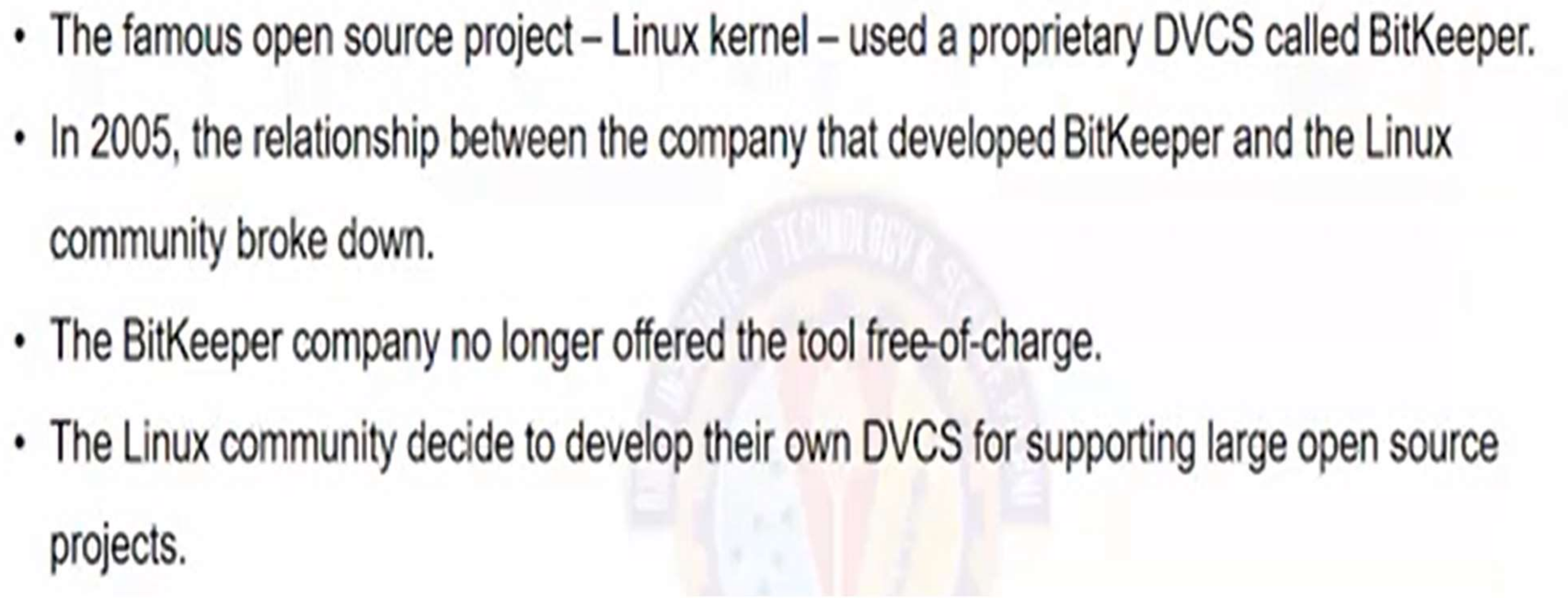
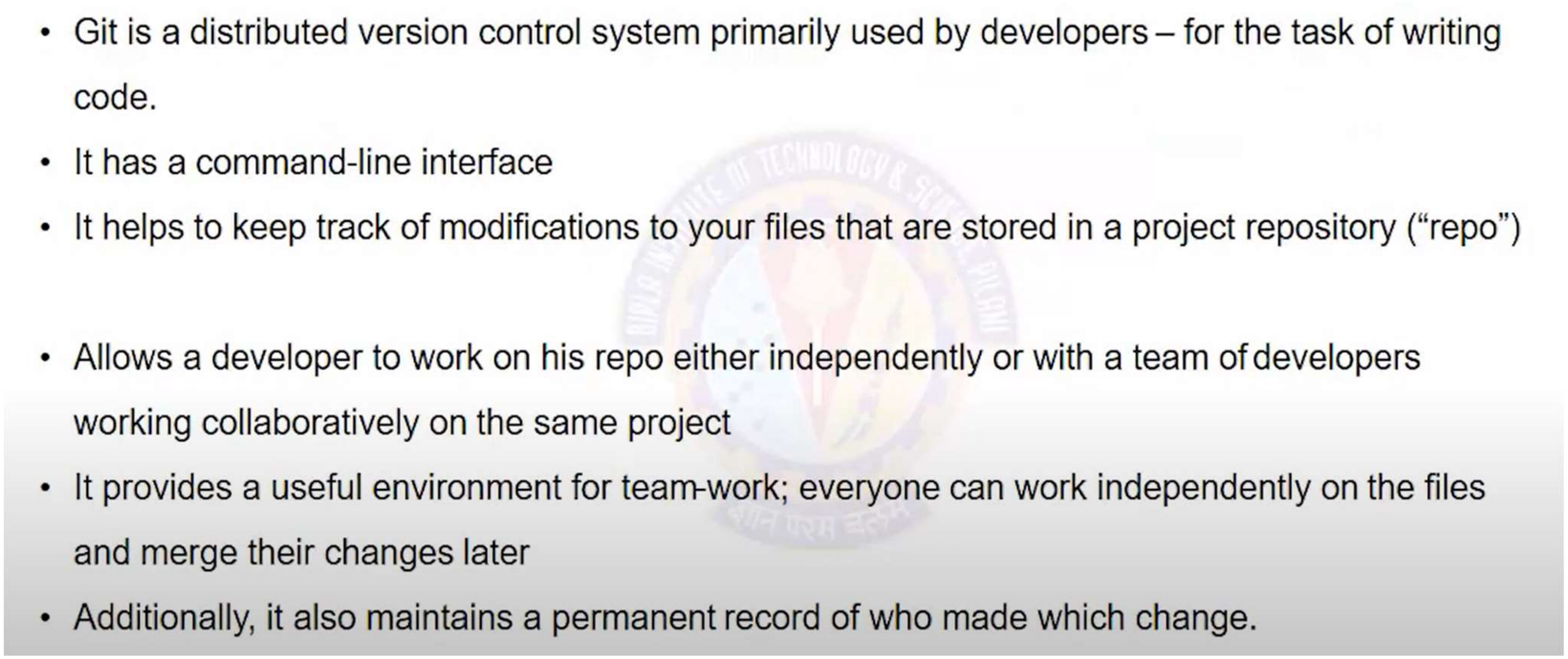
# Git and GitHub

## History of Git



- The famous open source project – Linux kernel – used a proprietary DVCS called BitKeeper.
- In 2005, the relationship between the company that developed BitKeeper and the Linux community broke down.
- The BitKeeper company no longer offered the tool free-of-charge.
- The Linux community decide to develop their own DVCS for supporting large open source projects.

