**What is Git?**

**What is Git?**

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

Git is a distributed version control system for **tracking changes in source code** during software development. It is designed for coordinating work among programmers, but it can be used to track change.

Git has the functionality, performance, security and flexibility that most teams and individual developers need. Git is the most broadly adopted tool of its kind. This makes Git attractive for the following reasons. Vast numbers of developers already have Git experience and a significant proportion of college graduates may have experience with only Git.

just memorize these Shell commands and type them to sync up. if you get errors, save your work elsewhere, delete the Project and download a fresh copy.

### Brief History of Git

As with many great things in life, Git began with a bit of creative destruction and fiery controversy.

For most of the lifetime of the Linux kernel maintenance (1991–2002), changes to the software were passed around as patches and archived files. **In 2002**, the Linux kernel project began using a proprietary Version Control System called **BitKeeper.**

In 2005, the relationship between the community that developed the Linux kernel and the commercial company that developed BitKeeper broke down, and the tool’s free-of-charge status was revoked. This prompted the Linux development community (and in particular **Linus Torvalds**, the creator of Linux) to develop their own tool based on some of the lessons they learned while using BitKeeper. **That's how Git was borned**.

### More About Git

Git is an example of version control. Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.

It allows you to:

* Revert files to previous state,
* Revert entire project back to previous state,
* Compare changes over time,
* See who modified what? **And much more...**

It means if you screw things up or lose files, you can easily recover.

Use cases:

* Individual development,
* Collaborative development,
* Offline usage.

Why Git?

* Everything is local (full history tree available offline),
* Everything is fast,
* Snapshots, not diffs,
* It is distributed not centralized,
* Great for those who hate: CVS/SVN (earlier [version control systems](https://lms.clarusway.com/mod/lesson/view.php?id=642" \o "Version Control Systems)).

### What is a Version Control System (VCS)?

A **V**ersion **C**ontrol **S**ystem (also called Revision/Source Control System) is a software designed to record changes made to files over time so that you can recall specific versions later. It's like having an unlimited undo/redo feature under your belt.  **It allows you to revert selected files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.** It's a very powerful tool for your project.

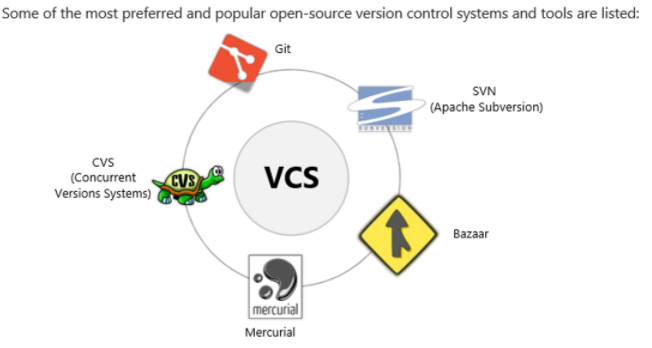
A VCS is a tool to track changes of source code and learning

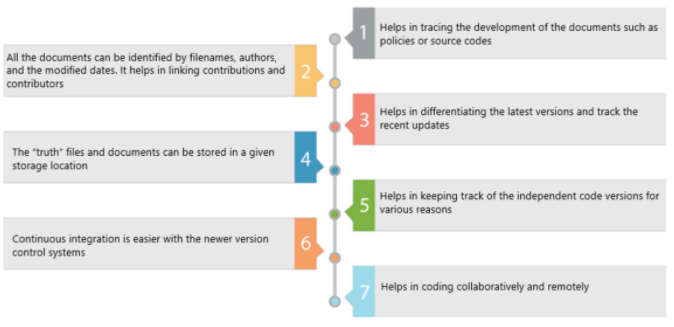
* What had changed
* When it changed
* Why it changed
* Who changed it

### Need of Version Control Systems

There are many reasons to use a **Version Control System (VCS)** in a system. Here are the reasons to use it.

* Files need to be stored somewhere.
* You can store them anywhere you like, but if you store them in a VCS, you never lose them.
* Every reported change (check-in) associated with a file are also available.
* All the previous versions of a file can be easily extracted/restored.



