Introduction to Git

SSD Lab-1

What is Git?

Git is a version control system used for tracking changes in computer files. It is generally used for source code management in software development.

What is Version Control System ?

A **Version Control System (VCS)** is a tool that helps you keep track of changes to your files over time. It’s like a "time machine" for your project.

In short, a VCS helps you:

- Save your progress.
- Recover from mistakes.
- Work safely on new features.

```c
#include <stdio.h>

int add(int a, int b)
{
    return a+b;
}

int main()
{
    int a=5;
    int b=3;
    printf("%d\n",add(a,b));
}
```

```c
#include <stdio.h>

int add(int a, int b)
{
    return a+b;
}
int sub(int a, int b)
{
    return a-b;
}

int main()
{
    int a=5;
    int b=3;
    printf("%d\n",add(a,b));
    printf("%d\n",sub(a,b));
}
```

```c
#include <stdio.h>

int calc(int a, int b,char sign)
{
    if(sign=='+')
    {
        return a+b;
    }
    else if(sign=='-')
    {
        return a-b;
    }
    else
    {
        return -1;
    }
}

int main()
{
    int a=5;
    int b=3;
    printf("%d\n",calc(a,b,'+'));
    printf("%d\n",calc(a,b,'-'));
}
```

An Example

Why is Version Control System Needed ?

Save time by preventing data loss.

Make it easy to fix mistakes.

Enable smooth experimentation.

Keep your project history organized.

Support collaboration.

Do You Have Git???
Let’s Install

Before installing Git, let's check if it's already installed on your system.

Open the Terminal

- **Linux**: Open the **Terminal** using `Ctrl + Alt + T`.
- **Mac**: Open **Terminal** from the Applications > Utilities folder.
- **Windows**: Open **Command Prompt** or **Git Bash** (if installed).

Execute command “`git --version`” in terminal/CMD

- If you see a version number (e.g., `git version 2.x.x`), Git is already installed.
- If you see an error saying `'git' is not recognized`, Git is not installed, and you'll need to install it using the steps below.

Git on windows

➔ **Download Git for windows .exe file**

Go to following link and download the relevant git installer file:

https://git-scm.com/downloads/win

➔ **Install Git**

Execute/ Run the .exe file and follow the instructions of the installer (Try to proceed with default or recommended choices).

➔ **Verify the Installation**

After installation, you can launch Git Bash from the start Menu or desktop Shortcut (if created during installation).

Execute command “git --version” in terminal or Git Bash

➢ If you see a version number (e.g., `git version 2.x.x`), Git is successfully installed.
➢ If you see an error saying `'git' is not recognized`, Git is not installed correctly, and you need to recheck and install it again.

Git on Mac

Using Homebrew (Package Manager)

➔ Install Homebrew (if not already installed)

Open Terminal and type the following command:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

You can find more details on Homebrew installation at https://brew.sh/.

➔ Install Git using Homebrew

Execute command “brew install git” in terminal

➔ Verify the Installation

Execute command “git --version” in terminal

- ➢ If you see a version number (e.g., `git version 2.x.x`), Git is successfully installed.
- ➢ If you see an error saying `'git' is not recognized`, Git is not installed correctly, and you need to recheck and install it again.

Git in Linux

➔ **Update Package Lists**

Open Terminal and type the following command:

“sudo apt-get update”

➔ **Install Git using apt-get/dnf**

For Ubuntu/Debian-based Systems

Execute command “sudo apt-get install git-all”

For Fedora-based Systems

Execute command “sudo dnf install git-all”

➔ **Verify the Installation**

Execute command “git --version” in terminal

➢ If you see a version number (e.g., `git version 2.x.x`), Git is successfully installed.
➢ If you see an error saying `'git' is not recognized`, Git is not installed correctly, and you need to recheck and install it again.

