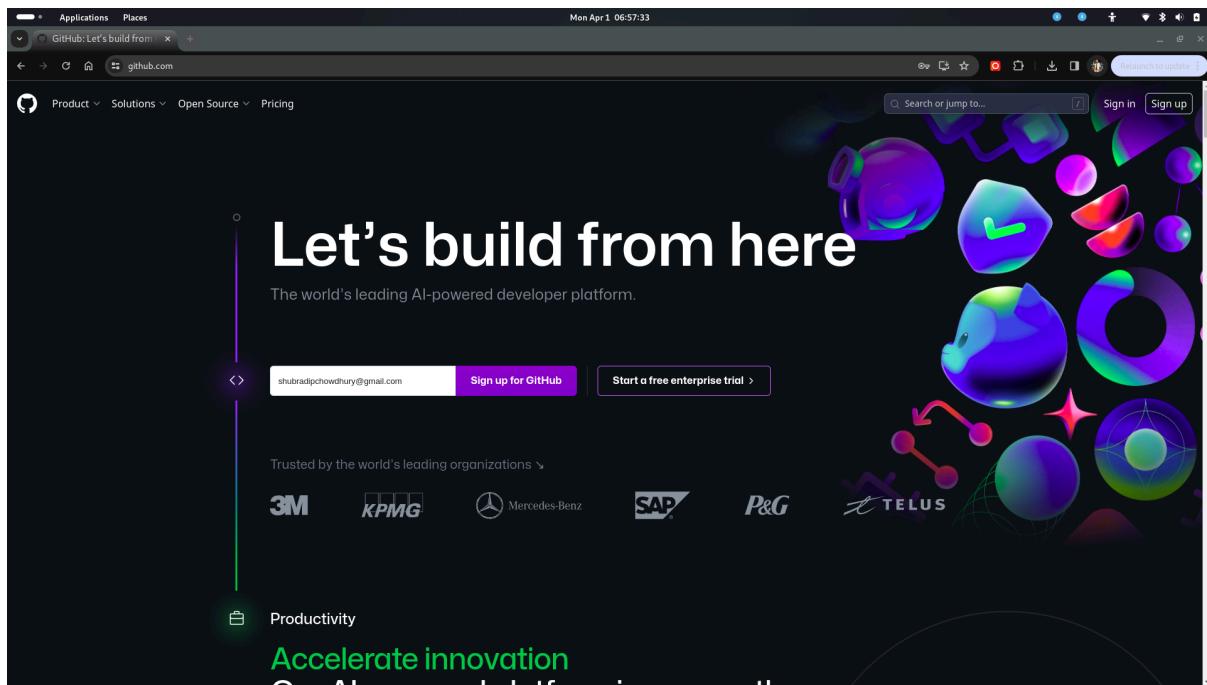


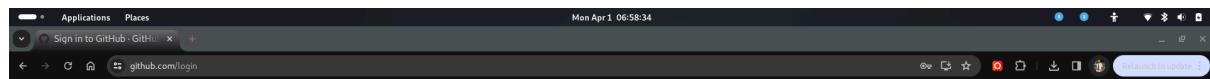
ASSIGNMENT-8

Problem Statement: Deploy a project from a local machine to GitHub and vice-versa.

The steps to deploy a project from local machine to GitHub are as follows: -

1. The initial and most important step involves downloading Git, a DevOps tool, and establishing a GitHub account.
2. Log in to your GitHub account using your login details.





Sign in to GitHub

Username or email address

shubradipchowdhury@gmail.com

Password

Forgot password?

.....

