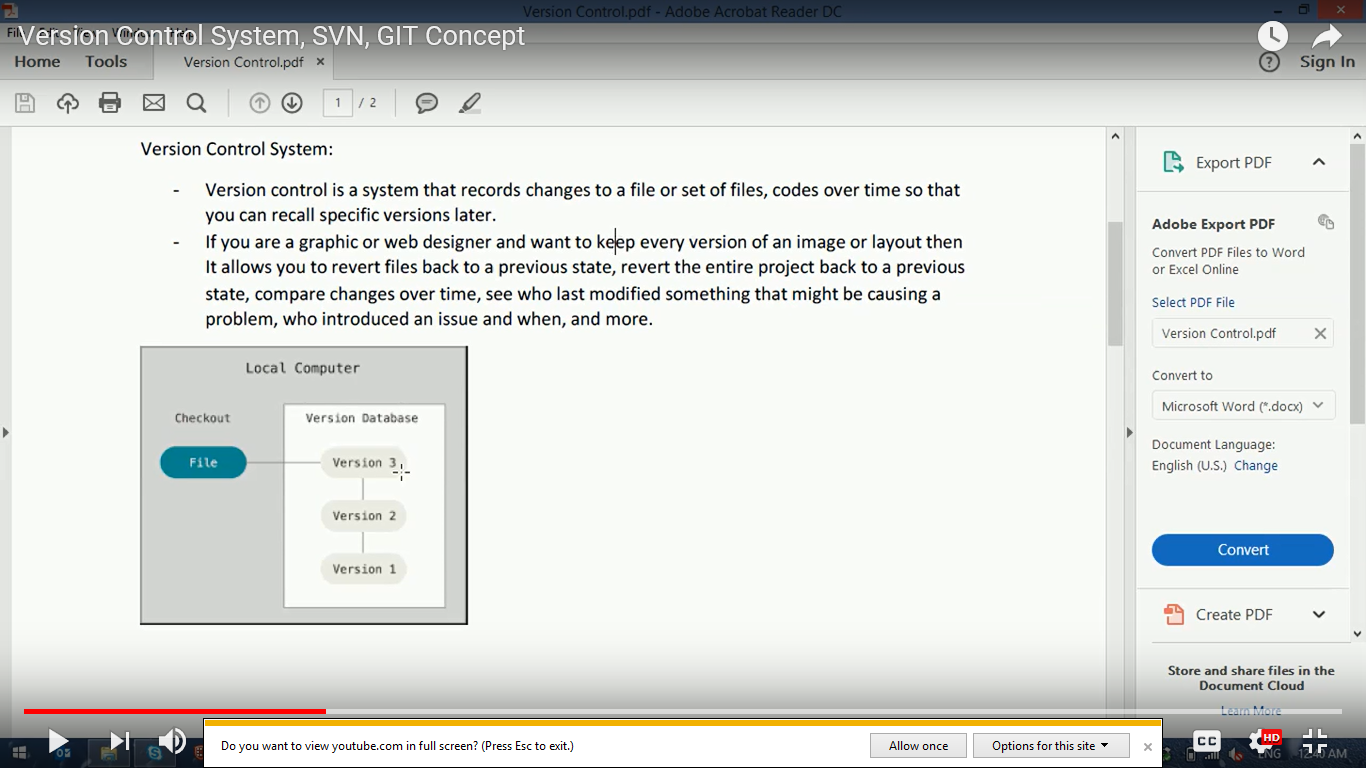
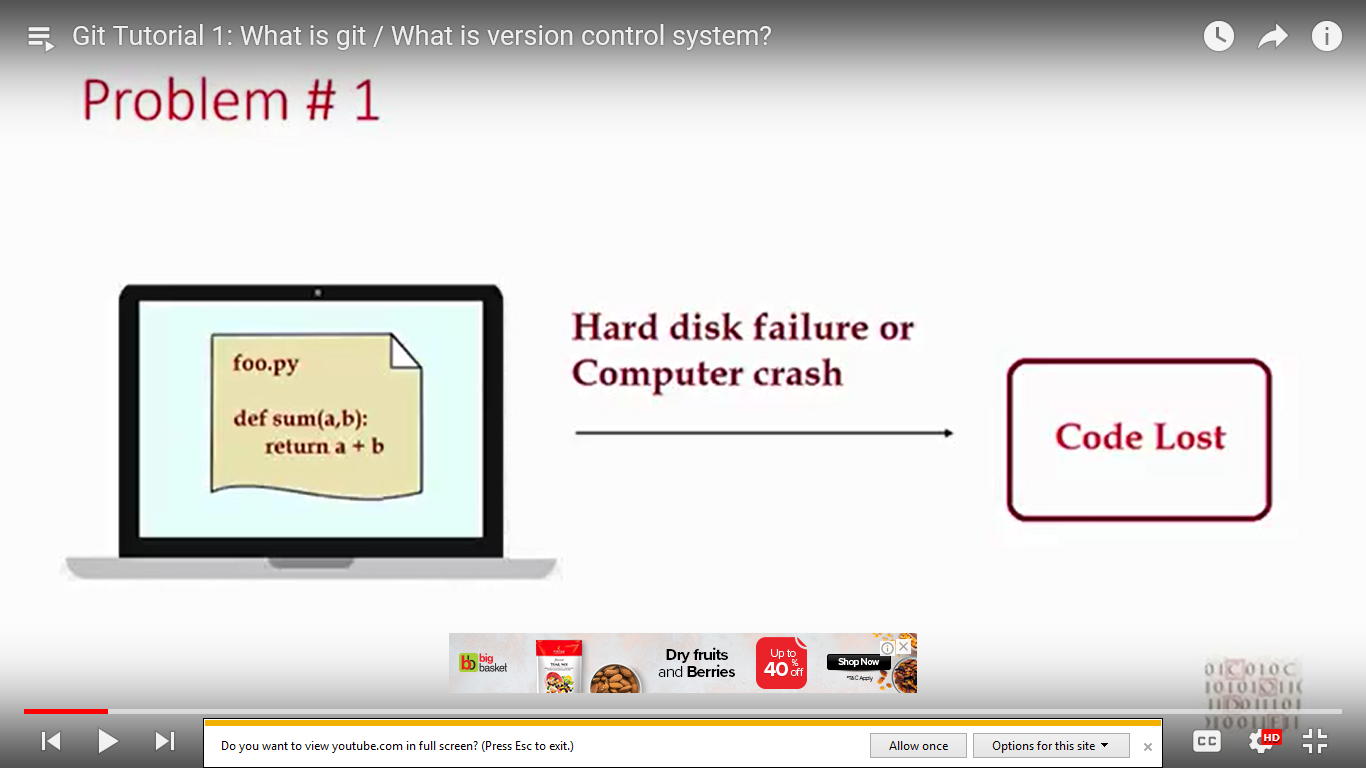
**What is GIT ?**