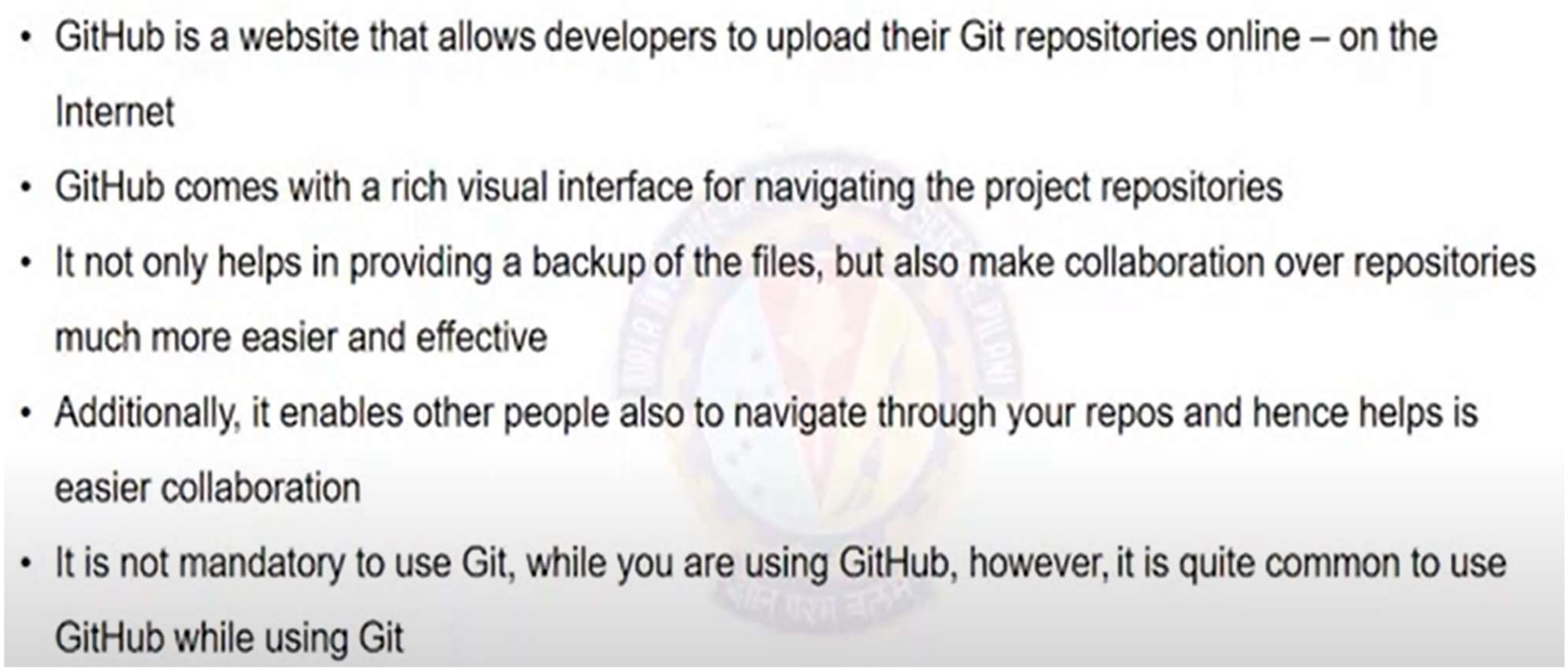
## What is Git?



- Git is a distributed version control system primarily used by developers – for the task of writing code.
- It has a command-line interface
- It helps to keep track of modifications to your files that are stored in a project repository (“repo”)

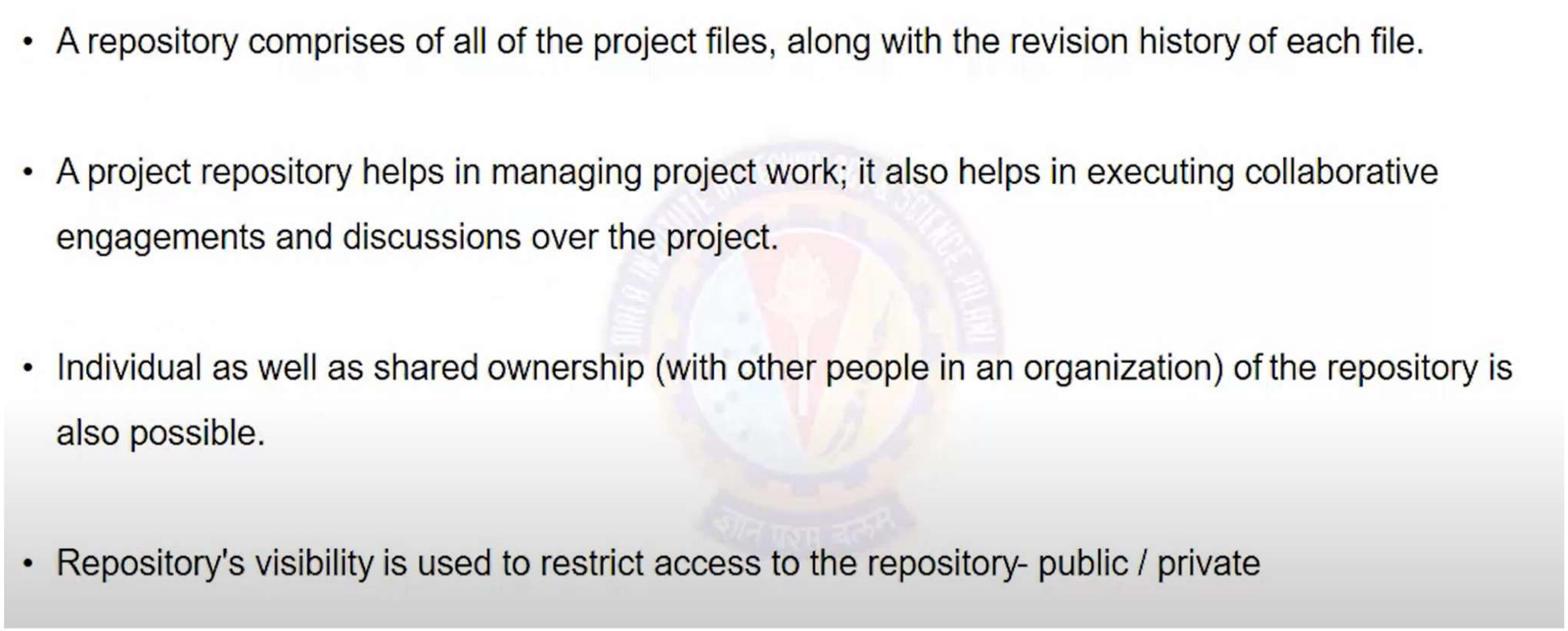
- Allows a developer to work on his repo either independently or with a team of developers working collaboratively on the same project
- It provides a useful environment for team-work; everyone can work independently on the files and merge their changes later
- Additionally, it also maintains a permanent record of who made which change.

