

A *local non-bare Git repository* is typically called *local repository*.

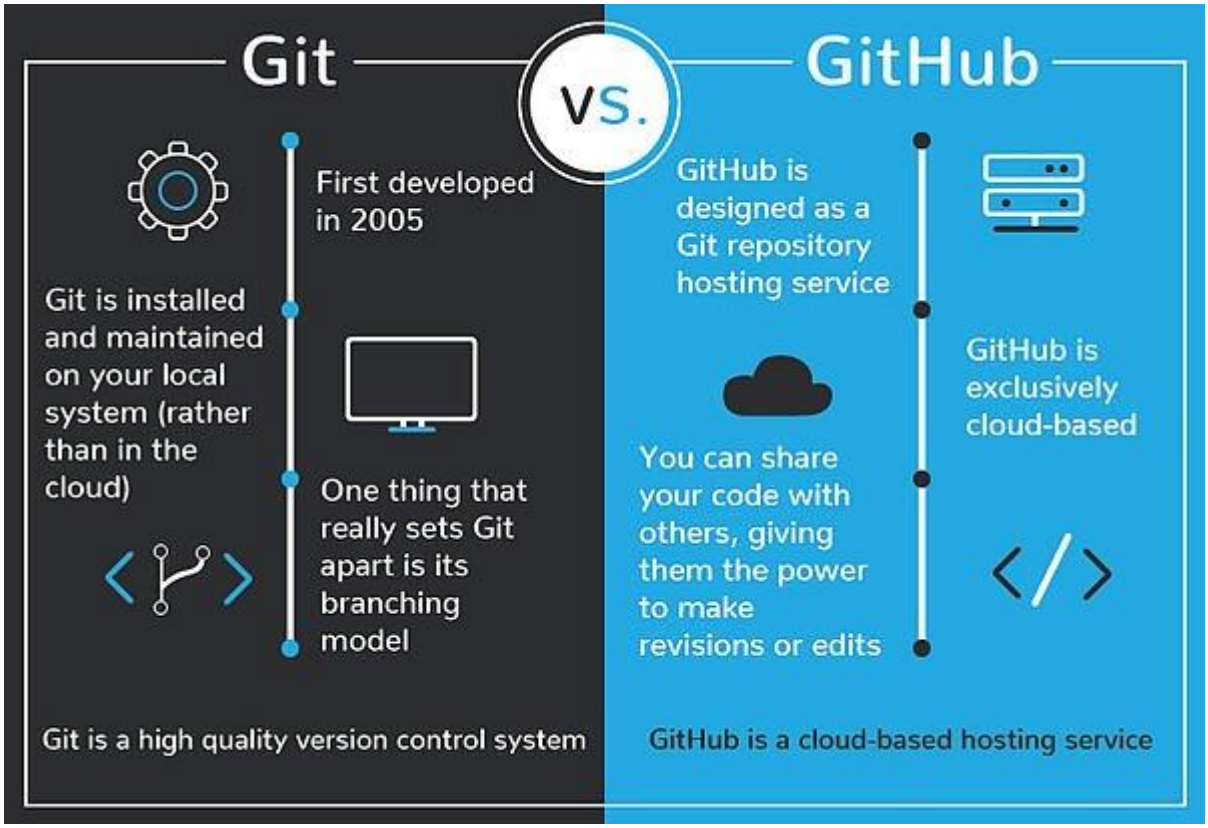
Git allows the user to synchronize the local repository with other (remote) repositories.

Users with sufficient authorization can send new versions of their local repository to the remote repositories via the *push* operation. They can also integrate changes from other repositories into their local repository via the *fetch* and *pull* operation.

Every local repository has a *working tree*. The files in the working tree may be new or based on a certain version from the repository. The user can change and create files or delete them.

After doing changes in the working tree, the user can capture new versions of the files in the Git repository. Or the user can restore files to a state already captured by Git

Difference between GIIT and Github



alternatives to github

10 Best GitHub Alternatives to Host Open Source Projects (tecmint.com)

Installing and customizing GIT

1.checking git version

git –version

2.

creating a local repository

2. git init

3.customizing git

git config user.email "peterone23four@gmail.com"

4.git config color.ui true

git config color.status auto

git config color.branch auto

setting default editor

git config core.editor notepad

pushing changes to remote repository

1.create a readme file in git local repository

```
echo 'TODO: Add contents for README' > README
```

2.check status using

status -s

```
g.it status -s
```

```
4.Add file for staging

git add .
```

```
5.commit file to be pushed

git commit
```

```
6.push file to repository

 adding a remote repository address

git remote set-url origin https://<username>:<access
token>@<repository address>

git push origin master
```

```
Cloning remote repository



git clone <remote repository address >
```