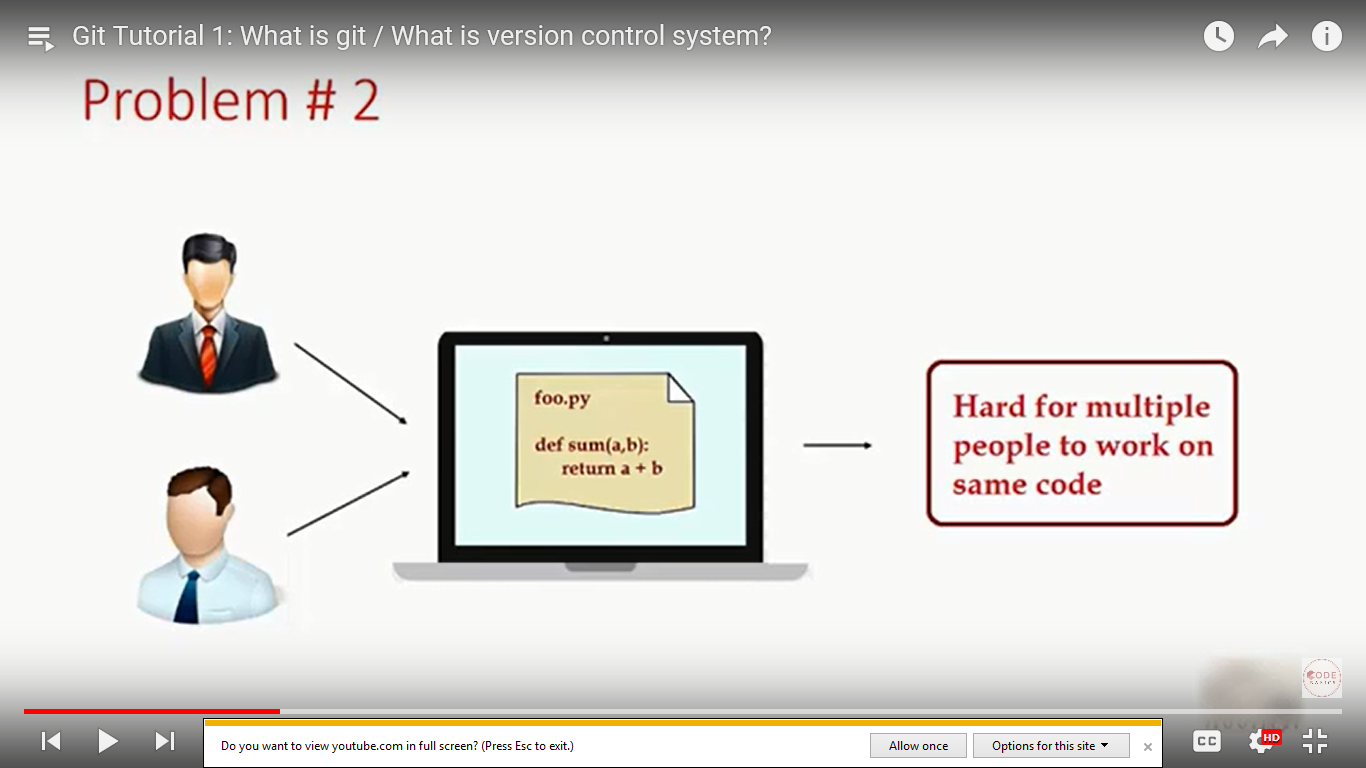
* + Git is a version control system (VCS) for tracking changes in computer files and coordinating work on those files among multiple people.
  + It is primarily used for software development, but it can be used to keep track of changes in any files.
  + This course enables you to learn and solve versioning problems with your files and codes.   
      