Sign in

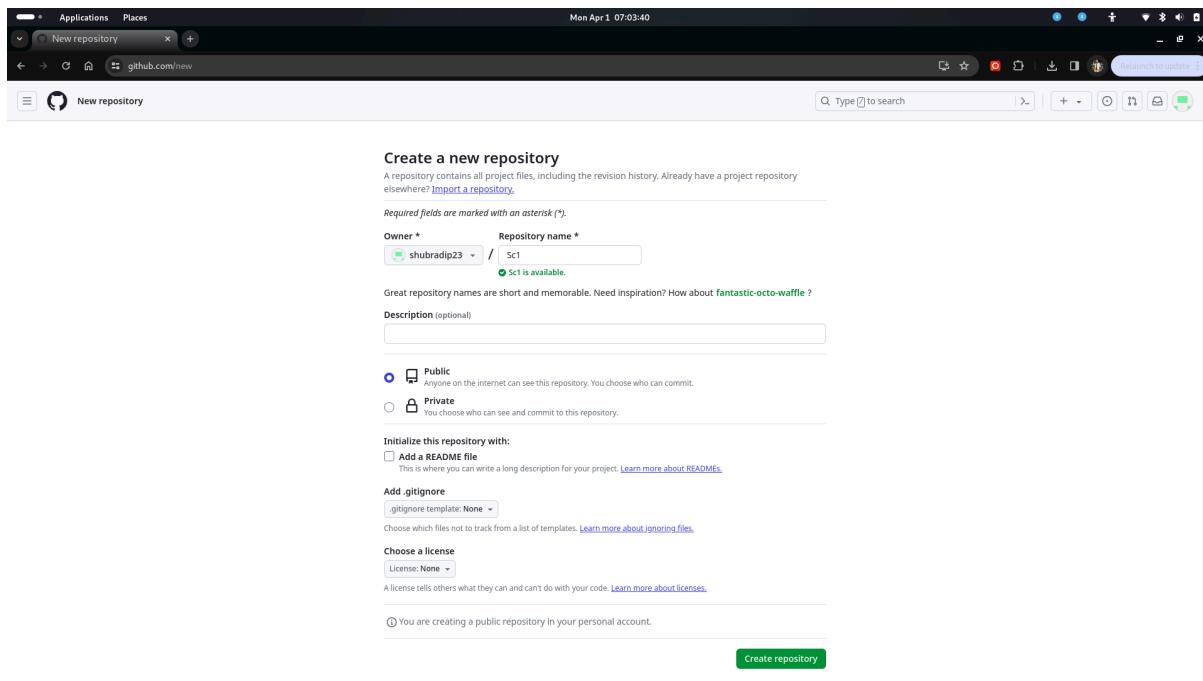
[Sign in with a passkey](#)

New to GitHub? [Create an account](#)

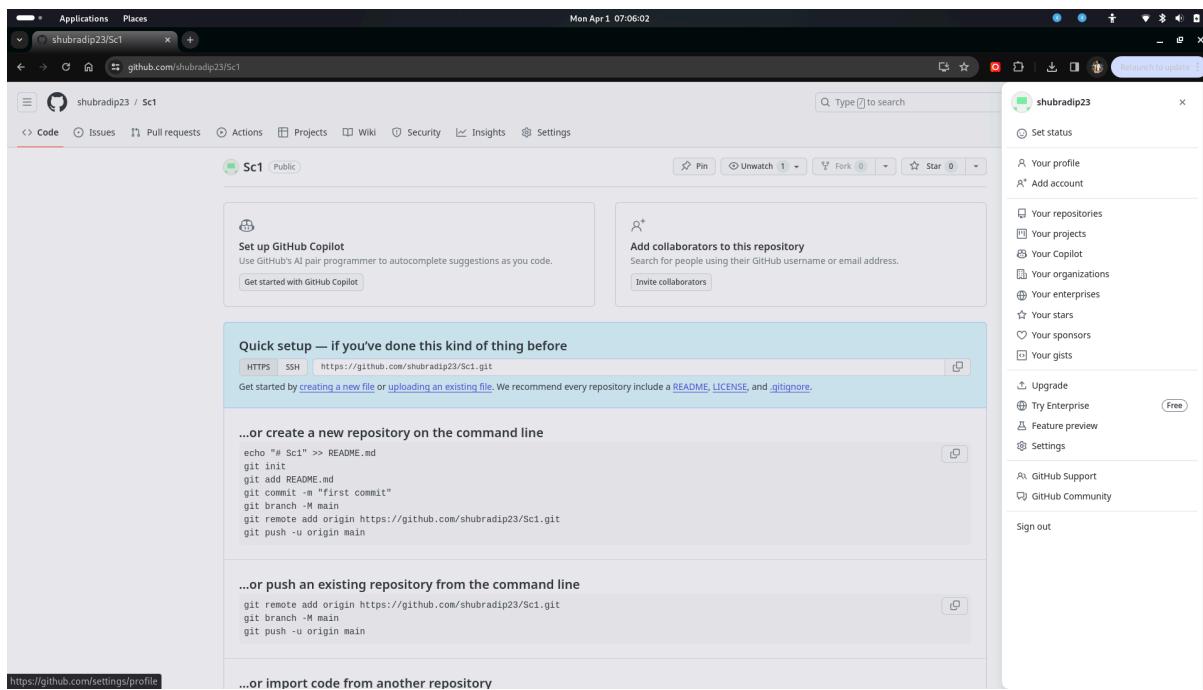
[Terms](#) [Privacy](#) [Docs](#) [Contact GitHub Support](#) [Manage cookies](#) [Do not share my personal information](#)