## What is GitHub



- GitHub is a website that allows developers to upload their Git repositories online – on the Internet
- GitHub comes with a rich visual interface for navigating the project repositories
- It not only helps in providing a backup of the files, but also make collaboration over repositories much more easier and effective
- Additionally, it enables other people also to navigate through your repos and hence helps is easier collaboration
- It is not mandatory to use Git, while you are using GitHub, however, it is quite common to use GitHub while using Git

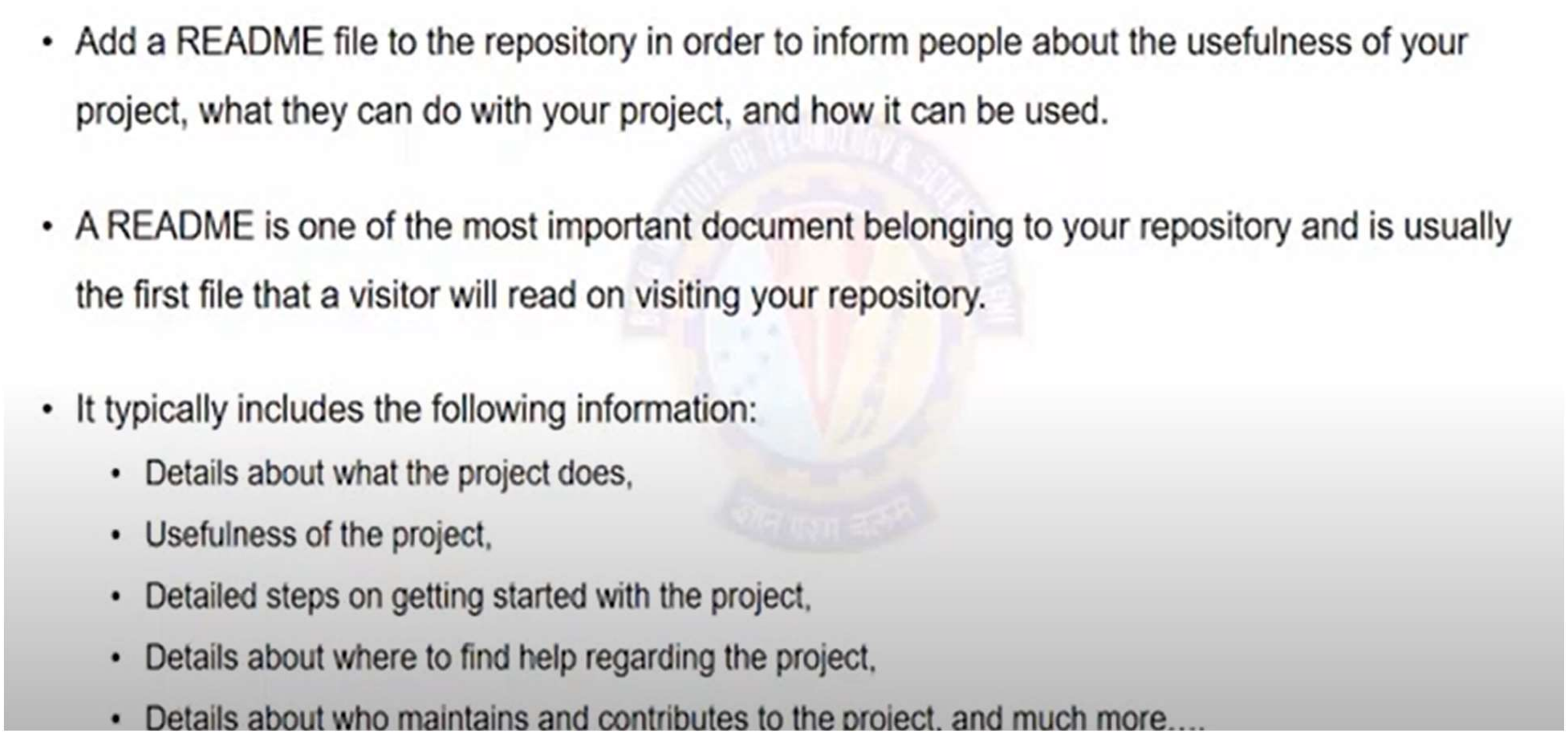
## What is a repository?



- A repository comprises of all of the project files, along with the revision history of each file.
- A project repository helps in managing project work; it also helps in executing collaborative engagements and discussions over the project.
- Individual as well as shared ownership (with other people in an organization) of the repository is also possible.
- Repository's visibility is used to restrict access to the repository- public / private