    All these concepts are presented in an easy to understand manner, using demos and assignments to clarify the concepts and present the actual method of implementation.

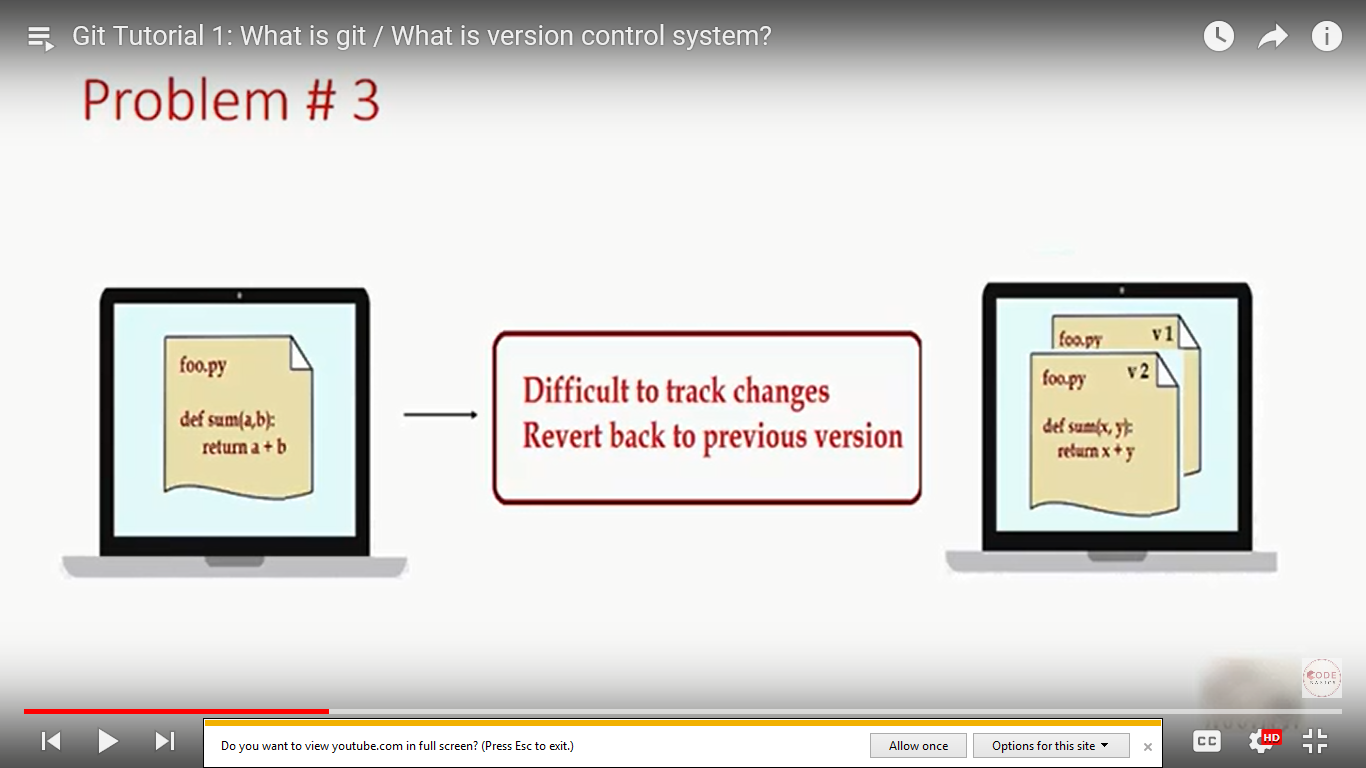
**Scope**

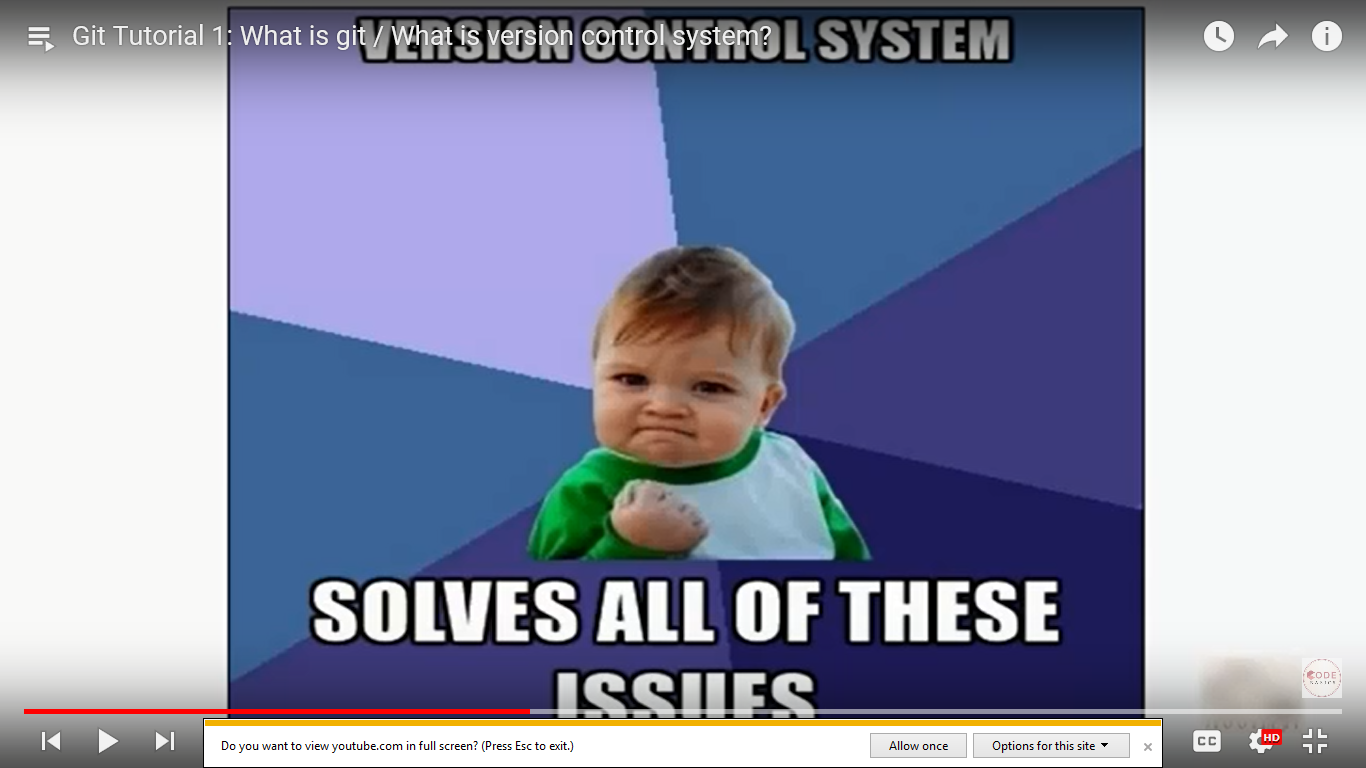
* + Git course offered by DACME-TLS will enable you to:
  + Understand distributed version control system and its features
  + Set-up Git in your system
  + List the three-stage workflow in Git
  + Create branches and track files
  + Create a repository in Git and GitHub
  + Describe merging, cloning, rebasing, among others
* The following can be done after completing this course
  + Install Git and configure it
  + Create a new file and track it across the three stage workflow
  + Revert to earlier commit, deleting/ignoring Git files
  + Create an account in GitHub and move the files to remote repository
  + Collaborate between local and remote repository
  + Merge different branches of a file, handle conflicts
  + Use the issue tracker in GitHub to collaborate among developers
  + Stash file versions and revert back to them
  + Rebase the files in the repository
  + Clone the repository and collaborate the files across developers











* + Git is a version control system (VCS) for tracking changes in computer files and coordinating work on those files among multiple people.
  + It is primarily used for software development, but it can be used to keep track of changes in any files.
  + It is distributed and Open Source

