Docsify setup document

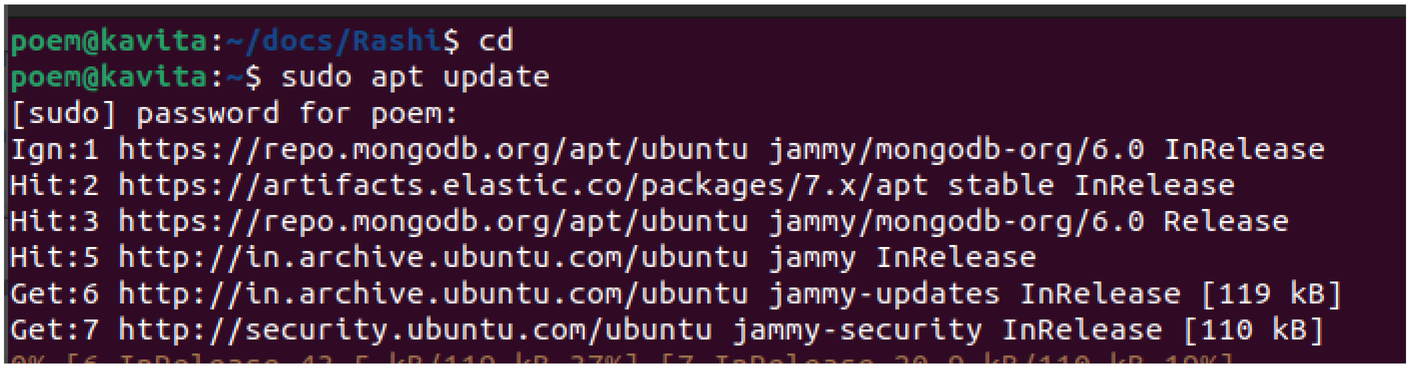
Overview

> podman is a containerization tool for Linux-based operating systems. You can follow these steps to install Podman on Ubuntu:

system updates:

- First, update your system so that you have the latest package progress and dependencies:

> sudo apt update

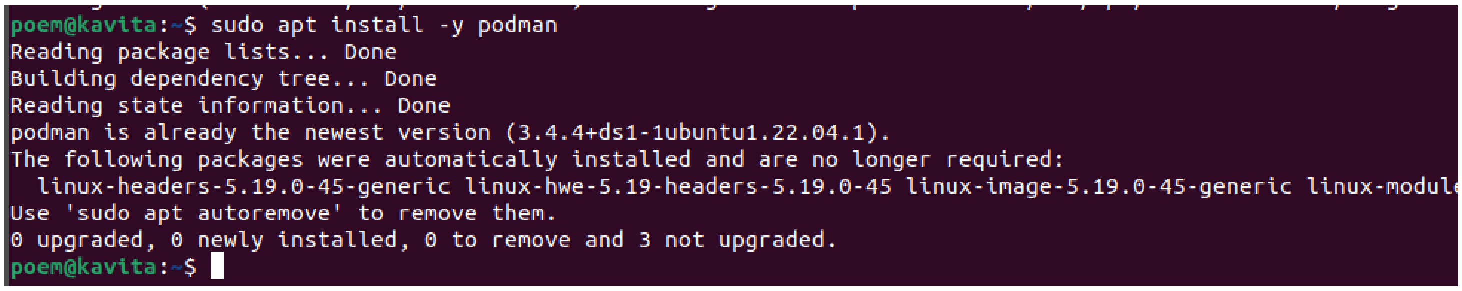


```
poem@kavita:~/docs/Rashi$ cd
poem@kavita:~$ sudo apt update
[sudo] password for poem:
Ign:1 https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0 InRelease
Hit:2 https://artifacts.elastic.co/packages/7.x/apt stable InRelease
Hit:3 https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0 Release
Hit:5 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
```

Install Podman

- Now install the podman package

> sudo apt install -y podman



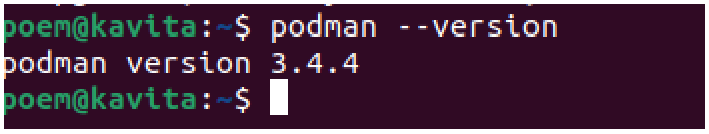
```
poem@kavita:~$ sudo apt install -y podman
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
podman is already the newest version (3.4.4+ds1-1ubuntu1.22.04.1).
The following packages were automatically installed and are no longer required:
  linux-headers-5.19.0-45-generic linux-hwe-5.19-headers-5.19.0-45 linux-image-5.19.0-45-generic linux-module
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
poem@kavita:~$
```

- By doing this flag ***"(-y)"*** you will not need to avoid any confirmation in the installation process.

Check Podman Version

- podman is installed, you can check its version using the podman --version command:

> podman --version



```
poem@kavita:~$ podman --version
podman version 3.4.4
poem@kavita:~$
```

- If Padman is successfully installed in your system, then it will show you the version number.

2. Setup docsify in podman container

- To set up Docsify in Podman container you need to follow below steps:

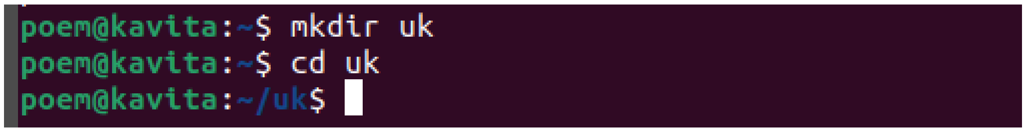
Create a Directory for Docsify

- Create a directory where you'll keep your Docsify documentation files. For example

> - You can create a new folder. then you can use the "mkdir folder_name" command in Terminal to create a new folder.

> - "mkdir uk"

> - Navigate Inside the Folder
>   "cd uk"

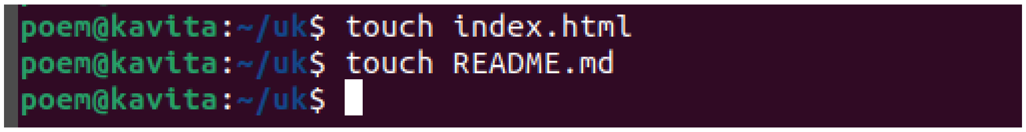


```
poem@kavita:~$ mkdir uk
poem@kavita:~$ cd uk
poem@kavita:~/uk$
```

Create the File and Open it for Editing:

> "touch index.html"

> touch README.md

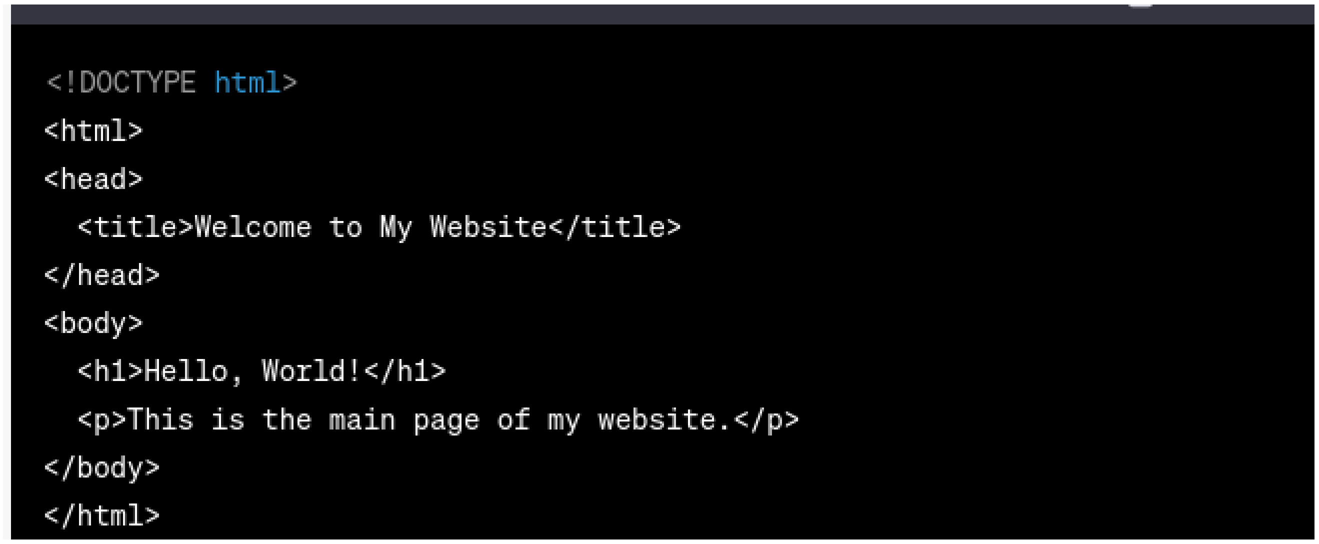


```
poem@kavita:~/uk$ touch index.html
poem@kavita:~/uk$ touch README.md
poem@kavita:~/uk$
```

