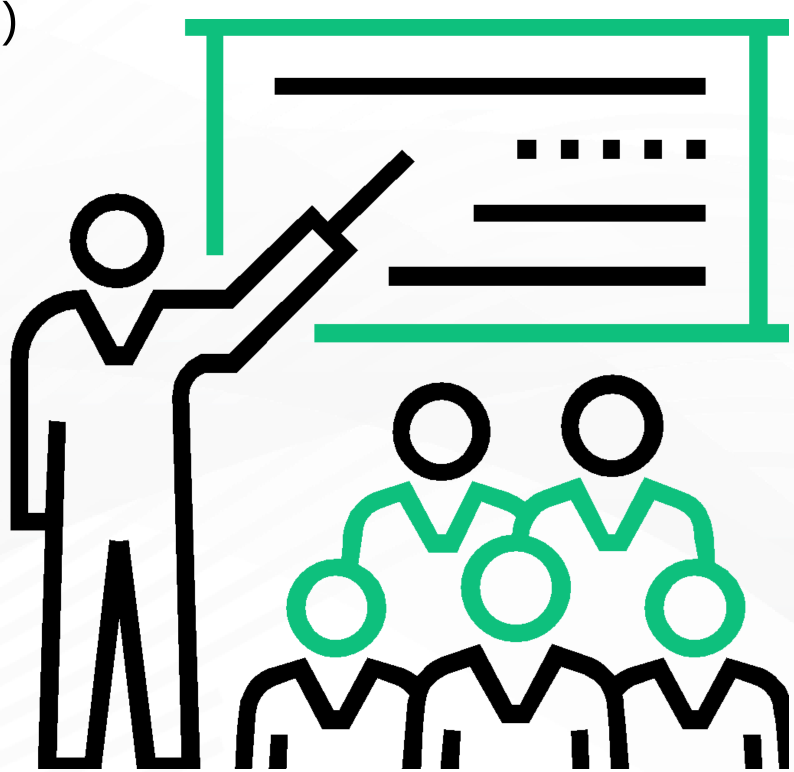
Version Control System vs Distributed Version Control System

Module Objectives

At the end of this module, you will be able to:

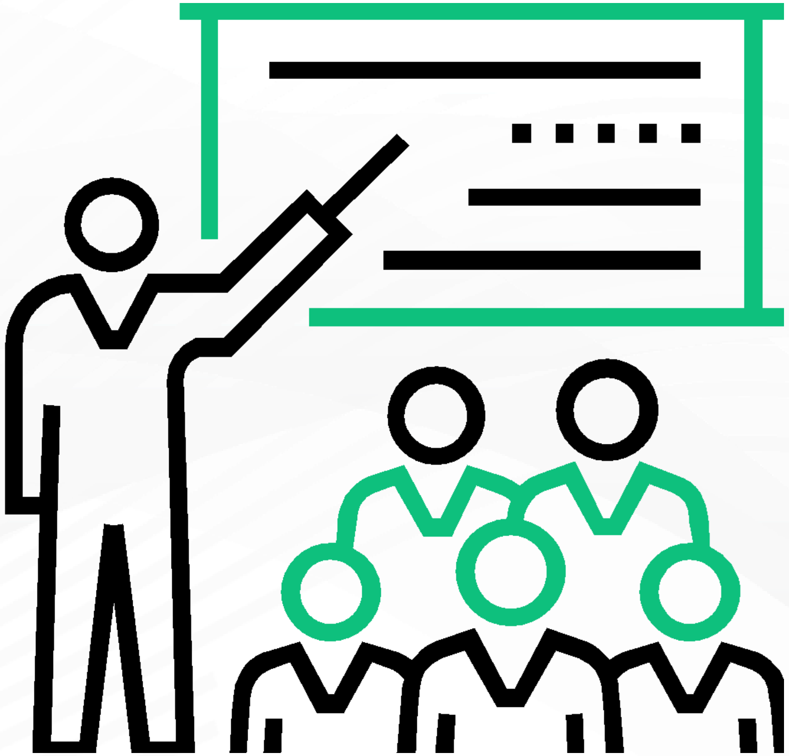
- Explain how local version control works
- Discuss the basics of Centralized Version Control Systems (CVCS)
- Describe the basics of Distributed Version Control Systems (DVCS)
- Enlist the advantages of DVCS, such as:
  - Private Workspace
  - Easier merging
  - Easy to scale horizontally
- List the disadvantages of DVCS
- Explain how CVCS and DVCS compare with each other
- Describe the working of the multiple repositories model
- Learn how to reset the local environment
- Identify how revert operation is used to undo the changes



Module Topics

Let us take a quick look at the topics that we will cover in this module:

1. Local repository
2. Advantages of distributed version control system
3. The multiple repositories model
4. Completely resetting local environment
5. Revert - canceling out changes

