Tech Global

www.techglobalschool.com

Table of Contents

CLI-GIT-VCS Questions and Answers **1**
What are linux commands and can you give some examples that you use? 1
What is GIT vs VCS? 1
What are the differences between GIT and hosting platforms like GitHub? 2
What is SSH Key? 3
What is GIT branching? 3
What are some GIT commands you use? 3
What is a Pull Request? 4
What is merge conflict and how to resolve it? 4

CLI-GIT-VCS Questions and Answers

What are linux commands and can you give some examples that you use?

- Linux commands are instructions that you can give to a computer running the Linux operating system.
- They allow us to perform various tasks, like managing files, folders, and programs.
- Some examples:
    - **pwd**: use this command to check which directory you are working on.
    - **cd**: use this command to open the specified directory – navigate to the specified directory.
    - **cd ..**: use this command to move up to the parent directory– navigate to the parent directory
    - **cd /**: use this command to move up to the starting directory– root directory.
    - **cd ~**: use this command to move to the home directory.
    - **clear**: use this command to clear working terminal or command prompt.
    - **mkdir**: use this command to create a new directory.
    - **rmdir**: use this command to remove an existing empty directory.
    - **touch:** use this command to create a new file.
    - **echo content > fileName:** use this command to create a new file with an initial text.

What is GIT vs VCS?

- GIT (Global Information Tracker) is the name of the version control system (VCS).
- GIT is a software for tracking changes in any set of files, and usually used for coordinating work among programmers to collaboratively develop the source code.
- **Version Control System Types**:

1. **Centralized** - one central server and many users.
2. **Distributed** - local repositories added for each user.

Centralized Version Control System:

- There is a central server that stores the entire history of the project, including all the files and their versions.
- Developers check out a copy of the project from the central server to work on it.
- Common examples include CVS (Concurrent Versions System) and Subversion (SVN).

Advantages:

- Centralized control and management.
- Easy access control and permissions.
- Users can work offline but commits require network access.

Disadvantages:

- Single point of failure (if the central server goes down, work is disrupted).
- Slower performance for certain operations.
- Limited branching and merging capabilities.

Distributed Version Control System:

- Each developer has a complete copy of the project, including the entire history, on their local machine.
- Developers can work independently and commit changes to their local copy without needing constant network access.
- Changes can be synchronized and merged with other developers' copies when needed.
- Common examples include Git, Mercurial, and Bazaar.

Advantages:

- No single point of failure (no central server dependency).
- Faster performance, especially for common operations.
- Strong branching and merging capabilities.
- Offline work is seamless.

Disadvantages:

- Can be more complex to set up and manage, especially for large projects.

What are the differences between GIT and hosting platforms like GitHub?

- **GIT** is a software for tracking changes in any set of files, and usually used for coordinating work among programmers to collaboratively develop the source code.
- **GIT** is an open-source tool that allows.
    - team members to collaborate.
    - track and revert changes.
- **GitHub** is a platform where developers store their projects as repositories and have additional features to help them to collaborate to develop projects.
- Similar hosting websites: **GitLab, Bitbucket.**

What is SSH Key?

- An SSH key, in the context of services like GitHub, GitLab, and Bitbucket, is a secure and convenient way to authenticate and establish a secure connection between your local development environment (your computer) and the remote repository hosted on one of these platforms.
- SSH keys are an alternative to using a username and password for authentication, and they offer several benefits:
  - **Security**: SSH keys are more secure than passwords because they are cryptographically generated and provide a stronger level of protection against unauthorized access.
  - **Convenience**: Once set up, you don't need to enter your password each time you interact with a remote repository. It simplifies the authentication process.
  - **Automation**: SSH keys are essential for automated processes, such as continuous integration and deployment (CI/CD) pipelines, which require secure, passwordless authentication.

What is GIT branching?

- **Git branching** is a fundamental concept in GIT, and it plays a crucial role in managing and organizing your codebase.
- It allows us to create a separate line of development, known as a branch, from the main codebase (often referred to as the "master" or "main" branch).
- Each branch can have its own set of changes, features, or bug fixes, while the main branch remains stable.

What are some GIT commands you use?

git branch: list the existing branches.
git branch <branch-name>: create a new branch with the specified name.
git branch -m <new-branch-name>: rename the branch with the specified new name.
git checkout <branch-name>: switch to the specified branch
git checkout -b <branch-name>: create and switch to the created branch.
git branch –d <branch-name>: delete the specified branch.
git merge <branch-name>: merge changes from the specified branch.
git push origin <branch-name>: push changes from the specified branch (first time push).
git push: push changes from an existing branch (already pushed before at least once).
git pull: fetch changes from the remote repository including the main/master branch.

What is a Pull Request?

- A pull request – also referred to as a merge request – is known as a contributor/developer is ready to begin the process of merging new code changes with the main project repository.
- It is basically after you are done with your changes on your own branch, you request your changes to be merged with main remote branch.
- Steps to create a Pull Request
- Do your changes in a repository with your own branch.
- Push your code to your remote branch.
- Go to hosting website.
- Create a pull request from your branch to main branch (add reviewers if required as a policy)
- After PR is created, your reviewer (if you add any) or yourself can merge the code to main branch (master).

What is merge conflict and how to resolve it?

- This is a common issue when multiple people work on the same project.
- A **merge conflict** occurs when Git is unable to automatically merge two branches because they have competing changes.
- It occurs when multiple people make changes on the same file.
- We need to resolve the conflict before merging it to main branch (master).

To resolve a merge conflict, follow these steps:

- **Checkout the Branch:** First, make sure you are on the branch where the conflict needs to be resolved. You can do this using <**git checkout branch-name>**.
- **Pull the Latest Changes:** It's a good practice to ensure your branch is up to date with the latest changes from the remote repository by running **git pull**.
- **Locate and Open the Conflict File:** Git will indicate the conflicted files. Open the file(s) in a text editor.
- **Review the Conflict:** Inside the file, Git will mark the conflicting lines with special markers such as **<<<<<<<, =======,** and **>>>>>>>**. These markers separate the conflicting changes. Review the conflicting sections to understand what each contributor intended.
- **Manually Resolve the Conflict:** Edit the file to retain the changes you want to keep and remove the markers. This may involve combining, rewriting, or discarding code as needed.
- **Save the File:** After resolving the conflict, save the file.
- **Add and Commit the Changes:** Use **git add** conflicted-file to stage the resolved file, and then commit the changes using **git commit**.

Push the Changes: If you resolved the conflict while working on a branch with a corresponding Pull Request, push the changes to the remote repository using **git push**.