- Upon successfully logging in, initiate the creation of a new repository by selecting the appropriate option on the left side of the window panel.

- Choose an appropriate name for your repository (for example, we'll use "sc1") and ensure that the repository is set to public. Simply click on "**create repository**" without making any additional modifications.



5. After creating a repository, navigate to your account on the right-hand side, and then locate and click on "**Settings**."



6. Scroll down and locate "**Developer settings**" on the left-hand side, then click on it.

The screenshot shows the GitHub profile settings page under the 'Developer settings' tab. On the left, there's a sidebar with links like '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', and '< Developer settings'. The main area has sections for 'Pronouns' (set to 'Don't specify'), 'URL' (empty), 'ORCID ID' (with a note about ORCID.org and a 'Connect your ORCID ID' button), 'Social accounts' (four empty fields for linking social profiles), 'Company' (empty field with a note about @mentioning a company), 'Location' (empty field), and a checkbox for 'Display current local time'. Below these is a note about optional fields being deletable and a privacy statement. At the bottom is a green 'Update profile' button.

7. Select "**Personal Access Tokens**" and choose "**Tokens (classic)**" from the dropdown list.

The screenshot shows the GitHub developer settings page under the 'Tokens' tab. On the left, there's a sidebar with 'GitHub Apps', 'OAuth Apps', 'Personal access tokens' (which is the active tab, indicated by a green 'Beta' badge), 'Fine-grained tokens', and 'Tokens (classic)'. The main area is titled 'GitHub Apps' with a sub-section for 'Personal access tokens'. It contains a note about building GitHub Apps, a 'Register a new GitHub App' link, and developer documentation. At the bottom are links for 'Terms', 'Privacy', 'Security', 'Status', 'Docs', 'Contact', 'Manage cookies', and a 'Do not share my personal information' checkbox. A footer at the very bottom includes the GitHub logo, a copyright notice for 2024, and links for 'Type to search' and other browser controls.

8. Click on "**Generate new token**" above, and then choose "**Generate new token (classic)**" from the dropdown list.

The screenshot shows the GitHub 'Personal Access Tokens' settings page. On the left, there's a sidebar with 'Personal access tokens' selected. The main area is titled 'Personal access tokens (classic)' and shows a list of tokens. A modal window is open over the list, titled 'Generate new token (classic)'. It contains two buttons: 'Generate new token (Beta)' and 'Generate new token (classic)'. Below these buttons, it says 'For general use'. At the bottom of the modal, it notes '⚠ This token has no expiration date.' The URL in the browser bar is https://github.com/settings/tokens/new.