Basic Git Commands

Setup & Configuration

`git config`

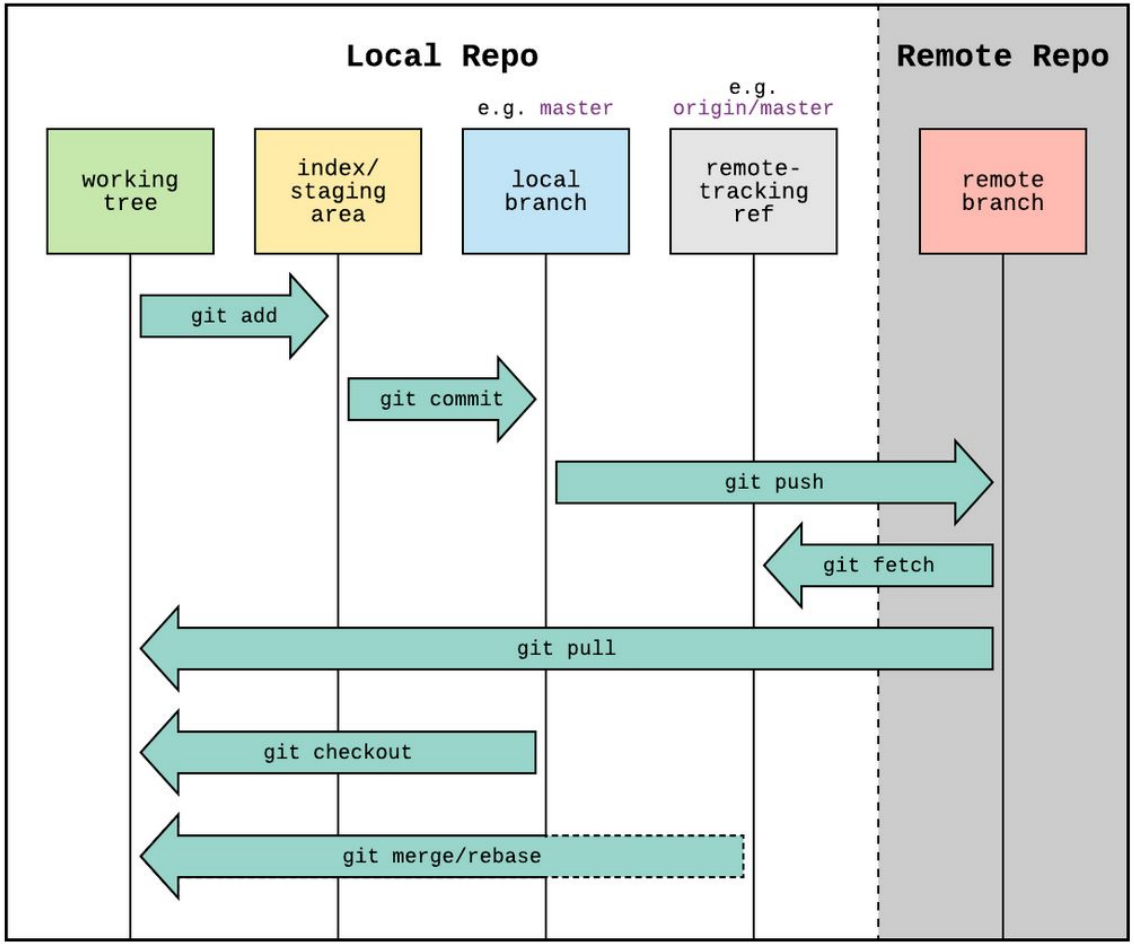
- Set user information and preferences.
- Examples:
  - **`git config --global user.name "Your Name"`**
  - **`git config --global user.email "your.email@example.com"`**

`git init`

- Initialize a new Git repository in the current directory.

`git clone`

- Clone an existing repository from a remote location.
- Example: **`git clone <repository_url>`**


Local Repo
Remote Repo
e.g. master
e.g. origin/master
working tree
index/ staging area
local branch
remote-tracking ref
remote branch
git add
git commit
git push
git fetch
git pull
git checkout
git merge/rebase

Basic Workflow

`git status`

- Show the current status of the working directory and staging area.