- You can use a text editor like vim to create and edit the file. Open your terminal and run:

- Then after that we have to write the code of html in Index.html

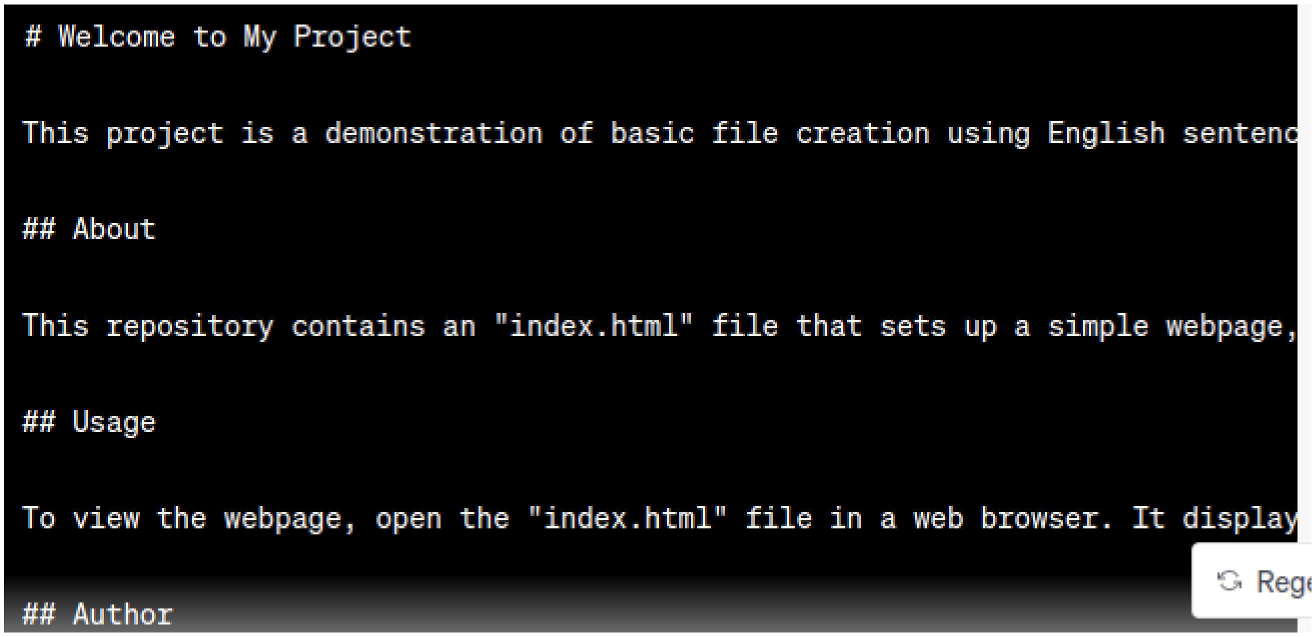
> vim index.html



```
<!DOCTYPE html>
<html>
<head>
  <title>Welcome to My Website</title>
</head>
<body>
  <h1>Hello, World!</h1>
  <p>This is the main page of my website.</p>
</body>
</html>
```

Create a file named "index.html" in your preferred text editor and paste the following content

> vim README.md



```
# Welcome to My Project

This project is a demonstration of basic file creation using English sentenc

## About

This repository contains an "index.html" file that sets up a simple webpage,

## Usage

To view the webpage, open the "index.html" file in a web browser. It display

## Author
```

Create a file named "Readme.md" in your preferred text editor and paste the following content

Create Dockerfile

```
FROM node:latest
LABEL description="A demo Dockerfile for build Docsify."
WORKDIR /docs
RUN npm install -g docsify-cli@latest
EXPOSE 3000/tcp
ENTRYPOINT docsify serve .
```

Run the New Container and Set the Directory

- Run the new container and enter the desired directory where you want to work with Docsify. Replace **/path/to/your/directory** with the actual path to your desired directory:

> ***let me explain the options -d, -p, and -v that are used in Podman containers***

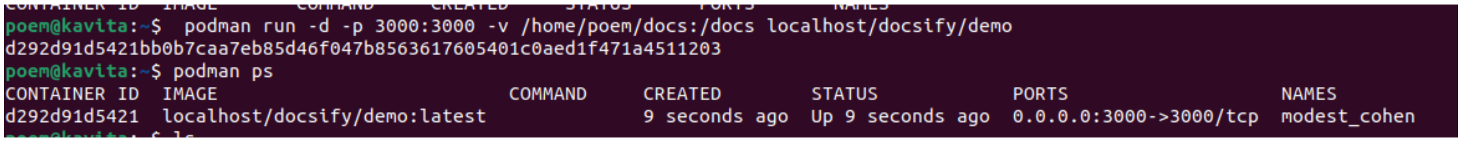
> - ***"-d"*** (Detach Mode): This option allows the container to run in the background, meaning the container will run as a separate process and free up your terminal. It's useful for users who don't want to see the container's output in the terminal.

> - ***"-p"*** (Port Forwarding): This option enables port forwarding between the container and the host system. This means you can access the services running inside the container without external categorization.

> - ***"-v"*** (Volume Mount): This option allows you to mount a file or directory between the container and the host system. As a result, you can use your host file system within the container, enabling data sharing for your container.

- ***First, create a Podman container for Docsify.***

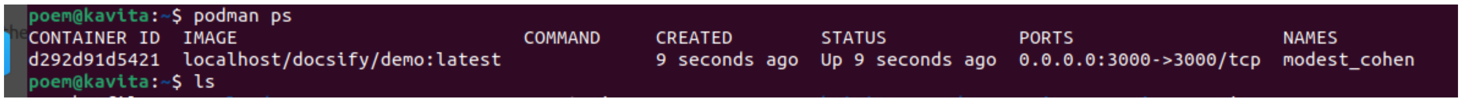
> Podman run -d -p 3000:3000 -v /home/poem/docs:/docs localhost/docsify/demo



```
poem@kavita:~$  podman run -d -p 3000:3000 -v /home/poem/docs:/docs localhost/docsify/demo
d292d91d5421bb0b7caa7eb85d46f047b8563617605401c0aed1f471a4511203
poem@kavita:~$ podman ps
CONTAINER ID  IMAGE                          COMMAND     CREATED        STATUS            PORTS                   NAMES
d292d91d5421  localhost/docsify/demo:latest              9 seconds ago  Up 9 seconds ago  0.0.0.0:3000->3000/tcp  modest_cohen
```

- Check if the container is created with the “podman ps” command.

