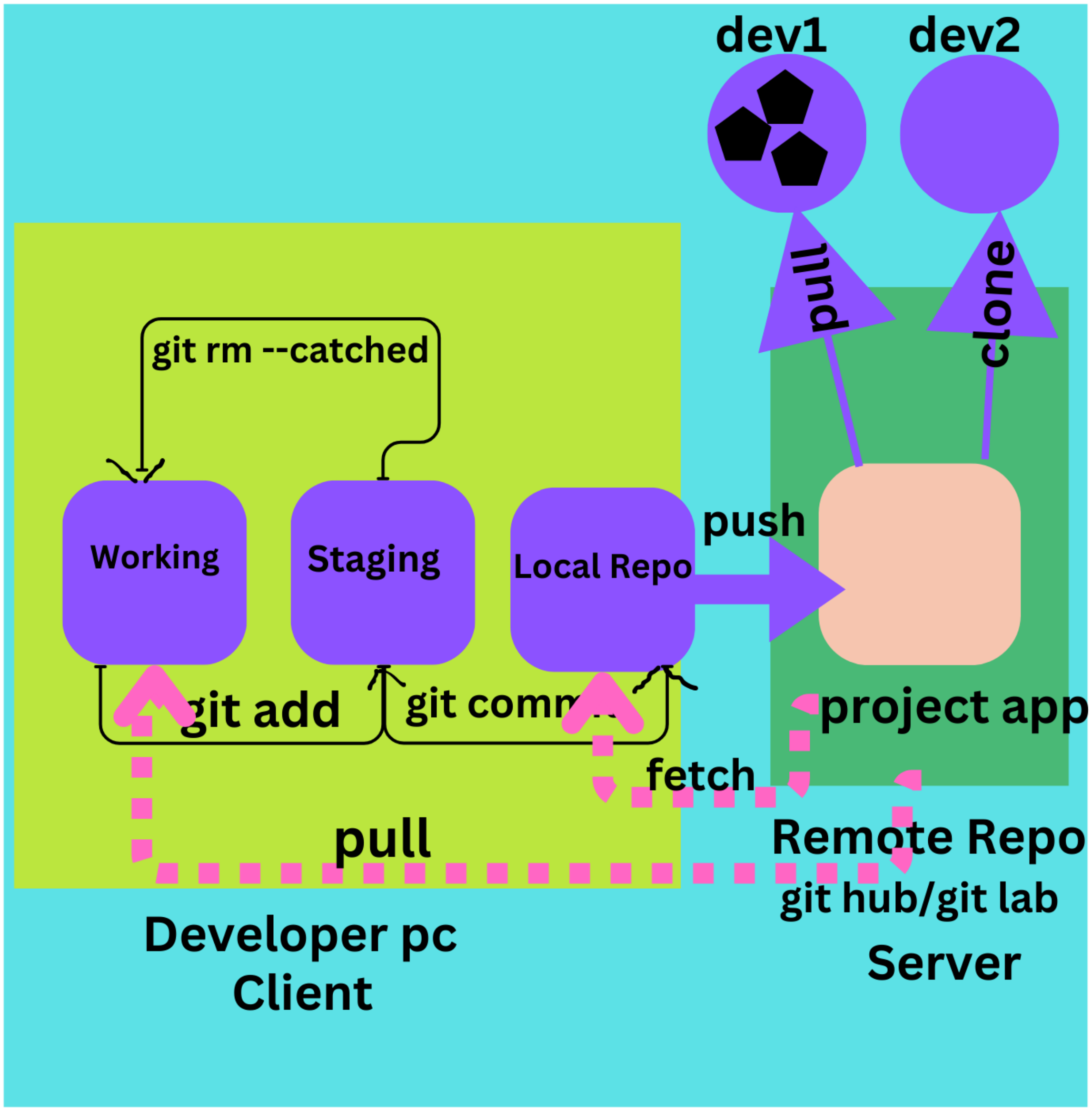
Git and Github

Git is a distributed version control system used for tracking changes in source code during software development. It allows multiple developers to work on a project simultaneously, coordinating their contributions without overwriting each other's work. Here are some key features and concepts

Architecture of git:



Git architecture contains 3 stages

1. **Working**- Where developer can work or write source laptop.

