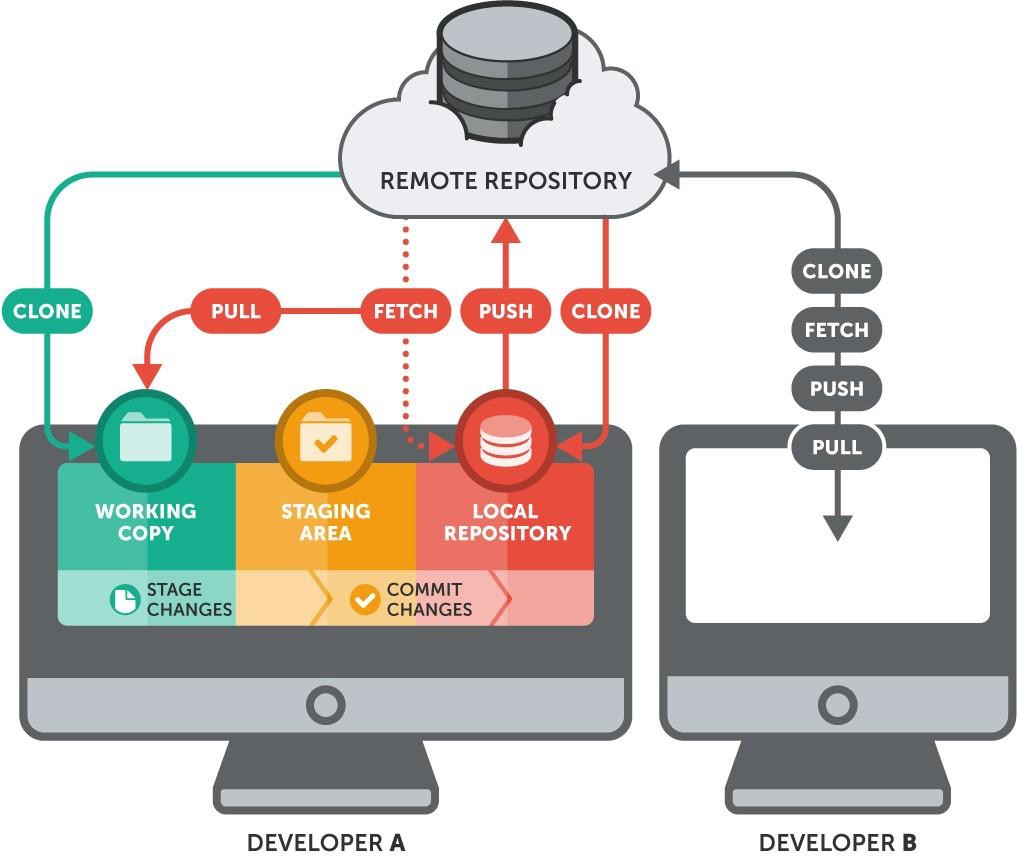
**WEEK-3: USE VERSION CONTROL SYSTEMS COMMAND TO CLONE, COMMIT, PUSH, FETCH, PULL, CHECKOUT, RESET, AND DELETE REPOSITORIES**

**A version control system (VCS)** is a software tool that helps manage changes to source code over time. It allows multiple developers to collaborate on a project, track changes, and revert to previous versions when needed. Version control systems are essential in software development to maintain code integrity, facilitate teamwork, and manage the complexity of evolving projects.

**Git** is one of the most popular distributed version control systems. Developed by Linus Torvalds in 2005, Git provides a distributed and decentralized model where each developer has a complete copy (clone) of the entire repository on their local machine. This allows developers to work offline, commit changes locally, and synchronize their work with others using remote repositories.

**GitHub** is a web-based hosting service for **Git repositories.**



Lets Now, Understand the working of **Git**

### 1. Working Directory

**Definition:** The working directory is where you make changes to your files. It contains the files you are currently editing and working on.