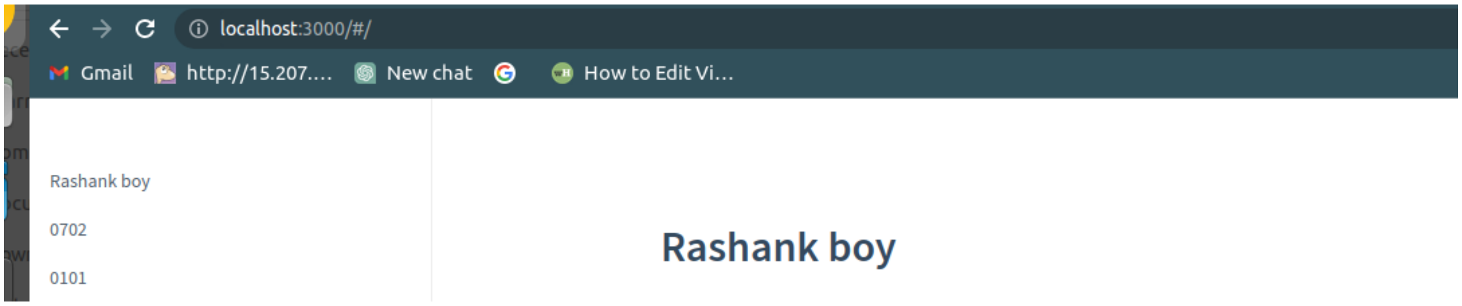
> Podman ps



```
poem@kavita:~$ podman ps
CONTAINER ID  IMAGE                          COMMAND     CREATED         STATUS             PORTS                   NAMES
d292d91d5421  localhost/docsify/demo:latest              9 seconds ago   Up 9 seconds ago   0.0.0.0:3000->3000/tcp  modest_cohen
poem@kavita:~$ ls
```

Access your Docsify documentation

- Open your web browser and enter http://localhost:3000 in the address bar. This will allow you to view your Docsify documentation served from the container. You should see the Docsify interface displaying your documentation content.



- **Docsify use the reference link for additional information** https://docsify.js.org/#/

3. started with GitHub

Step 1.

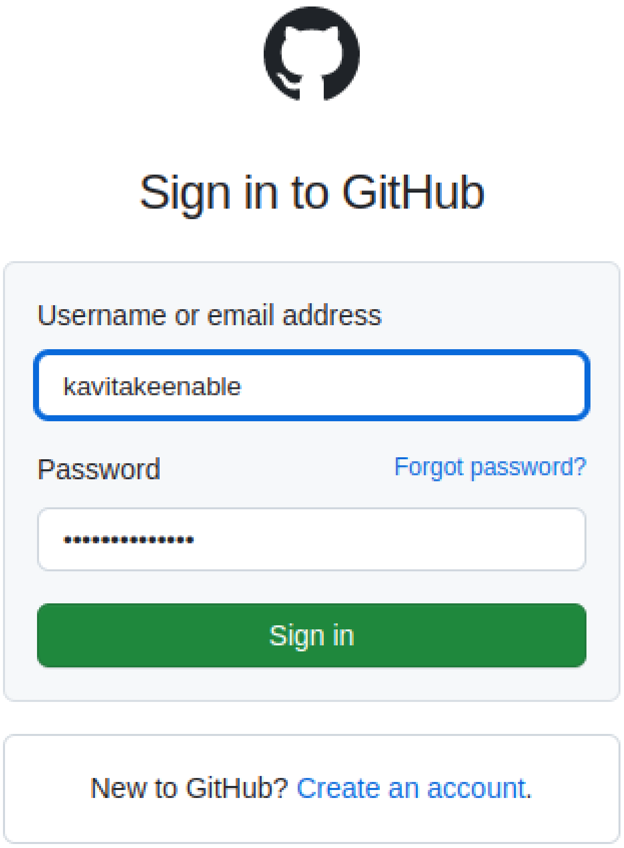
Login github

Now go to GitHub but how to login to GitHub account is given below

Here's a step-by-step guide to hosting Docsify docs on GitHub

Login ya Sign in Github Account

> Log in to your GitHub account.



Sign in to GitHub

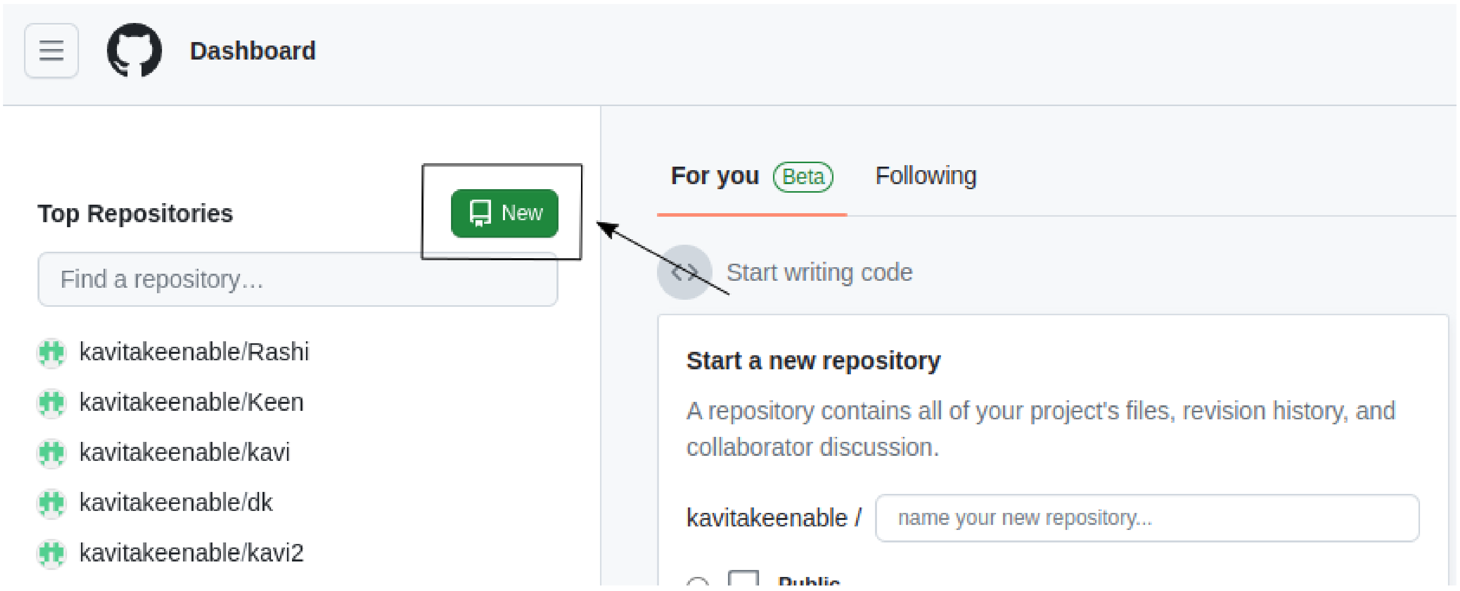
Username or email address

kavitakeenable

Password

Create a New Repository

On GitHub, click the "New" button to create a new repository. Give it a name of your choice



kavitakeenable/Rashi

kavitakeenable/Keen

kavitakeenable/kavi

kavitakeenable/dk

kavitakeenable/kavi2

Start a new repository

A repository contains all of your project's files, revision history, and collaborator discussion.

kavitakeenable / name your new repository...