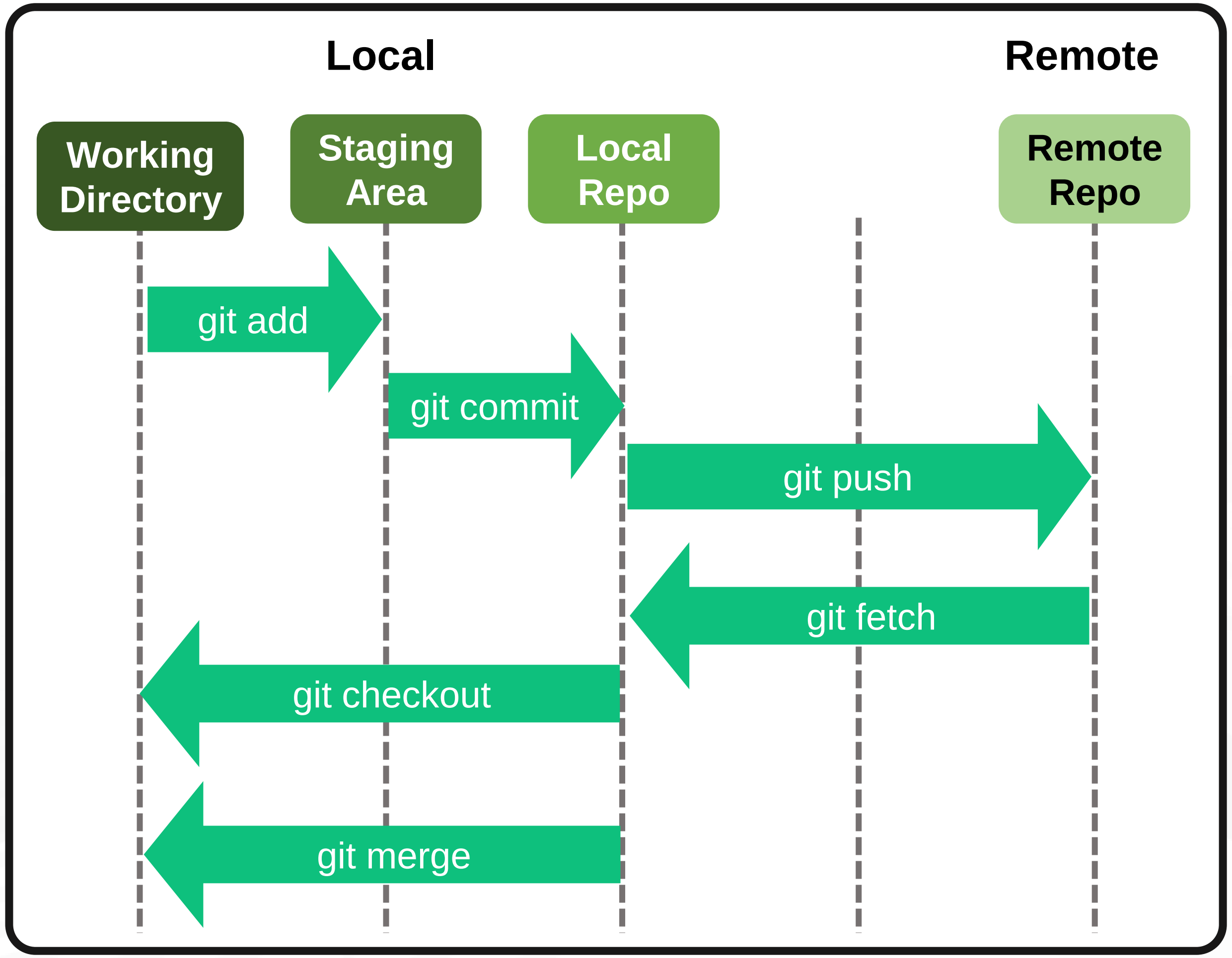


1.1 Local Repository

Following are the key details of the local repository:

- Local version control keeps track of the files within the local system.
- Local version control systems save a series of patches, but collaboration or branching is almost impossible with local repositories.
- A local repository, from a DVCS viewpoint is a collection of files which originate from a certain version of the repository.
- The collection of files is called the working tree or the checkout.

