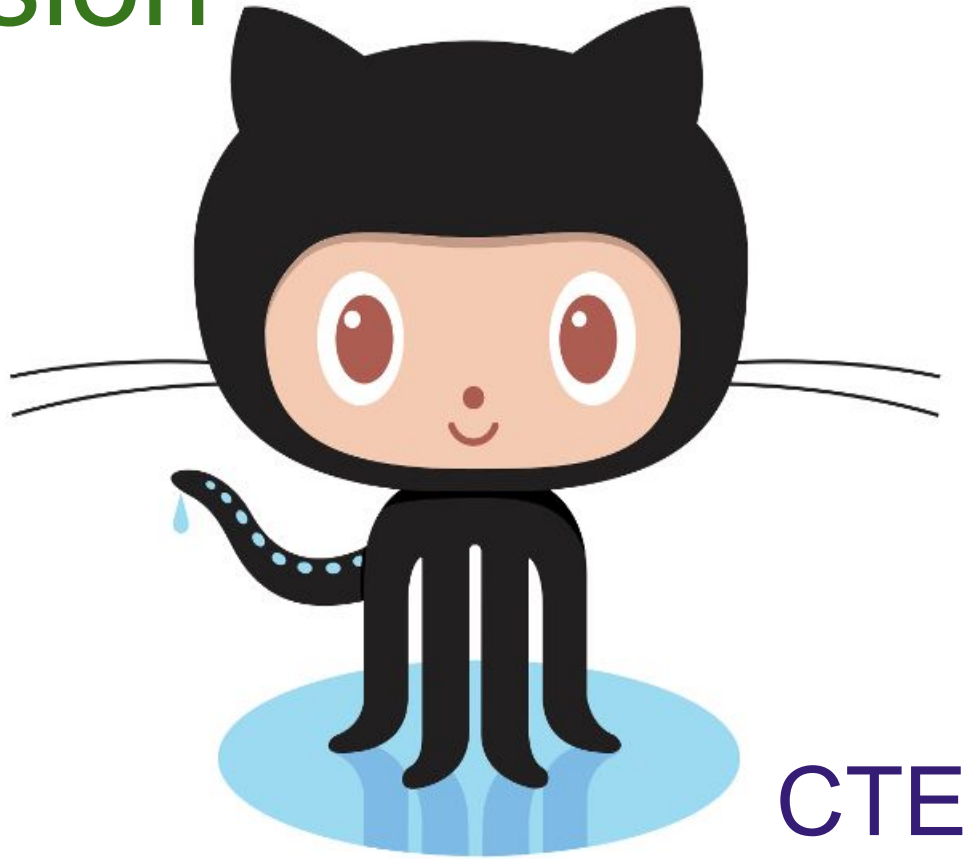
Git Session



CTE Python

Version Control System

What is a VCS ?

Software tools that help developers keep track of changes made to source code over time. It keeps track of these modifications in a special kind of database. If a mistake is made, developers can compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

Why do we need it ?

Developing software without using some sort of version control is risky, because even one bad change could potentially make the software corrupt/bug-ridden. It also enables developers to work faster because of improved communication amongst team members.

Why Git ?

There are many VCSs out there but mostly Git is preferred to the others. There are many reasons for this, but one of the main reasons is that it is a Distributed Version Control System.

