**Git version control**

**Explain in detail the git version control system explaining why and when we use it and all the related terms**



**Why we use:**

In an environment where people work together for the same purpose, Git becomes a well known tool that developers can use to handle changes to their codes with ease. Given that it is a distributed system, every participant in this project will not only have access to the complete history of all its files but also flexibility is enhanced particularly during offline or remote-related tasks.

**When we use:**

In any project development, and we want to keeping track of changes, managing multiple versions of code, and collaborating seamlessly across teams is very important.

**Terms:**

* Commit
* Repository / repo
* Working Directory
* Pull
* Push
* clone
* Staging Area
* Local Repositry
* Remote
* Branch
* Conflict
* Tag
* Checkout
* Head
* Origin
* Master
* Merge

**Mention all the cloud repositories that uses git version control and compare between them**

**GitHub**

GitHub is the most popular platform.

Features: It offers repository management, code review, issue tracking, and integrated CI/CD tools. GitHub Copilot, an AI pair programmer, is a standout feature.

Integrations: Integrates well with Microsoft platforms like Azure and supports connections with AWS and other services.

Ideal For: Open-source projects, remote teams, and collaborative development.

**GitLab**

GitLab provides a comprehensive suite of DevOps tools, including security testing and monitoring.

Features: Similar to GitHub, it offers repository hosting, code review, and CI/CD pipelines. It also has a strong focus on DevSecOps.

Integrations: Works well with various CI/CD tools and other DevOps platforms.

Ideal For: Teams looking for an all-in-one DevOps platform.

**Bitbucket**

Bitbucket is part of the Atlassian suite, making it ideal for teams using Jira and Confluence.

Features: Provides repository hosting, code review, and CI/CD tools. It supports both Git and Mercurial repositories.

Integrations: Seamlessly integrates with other Atlassian products.

Ideal For: Teams already using Atlassian tools.

**Azure DevOps Repos**

Azure DevOps is tightly integrated with other Microsoft services, especially Azure.

Features: Offers a full suite of DevOps tools, including Azure Repos for version control, Azure Pipelines for CI/CD, and Azure Boards for project management.

Integrations: Best suited for teams using Microsoft technologies.

Ideal For: Enterprises and teams heavily invested in the Microsoft ecosystem.

**GCP (Google Cloud Platform)**

GCP offers Cloud Source Repositories, which are fully managed private Git repositories.

Features: Provides secure storage, collaboration capabilities, and integration with other Google Cloud services. It supports CI/CD through Cloud Build and other tools.

Integrations: Works seamlessly with other GCP services and tools.

Ideal For: Teams using Google Cloud for their infrastructure and services.

**AWS Cloud**

AWS offers CodeCommit, a fully managed source control service that hosts secure Git repositories.

Features: Provides secure storage, collaboration capabilities, and integration with other AWS services. It supports CI/CD through AWS CodePipeline and other tools.

Integrations: Best suited for teams using AWS for their infrastructure and services.

Ideal For: Teams heavily invested in the AWS ecosystem.

**State all the commands that you know with the explaining of each function of them**

**git config --global user.name “Your Name”**

It is used to set the name that will be attached to your commits and tags.

**git config --global user.email “you@example.com”**

It is used set the e-mail address that will be attached to your commits and tags.

**git init**

It is used to create an empty Git repository.

**git add "file name"**

It is used to add specific to the staging area.

**git add .**

It is used to add all files in the working directory to the staging area.

**git commit**

It is used to makes sure that the changes are saved to the local repository.

**git status**

It is used to know the current state of the repository.

**git clone <url>**

It is used to create a local working copy of an existing remote repository.

**git pull**

It is used to fetch and merge changes from the remote repository to the local repository.

**git push**

It is used to transfer the commits or pushing the content from the local repository to the remote repository.

**git diff**

It is used to show changes between working directory and staging area.

**git log**

It is used to show the order of the commit history for a repository.

**git remote**

It is used to manage set of tracked repositories

**git remote add [alias] [url]**

It is used to add a git URL as an alias

**git diff --staged [file]**

It is used to shows any changes between the staging area and the repository.

**git log --oneline**

It is used to view the commit history in your Git repository.