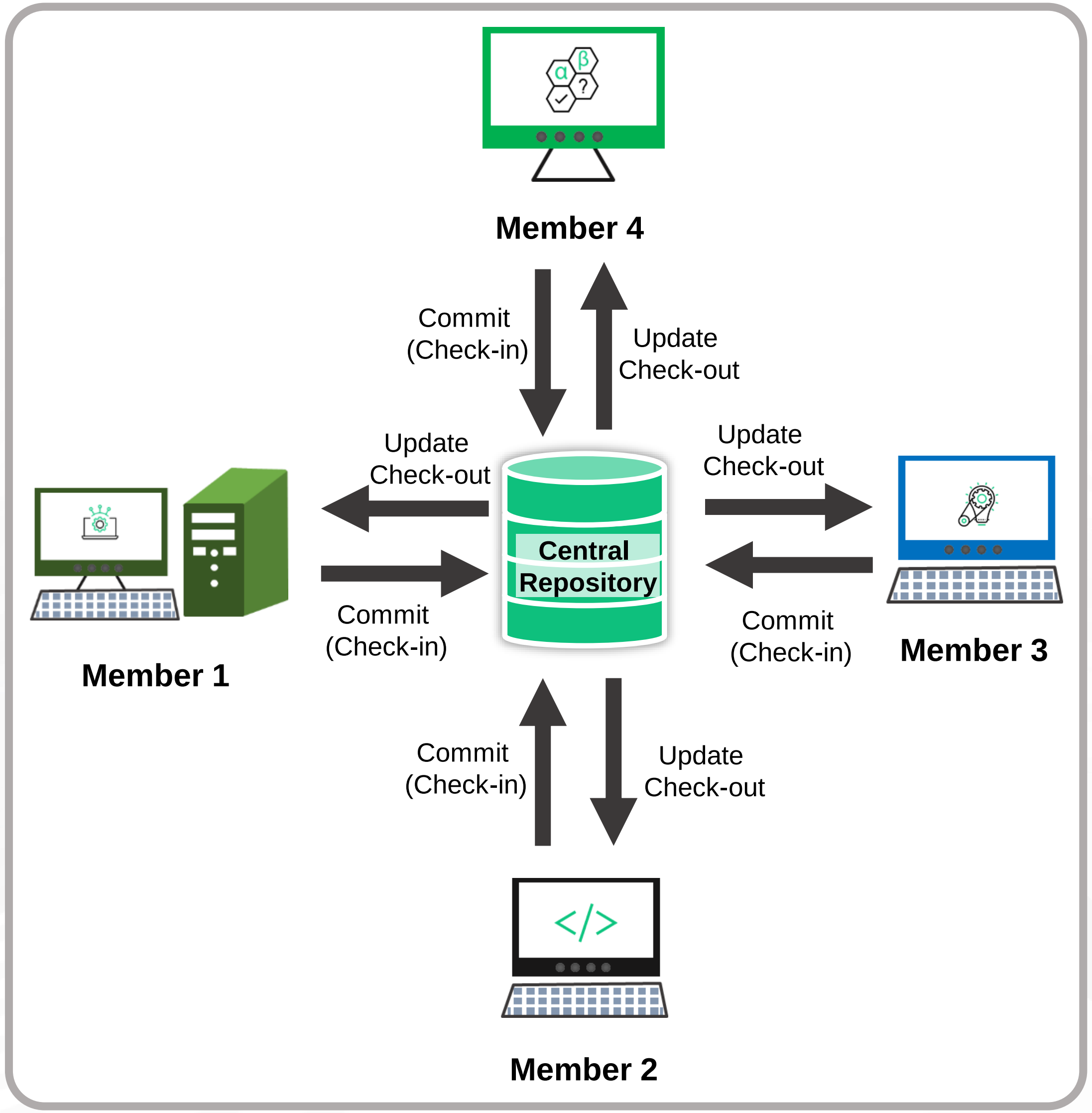
The picture explains how the changes are made to the local repository and pushed to the remote repository.



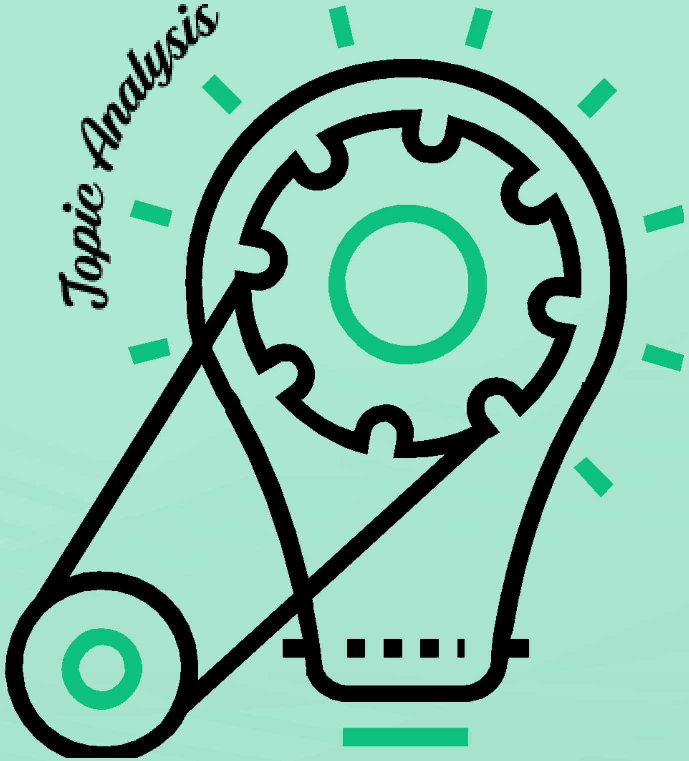
1.2 Centralized Version Control System (CVCS)

Following are the key details of the Centralized Version Control System (CVCS):

- Centralized version control systems keep a single copy of the project on a central server and the developers commit their changes to this central copy of the project.
- Updates to the code is recorded in the central system; when developers pull the code, changes are automatically applied to the files that were changed and developers pull this updated version.
- CVCSs eliminate the need to keep multiple copies of their files on their hard drives.
- The version control system will communicate with the central system and retrieve any version on the go.
- Some examples of CVCS are CVS, Subversion (SVN), Perforce, etc.



What did You Grasp?