Types of VCS

1. Local VCS
2. Central VCS
3. Distributed VCS

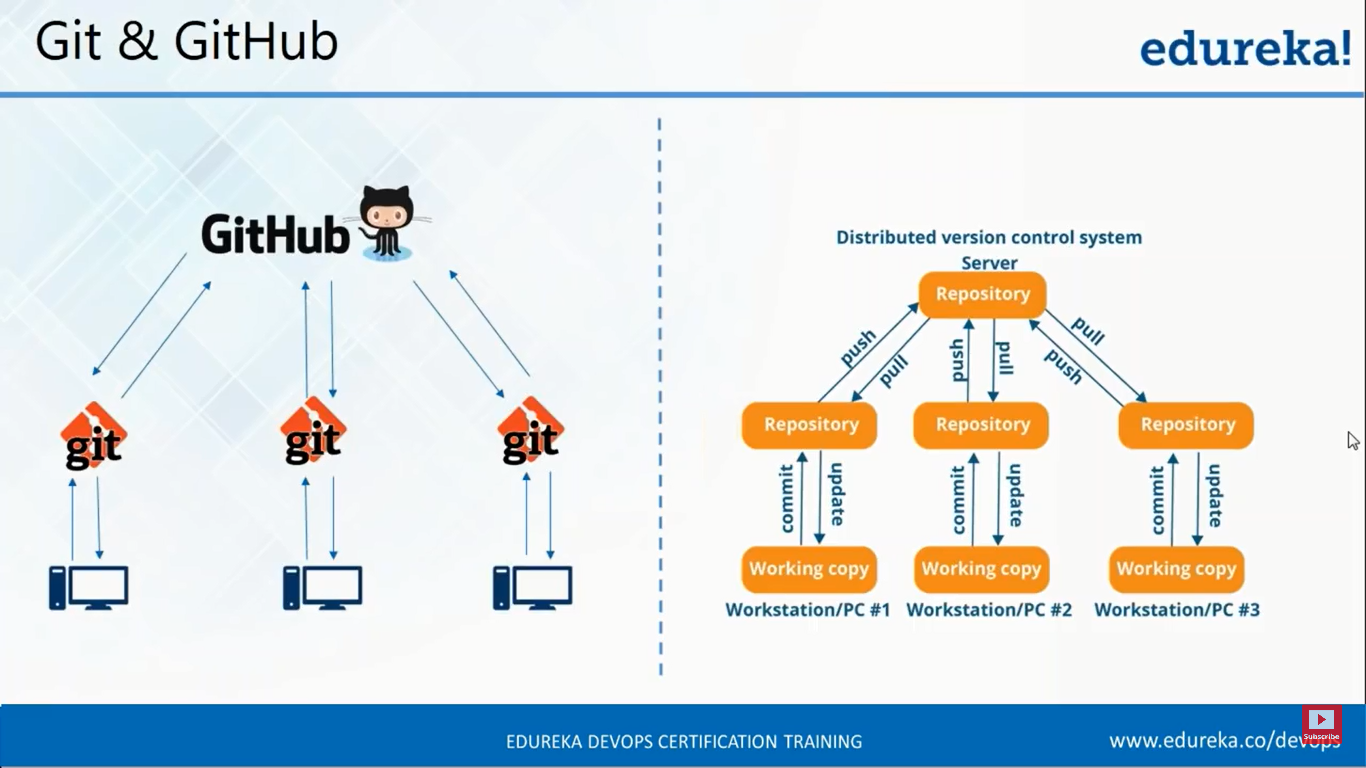
## Advantages of Distributed VCS over Centralized VCS

The act of cloning an entire repository gives distributed version control tools several advantages over centralized systems:

* Performing actions other than pushing and pulling changesets is extremely fast because the tool only needs to access the hard drive, not a remote server.
* Committing new changesets can be done locally without anyone else seeing them. Once you have a group of changesets ready, you can push all of them at once.
* Everything but pushing and pulling can be done without an internet connection. So you can work on a plane, and you won’t be forced to commit several bugfixes as one big changeset.
* Since each programmer has a full copy of the project repository, they can share changes with one or two other people at a time if they want to get some feedback before showing the changes to everyone.

