**19 October Assignment:**

1. **Differences between Git and Github.**

| **Aspect** | **Git** | **GitHub** |
| --- | --- | --- |
| **Type** | **Distributed Version Control System (DVCS)** | **Web-based Git repository hosting platform** |
| **Location** | **Operates on your local computer** | **Hosts Git repositories in the cloud** |
| **Network Dependency** | **Doesn't require an internet connection** | **Requires an internet connection** |
| **User Interface** | **Command-line tool** | **Web-based graphical user interface** |
| **Repository Management** | **Manages repositories locally** | **Provides a platform for remote hosting** |
| **Collaboration Features** | **Limited to local team collaboration** | **Offers collaborative tools (e.g., pull requests, issues, project boards) for distributed teams** |
| **Access Control** | **Basic file permissions and SSH keys for local repositories** | **Offers granular access control, team management, and user permissions** |
| **Social Features** | **None** | **Allows users to follow, star, and contribute to projects** |
| **Issue Tracking** | **Not included** | **Built-in issue tracking and project management** |
| **Pricing** | **Free and open-source** | **Offers both free and paid plans with additional features** |
| **Alternatives** | **Can be used with other Git hosting platforms** | **One of several Git hosting platforms, including GitLab and Bitbucket** |

1. **Navigating from one version to another- GIT reset.**

**In Git, the git reset command is used to move the HEAD (the current branch's reference) and the branch pointer to a specific commit, effectively allowing you to navigate from one version of your project to another. There are three primary modes of git reset: soft, mixed, and hard, each with different effects on your working directory, staging area, and commit history. Here's an overview of these modes:**

* **Soft Reset (git reset --soft)**
  + **When you perform a soft reset, the branch pointer and HEAD move to the specified commit, but your working directory and staging area remain unchanged.**
  + **This is useful when you want to "uncommit" the most recent commit, preserving the changes in your working directory and allowing you to make additional modifications before committing again.**
* **Example:**
* **git reset --soft HEAD~1**
* **Mixed Reset (git reset --mixed, default if no mode is specified)**
  + **A mixed reset, which is the default if you don't specify a mode, moves the branch pointer and HEAD to the specified commit and also updates the staging area to match the specified commit.**
  + **This is useful when you want to uncommit and unstage changes, allowing you to make different selections before committing again.**
* **Example:**
* **git reset HEAD~1**
* **Hard Reset (git reset --hard)**
  + **A hard reset moves the branch pointer, HEAD, and staging area to the specified commit, effectively discarding all changes made after that commit.**
  + **Be cautious with this mode, as it permanently discards uncommitted changes and staged changes. It should be used with care.**
* **Example:**

**git reset --hard HEAD~1**

**Remember to use git reset with caution, especially when performing a hard reset, as it can result in data loss if not used carefully. It's a good practice to create a backup branch or commit your changes before performing reset operations to avoid accidentally losing work.**