1. Which of the following commands is used to move the working copy to the staging area?
   - **A) git add**
   - **B) git commit**
   - **C) git push**
   - **D) git fetch**

2. State True or False

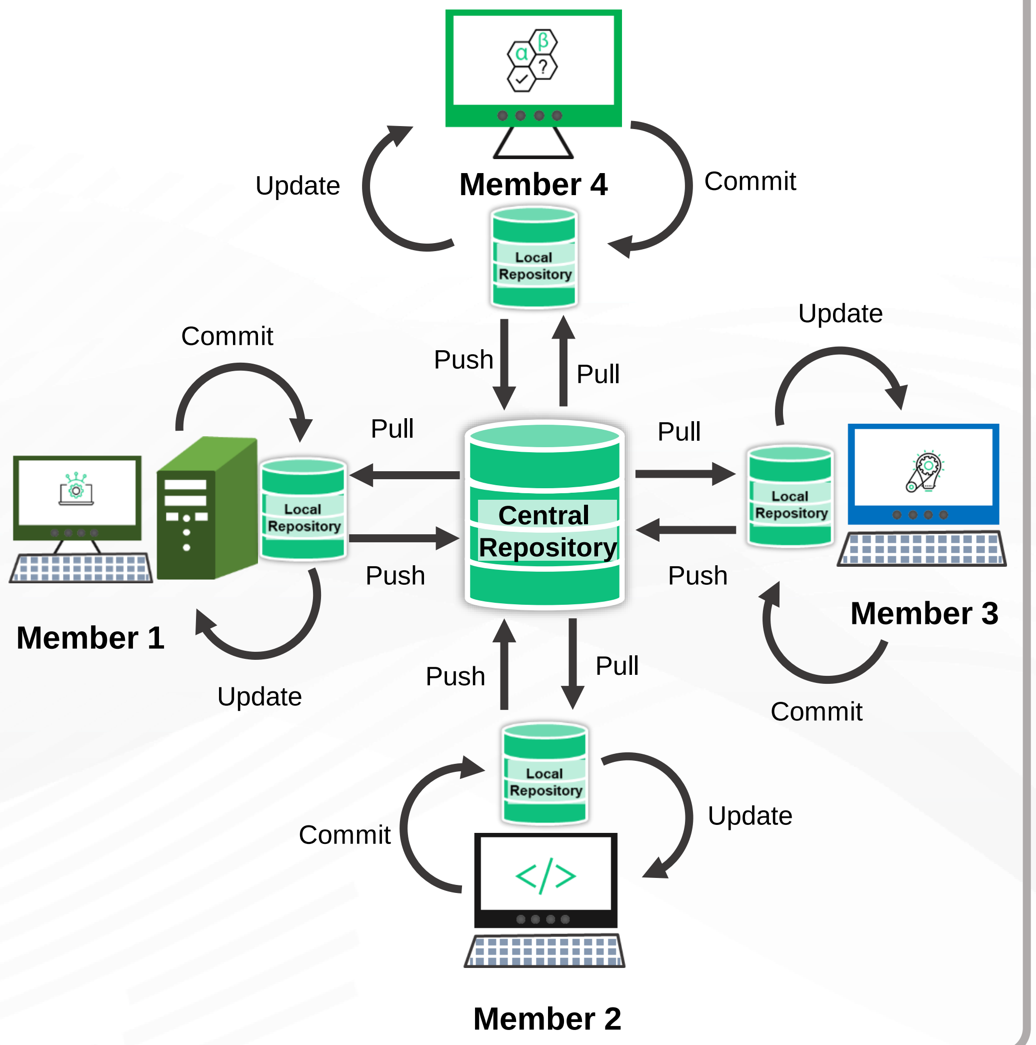
   There are multiple central repositories in a CVCS.
   - **A) True**
   - **B) False**

2.1 Distributed Version Control System (DVCS)

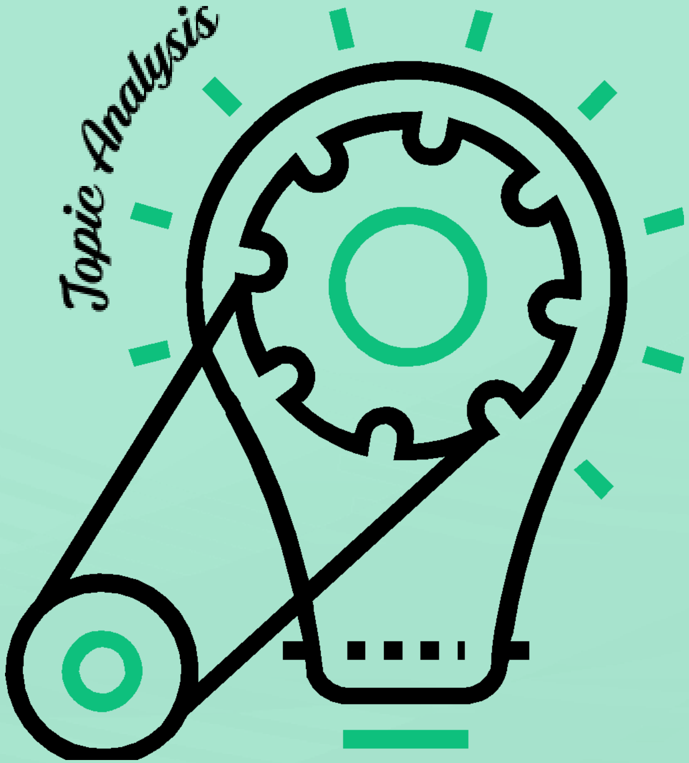
Following are the key details of the Distributed version control system (DVCS):

- A centralized version control system (CVCS) has a single repository to be accessed by the users.
- Distributed version control system (DVCS) replicates the repository onto each user's machine, i.e., each user has a self-contained first-class repository.
- There is no need for a privileged master repository, though teams have it by convention, for doing continuous integration.