Public

Step 2: Fill Repository Details

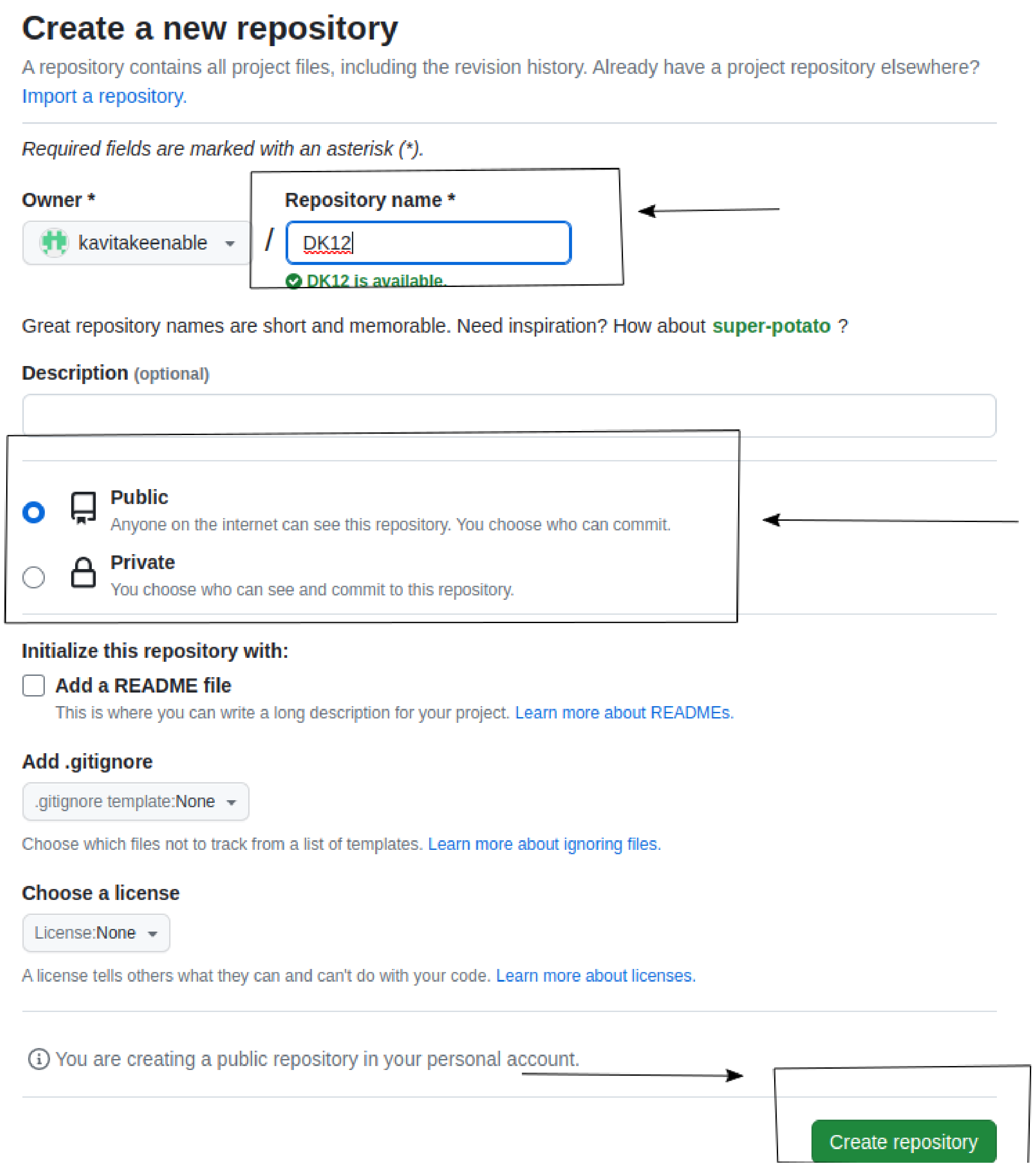
> 1. In the ***"Repository name"*** box, type "DK12"

> 2. In the ***"Description"*** box, type a short description.

> 3. Select whether your repository will be ***Public or Private.***

> 4. Select Add a README file.

> 4. Click ***Create repository.***



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 * kavitakeenable / Repository name * DK12

DK12 is available.

Great repository names are short and memorable. Need inspiration? How about super-potato ?

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.

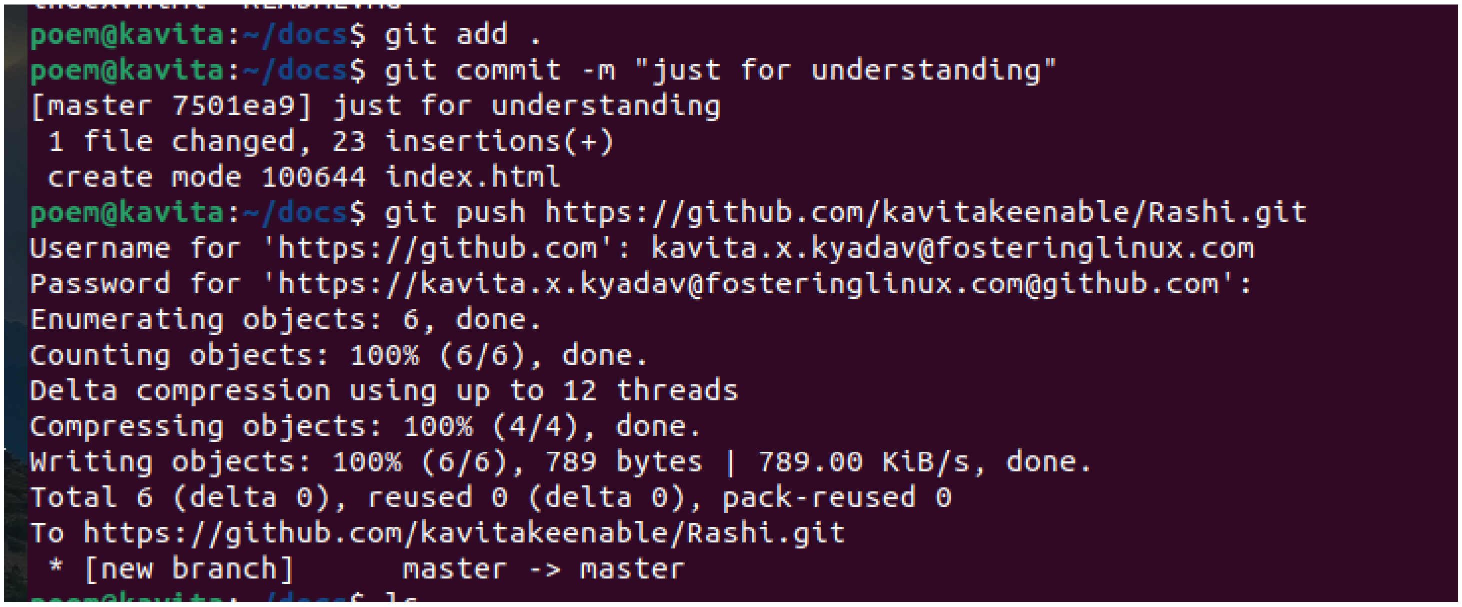
Step 3 : Stage and Commit Changes

use of the Git commands - commit, clone, push, and pull

> ***git add*** to stage changes, ***commit*** them with git commit, and push them to the remote repository with git push. Has this process enabled you to integrate your Doxify documentation into a GitHub repository and manage it under centralised version control

> - **Commit** : The git commit command is used to permanently store the changes made to code. Whenever you make changes to your code that you want to keep, you create a "commit" that contains a brief description of the changes you made. This helps in keeping track of the project's history.

> - **Push** : The git push command is used to upload the committed changes from your local computer to a remote Git repository. When you make changes to your code and commit them, you need to push those changes to your remote repository so that others can see and access them.