`git add`

- Stage changes for the next commit.
- Examples:
  - **`git add <file>`**
  - **`git add .`** (stage all changes)

`git commit`

- Save changes to the repository with a message.
- Example: **`git commit -m "Commit message"`**

Basic Workflow

`git push`

- Push commits to the remote repository.
- Example: **`git push origin main`**

`git pull`

- Fetch and merge changes from the remote repository.
- Example: **`git pull origin main`**

`git fetch`

- Fetch updates from the remote repository without merging.

Branching & Merging

`git branch`

- List, create, or delete branches.
- Examples:
  - `git branch` (list branches)
  - `git branch <branch_name>` (create a new branch)
  - `git branch -d <branch_name>` (delete a branch)

`git checkout`

- Switch branches or restore files.
- Examples:
  - `git checkout <branch_name>`
  - `git checkout -b <new_branch_name>` (create and switch to a new branch)

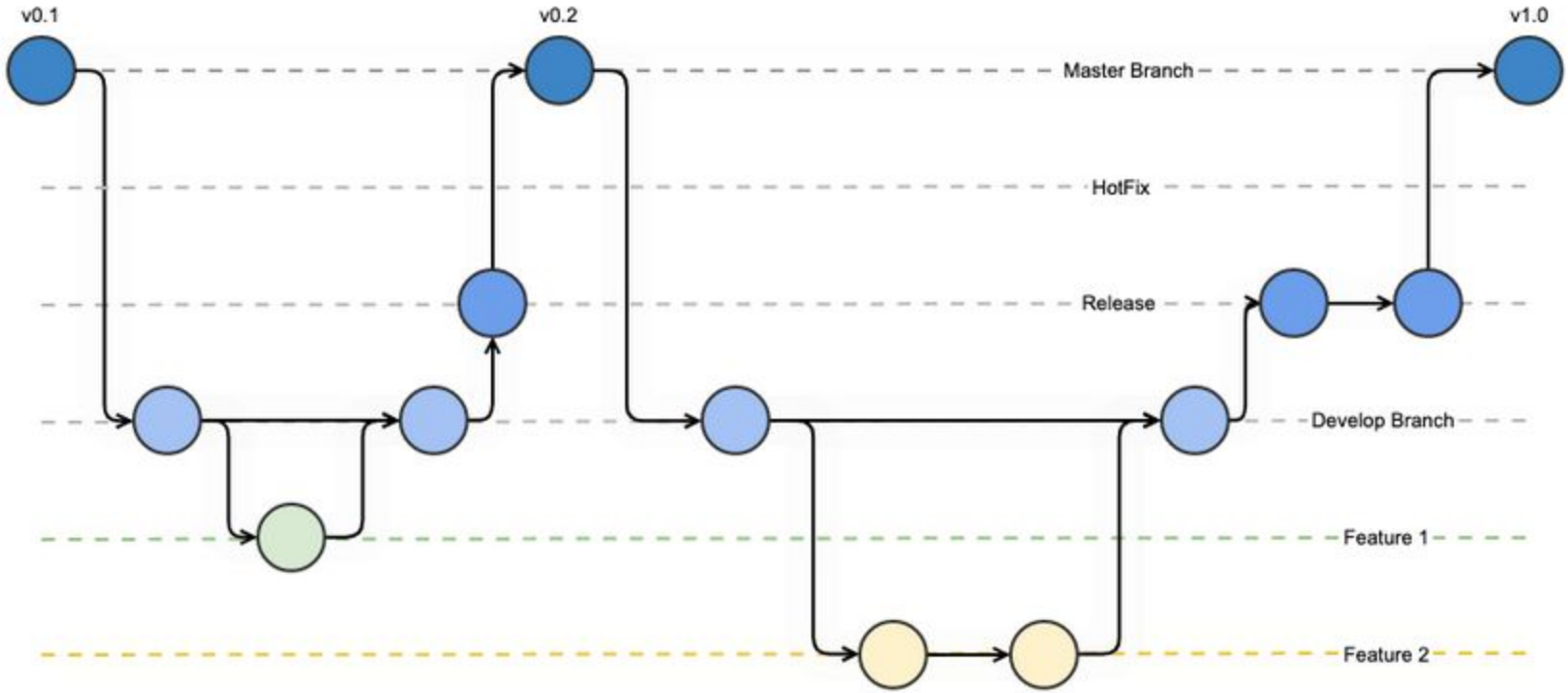
Branching & Merging

`git switch`

- Switch branches (preferred over `checkout` for branch switching).
- Example: `git switch <branch_name>`

`git merge`

- Merge a branch into the current branch.
- Example: `git merge <branch_name>`


v0.1
v0.2
v1.0
Master Branch
HotFix
Release
Develop Branch
Feature 1
Feature 2

Inspecting History

`git log`

- View the commit history.
- Example: `git log --oneline --graph` (compact graphical view)

`git show`

- Show details of a specific commit.
- Example: `git show <commit_hash>`

`git diff`

- Show differences between changes.
- Examples:
  - `git diff` (unstaged changes)
  - `git diff --staged` (staged changes)

Undoing changes

`git restore`

- Undo changes in the working directory.
- Examples:
  - `git restore <file>`
  - `git restore --staged <file>` (unstage changes)

`git reset`

- Undo commits or changes.
- Examples:
  - `git reset <file>` (unstage a file)
  - `git reset --soft <commit>` (keep changes, reset HEAD)
  - `git reset --hard <commit>` (discard changes)

`git revert`

- Create a new commit to reverse a specific commit.
- Example: `git revert <commit_hash>`

About GitHub