DVCS Work



What did You Grasp?



1. State True or False

   The working copy contains only the partial code of the central repository.

   A) True

   B) False

2.2 Advantages of Distributed Version Control System

Advantages

Following are the advantages of DVCS:

- Other than push and pull, all actions can be performed very quickly, since it is the hard drive, and not the remote server that is accessed every time.
- Changesets can be committed to the local repository first and then a group of these changesets can be pushed to the central repository in a single shot.
- Only the pushing and pulling activities need internet connectivity; everything else can be managed locally.
- Every developer has a complete copy of the entire repository and the impact any change can be checked locally before the code is pushed to the central repository.
- DVCS is built to handle changes efficiently, since every change has a Global Unique Identifier (GUID) that makes it easy to track.
- Tasks like branching and merging can be done with ease, since every developer has their own branch and every shared change is like reverse integration
- DVCS is very easy to manage compared to CVCS.

2.3 Private Workspace

Following are the key details of a private workspace:

- DVCS offers each developer a private copy of the complete repository.
- By giving developers a private copy of the entire repository, the DVCS opens up much more flexibility for the kind of things they can do in their private workspace.
- With DVCS, a developer can do frequent commits as often as they want to.
- This option proves advantageous for a solitary developer, who never has to worry about coordinating with others and managing the maintenance overhead.
- With private workspace, one can commit incomplete functionality regularly to the local repository to check point without affecting other users.

2.4 Easier Merging

Following are the key details of merging:

- Branching is generally an easy thing to do, but merging is not.
- People using a CVCS tend to avoid branching because most of those centralized tools aren’t very good at merging.
- Merging in a DVCS is less error-prone, since they keep the developer’s changes distinct from the intended merge in order to get the changes committed.
- DVCS deals with whole-tree branches, not directory branches. The path names in the tree are independent of the branch. This improves interoperability with other tools.

2.5 Easy to Scale Horizontally

Easy to Scale Horizontally

- A DVCS has much more modest hardware requirements for a central server.
- Users don’t interact with the server unless they need to push or pull.
- All the heavy lifting happens on the client side so the server hardware can be very simple indeed.
- With a DVCS, it is also possible to scale the central server by turning it into a server farm.
- Instead of one large server machine, you can add capacity by adding more small server machines, using scripts to keep them all in sync with each other.

2.6 Disadvantages of Distributed Version Control System

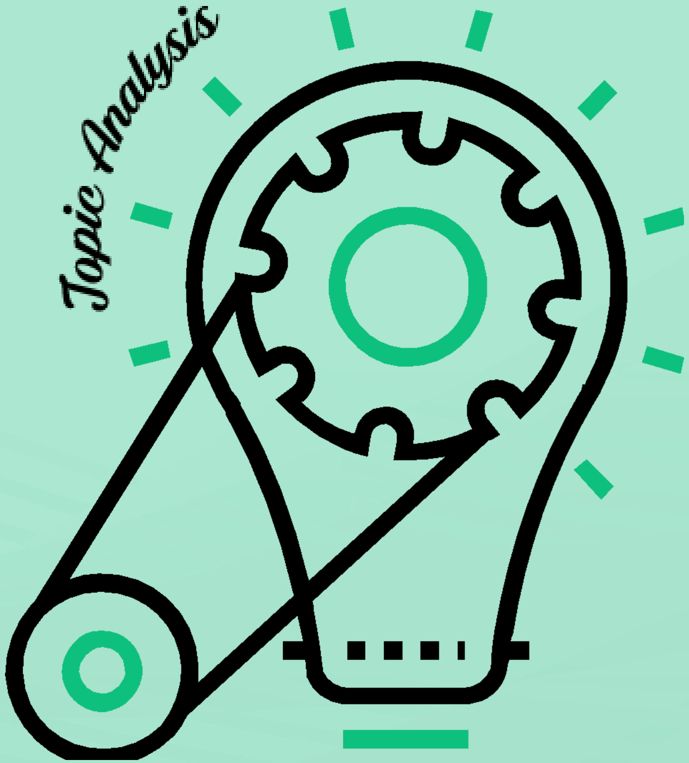
Disadvantages

Following are the disadvantages of DVCS:

- With many projects, large binary files that are difficult to compress, will occupy more space.
- Projects with a long history, i.e., a large number of changesets may take a lot of time and occupy more disk space.
- With DVCS, a backup is still needed, since the latest updated version may not be available to all the developers.
- Though DVCS doesn’t prevent having a central server, not having a central server might cause confusions in identifying the right recent version.
- Though every repo has its own revision numbers, releases have to be tagged with appropriate names to avoid confusions.

Module 3: Version Control System vs Distributed Version Control System

What did You Grasp?