**Example:** If you have a project folder on your computer with files like webapp.html, sample.txt, the folder and its contents are your working directory.

### 2. Staging Area

**Definition:** The staging area (also known as the index) is a place where you prepare changes before committing them.

**Example:** Suppose you modified webapp.html and sample.txt. If you only want to commit changes made to webapp.html for now, you would add webapp.html to the staging area using git add webapp.html. The changes to sample.txt remain in the working directory but are not yet staged.

### 3. Local Repository

**Definition:** The local repository is where Git stores your commits, branches, and history on your local machine. It’s the place where you commit your changes and manage your project.

**Example:** After staging and committing changes, the Git records are stored in your local repository. You can check the history of your commits with git log, and your commits are saved in your local .git directory.

### 4. Remote Repository

**Definition:** The remote repository is a version of your project stored on a server or a platform like GitHub, GitLab, or Bitbucket. It allows you to share and collaborate on your project with others.

**Example:** If you push your local commits to a repository on GitHub using git push origin master, you are updating the remote repository at https://github.com/username/repository.git with your local changes.

**Install Git:** Open a terminal and type the following command:

**sudo apt-get install git**

Enter your password when prompted and Git will be installed on your system.

**Create a git using your mail ID – github.com in URL**

**Configure Git:** After Git is installed, you need to configure it with your name and email.

**git config --global user.name "Your Name"**

**git config --global user.email "[your-email@example.com](mailto:your-email@example.com)"**

[Note: **--global**: This flag means you are setting this configuration for all repositories on your computer. If you use **--local** instead, it would only apply to the current repository.]

Replace "Your Name" with your actual name and "your-email@example.com" with your actual email address.

**Verify Configuration:** To verify that Git has been configured correctly, you can use:

**git config --list**

This command will display your configured user.name and user.email.

**Git Commands:**

git help commit

git help <command>

1. **git init:** Initialize a new Git repository.

**mkdir myproject //also myproject repository in github.com**

**cd myproject**

**git init**

2. **git clone:**  git clone is a command used to make a copy of a repository from a remote server (like GitHub) to your local computer.

**git clone [https://github.com/sriludone/CC\_app.git](https://github.com/OpenAI/gpt-3.5.git)**

**create a webapp.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Cloud Computing Lab</title>

<style>

body {

font-family: Arial, sans-serif;

background-color: #e0f7fa;

color: #333;

margin: 0;

padding: 0;

}

header {

background-color: #00796b;

color: white;

padding: 10px;

text-align: center;

}

nav {

background-color: #004d40;

color: white;

padding: 10px;

text-align: center;

}

nav a {

color: white;

text-decoration: none;

padding: 10px;

}

nav a:hover {

background-color: #00332d;

}

.container {

padding: 20px;

}

section {

margin: 20px 0;

}

footer {

background-color: #00796b;

color: white;

text-align: center;

padding: 10px;

position: fixed;

width: 100%;

bottom: 0;

}

</style>

</head>

<body>

<header>

<h1>Cloud Computing Lab</h1>

</header>

<nav>

<a href="#profile">Profile</a>

<a href="#manual">Manual</a>

<a href="#explore">Explore</a>

<a href="#commands">Commands</a>

</nav>

<div class="container">

<section id="profile">

<h2>Profile</h2>

<p>Cloud computing is a technology that allows you to use computing resources like servers, storage, and applications over the internet instead of having to own and manage physical hardware and software. </p>

</section>

<section id="manual">

<h2>Manual</h2>

<p>The manual provides instructions on how to use the cloud computing resources available in the lab. It covers how to access services, configure settings, and troubleshoot common issues.</p>

</section>

<section id="explore">

<h2>Explore</h2>

<p>Explore various cloud computing technologies and services. This section provides an overview of available tools and resources for further learning and experimentation.</p>

</section>

<section id="commands">

<h2>Commands</h2>

<p>There are some common commands used in cloud computing environments: clone, commit, push, fetch, pull, checkout, reset, and delete repositories </p>

</section>

</div>

<footer>

<p>&copy; 2024 Cloud Computing Lab</p>

</footer>

</body>

</html>

**firefox webapp.html**

**3. git mv:** Change the file name

**git mv file.txt hello.txt**

**4. git add:**  adding files from working directory to staging area for the next commit.

**git add webapp.html**

**5. git commit:** Record changes to the repository.

**git commit -m "webapp.html"**

6. **git branch:**  a branch in Git is like a separate version of your project. It allows you to make changes or add new features without affecting the main version of your project.