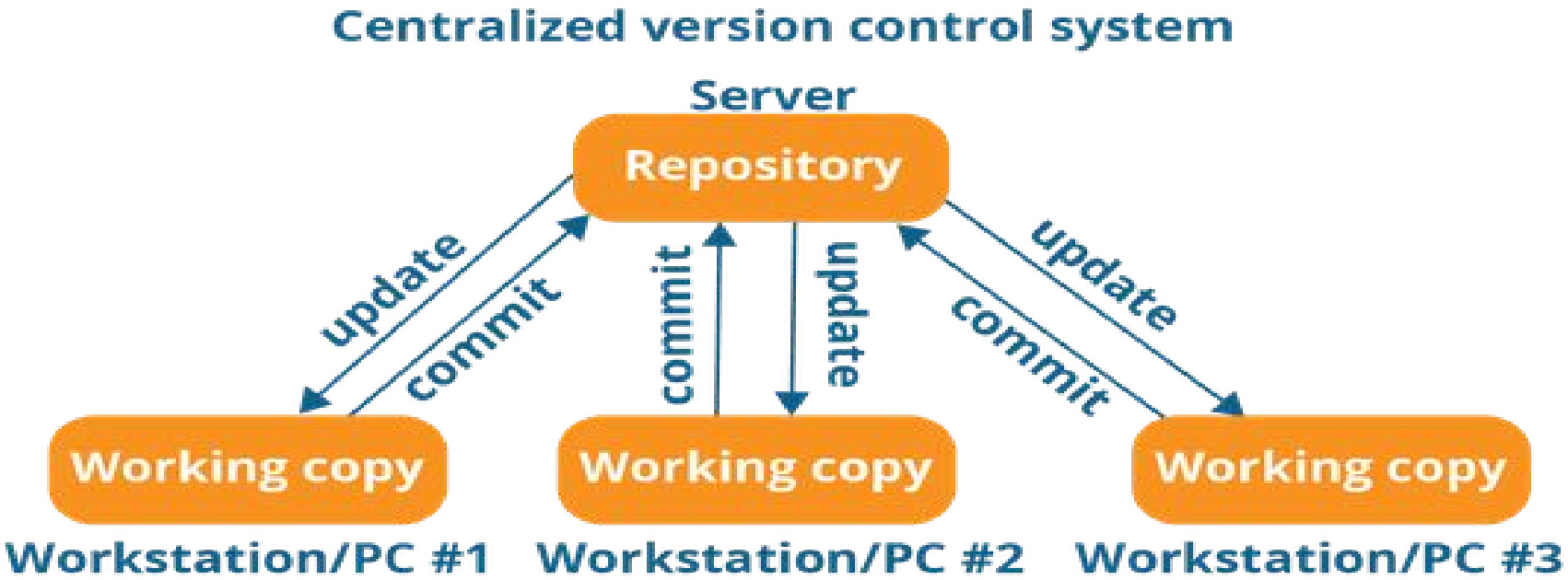
Let us understand the what is meant by the following terms :

- Centralized VCS
- Distributed VCS

and why a DVCS is better than a CVCS.

Centralized VCS

It uses a central server to store all files and enables team collaboration. Developers can directly access a central server and change files on the repository(collection of files and folders) on that server.