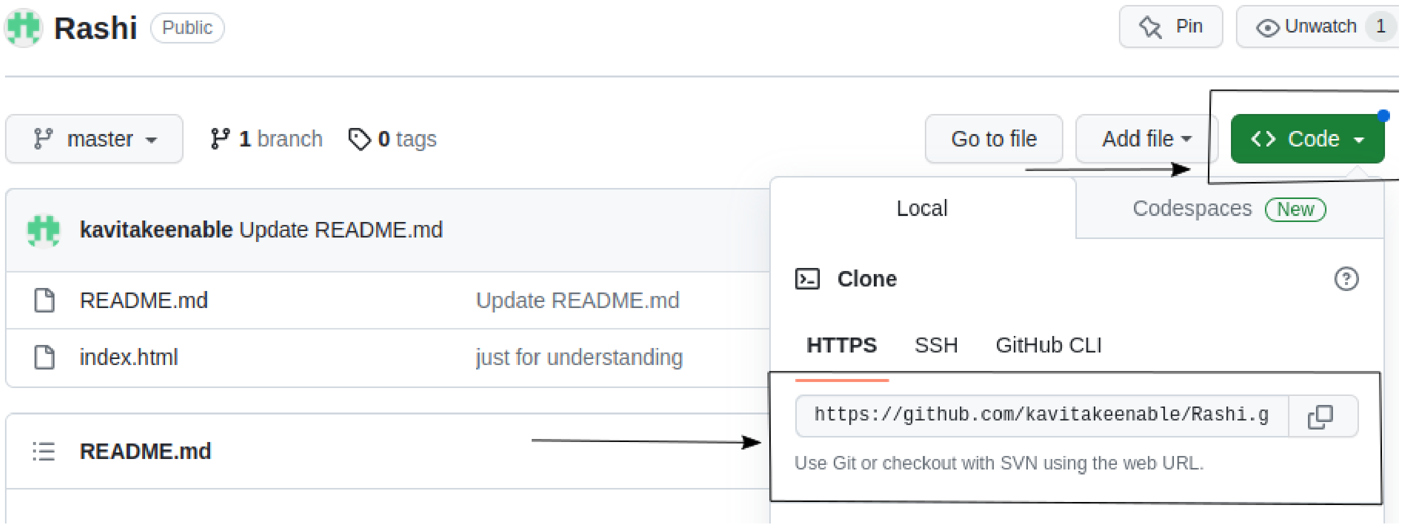


```
poem@kavita:~/docs$ git add .
poem@kavita:~/docs$ git commit -m "just for understanding"
[master 7501ea9] just for understanding
 1 file changed, 23 insertions(+)
 create mode 100644 index.html
poem@kavita:~/docs$ git push https://github.com/kavitakeenable/Rashi.git
Username for 'https://github.com': kavita.x.kyadav@fosteringlinux.com
Password for 'https://kavita.x.kyadav@fosteringlinux.com@github.com':
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 789 bytes | 789.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kavitakeenable/Rashi.git
 * [new branch]      master -> master
```

Step 4 : Clone the Repository

Choose the "Clone" option on GitHub to get the repository URL. Copy this URL

> - **Clone**: The git clone command is used to copy the entire remote Git repository onto your local machine. It's a way to get a complete copy of all files and history from the remote repository onto your system. This is often used when you want to start working on a project that is hosted remotely.



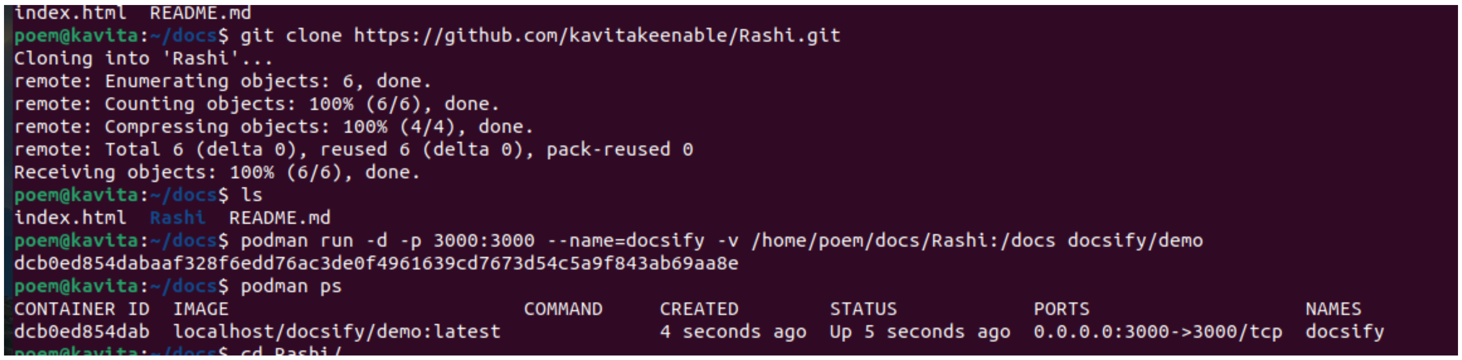
kavitakeenable Update README.md

README.md   Update README.md

index.html   just for understanding

README.md

```
https://github.com/kavitakeenable/Rashi.g
```

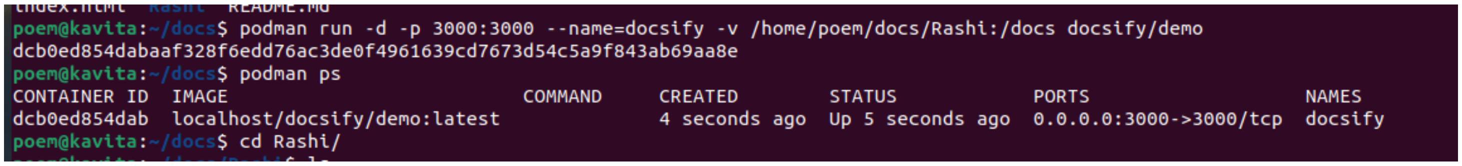


```
index.html  README.md
poem@kavita:~/docs$ git clone https://github.com/kavitakeenable/Rashi.git
Cloning into 'Rashi'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
poem@kavita:~/docs$ ls
index.html  Rashi  README.md
poem@kavita:~/docs$ podman run -d -p 3000:3000 --name=docsify -v /home/poem/docs/Rashi:/docs docsify/demo
dcb0ed854dabaaf328f6edd76ac3de0f4961639cd7673d54c5a9f843ab69aa8e
poem@kavita:~/docs$ podman ps
CONTAINER ID  IMAGE                         COMMAND     CREATED        STATUS            PORTS                   NAMES
dcb0ed854dab  localhost/docsify/demo:latest             4 seconds ago  Up 5 seconds ago  0.0.0.0:3000->3000/tcp  docsify
```

Step 5 Integration docsify and gitHub

- To integrate Docsify with GitHub and ensure seamless integration, include the repository name in the URL path.