The working directory is where you modify files. These files can be in different states:

- **Untracked**: New files that haven't been added to Git.
- **Tracked**: Files that Git is aware of.

2. **Stages**- This is second stage. when your work is completed in working then file need to move staging area. Here we can use "**git add"** command. The staging area is a file (also called the index) that stores information about what will go into your next commit. It's an intermediate area where changes can be formatted and reviewed before they are actually committed.
3. **Local repo**: A commit is a snapshot of the changes in the repository. Each commit has a unique ID (a SHA-1 hash) and contains metadata such as the author, date, and commit message.

1. **what is git?**

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 a distributed version control system used for tracking changes in source code during software development. It allows multiple developers to work on a project simultaneously, coordinating their contributions without overwriting each other's work.

2. **what do you understand by the term 'version control system' ?**

   Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are **software tools that help software teams manage changes to source code over time**.

   Version Control System (VCS) is a tool that helps manage changes to files, particularly source code, over time. It tracks and records modifications, allowing multiple people to collaborate on a project without interfering with each other's work.

3. **Different types of version control system?**

   The two most popular types of version or revision control systems 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.

4. **What is github?**

   GitHub is where over 100 million developers shape the future of software, together. Contribute to the open source community, manage your Git repositories

   GitHub is a web-based platform that uses Git for version control and provides additional collaboration features for software development projects. It allows developers to store, manage, track, and collaborate on code

5. Mention some popular git hosting service
   a. GitLab
   b. Bitbucket
   c. GitHub
   d. SourceForge
   e. AWS CodeCommit
   f. Azure DevOps Server
   g. AWS Elastic Beanstalk
   h. Assembla
   i. Codebase

6. **what benefit comes with using git?**

   One of the biggest advantages of Git is its **branching capabilities**.

   Unlike centralized version control systems, Git branches are cheap and easy to merge.

   This facilitates the feature branch workflow popular with many Git users.

Feature branches provide an isolated environment for every change to your codebase.

Using Git, a distributed version control system, offers numerous benefits for managing and collaborating on software development projects:

1. **Version Control**:
    - **History Tracking**: Git maintains a complete history of every change made to the project, allowing developers to revert to previous versions if needed.
    - **Branching and Merging**: Developers can create branches to work on features or fixes independently and then merge them back into the main codebase.
2. **Collaboration**:
    - **Distributed Development**: Each developer has a complete local copy of the repository, allowing them to work offline and collaborate asynchronously.
    - **Pull Requests and Code Reviews**: Git supports workflows like pull requests, enabling code reviews and discussions before changes are integrated into the main codebase.
3. **Backup and Redundancy**:
    - **Distributed Repositories**: Since every developer has a full copy of the repository, there is built-in redundancy, reducing the risk of data loss.
4. **Efficiency and Performance**:
    - **Speed**: Git operations are generally fast due to local repository operations.
    - **Storage**: Git compresses and efficiently stores data, minimizing the storage footprint.
5. **Integration with Tools and Services**:
    - **CI/CD Integration**: Git integrates well with continuous integration and continuous deployment (CI/CD) tools, automating testing and deployment workflows.

- **Compatibility**: Many development tools and platforms support Git, making it easier to integrate into existing workflows.

6. **Flexibility and Customization**:
    - **Custom Workflows**: Teams can implement various workflows (e.g., Gitflow, GitHub flow) that suit their development practices.
    - **Hooks**: Git provides hooks to automate tasks like code quality checks, documentation generation, and more.
7. **Open Source and Community Support**:
    - **Community Contributions**: Being open-source, Git has a large community that contributes to its development, ensuring it remains up-to-date and feature-rich.

    - **Extensive Documentation**: There is extensive documentation and a wealth of resources available for learning and troubleshooting Git.

These benefits make Git a powerful tool for both individual developers and large teams, supporting efficient and effective software development practices.