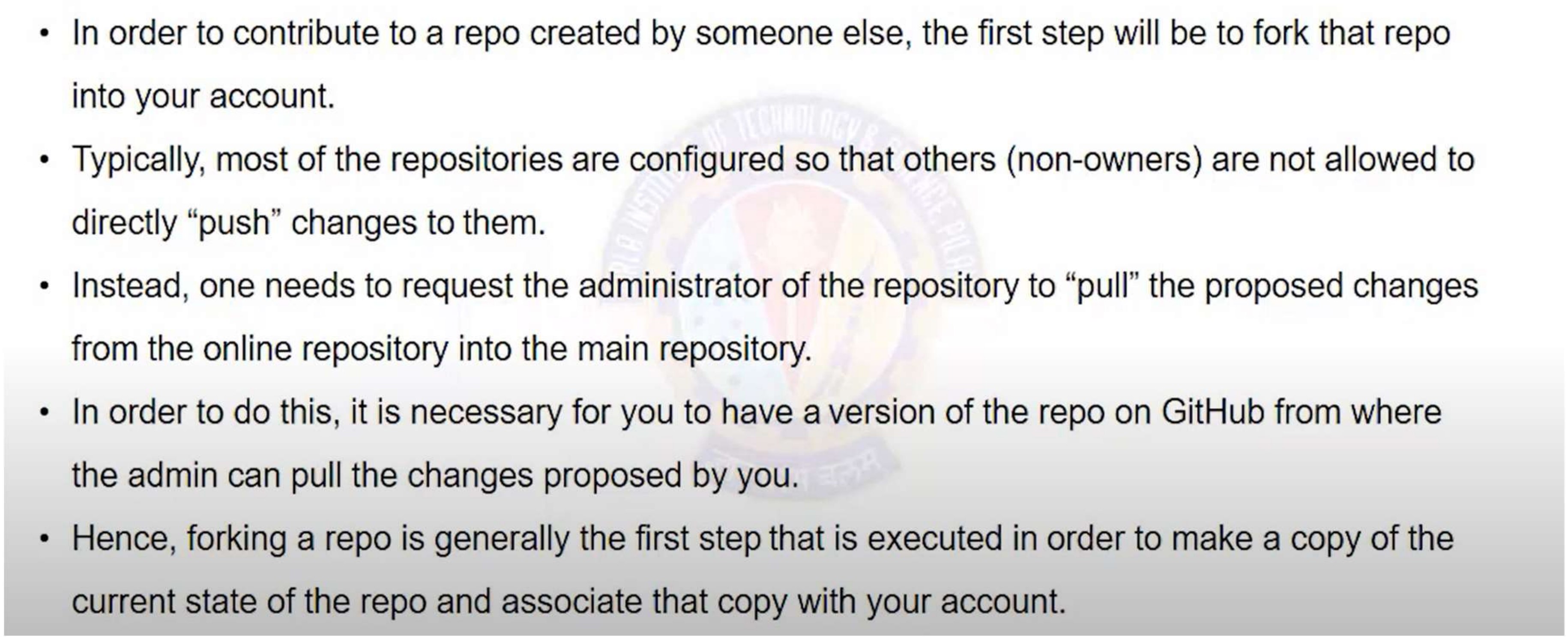
- Creating a repository
- Visibility

## Adding Readme.md



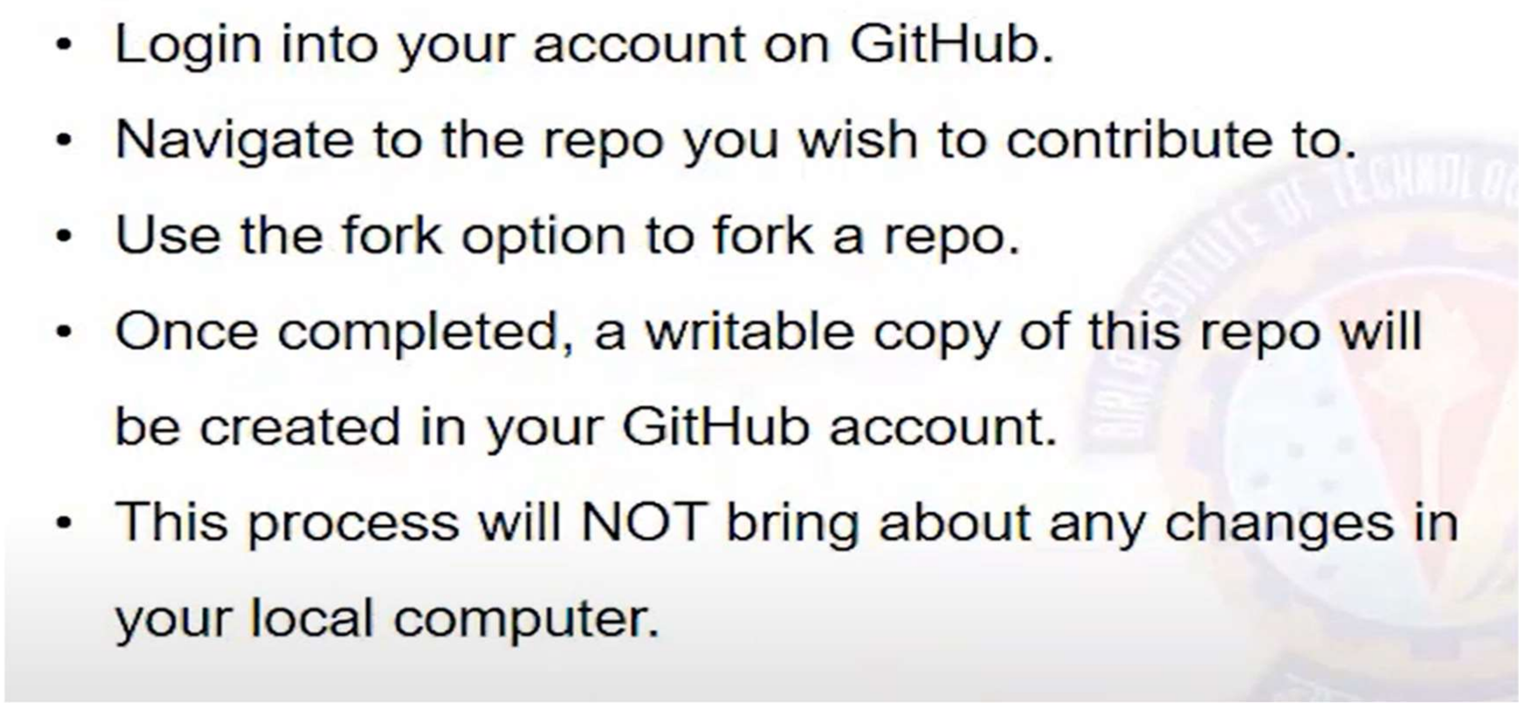
- Add a README file to the repository in order to inform people about the usefulness of your project, what they can do with your project, and how it can be used.
- A README is one of the most important document belonging to your repository and is usually the first file that a visitor will read on visiting your repository.
- It typically includes the following information:
  - Details about what the project does,
  - Usefulness of the project,
  - Detailed steps on getting started with the project,
  - Details about where to find help regarding the project,
  - Details about who maintains and contributes to the project, and much more....