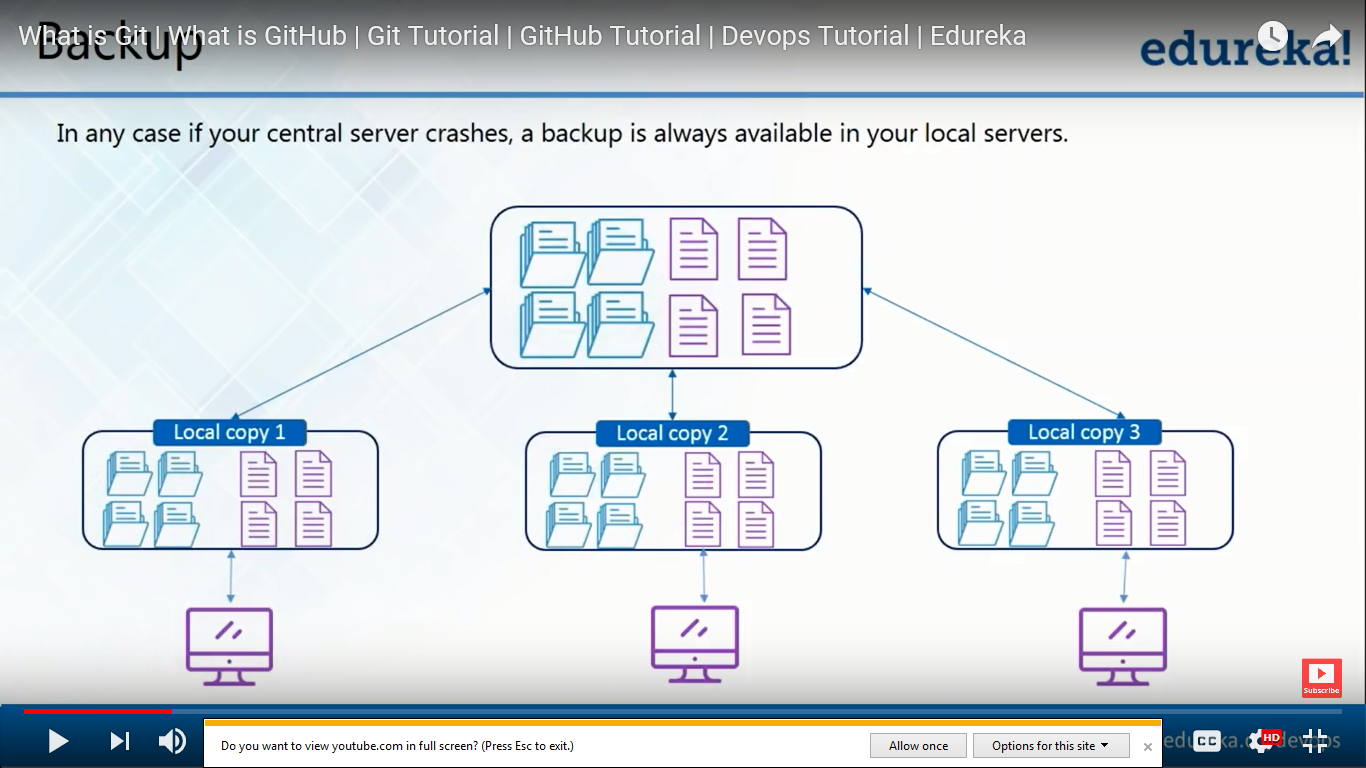
## Disadvantages Compared to Centralized Version Control