7. **what is git repository?**

    A Git repository is **a central storage location for managing and tracking changes in files and directories**.

8. **how can you initialize a repository in git?**

    To create a new repo, you'll use the **git init command**. git init is a one-time command you use during the initial setup of a new repo. Executing this command will create a new .git subdirectory in your current working directory. This will also create a new main branch.

9. **Is git centralized or distributed version control system ? what is difference between them?**

   Git is a distributed version control system (DVCS).

   ## Centralized Version Control System (CVCS)

   1. **Architecture**: In a CVCS, there is a single central repository where all the versions of the code are stored.
   2. **Server Dependency**: Developers commit their changes directly to this central server. If the server goes down, no one can commit or retrieve updates.
   3. **Example**: Subversion (SVN), CVS (Concurrent Versions System).
   4. **Advantages**:
      - Easier to understand for beginners due to the simplicity of having one central repository.
      - Easier to manage user permissions and access control.
   5. **Disadvantages**:
      - Single point of failure: If the server crashes, all version history could be lost unless backups are in place.
      - Requires a constant network connection to the central server to commit or update changes.

   ## Distributed Version Control System (DVCS)

   1. **Architecture**: In a DVCS like Git, every developer has a full copy of the repository, including its entire history.
   2. **Local Repositories**: Developers can commit changes locally and later synchronize with other repositories. This allows for offline work and more robust backups.
   3. **Example**: Git, Mercurial.
   4. **Advantages**:

- Enhanced redundancy: Every developer's copy of the repository acts as a backup.
- Supports offline work: Developers can commit changes without needing a network connection.
- Easier branching and merging: DVCSs are often designed with branching and merging as first-class operations, making them more efficient.

5. **Disadvantages**:
    - Initial learning curve can be steeper for beginners due to the distributed nature.
    - Potential for more complex workflows, especially in large teams, which may require additional tools or processes to manage effectively.

In summary, Git is a distributed version control system, offering advantages like offline work, redundancy, and more efficient branching and merging compared to centralized systems.

10. **what are the git commands that you use to commit changes to your remote repository?**

To commit changes to a remote repository in Git, you typically follow these steps and use the following commands:

a. **Initialize a Git repository (if not already initialized)**: git init

b. **Add files to the staging area**:

```
git add <file>      # Add a specific file
git add .           # Add all changes in the current directory
```

c. **Commit the changes**: git commit -m "Your commit message"

d. **Add a remote repository (if not already added)**:

```
git remote add origin <remote_repository_URL>
```

e. **Push the changes to the remote repository**:

```
git push origin <branch_name>
```

Here’s a brief explanation of each step:

1. git init : Initializes a new Git repository in your current directory.
2. **git add** : Stages the changes you want to commit. You can add individual files or use `.` to add all changes.
3. **git commit** : Records the changes in the local repository with a descriptive message.
4. **git remote add origin <remote_repository_url>**: Links your local repository to a remote repository, typically hosted on a service like GitHub, GitLab, or Bitbucket.
5. **git push -u origin <branch_name>**: Uploads your local commits to the remote repository on the specified branch.
6. **git status-** check the file status like tracked or untracked
7. **git rm—— cached <file_name>:-** when you want get back from staging area to working this command is used.
8. **git restore :-**If we changed something in file at staging area then we dont want to keep that chnges in our file at that time we can use this command
9. **git branch -M main :-** If we want to change the branch master to main then we can use this command.
10. **git clone <url> :-** Hole data is copied from remote repo to working repo.
11. **git pull :-** If you want to pull perticular data or only changed data then we can use this command

11. **what are the difference between git fetch and git push?**
    - **git fetch** :
        - Downloads data from the remote repository.
        - Does not affect your working directory or current branch.
        - Allows you to review changes before merging them into your local repository.

- **git push** :
    - Uploads your local commits to the remote repository.
    - Updates the remote branch with your local changes.
    - Shares your changes with others by making them available in the remote repository.

In summary, git fetch is used to update your local repository with changes from a remote repository without altering your working directory, while git push is used to upload your local changes to a remote repository, making them available to others.