

SEPM EXPERIMENT 2

NAME: Meet Kadam

DIV: T12

ROLL NO: 43

AIM: To understand version control system, install git and create a github account

THEORY:

Understanding version control systems (VCS) like Git can seem daunting at first, but it's much easier once you break it down step by step. Here's a simple theory on how to approach learning and understanding version control:

1. Think of Your Project as a Book

- Imagine your project as a book that's constantly being written and edited by different people (or even yourself).
- Every time you make a change to the book (add a chapter, edit a paragraph), you might want to track those changes.
- Version control allows you to do this by keeping a history of every change made to the "book" and even allowing you to go back and read earlier drafts if needed.

2. Understanding the Key Concepts

- **Repository:** This is like the bookshelf where all versions of your book (or project) are stored.
- **Commit:** Every time you make a change and save it, it's like writing a new page in the book and marking it with a timestamp. These "pages" (commits) are stored and can be revisited or reverted.

- **Branch:** Imagine you want to explore a new storyline in your book, but you don't want to mess with the main plot. A branch allows you to write the new storyline in parallel, without changing the main book.
- **Merge:** When you're happy with your alternate storyline, you can combine it back into the main plot. This is where changes from one branch get merged into another.
- **Clone:** This is like making a copy of the book from the bookshelf. You take it home (to your local computer) and start working on it yourself.
- **Push/Pull:** After making changes to your local copy (cloned repo), you push your changes to the central repository (like publishing your edited chapters), and you can pull down updates that other collaborators have made to keep your copy up to date.

3. The Workflow

- **Create/Clone Repository:** You start with a repository, either by creating a new one or cloning an existing one.
- **Edit Locally:** You make changes to files on your local computer (write or modify content).
- **Stage Changes:** Before committing, you stage the changes (decide which changes are going to be committed).
- **Commit:** You save your work, marking it with a message that explains what you've done.
- **Push to Central Repository:** You push your changes to the central repository so that others can access your work.
- **Pull from Repository:** You fetch updates from others' changes to ensure you have the most recent version.

4. Branching and Merging

- Branching allows you to experiment freely and work on separate parts of a project in isolation. When you're ready, you can merge these changes into the main project.

- It's a little like writing a first draft (branch), refining it and then blending it into the final book (main branch).

5. Practice, Practice, Practice

- Just like learning a new language or any other skill, the more you use version control, the easier it becomes. Start with small projects, and as you gain comfort, expand into more complex workflows.

6. Use Visual Tools

- Tools like GitHub Desktop, GitKraken, or Sourcetree provide graphical interfaces to help visualize your branches and commits. This can make it easier to understand what's going on without diving into the command line.

1. Installing Git For

Windows:

1. Go to the official Git website: git-scm.com.
2. Click on the "Download" button for Windows.
3. Once the .exe file is downloaded, run the installer.
4. During installation, you can leave most of the default settings, but pay attention to these options:
 - **Choosing the editor:** Git will prompt you to select an editor (e.g., Vim, Nano, Notepad++). If you're unsure, the default is fine, or you can select your preferred editor.
 - **Adjusting your PATH:** Choose "Git from the command line and also from 3rd-party software."
 - **Line endings:** Choose the recommended option (usually "Checkout Windows-style, commit Unix-style line endings").
5. Once installed, open Git Bash (you'll see it in your Start Menu). You should now have Git installed!

For macOS:

1. Open the **Terminal**.
2. Type `git --version` and press Enter. If Git is not already installed, macOS will prompt you to install it.
3. Follow the on-screen instructions to install Git (usually via Xcode Command Line Tools).
4. Once installed, check by typing `git --version` again to verify.

For Linux:

1. Open your terminal.
2. For **Ubuntu/Debian**-based distributions, type: `bash` Copy `sudo apt update`
`sudo apt install git`

For **Fedora**-based distributions,

type: `bash` Copy

`sudo dnf install git`

3. Once installed, type `git --version` to verify that it was installed successfully.

2. Creating a GitHub Account

1. **Go to GitHub:** Head to github.com.
2. **Sign up:** Click the "Sign up" button in the top-right corner.
 - Enter your username, email address, and password.
 - GitHub will ask you to verify your email, so go check your inbox for a confirmation link.
3. **Choose a Plan:** GitHub offers a free plan, which is great for most users. Select the free plan (or another plan if needed).

4. **Verify Account:** You may need to verify that you're not a robot by completing a CAPTCHA.
 5. **Complete Setup:** GitHub will guide you through a few additional steps, like configuring your preferences for notifications, etc.
 6. Once completed, you can log into your account!
-

Setting Up Git with GitHub

Once you have both Git installed and a GitHub account created, you need to configure Git to use your GitHub account.

1. **Set your name and email in Git** (important for commit history): Open a terminal or Git Bash and type the following commands (replace with your name and email):

```
bash Copy git config --global user.name "Your Name" git
```

```
config --global user.email "your_email@example.com"
```

This associates your name and email with your commits.