## Forking a repo



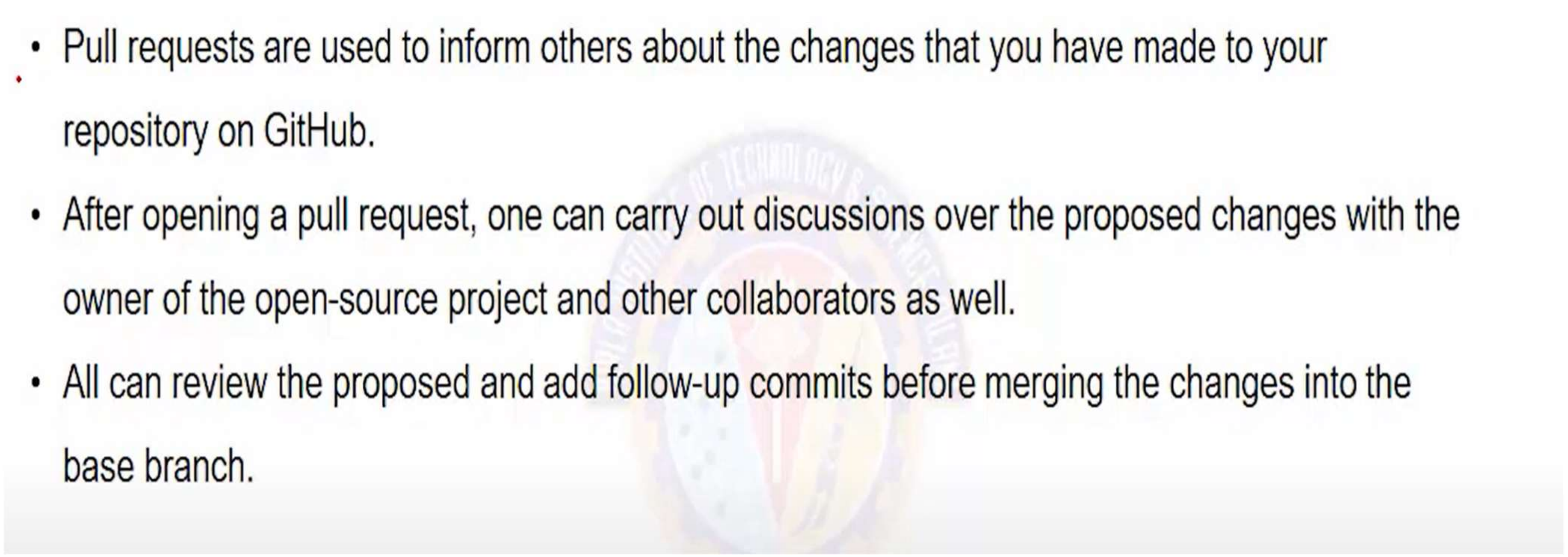
- In order to contribute to a repo created by someone else, the first step will be to fork that repo into your account.
- Typically, most of the repositories are configured so that others (non-owners) are not allowed to directly “push” changes to them.
- Instead, one needs to request the administrator of the repository to “pull” the proposed changes from the online repository into the main repository.
- In order to do this, it is necessary for you to have a version of the repo on GitHub from where the admin can pull the changes proposed by you.
- Hence, forking a repo is generally the first step that is executed in order to make a copy of the current state of the repo and associate that copy with your account.

## How to fork a repo?



- Login into your account on GitHub.
- Navigate to the repo you wish to contribute to.
- Use the fork option to fork a repo.
- Once completed, a writable copy of this repo will be created in your GitHub account.
- This process will NOT bring about any changes in your local computer.

## What is a Pull Request?



- Pull requests are used to inform others about the changes that you have made to your repository on GitHub.
- After opening a pull request, one can carry out discussions over the proposed changes with the owner of the open-source project and other collaborators as well.
- All can review the proposed and add follow-up commits before merging the changes into the base branch.

- Creating a Pull Request
- Merging[No Conflict and Conflict]