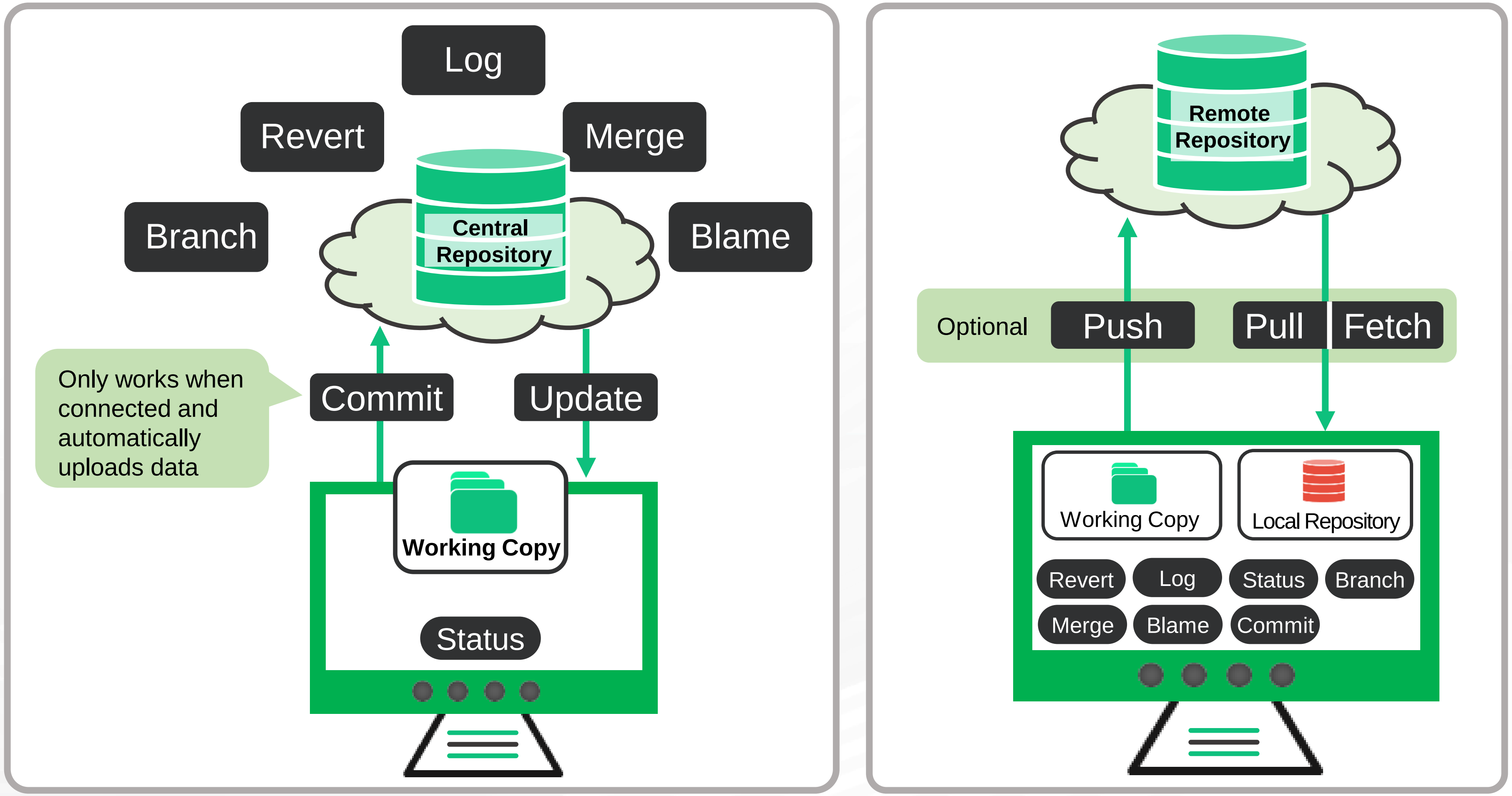
1. Which of the following statements is true?
   - **A) In a DVCS, the central server is accessed every time an update is done to the code**
   - **B) Push and pull activities in a DVCS require internet connectivity**
   - **C) Merging in a DVCS is hard compared to CVCS**
   - **D) The idea of private workspace is unique to DVCS**

2. State True or False.

   Having a central server is still possible with DVCS.
   - **A) True**
   - **B) False**

3.1 Centralized vs Distributed Version Control Systems

The following picture highlights the difference between centralized and distributed version control systems.



Only works when connected and automatically uploads data

source: Scriptcrunch

3.2 Comparison of CVCS and DVCS

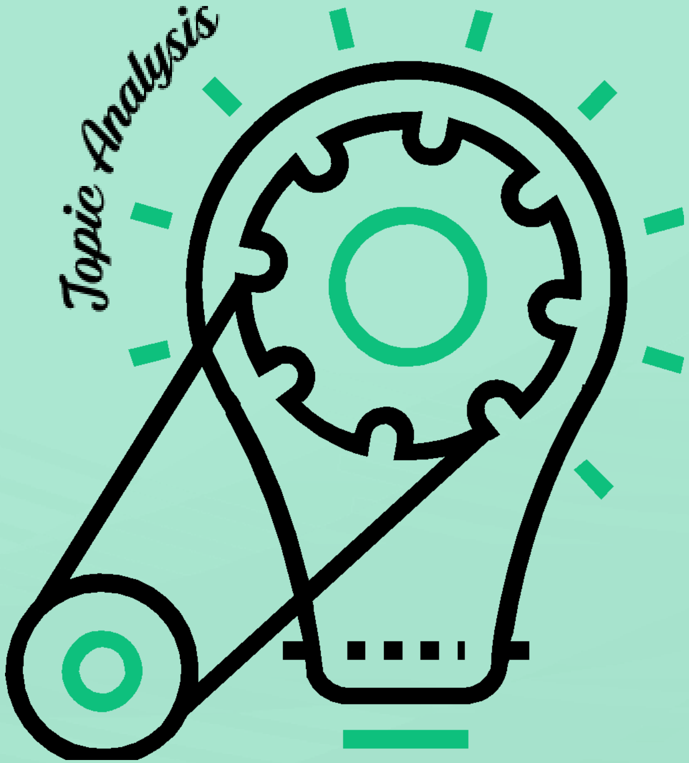
CVCS	DVCS
CVCS focuses on synchronizing, tracking, and backing up files.	DVCS focuses on sharing changes; every change has a guid or unique id.
CVCS works based on a client-server relationship, with the source repository located on one single server, providing access to developers across the globe.	Every developer has one local copy of the source code repository, in addition to the central source code repository.
Recording/downloading and applying a change are separate steps in a centralized system, they happen together.	Distributed systems have no forced structure. You can create "centrally administered" locations or keep everyone as peers.
CVCS relies on internet connectivity for access to the server.	DVCS enables working offline. Apart from push and pull actions, everything is done locally.

