Git_Quiz

Q1 - What is a version control system?
Ans- The process of monitoring and managing changes to software code is known as version control, commonly referred to as source control. Software technologies called version control systems assist software development teams in tracking changes to source code over time.

Q2 - Why did a version control system develop? What were the necessities?
Ans- To keep track of changes and ensure that everyone on the team is working on the correct version, version control is essential. For any code, files, or assets that several team members will collaborate on, version control software should be used.
More than just managing and tracking files is required. You should be able to create and deliver things more quickly. This is particularly crucial for DevOps teams.
This is due to using the appropriate one:

- increase in visibility
- promotes global team collaboration.
- increases product delivery speed

Q3 - Define the different types of version control systems.
Ans- The two most popular types are centralized and distributed. Centralized version control systems store all the files in a central repository, while distributed version control systems store files across multiple repositories.

1. Distributed
   A distributed version control system (DVCS) allows users to access a repository from multiple locations. DVCSs are often used by developers who need to work on projects from multiple computers or who need to collaborate with other developers remotely.

2. Centralized
   A centralized version control system (CVCS) is a type of VCS where all users are working with the same central repository. This central repository can be located on a server or on a developer's local machine. Centralized version control systems are typically used in software development projects where a team of developers needs to share code and track changes.

Q4 - List a few differences between the two version control system types.
Ans-

Centralized Version Control	**Distributed version Control**
Centralized version control is the simplest form of version control in which the central repository of the server provides the latest code to the client machines.	Distributed version control is a form of version control where the complete codebase is mirrored on every developer's computer.
There are no local repositories.	There are local repositories.
Works comparatively slower.	Works faster.
Consider the entire columns for compression.	Consider columns as well as partial columns.
Focuses on synchronizing, tracking, and backing up files.	Focuses on sharing changes.
A failure in the central server terminates all the versions.	A failure in the main server does not affect the development.

Q5 - What is Git?
Ans- Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
Git is simple to use, has a small environmental impact, and performs extremely quickly. With capabilities like affordable local branching, handy staging areas, and numerous workflows, it surpasses SCM solutions like Subversion, CVS, Perforce, and ClearCase.

Q6 - List a few features of Git.
Ans- Some addictive features of git are:-

1. Tracks history
2. Free and open source
3. Supports non-linear development
4. Creates backups
5. Scalable
6. Supports collaboration
7. Branching is easier
8. Distributed development

Q7 - State any three commands of Git and why we use them.
Ans-

1. **Git clone-** Git clone is a command for downloading existing source code from a remote repository (like Github, for example). In other words, Git clone basically makes an identical copy of the latest version of a project in a repository and saves it to your computer.
   Syntax- git clone <https://name-of-the-repository-link>

2. **Git Version-** The Git version command gives us the current installed version of git.
   Syntax- git --version

3. **Git status-** The Git status command gives us all the necessary information about the current branch.
   Syntax- git status

Q8 - Is Git the same as Github? Why or Why not?
Ans- No, because Git is a software and Github is a Service. It cannot be the same as Git is a command line tool and Github is a graphical user interface. We can install Git on the local system but Github is hosted on the web, it is exclusively cloud-based.

Q9 - What is the command to get the installed version of Git?
Ans- The Git version command gives us the current installed version of git. **Syntax-** git --version

Q10 - What is the command to add all files and changes of the current folder to the staging environment of the Git repository?
Ans-
[git add .] is used to get all the changes to staging area.

Q11 - What is the difference between git status and git log commands?
Ans- Git Status is a useful command for us to gather information about the branch we're currently working on.
The status output doesn't show you any information about the commit history however. To do this, we need to use git log

Q12 - What is the command to initialize Git on the current repository?
Ans- [git init] -command is used to initialize git on the current repository.

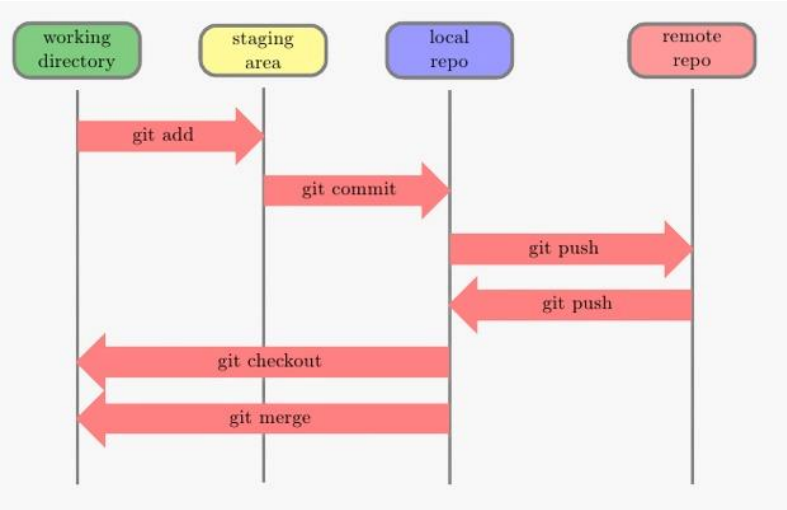
Q13 - What are the different states of a file in Git? Explain them along with the associated commands.
Ans- Each file in the working directory can be in one of two states:
Tracked: A tracked file is a file that was included in the last snapshot. These are files known to Git. Each track file can be in one of three substates: modified, staged, or committed.
Untracked: Untracked files are all files in the working directory that are not in the last snapshot and are not in the staging area.

Q14 - Git automatically adds new files to the repository and starts tracking them. True or False? Give reasons.
Ans- NO.
Git does not add them to the repository as you need to run (git add) command to add it and then push it to the repo.