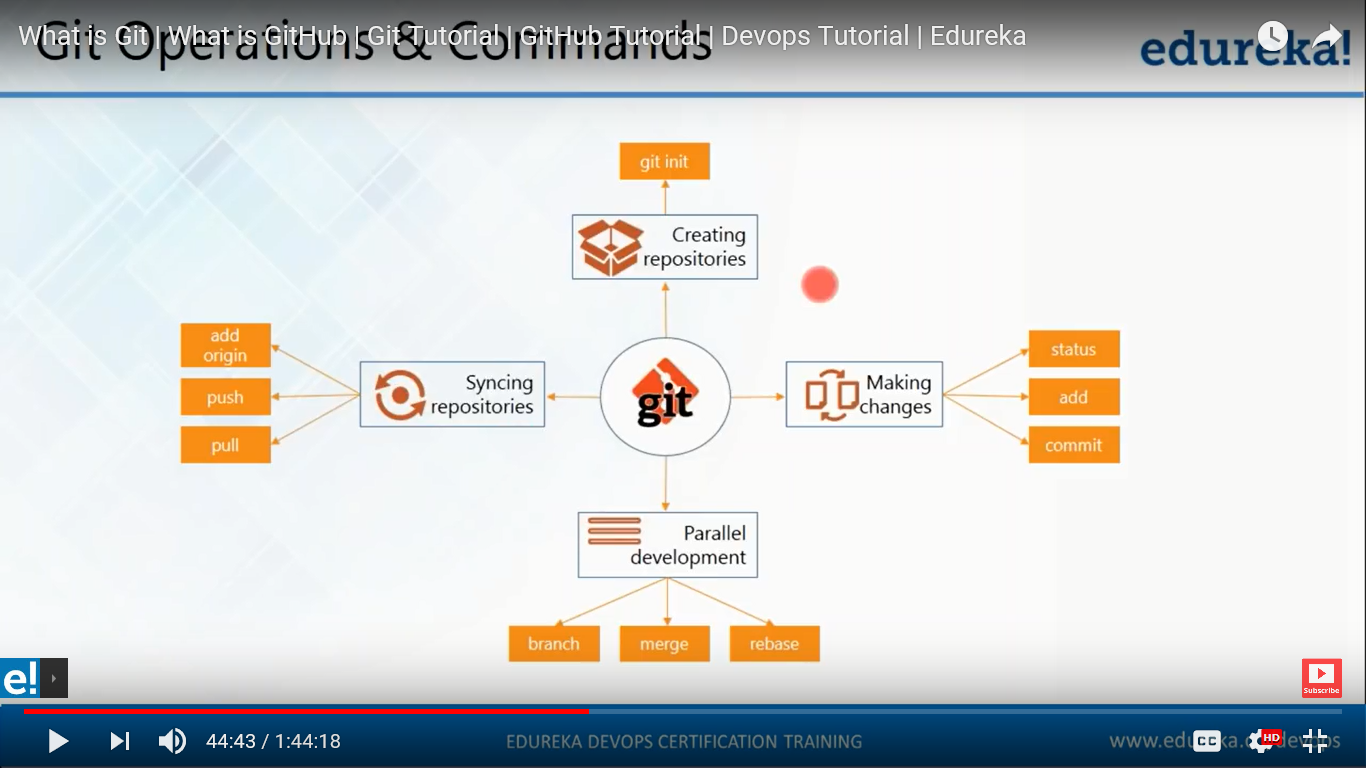
Local repository space occupation is the main disadvantage



