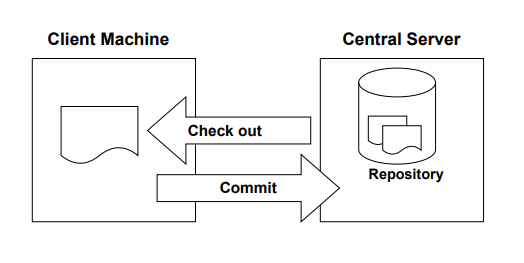
**Difference between a Centralised Version Control System (CVCS) and a Distributed Version Control System (DVCS)**

A version control system is a software that allows you to manage changes to assets (codebase, files) over a period of time. Centralised and Distributed are the two main types of version control systems. The fundamental difference between these two lies in how they −

* Manage the repositories
* Manage the content workflow

**Centralised Model**

Centralised version control systems follow a server-client model. The server holds a single central copy of the project along with a history of changes made to the code base over a period of time. The basic workflow to work with a centralized version control system is given below −



When a user wants to work with a file, they connect to the server via a client machine and download a local copy of the file. The user works on the copy that is available on the user’s local system and finally sends changes made to the file or codebase to the central server. It is necessary that the central server is always reachable to be able to perform source code management. Some common **centralized version control systems that are used include Concurrent Version Systems (CVS), Subversion (SVN), and Perforce**.

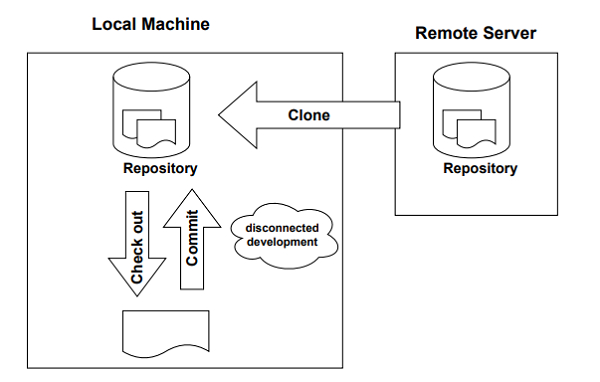
One of the main drawbacks of the centralized version control system is that the central server is a single point of failure. This means that if the central server fails, no operations can be performed on the repository as it resides on the server and hence changes made to the code cannot be tracked. Also, when working in larger teams, the central server and repository can become a bottleneck. This is where the Distributed Version Control Systems comes to rescue.

**Distributed Version Control System**

Unlike a Centralised Version Control System that is hierarchical, distributed version control systems follow a more egalitarian approach. Just like in a Centralized Version Control System, you checkout a copy of the repository, make changes, and check it back in. The main difference between the two types is that Centralized version control systems keep the history of changes on a central server. Everyone requests the latest version of the work and pushes the latest changes to the central server.

In case of a Distributed Version Control System, each collaborator will have the complete repository on the local machine including the complete revision history, commit information. The collaborator can push his version of the code or new change sets locally. This helps avoid failure due to a crash of the central versioning server. Once a feature or a group of changesets are ready, the collaborator can then push all of them to the central server at once. In other words, unlike centralized version control systems, it is not necessary to be online to change revisions or add changes to the work. Moreover, a collaborator can choose to share his code with selected people for feedback before finalizing and making his code available to the entire team.

The following diagram shows the operations that can be performed in a distributed version control system.



In Git’s terminology the central server is called a repository. The copy of the files downloaded by a collaborator from the repository is called a clone. A local repository is one that is copied from the server. A developer works on the clone. When all features are made and tested properly the developer can then sync the final codebase with the remote repository.

# Difference between Centralized Version Control and Distributed Version Control

## Centralized Version Control

**Centralized Version Control** is a version control system using server/client model and server contains all the history of source code.

## Distributed Version Control

**Distributed Version Control** is a version control where each client can have same copy of source code as server has and both server and client maintain history of source code.

Following are the important difference between Centralized Version Control and Distributed Version Control.

| **Sr. No.** | **Key** | **Centralized Version Control** | **Distributed Version Control** |
| --- | --- | --- | --- |
| 1 | Working | In CVS, a client need to get local copy of source from server, do the changes and commit those changes to central source on server. | In DVS, each client can have a local branch as well and have a complete history on it. Client need to push the changes to branch which will then be pushed to server repository. |
| 2 | Learning Curve | CVS systems are easy to learn and set up. | DVS systems are difficult for beginners. Multiple commands needs to be remembered. |
| 3 | Branches | Working on branches in difficult in CVS. Developer often faces merge conflicts. | Working on branches in easier in DVS. Developer faces lesser conflicts. |
| 4 | Offline Access | CVS system do not provide offline access. | DVC systems are workable offline as a client copies the entire repository on their local machine. |
| 5 | Speed | CVS is slower as every command need to communicate with server. | DVS is faster as mostly user deals with local copy without hitting server everytime. |
| 6 | Backup | If CVS Server is down, developers cannot work. | If DVS server is down, developer can work using their local copies. |