Q15 - What is the command to commit the staged changes for the Git repository?
Ans- [git commit]

Q16 - What is the command to commit with the message "New email"?
Ans- [git commit -m " New email"]

Q17 - Draw the full workflow of Git and describe the diagram.
Ans-



1. Working directory - where the user does all their work.
2. Staging Area – Where users list changes made to the working directory before creating (committing) a snapshot.
3. repository – where Git keeps these changes as different versions of the project.

Basic Workflow

1. Create a local git repository.To start tracking the project, type git init into your terminal.

Git Status This isn't necessarily a step in the process, but running git status will help to keep track of progress within the workflow.

2. Add file to the staging area When run git init, it doesn't mean that any changes are being recorded yet. To tell git to start tracking a file, you must run git add <filename>.
3. Commit your changes When your file is in the staging area.
   To Review: The basic git workflow consists of creating a local repository (mkdir), adding that to the staging area (git add), and committing your changes (git commit).

Q18 - What is a branch in Git?

Ans- A branch is a different version of the repository than the main working project. This is a feature available in most modern version control systems.

A Git project can have multiple branches.

These branches are pointers to snapshots of changes.

Q19 - What is the command to create a new branch named "new-email"?

Ans- There are many ways to create a new branch:

Syntax- git branch new-email

Q20 - What is the command to move to the branch named "new-email"?

Ans- [git checkout new-email]

Q21 - What is the option, when moving to a branch, to create the branch it if it does not exist?

Ans- There are 2 ways:

- git checkout -b [branch name]
- git switch -c [branch name]

Q22 - What does the git init command does?

Ans- Create a new, empty repository using the git init command.

Used to build an existing project as a git project.

Used to initialize.

The git init command creates a .git subdirectory in the current working directory. This newly created subdirectory contains all the required metadata

Q23 - What is a fork? How is it different from clone in Git? How do you fork and clone a repository?

Ans- Fork creates a completely independent copy of a Git repository.

- Unlike forks, Git clones create linked copies that stay in sync with the target repository.

Steps to fork the repo:

1. On GitHub.com, navigate to the repository.
2. In the top-right corner of the page, click Fork.
3. Select an owner for the forked repository.
4. Optionally, add a description of your fork.
5. Create fork.

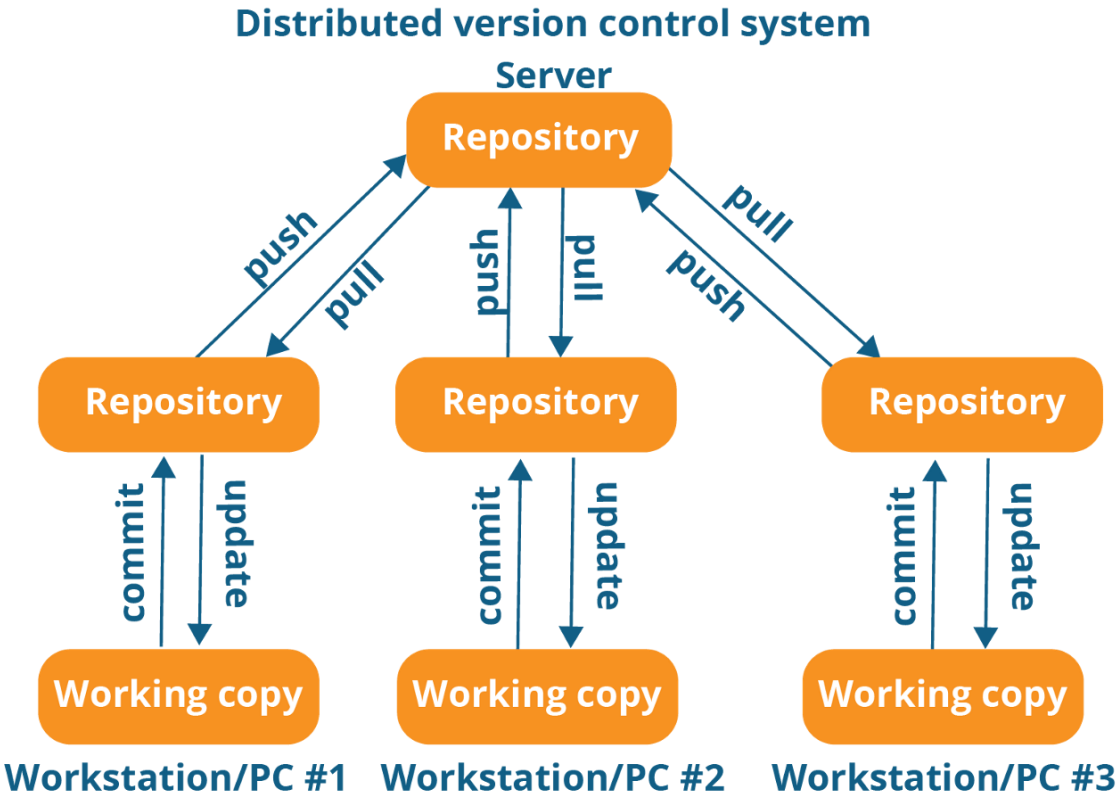
Q24 - What does 'push' mean in Git? Give the command.

Ans- The term push refers to uploading the contents of your local repository to a remote repository.

- Pushing is the act of transferring commits from your local repository to a remote repository.
- Pushing may overwrite your changes.
- git push origin [branch]

Q25 - Draw the standard architecture of two types of version control systems. (Hint - server project repo, working directories etc.) Explain the diagrams.

Ans-



THANK YOU