git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com





**Getting & Creating Projects**

| **Command** | **Description** |
| --- | --- |
| git init | Initialize a local Git repository |
| git clone ssh://git@github.com/[username]/[repository-name].git | Create a local copy of a remote repository |

**Basic Snapshotting**

| **Command** | **Description** |
| --- | --- |
| git status | Check status |
| git add [file-name.txt] | Add a file to the staging area |
| git add -A | Add all new and changed files to the staging area |
| git commit -m "[commit message]" | Commit changes |
| git rm -r [file-name.txt] | Remove a file (or folder) |

**Branching & Merging**

| **Command** | **Description** |
| --- | --- |
| git branch | List branches (the asterisk denotes the current branch) |
| git branch -a | List all branches (local and remote) |
| git branch [branch name] | Create a new branch |
| git branch -d [branch name] | Delete a branch |
| git push origin --delete [branchName] | Delete a remote branch |
| git checkout -b [branch name] | Create a new branch and switch to it |
| git checkout -b [branch name] origin/[branch name] | Clone a remote branch and switch to it |
| git checkout [branch name] | Switch to a branch |
| git checkout - | Switch to the branch last checked out |
| git checkout -- [file-name.txt] | Discard changes to a file |
| git merge [branch name] | Merge a branch into the active branch |
| git merge [source branch] [target branch] | Merge a branch into a target branch |
| git stash | Stash changes in a dirty working directory |
| git stash clear | Remove all stashed entries |

**Sharing & Updating Projects**

| **Command** | **Description** |
| --- | --- |
| git push origin [branch name] | Push a branch to your remote repository |
| git push -u origin [branch name] | Push changes to remote repository (and remember the branch) |
| git push | Push changes to remote repository (remembered branch) |
| git push origin --delete [branch name] | Delete a remote branch |
| git pull | Update local repository to the newest commit |
| git pull origin [branch name] | Pull changes from remote repository |
| git remote add origin ssh://git@github.com/[username]/[repository-name].git | Add a remote repository |
| git remote set-url origin ssh://git@github.com/[username]/[repository-name].git | Set a repository's origin branch to SSH |

**Inspection & Comparison**

| **Command** | **Description** |
| --- | --- |
| git log | View changes |
| git log --summary | View changes (detailed) |
| git diff [source branch] [target branch] | Preview changes before merging |
| Sum(int b , int c)  {  Return b+c;  }  Div()  {  }  Multi()  {  } |  |