Drawbacks of CVCS

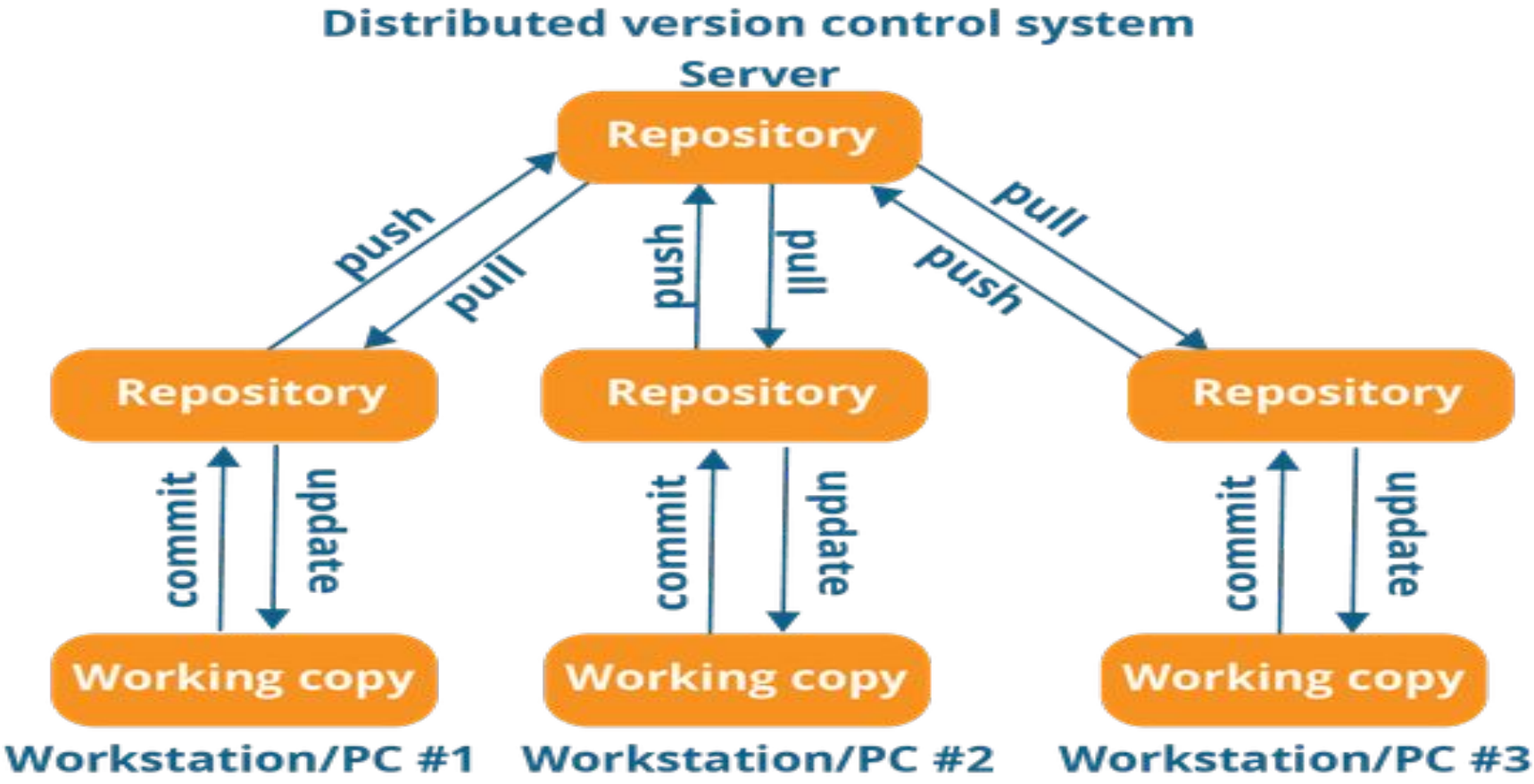
- It is not locally available, hence you need to be connected to network to perform any action.
- Since the project only exists on the central server, in the case of server storage gets corrupted, it will result in losing data of the entire project.
- Any unchecked changes to the project could result in software getting broken/bug-ridden, then a rollback to previous stable release has to be done.

Distributed VCS

They do not necessarily rely on a central server to store all the versions of a project file.

In DVCS, every developer has a local copy or “clone” of the main server repository (i.e. every contributor maintains a local repository of their own which contains all the files/folders present in the main repository).

Workflow for DVCS



DVCS for the win !

- All operations (except push & pull) are very fast because only hard drive needs to be accessed.
- Committing new changes can be done locally without affecting the main repository. When a developer is satisfied with his/her local changes all of them can be pushed at once.
- Every contributor has a copy of the repository, they can share changes with each other, if they wish to get feedback before changing the main repository.
- If data on the central server gets corrupted or lost, it can be easily recovered from any one of the contributor’s local repositories.

Basic Git Commands

- init - Create a local repository
- add - Adds file/s to the staging area
- commit - Records changes done locally into Git
- pull - Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
- push - Write local changes to remote branch of repository.

Now we are ready to move on to a live Git session !