## Workflow of a Pull based development model

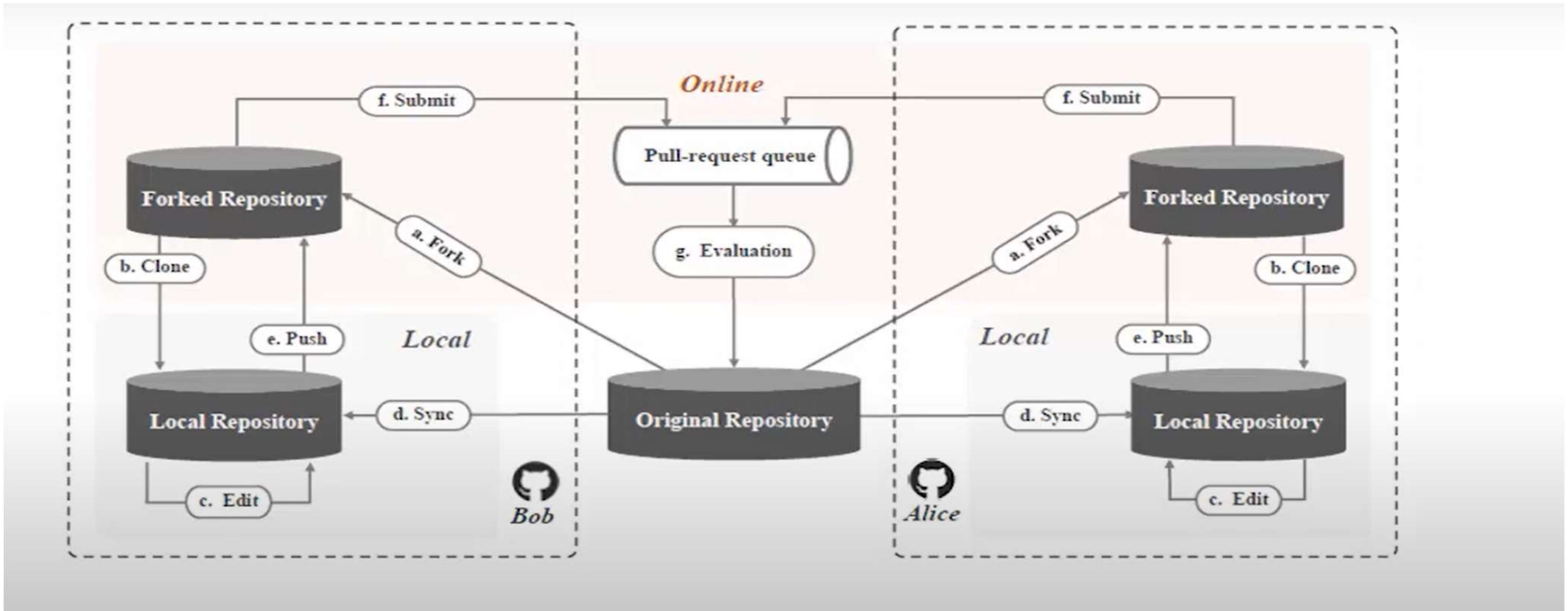
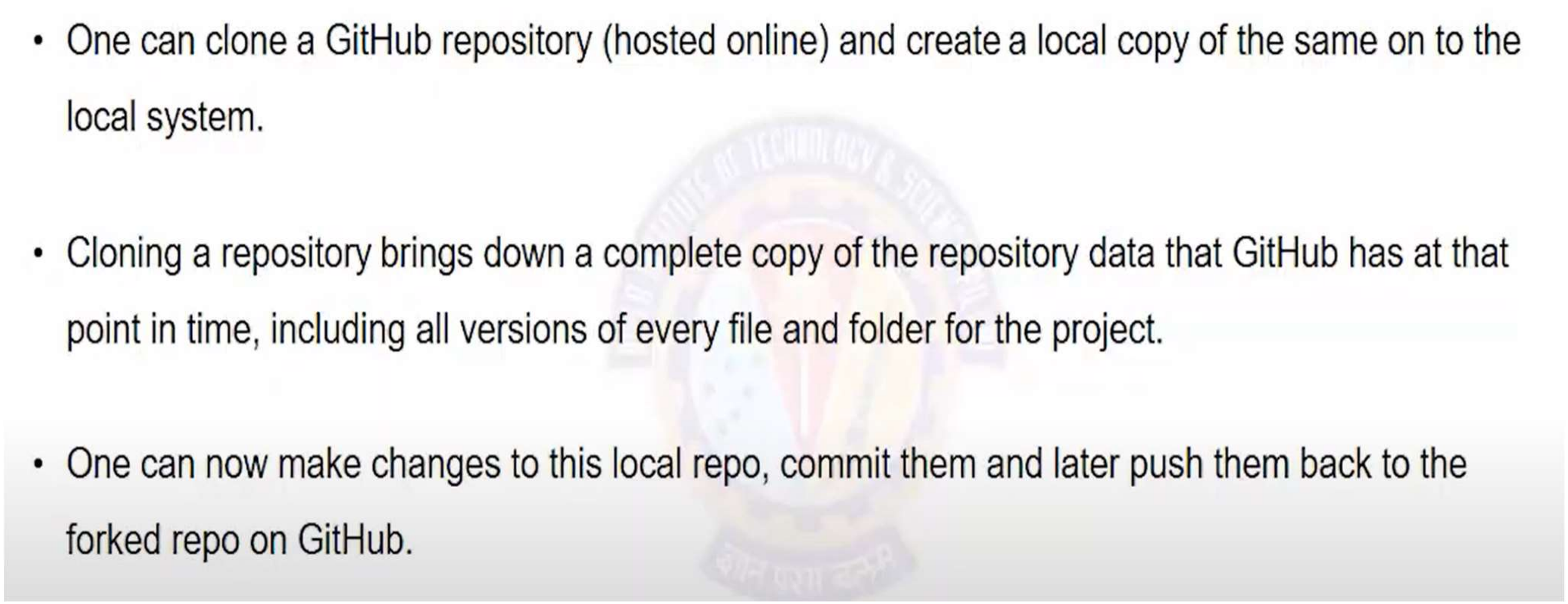


Figure depicting workflow of Pull based development model

## Cloning a repo



- One can clone a GitHub repository (hosted online) and create a local copy of the same on to the local system.
- Cloning a repository brings down a complete copy of the repository data that GitHub has at that point in time, including all versions of every file and folder for the project.
- One can now make changes to this local repo, commit them and later push them back to the forked repo on GitHub.

## Committing changes to a repo and pushing

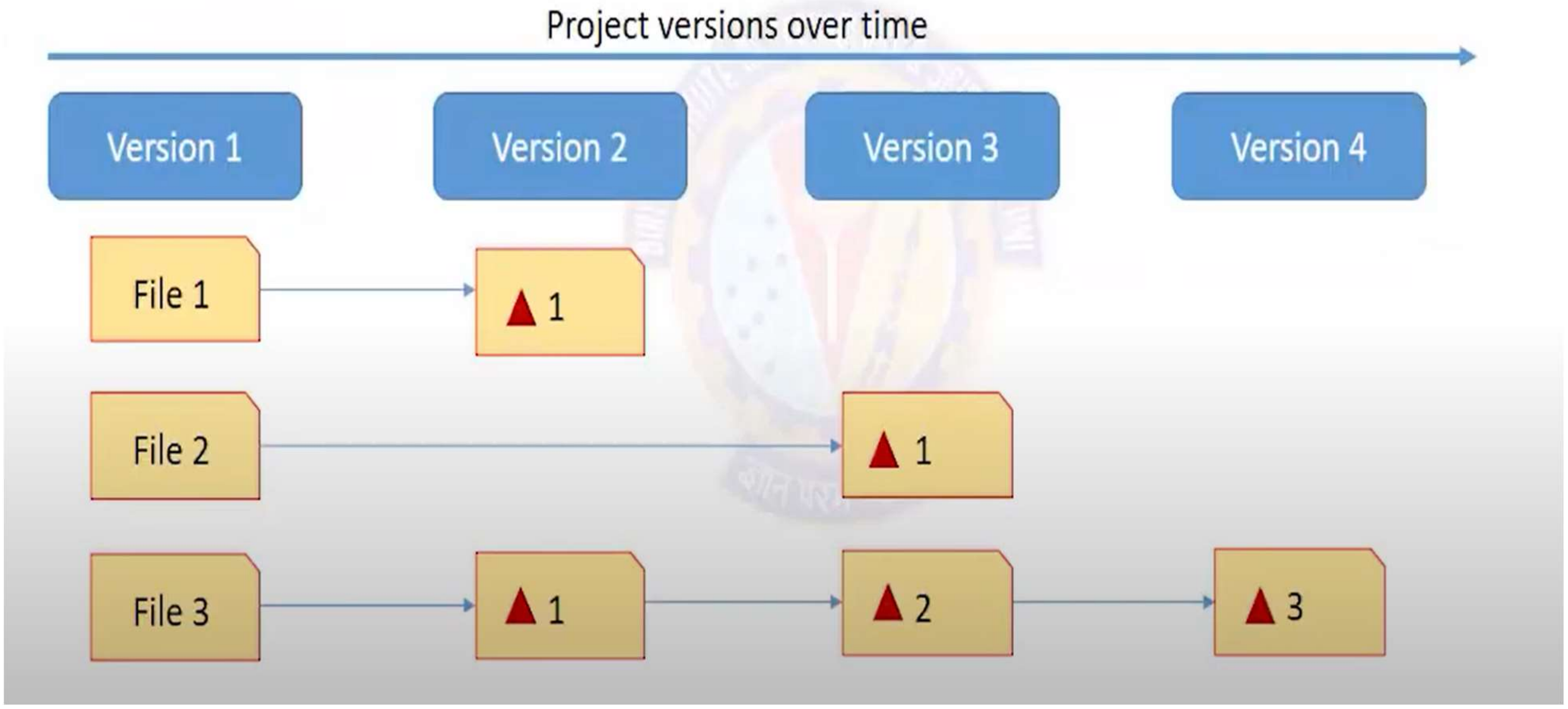
- Changes only local repository
- No changes to the remote repo on GitHub
- Push the changes [committed locally] to the remote repo on GitHub.