3.2 Comparison of CVCS and DVCS (Contd.)

CVCS	DVCS
CVCS requires contacting the server for every command and is relatively slow.	DVCS takes time only during the initial checkout as it mirrors the central repository on the local environment. Subsequent updates and commits are done in the local repository and so the process is faster.
CVCS allows checking out of a few lines of code, if the work has to be done in only a few modules.	DVCS doesn’t allow partial checkout and projects with a long history will take a long time and occupy more disk space.
CVCS is comparatively easy to understand for beginners.	DVCS has some complex processes for beginners.

Module 3: Version Control System vs Distributed Version Control System

What did You Grasp?



1. Which of the following statements is incorrect?

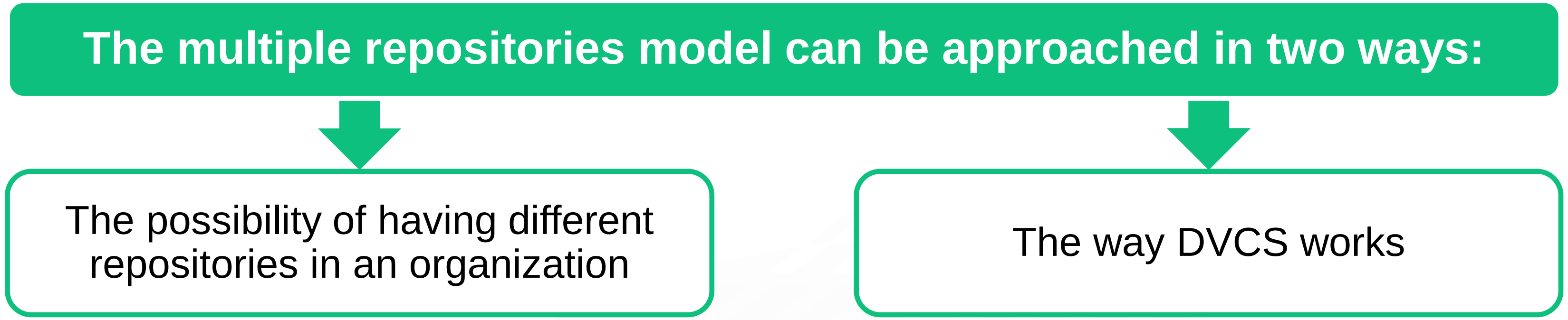
   A) DVCS works on a client-server relationship

   B) CVCS requires the Internet connection for access to the central server

   C) With CVCS is it possible to checkout portions of code

   D) Initial checkout is a time consuming process in DVCS

4.1 Multiple Repositories Model



The multiple repositories model can be approached in two ways:

The possibility of having different repositories in an organization

The way DVCS works

The way DVCS works:

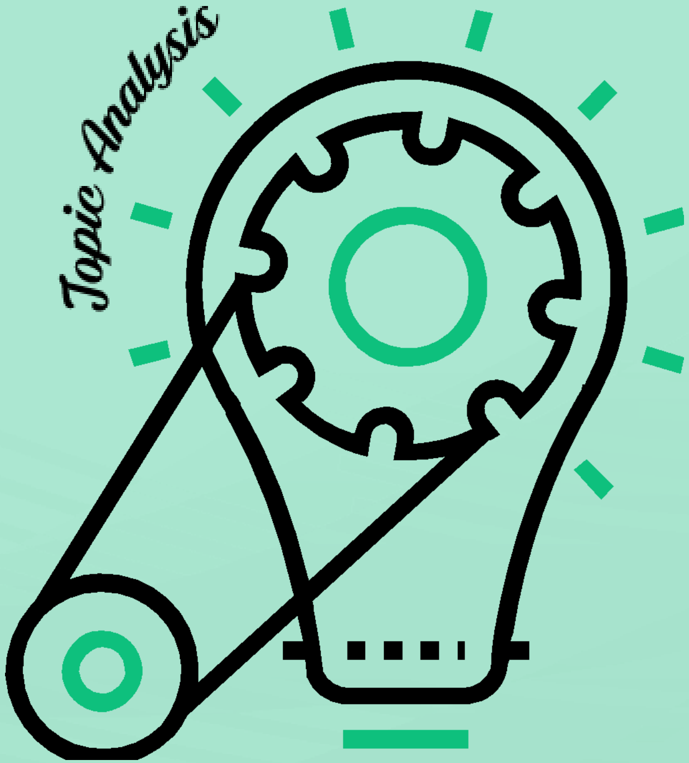
- Unlike CVCSs, DVCSs utilize multiple repositories, one central repository and multiple local repositories in developers’ local machines.
- All local repositories are exact replicas of the central repository, in the sense that they have the complete history of changes.
- Changes are done to the local repositories first and are then merged with the central repository.