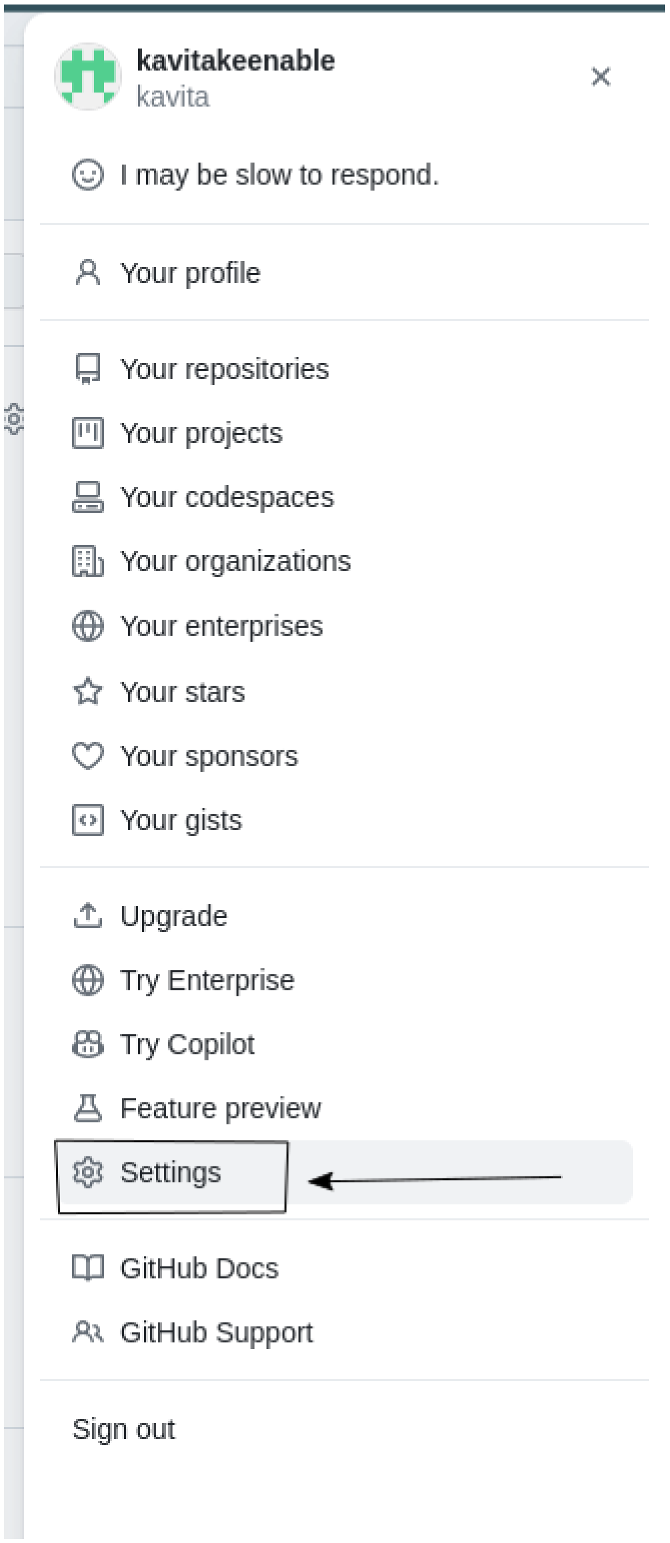
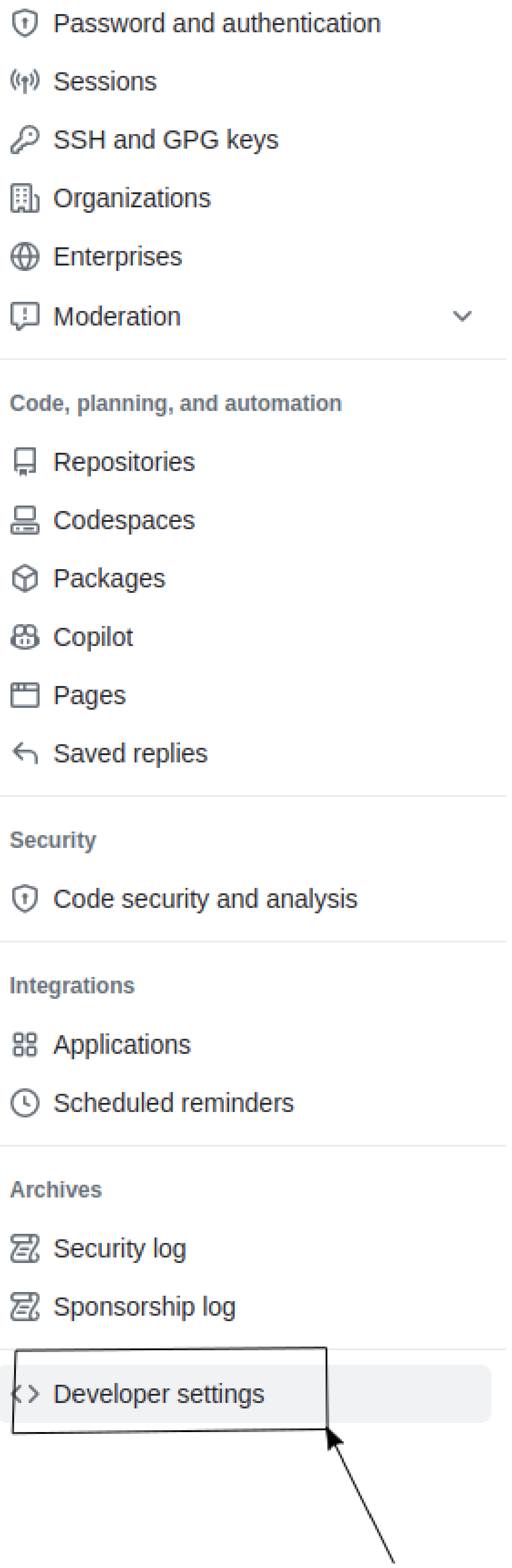
> Podman run -d -p 3000:3000 --name=docsify -v /home/poem/Rashi:/docs docsify/demo



```
poem@kavita:~/docs$ podman run -d -p 3000:3000 --name=docsify -v /home/poem/docs/Rashi:/docs docsify/demo
dcb0ed854dabaaf328f6edd76ac3de0f4961639cd7673d54c5a9f843ab69aa8e
poem@kavita:~/docs$ podman ps
CONTAINER ID  IMAGE                         COMMAND     CREATED        STATUS            PORTS                   NAMES
dcb0ed854dab  localhost/docsify/demo:latest             4 seconds ago  Up 5 seconds ago  0.0.0.0:3000->3000/tcp  docsify
poem@kavita:~/docs$ cd Rashi/
```

Step 6 Access Account Settings

> ***Navigate to your account settings. Look for your profile picture or username at the top-right corner and click on it to access your account settings.***
>
> - Within the account settings, search for a section related to "Tokens," "Security," or ***"Developer Settings."***

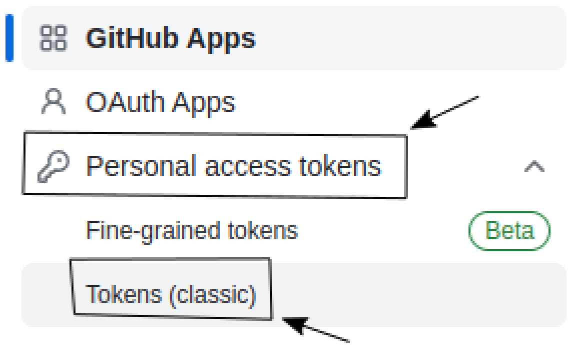
profile account	Setting
 kavitakeenable kavita I may be slow to respond. Your profile Your repositories Your projects Your codespaces Your organizations Your enterprises Your stars Your sponsors Your gists Upgrade Try Enterprise Try Copilot Feature preview Settings GitHub Docs GitHub Support Sign out	 Password and authentication Sessions SSH and GPG keys Organizations Enterprises Moderation Code, planning, and automation Repositories Codespaces Packages Copilot Pages Saved replies Security Code security and analysis Integrations Applications Scheduled reminders Archives Security log Sponsorship log Developer settings

Create New Token

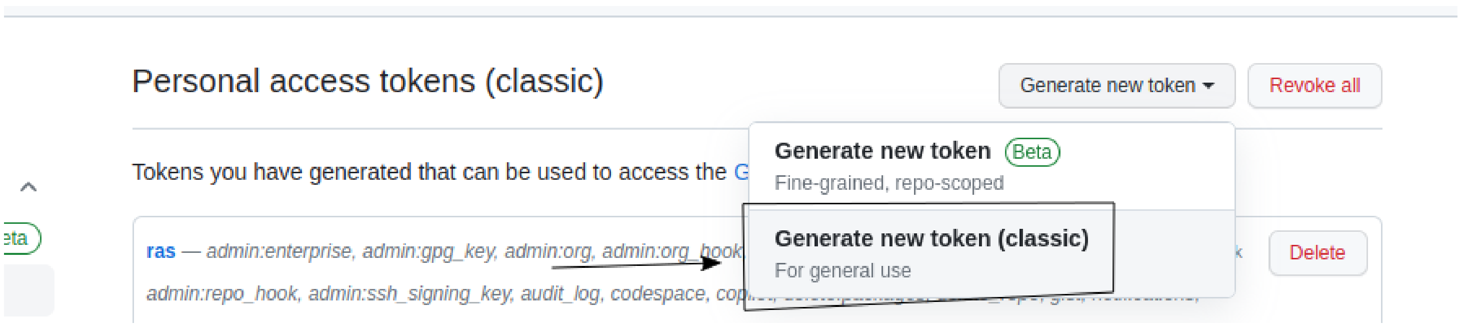
> - Choose the option to create a new personal access token.

> - Click the button to generate the token.