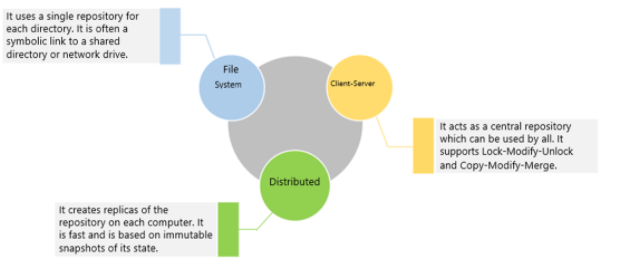


### Types of Version Control Systems

There are three types of version control systems:

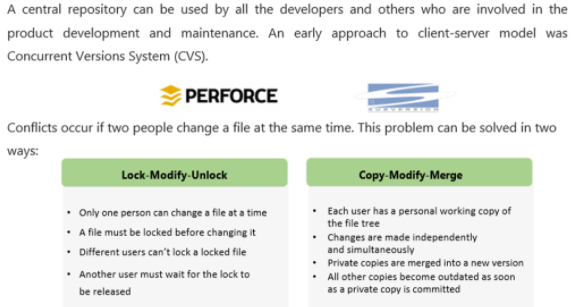
* **File-based,**
* **Client-server type,**
* **Distributed.**

File-based version control systems are obsolete and not much used. Distributed VCS's are more common on the market. Git is a distributed type of VCS.



### Client-Server Based Version Control Systems

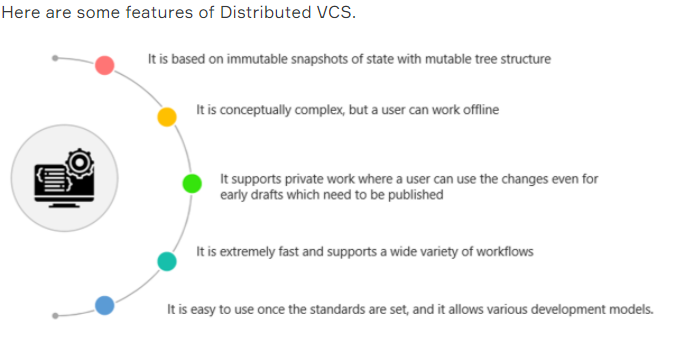
These are properties of Client-Server Based Version Control Systems.



### Distributed Version Control Systems

### \* En bilinenleri GIT ve Mercurial

Distributed version control systems create replicas of the repository on each computer. Every user has to work on a replica and can do so even being disconnected from the network. They are suited for large projects and independent developers who can work independently and commit the changes for merging.



### What is a Repository?

**A repository is a directory or storage space where your projects can live**. Sometimes it is shortened to “**repo**.” It can be local to a folder on your computer, or it can be a storage space on the cloud. You can keep code files, text files, image files, etc. inside a repository.

### Understanding How Git Manages Data

In a Git repository your file can reside in three main states: **Modified**, **Staged**, and **Committed**.

* **Modified** means that you have changed the file but have not committed it to your database (repo) yet.
* **Staged** means that you have marked a modified file in its current version to go into your next commit snapshot.
* **Committed** means that the data is safely stored in your local database.

This leads us to the three main sections of a Git project:

* The working tree,
* The staging area,
* The Git directory.

The working tree is a single checkout of one version of the project. These files are pulled out of the compressed database in the Git directory and placed on disk for you to use or modify.

The staging area is a file, generally contained in your Git directory, that stores information about what will go into your next commit. Its technical name in Git terminology is the “index”, but the phrase “staging area” works just as well.

The Git directory (.git) is where Git stores the metadata and object database for your project.

**Git nedir?**

* **Free and open source** verison control system
* Used for **tracking changes in source code during software development and coordinating work among programmers**.
* A distributed type of version control system

**Tarihçesi:**

* 1991-2002 linux geliştirme faaliyetleri yama ve arşiv dosyalarla yürütüldü,
* 2002’de Linux projesinde **Bitkeeper** adlı versiyon kontrol sistemi kullanılmaya başlandı, **2005’te** bu sitemin sahibi ile Linux topluluğu ilişkisi bozulunca, Linux ekibi kendi tool’unu geliştirdi.

**Sürüm kontrolü nedir?**

Sürüm kontrolü; daha sonra belirli sürümleri geri çağırabilmeniz için bir dosyada veya dosyalarda zaman içinde yapılan değişiklikleri kaydeden bir sistemdir. Şunları yapmanızı sağlar:

* Dosyaları önceki durumuna geri döndürmeye,
* Tüm projeyi önceki durumuna geri döndürmeye,
* Zaman içindeki değişiklikleri karşılaştırmaya
* Kimin değişiklik yaptığını görmeye,
* Ve daha fazlası...

Kullanım durumları:

* Kişisel Gelişim,
* İşbirlikçi geliştirme,
* Çevrimdışı kullanım.

