**Version Control System**

A **Version Control System (VCS)** is a software tool that helps manage changes to source code or any set of files over time. It allows multiple developers to collaborate on a project by keeping track of changes, managing conflicts, and maintaining a history of modifications. This makes it easier to revert to previous versions, compare changes, and coordinate updates.

**Version Control Systems:**

**Distributed Version Control Systems (DVCS)**:

* + In DVCS, every contributor has a complete copy of the repository, including the full history.
  + Advantages: No single point of failure, enables offline work, better collaboration.
  + Example: **Git**, **Mercurial**, **Bazaar**.

**Key Features of Version Control Systems:**

1. **Tracking Changes**: VCS records all changes made to a file or set of files, helping to keep a history.
2. **Collaboration**: Multiple people can work on the same project without overwriting each other's work.
3. **Branching and Merging**: Developers can create branches (parallel versions of a project) and merge them later, useful for managing features or experiments.
4. **Reverting**: Revert back to previous versions if bugs or issues occur.
5. **Conflict Management**: VCS helps detect conflicts when multiple users make changes to the same file and provides mechanisms to resolve them.

**Popular Version Control Systems:**

1. **Git**:
   * A distributed version control system, widely popular due to its flexibility, performance, and reliability.
   * Used with platforms like GitHub, GitLab, Bitbucket.
   * Git supports non-linear development through branching and merging.
2. **SVN (Apache Subversion)**:
   * A centralized system that allows users to track changes and version files.
   * It is well suited for smaller projects or those that need strong central control.

**Benefits of Using VCS:**

* **Backup**: VCS acts as a backup of your codebase.
* **Collaboration**: Teams can work simultaneously without stepping on each other's toes.
* **History**: You can view the history of a file and revert to any previous state.
* **Branching**: Developers can experiment with new ideas in separate branches and merge them back if successful.
* **Code Review**: VCS platforms like GitHub offer mechanisms for code review and pull requests, improving code quality.

**Common Version Control Terminology:**

1. **Repository**: A collection of files and their version history.
2. **Commit**: A record of changes made to the repository.
3. **Branch**: A parallel version of the repository for developing features separately.
4. **Merge**: Combining changes from one branch into another.
5. **Clone**: Making a copy of a repository (used in distributed systems like Git).
6. **Pull**: Fetching the latest changes from the central or remote repository.
7. **Push**: Sending your changes to the central or remote repository.
8. **Checkout**: Switching to a specific version or branch of a repository.

**Popular Hosting Platforms for Version Control:**

1. **GitHub**: A widely used platform for hosting Git repositories, offering collaboration tools like pull requests, issues, and CI/CD pipelines.
2. **GitLab**: Provides Git repository hosting with additional features like integrated CI/CD pipelines.
3. **Bitbucket**: Another Git hosting service with integrations for CI/CD and project management.

**How to Set Up the GitHub in the system**

1. Download the git from below URL

<https://git-scm.com/downloads/win>

2. Execute the .exe file for complete set up

3. Right Click on the My Computer and Select the properties

4. Click on Advanced system Settings

5. Click on Environment Variables to set Java runtime environment in Advanced tab

6. Go to System variable

7. Click edit button on Path of System Variable

8. Click New button and paste the value as C:\Program Files\Git\bin;C:\Program Files\Git\cmd

9. Click on button

10 Now Open the command prompt and write git and enter

Git Commands

#Git Commands

There will be two words

remote word means the code is present in git repo and visible to all

local word means the code is present in only your machine and not visible to

1. git fetch --all -->This will take all the remote branches/tags in local machine

2. git checkout <branch\_name>

2. git pull origin main --> This will take the latest code from main branch

3. git commit -m "text message" --> This will commit your changes with message but it will not push the code in remote

4. git push --> This will push the code in remote

#If you want to clone the AutomationTesting

1. Go to file of Intellij

2. Click on New button and select Project from Version Control

3. https://github.com/AutomationTesting0612/AutomationTesting.git

4. Click on clone button

5. This is For Testing