Personal Access tokens



Generate New token

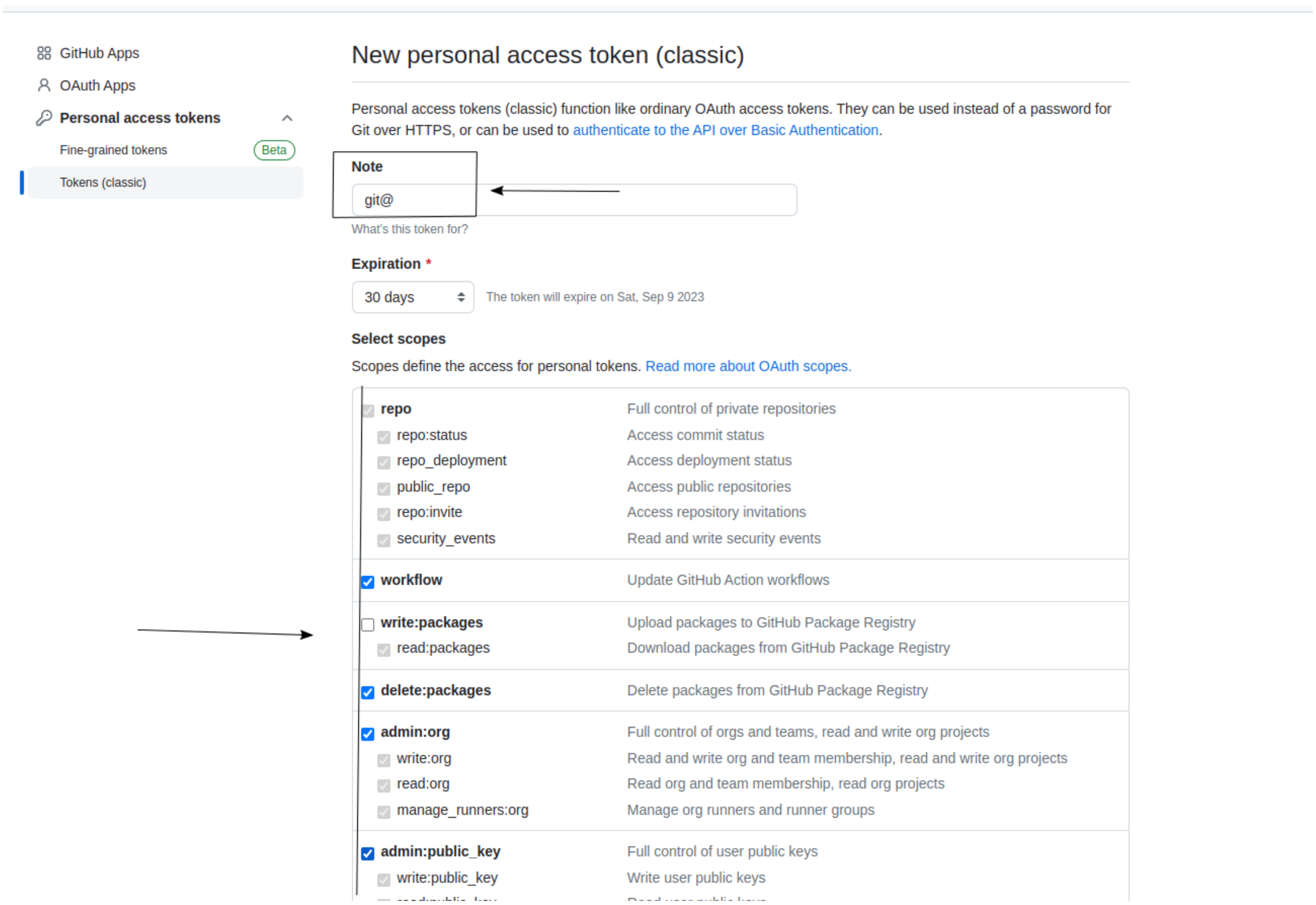

Personal access tokens (classic)
Generate new token
Revoke all
Generate new token
Beta
Fine-grained, repo-scoped
Generate new token (classic)
For general use
Delete

Tokens you have generated that can be used to access the G

ras — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook
admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, cop

Set Permissions

> Specify what the token can do. Choose permissions like accessing repositories or performing read/write actions.



GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens Beta
Tokens (classic)

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note: git@
What's this token for?

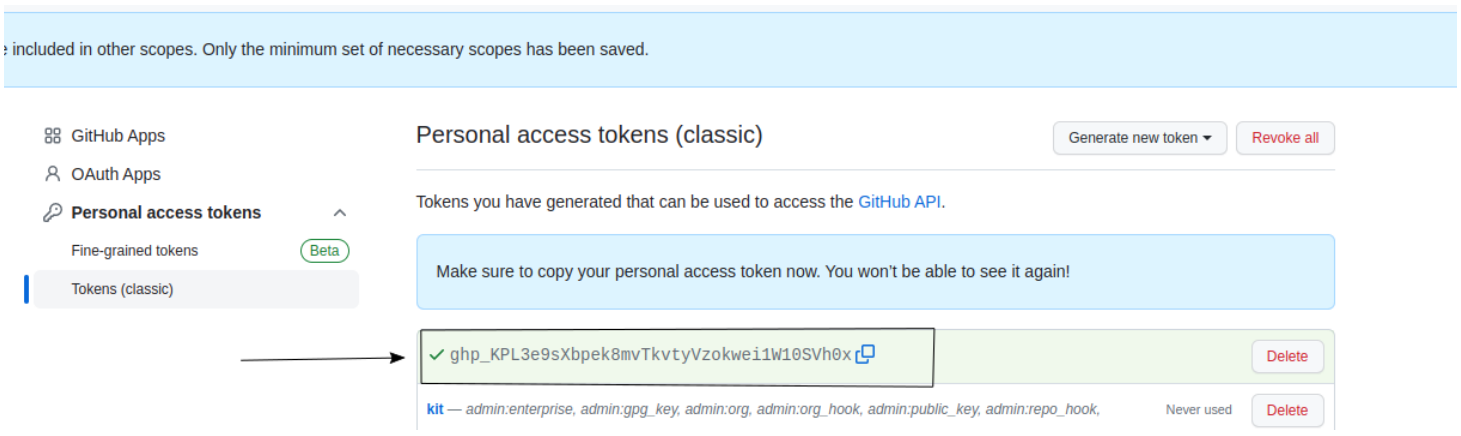
Expiration *: 30 days — The token will expire on Sat, Sep 9 2023

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

✓	**repo**	Full control of private repositories
✓	repo:status	Access commit status
✓	repo_deployment	Access deployment status
✓	public_repo	Access public repositories
✓	repo:invite	Access repository invitations
✓	security_events	Read and write security events
✓	**workflow**	Update GitHub Action workflows
☐	**write:packages**	Upload packages to GitHub Package Registry
✓	read:packages	Download packages from GitHub Package Registry
✓	**delete:packages**	Delete packages from GitHub Package Registry
✓	**admin:org**	Full control of orgs and teams, read and write org projects
✓	write:org	Read and write org and team membership, read and write org projects
✓	read:org	Read org and team membership, read org projects
✓	manage_runners:org	Manage org runners and runner groups
✓	**admin:public_key**	Full control of user public keys
✓	write:public_key	Write user public keys

- Once generated, you'll typically see the token displayed on your screen.
  Copy and Store Securely.



included in other scopes. Only the minimum set of necessary scopes has been saved.

GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens Beta
Tokens (classic)

Personal access tokens (classic) — Generate new token ▾ — Revoke all

Tokens you have generated that can be used to access the GitHub API.