**EX:**

**Main Branch (Master)**: This is your original branch with all the documents you currently have.

**New Branch**: You create a new branch to add your new files. This branch is like a separate copy of your book where you work on the new chapter.

**Steps:**

**1. Create a Branch:**

**git branch newapp**

**2. Switch to the Branch: switch to `newapp` to start working on it.**

**git checkout newapp**

**3. Add Changes: write your new chapter in this branch.**

**create another file sample.txt  
git add sample.txt**

**4. Merge the Branch: merge `newapp` back into the main branch so everyone can see the newapp.**

**git checkout master**

**git merge newapp**

**now, check the master branch**

**git checkout master**

**ls**

**5. Delete the Branch: After merging, you might delete the `newapp` branch if you no longer need it.**

**git branch -d newapp**

1. **git push:** Upload local changes to a remote repository.

we have 2 ways- by using username and password

**git push origin master**

-by creating token, follow bellow commands

**Check where your code will be pushed**:

git remote -v

This shows a list of remote repositories linked to your local repository. It helps you verify where you are pushing your changes.

**Add a new remote repository**:

git remote add origin <https://github.com/sriludone/myproject.git>

This command sets up a link to a remote repository called origin (which is a common name for the main remote repository) where you will push your changes.

**Update the remote repository URL**:

git remote set-url origin https://ghp\_Wyn7mLmRylz3y93C5iv2Hn0FUyE5je3Knymq@github.com/sriludone/myproject.git

This updates the URL of the remote repository, often used when you need to change the address or update credentials.

**Upload your changes**:

**git push origin master** //Once the remote is configured, you can push your changes

This uploads your local changes to the master branch of the remote repository called origin.

8. **git pull:** Fetch and merge changes from a remote repository.

**git pull origin master**

You and a friend are working on the same GitHub repository. You make changes to a file and push them to the remote repository. Your friend then makes their own changes and pushes those. Now, you want to update your local repository to include your friend's changes.

# First Navigate to your local repository directory

**cd sree/myproject**

# Fetch and merge changes from the remote repository's master branch into your local master branch

**git pull origin master**

**git pull:** This command combines `git fetch` and `git merge`. It fetches changes from the remote repository and then merges them into your current branch.

**origin:** This specifies the name of the remote repository (`origin` by default).

**master:** This specifies the branch from the remote repository that you want to pull changes from.

**Before git pull**: Your local repository might be out-of-date with the remote repository.

**After git pull:** Your local repository is updated with the latest changes from the remote repository, and any new commits from your friend are now part of your local branch.

9. **git fetch:**  Download changes from a remote repository without merging.

**git fetch origin**

10. **git checkout:** Switch branches or restore working tree files.

**git checkout brachname**

11. **git merge:** Merge changes from one branch into another.

**git checkout master**

**git merge newbranchname**

12. **git reset:** Undo changes and move the current branch to a previous state.

git reset --hard HEAD~1 //It requires atleaset 2 commits so create another file

**echo "Second commit" > secondfile.txt**

**git add secondfile.txt**

**git commit -m "Add secondfile.txt"**

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

13. **git status:**  Show the status of changes in the working directory.

**git status**

14. **git log:** Show the commit history.

**git log**

15. **git diff:** Show changes between commits, commit and working tree, etc.

**git diff**

16. **git rm:**  Remove files from the working directory and staging area.

**git rm file.txt**