## Development goals of Git

- Simplicity
- Speed
- Totally distributed
- Support for branching
  - Thousands of parallel branches can exist
- Ability to handle large software projects.
  - Eg: linux kernel.

- To achieve this-----------------------→ Git uses snapshot based VCS

## Delta based VCS [stores only the changes over time]



## Git uses snapshot based VCS

- Series of snapshots are stored instead of delta based approach.
  - Snapshot taken for every commit.
  - Reference to the snapshot is stored.
  - Unchanged files are not stored again.
    - Link/pointer to the earlier version which is already available.
  - Changed files are stored.

- How this approach is going to help Git?

# Snapshot based VCS[Git]

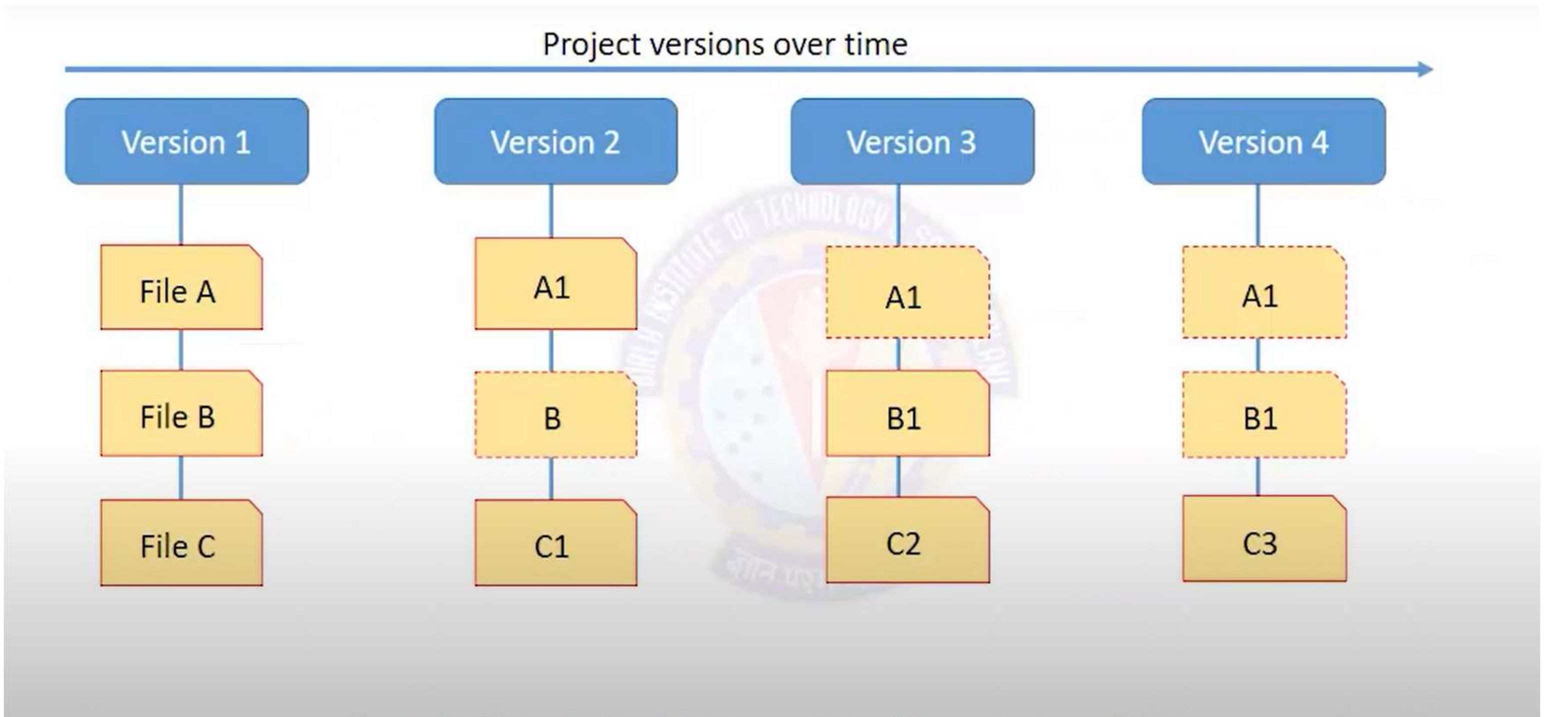


Figure depicting storing data as snapshots of the project over time

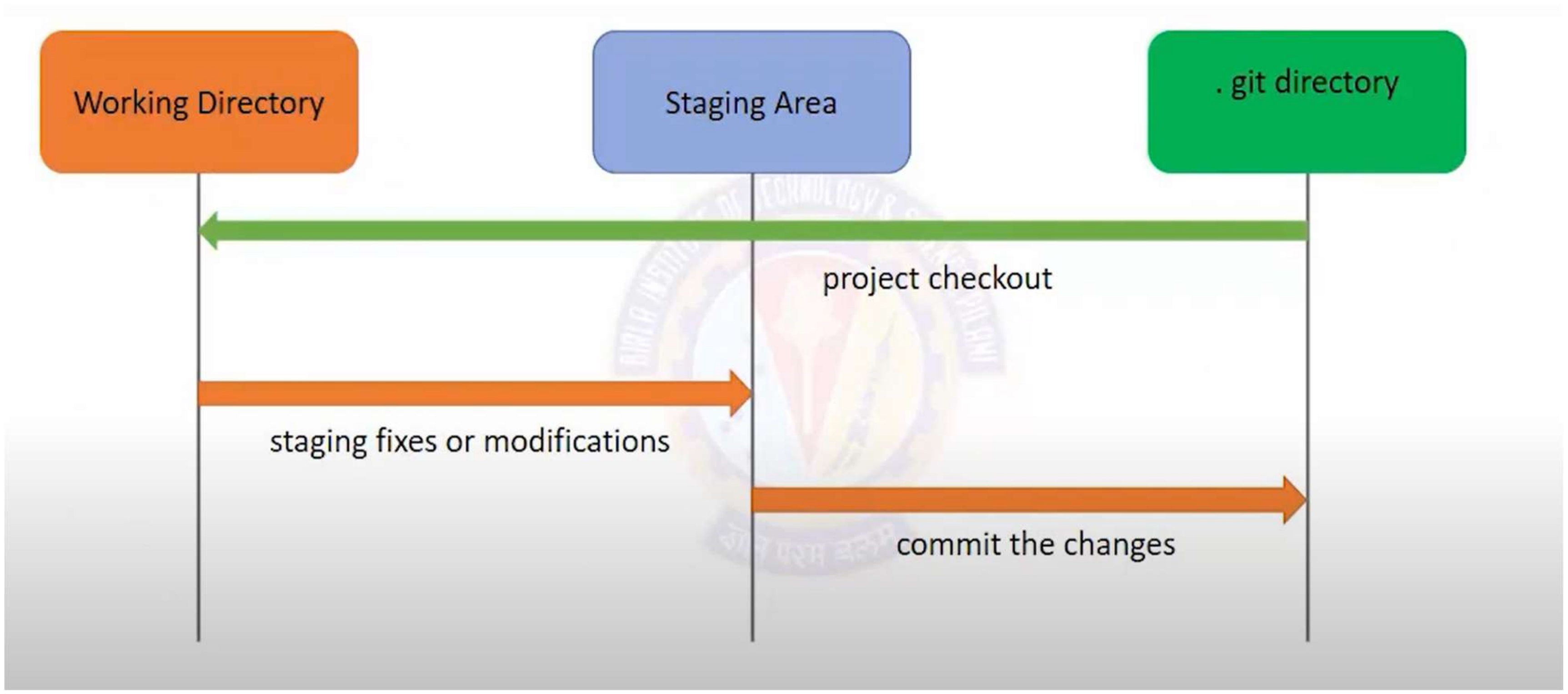
## Other Unique characteristics of Git

- Almost all operations are local[using only local resources]
  - Usually no information required from the server or other computers on the network.
    - Made possible by storing the entire history of the project locally.
  - Unless you are pushing your changes or updating to fetch the latest.