2. **Verify your configuration:** You can check if Git is correctly set up by typing:

```
bash Copy
```

```
git config --list
```

3. **Generate an SSH key (Optional but recommended):** If you want to push code securely to GitHub without entering your password every time, you should set up an SSH key. ○ Open the terminal and type:

```
bash Copy ssh-keygen -t ed25519 -C
```

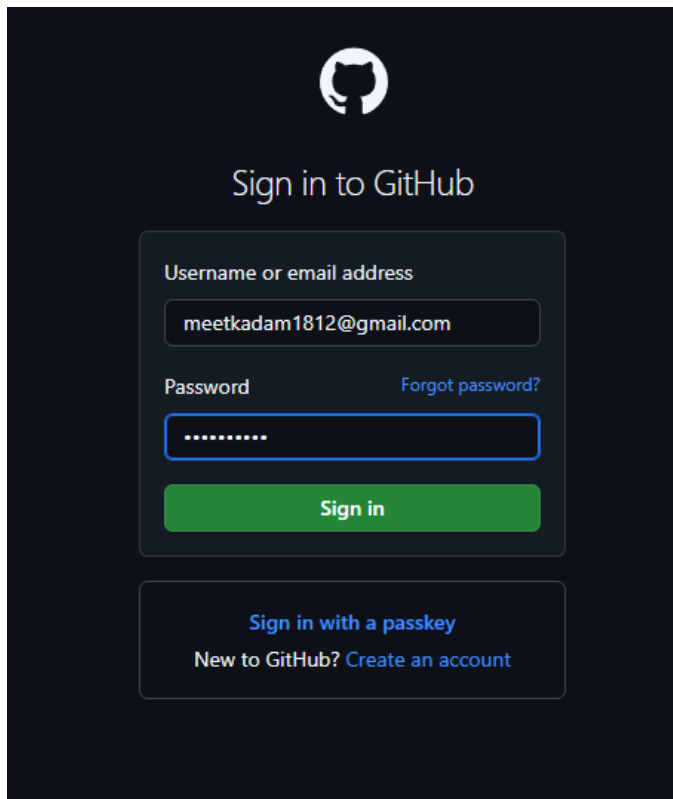
```
"your_email@example.com" Press Enter to accept the
```

default file location.

- Once generated, you can find your SSH key in the ~/.ssh folder.

- Add the key to your GitHub account by copying the SSH key and pasting it into your GitHub account settings under **SSH and GPG keys**.

If you prefer not to use SSH, you can use HTTPS to authenticate.



The image shows the GitHub sign-in page with a dark background. At the top is the GitHub logo. Below it, the text "Sign in to GitHub" is centered. The main form contains two input fields: "Username or email address" with the value "meetkadam1812@gmail.com" and "Password" with masked characters. A "Forgot password?" link is next to the password field. A green "Sign in" button is below the fields. At the bottom, there is a link "Sign in with a passkey" and a link "New to GitHub? Create an account".

Sign in to GitHub

Username or email address
meetkadam1812@gmail.com

Password [Forgot password?](#)
.....

Sign in

[Sign in with a passkey](#)
New to GitHub? [Create an account](#)



Device verification



Email

We just sent your authentication code via email to m*****@gmail.com. The code will expire at 11:59AM IST.

Device Verification Code

Verify

Having trouble verifying via email?

- [Re-send the authentication code](#)
- [Try GitHub Mobile for simplified device verification](#)

If you'd like to require verification on every sign in, consider enabling [two-factor authentication](#) on your account.

Overview Repositories Projects Packages Stars



RajmeetOP

Edit profile

Popular repositories

You don't have any public repositories yet.

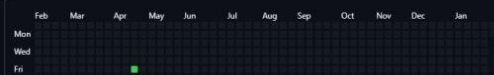
Customize your pins

1 contribution in the last year

Contribution settings

2025

2024



Learn how we count contributions

Less More

This is your contribution graph. Your first square is for joining GitHub and you'll earn more as you make [additional contributions](#). More contributions means a higher contrast square color for that day. Over time, your chart might start looking [something like this](#).

We have a quick guide that will show you how to create your first repository and earn more green squares!

[Read the Hello World guide](#)

Contribution activity

January 2025

RajmeetOP has no activity yet for this period.


Show more activity

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 MeetKadam

Repository name *

SEPM_meet_43

SEPM_meet_43 is available.

Great repository names are short and memorable. Need inspiration? How about **studious-octo-palm-tree** ?

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)


Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)


 You are creating a public repository in your personal account.

Create repository


 SEPM_meet_43

Public

 Pin

 Unwatch 1

 Fork 0

 Star 0



Start coding with Codespaces

Add a README file and start coding in a secure, configurable, and dedicated development environment.

Create a codespace






Add collaborators to this repository

Search for people using their GitHub username or email address.

Invite collaborators

Quick setup — if you've done this kind of thing before

 Set up in Desktop or  HTTPS  SSH https://github.com/MeetKadam/SEPM_meet_43.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# SEPM_meet_43" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/MeetKadam/SEPM_meet_43.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/MeetKadam/SEPM_meet_43.git
git branch -M main
git push -u origin main
```