4.2 Multiple Repositories for Different Services

Multiple Repositories for Different Services

- In mono repositories, the source code is placed in a single repository in an organization and developers have access to all code in a single shot.
- The Multiple repository concept refers to organizing the projects into their own separate repositories.
- DVCS like Git allows us to have multiple repositories, by which giving access to subsets of repositories on a need basis, becomes possible.
- Especially, products with multiple services can be handled efficiently by having separate repositories for each of them.
- While setting up continuous deployment projects, it is easier to let the repositories have their own processes for getting deployed.

What did You Grasp?



1. State True or False

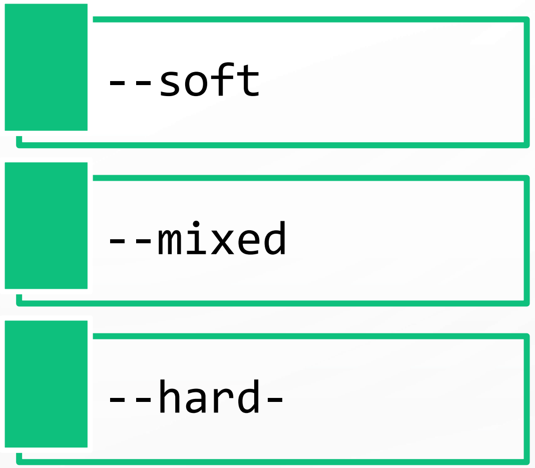
   With DVCS like Git, there is no room for having different repositories for different services.

   A) True

   B) False

5.1 Resetting the Local Environment

- Reset is one of the operations used for undoing the changes.
- We can understand the reset operation using Git. The Git reset operation has three basic forms of invocation that correspond to three commands, such as follows:



--soft

--mixed

--hard-

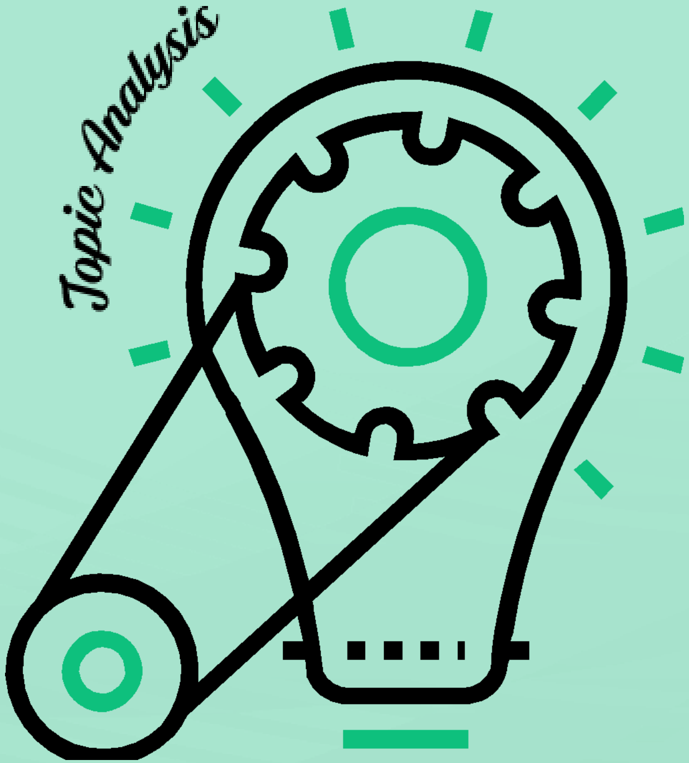
- These arguments correspond to three state management systems of Git, namely the Commit Tree (HEAD), the Staging Index and the Working Directory.
- The reset command moves both the HEAD and the branches to a specific commit.

5.2 Revert - Canceling out the Changes

Revert - Canceling out the Changes

- Revert is also one of the options for undoing the changes, but works in a different way compared to the traditional undo operation.
- The revert option doesn’t remove the commit from the project history, but it checks how to invert the changes introduced by the commit and appends a new commit with the resulting inverse content.
- The revert operation prevents Git from losing the history, which is very important to maintain the integrity of the version history.
- Revert option can be used when an inverse of a commit from the project has to be applied.

What did You Grasp?



1. Which of the following commands is used to move the HEAD and the branch refs to a specific commit?
   - **A) git reset**
   - **B) git move**
   - **C) git revert**
   - **D) git clean**

Module 3: Version Control System vs Distributed Version Control System

In a nutshell, we learnt:

1. How local version control works
2. A brief on centralized version control systems (CVCS)
3. A brief on distributed version control systems (DVCS)
4. Advantages of DVCS
   - ⇥ Private Workspace
   - ⇥ Easier merging
   - ⇥ Easy to scale horizontally
5. Disadvantages of DVCS
6. Comparison of CVCS and DVCS
7. Multiple repositories model
8. Resetting the local environment
9. Reverting - undoing the changes