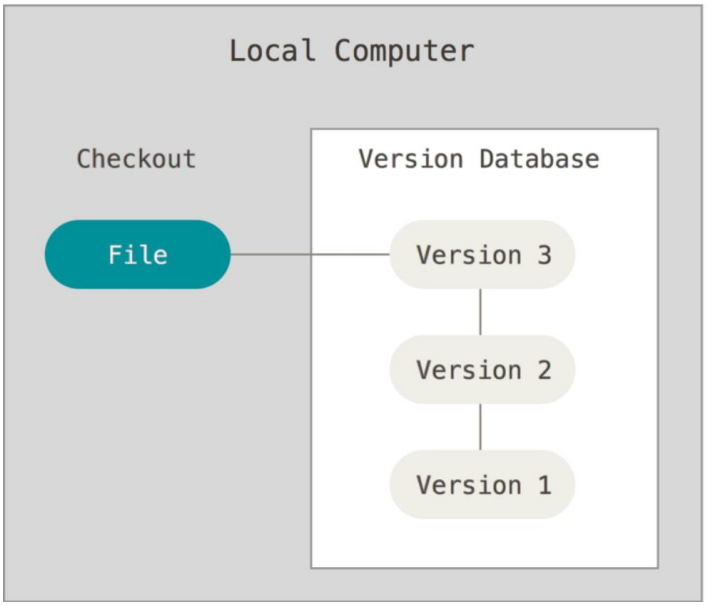
1. Version Control:

1) *Definition*:

Version control is a system that records to a file or set of files over time so the engineers are able to recall specific versions later. For example, in reality engineers are only to view the source code of certain file embedded in the computer, however, by using software of version control, *engineers can adjust and manage those files simultaneously.*

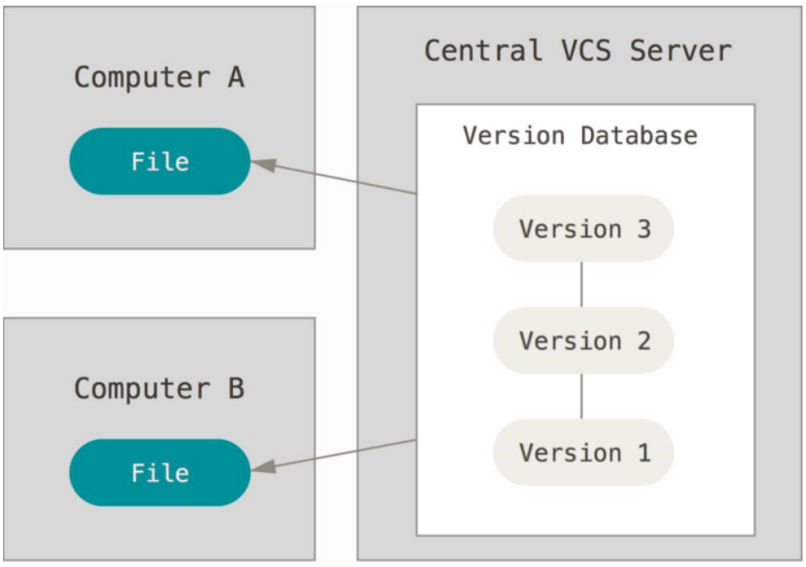
2) *Local Version Control systems*:

A lot of people intending to copy their valuable files into another directory, which seems extremely common due to its simplicity. However, *it might also be incredibly error-prone* because it is much easier to forget about which directory that those engineers put into and sometimes it might be accidentally write to the wrong file or copy over files that engineers do not mean to.



3) *Centralized Version Control Systems*:

The biggest demand for such system is that frequently engineers may confront with the challenge to *collaborate with other developers on the system*. By resolving such problem properly, Centralized Version Control Systems (CVCSs) were developed for such purpose. These systems have a sole sever containing all the versioned files and also a number of clients that check out files from that central place. For many years, this has been the standard for version control.



4) *Distributed Version Controls System*:
The meaning of Distributed Version Control Systems (DVCs, like Git, Mercurial, Bazaar or Darcs) is that clients do not only check out the latest snapshot of the files; rather, they fully mirror the repository, including its full history. Thus, if any severs are dead, and these systems were collaborating via the server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data.

2. Github and repository:

At macro-level, Github is a *website and cloud-based service* that helps developers store and manage their code, as well as trach and control changes to their codes. There are two main principles behind the exact definition of Github:

- Version Control (As the previous points illustrated)
- Git

For git is a *specific open-source* version control system created by Linus Torvalds in 2005. To be more specific, Git is a distributed version control system, meaning that the entire codebase and history is available on every developer's computer, which allows for relatively easing branching and merging processes.

Repository is usually used to organize a single project. Repositories can contain folders and files, images, videos, spreadsheets, and data sets. The most essential component of that is recommended by engineers is '*README*' or a file with information about the project. Moreover, steps of creating a personal repository is just as the followings illustrate:

1. *In the upper right corner, next to the avatar or identicon, click + and then select new repository.*

2. *Name the repository.*
3. *Write a short description.*
4. *Select initialize this repository with a README.*

3. **How to Git Clone:**

In general, one of the means to get Git Clone is via URLs, URLs contain information about the transport protocol, the address of the remote server and also the path to the repository. Relying on the transport protocol, some of this information might be occurring as absence. Moreover, Git may negatively support ssh, git, http, ftp, https, ftps and rync protocols. The following syntaxes may be used with them.

- ssh://[user@]host.xz[:port]/path/to/repo.git/
- git://host.xz[:port]/path/to/repo.git/
- http[s]://host.xz[:port]/path/to/repo.git/
- ftp[s]://host.xz[:port]/path/to/repo.git/
- rsync://host.xz/path/to/repo.git/

An alternative scp-like syntax may also be used with the ssh protocol:

- [user@]host.xz:path/to/repo.git/

The ssh and git protocols additionally support username expansion:

- ssh://[user@]host.xz[:port]/~[user]/path/to/repo.git/
- git://host.xz[:port]/~[user]/path/to/repo.git/
- [user@]host.xz:/~[user]/path/to/repo.git/

For local repositories, also supported by git negatively, the following syntaxes may be used:

- /path/to/repo.git/
- file:///path/to/repo.git/

Several examples:

- Clone from upstream:

```
$ git clone git://git.kernel.org/pub/scm/.../linux-2.6 my2.6 $ cd my2.6 $ make
```

- Make a local clone that borrows from the current directory, without checking things out:

```
$ git clone -l -s -n . ../copy $ cd ../copy $ git show-branch
```

- Clone from upstream while borrowing from an existing local directory:

```
$ git clone --reference my2.6 \          git://git.kernel.org/pub/scm/.../linux-2.7 \
my2.7 $ cd my2.7
```

- Create a bare repository to publish your changes to the public:

```
$ git clone --bare -l /home/proj/.git /pub/scm/proj.git
```

- Create a repository on the kernel.org machine that borrows from Linus:

```
$ git clone --bare -l -s /pub/scm/.../torvalds/linux-2.6.git \      /pub/scm/.../me/subsys-
2.6.git
```