GitHub is a code hosting platform for collaboration and version control.

Forking a repository means making a copy on your GitHub account.

Pull Requests - an important functionality provided by GitHub.

Steps for a pull request

1. Branch out / Fork the repo (most times)
2. Make changes
3. Pull request to merge

Connect your GitHub to git in your local Machine

Setting up GitHub Account

- Go to https://github.com/ and create an account if you don't already have one.
- If you already had it, Go to **Settings → Developer Settings → Personal Access Tokens → Tokens (classic) → Generate new token**.
- Give all the permissions that you want to have to the token and Set an expiration date and generate the token.
- Copy the token and store it somewhere(it won't be shown again).
- The token will be used as password while pushing the code online.

Configure GitHub locally

- Open a terminal or Git Bash.
- Enter following commands:
  - `git config --global user.name "Your Name"`
  - `git config --global user.email "your_email@gmail.com"`
- Verify the configuration
  - Enter ‘`git config --global --list`’ in terminal
  - It will get your details in the terminal if the config is successful.
- Get rid of entering password every time (optional)
  - Enter ‘`git config --global credential.helper store`’ in terminal
  - When you use your token as password for the first time it will store it for further interactions.

Add your ssh key to your GitHub (Optional)

- Go to Terminal
- Enter following commands if you don't already have ssh key:
  - `ssh-keygen -t ed25519 -C "your-email@example.com"`
  - `eval "$(ssh-agent -s)"`
  - `ssh-add ~/.ssh/id_ed25519`
  - `cat ~/.ssh/id_ed25519.pub`
- Copy the ssh key you got by above commands
- Go to GitHub and Go to **Settings → SSH and GPG keys → New SSH key**.
- Paste the copied key and give it a title, then click **Add SSH Key**.
- Verify the connection by executing command `ssh -T git@github.com` in your terminal

That's it! You're all set to work seamlessly with GitHub

Activity!

Go to Moodle to access the Activity Document.

Try this website, It’s fun

https://learngitbranching.js.org/