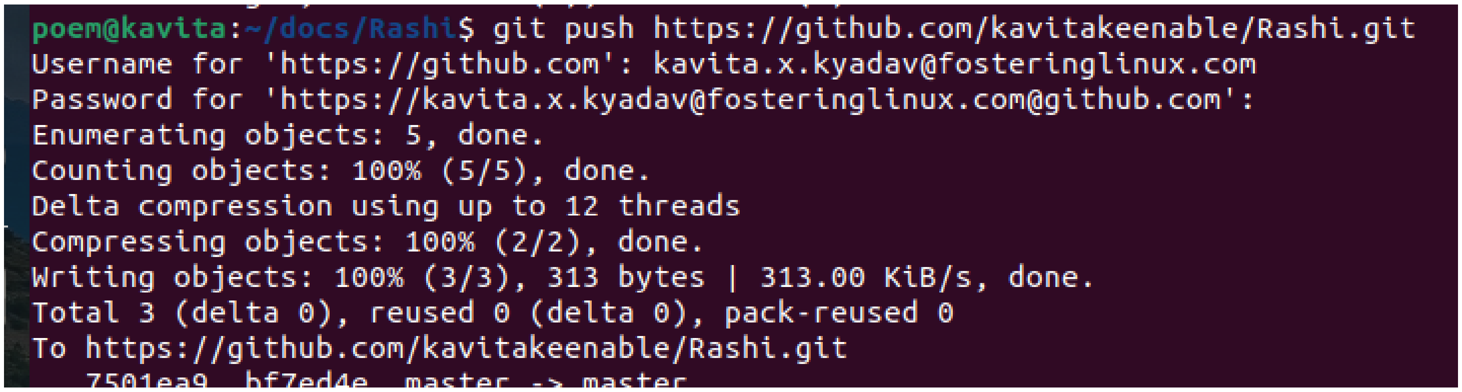
Make sure to copy your personal access token now. You won't be able to see it again!

```
✓ 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
```
Delete

kit — admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, — Never used — Delete

- Copy the token to your clipboard.
  Store the token in a safe place, like a password manager or a secure document.

Similarly you can push when you have to update any code you will push and when you have to take any code from remote server then you will pull



```
poem@kavita:~/docs/Rashi$ git push https://github.com/kavitakeenable/Rashi.git
Username for 'https://github.com': kavita.x.kyadav@fosteringlinux.com
Password for 'https://kavita.x.kyadav@fosteringlinux.com@github.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/kavitakeenable/Rashi.git
   7501ea9..bf7ed4e  master -> master
```

- **GitHub use the reference link for additional information**
  https://docs.github.com/en/get-started/quickstart/hello-world

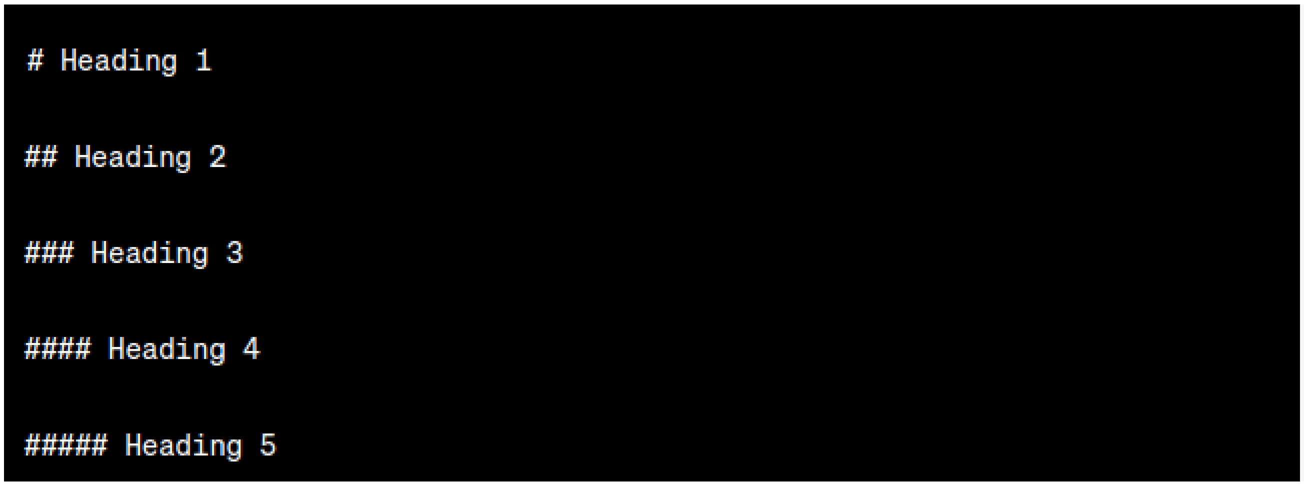
3.Markdown Basic Syntax

Overview

> Markdown is a plain text formatting language used to create simple and structured documents. It is commonly used for web content, readme files, documentation, email, and various other types of text-based content. Below I'm doing some common Markdown syntax examples and their explanations:

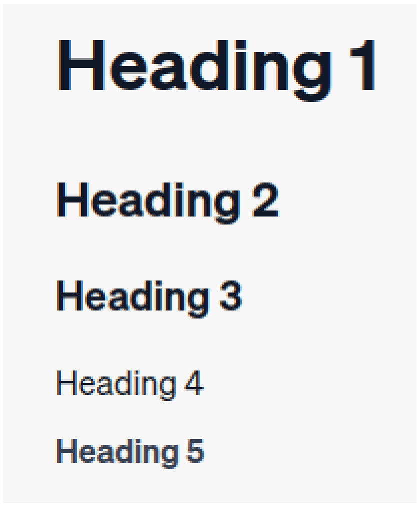
1. Headings

> Headings are created using the # symbol, with one # indicating Heading 1 and six # indicating Heading 6.



```
# Heading 1

## Heading 2

### Heading 3

#### Heading 4

##### Heading 5
```

output


Heading 1

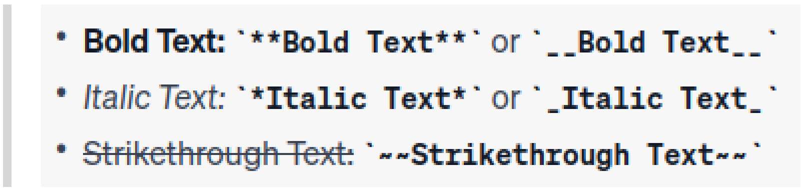
Heading 2

Heading 3

Heading 4

Heading 5

2. Text Formatting

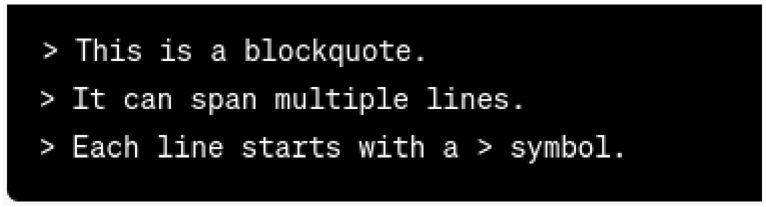

- **Bold Text:** `**Bold Text**` or `__Bold Text__`
- *Italic Text:* `*Italic Text*` or `_Italic Text_`
- ~~Strikethrough Text:~~ `~~Strikethrough Text~~`

Output

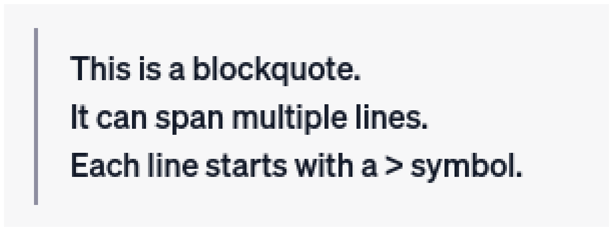
> Bold Text: **Bold Text** or **Bold Text**
> Italic Text: *Italic Text* or *Italic Text*
> Strikethrough Text: ~~Strikethrough Text~~

3. Blockquota

To create a blockquote in Markdown, you use the > symbol. Place the > symbol before the text you want to include in the blockquote.


```
> This is a blockquote.
> It can span multiple lines.
> Each line starts with a > symbol.
```

Output



> This is a blockquote.
> It can span multiple lines.
> Each line starts with a > symbol.

- **Markdown Use the reference link for additional information**
  https://www.markdownguide.org/cheat-sheet/