  - Entire history of the project can be fetched from the local repo.

## 3 states in Git: Modified, Staged, Committed

- **Modified**: Changed but not yet committed.[one or more files]
- **Staged**: Marked as modified and marked to go into the next commit snapshot.
- **Committed State**: File is safely stored in the local database/repo.

## The 3 sections of a Git Project.



## Why staging area is important?

## How its done?

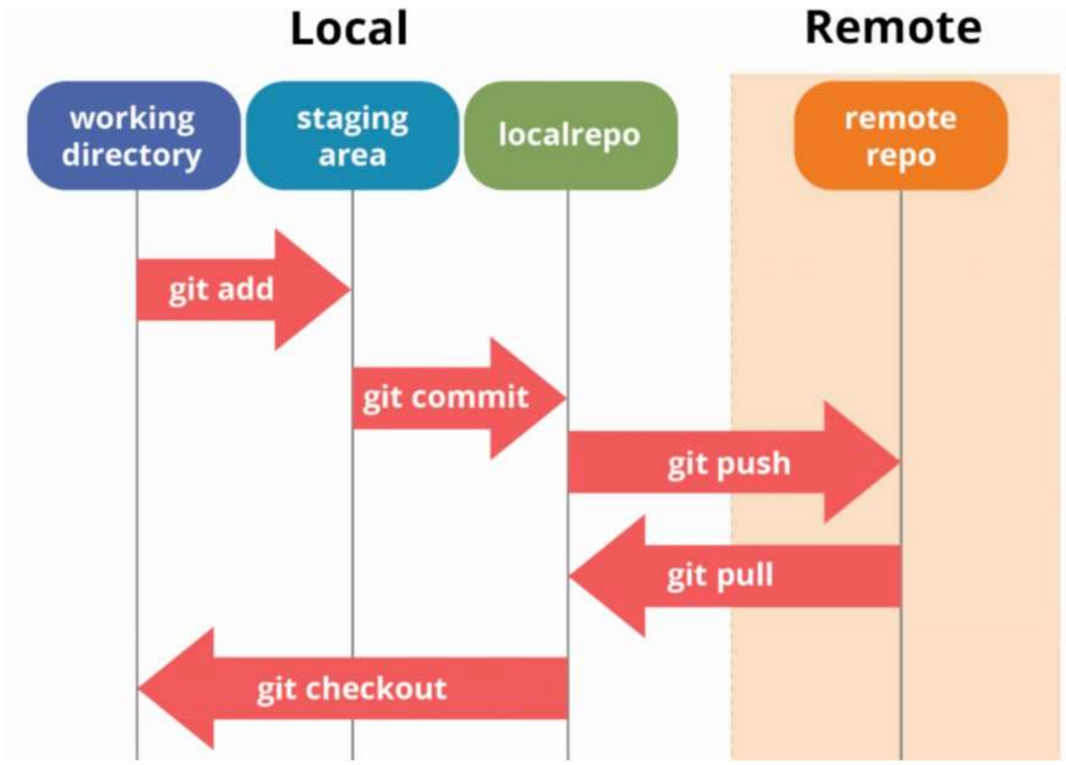
- Working directory is a single checkout of the latest version of the project.[from the compressed git directory]
- Staging area is a file[contains info about staged files].
- Git directory stores the metadata about the project.[downloaded to the local system when a project is cloned].

## Git Workflow

1> Modified files exist in the working directory

2> files are selectively staged to the staging area which are going to be part of the next commit.

3> Once commit is executed, the staged files are permanently stored in the snapshot in Git directory.



Local
Remote
working directory
staging area
localrepo
remote repo
git add
git commit
git push
git pull
git checkout