9. Enter an appropriate token name (for example, "**Token5**"), and select the option "**No expiration**" instead of choosing any time slot.

The screenshot shows the 'New personal access token (classic)' creation form. In the 'Note' field, 'token5' is entered. Under 'Expiration', the dropdown is set to 'No expiration'. A note below the dropdown states 'GitHub strongly recommends that you set an expiration date for your token to help keep your information secure. [Learn more](#)'. The 'Select scopes' section is expanded, showing various permission checkboxes. Some scopes are grouped under headings like 'repo', 'workflow', 'write:packages', etc. The URL in the browser bar is https://github.com/settings/tokens/new.

10. Ensure to check all the outer checkboxes provided below, ensuring that none is left unchecked.

The screenshot shows a list of OAuth scopes for a new personal access token. The 'repo' scope is selected. Other scopes listed include workflow, write:packages, delete:packages, admin:org, admin:public_key, admin:repo_hook, admin:org_hook, gist, notifications, user, and delete_repo.

Scope	Description
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
read:public_key	Read user public keys
admin:repo_hook	Full control of repository hooks
write:repo_hook	Write repository hooks
readrepo_hook	Read repository hooks
admin:org_hook	Full control of organization hooks
gist	Create gists
notifications	Access notifications
user	Update ALL user data
read:user	Read ALL user profile data
user:email	Access user email addresses (read-only)
user:follow	Follow and unfollow users
delete_repo	Delete repositories

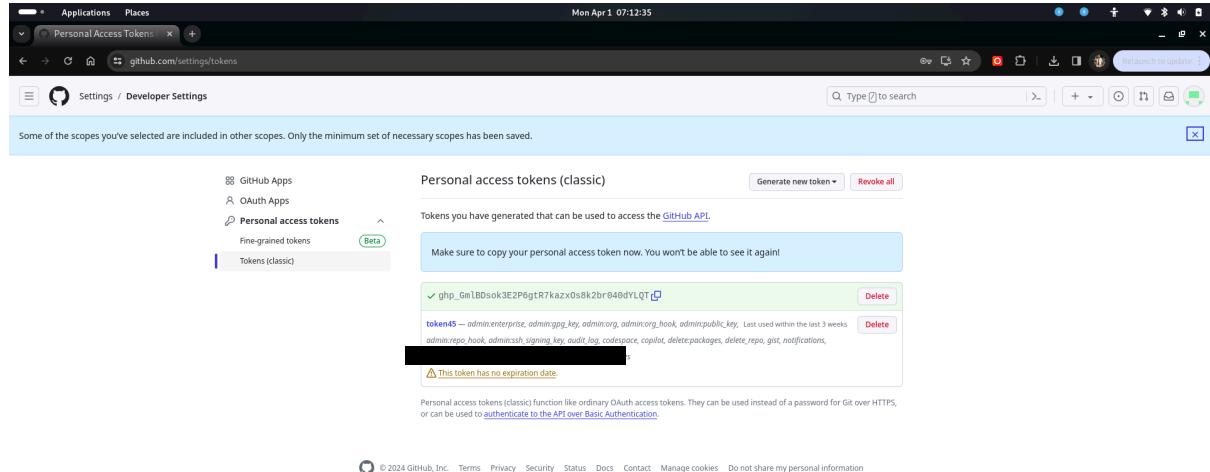
11. Continue checking all the outer checkboxes until the end, ensuring thorough coverage. Finally, click on "**Generate token**". If there is any redundancy in the token name, an error will be displayed at the top.

The screenshot shows a list of OAuth scopes for a new personal access token. The 'user', 'delete_repo', 'write:discussion', 'admin:enterprise', 'audit_log', 'codespace', 'copilot', 'project', 'admin:gpg_key', 'admin:ssh_signing_key', and 'read:ssh_signing_key' scopes are selected. The 'user' scope includes sub-options like read:user, user:email, and user:follow.

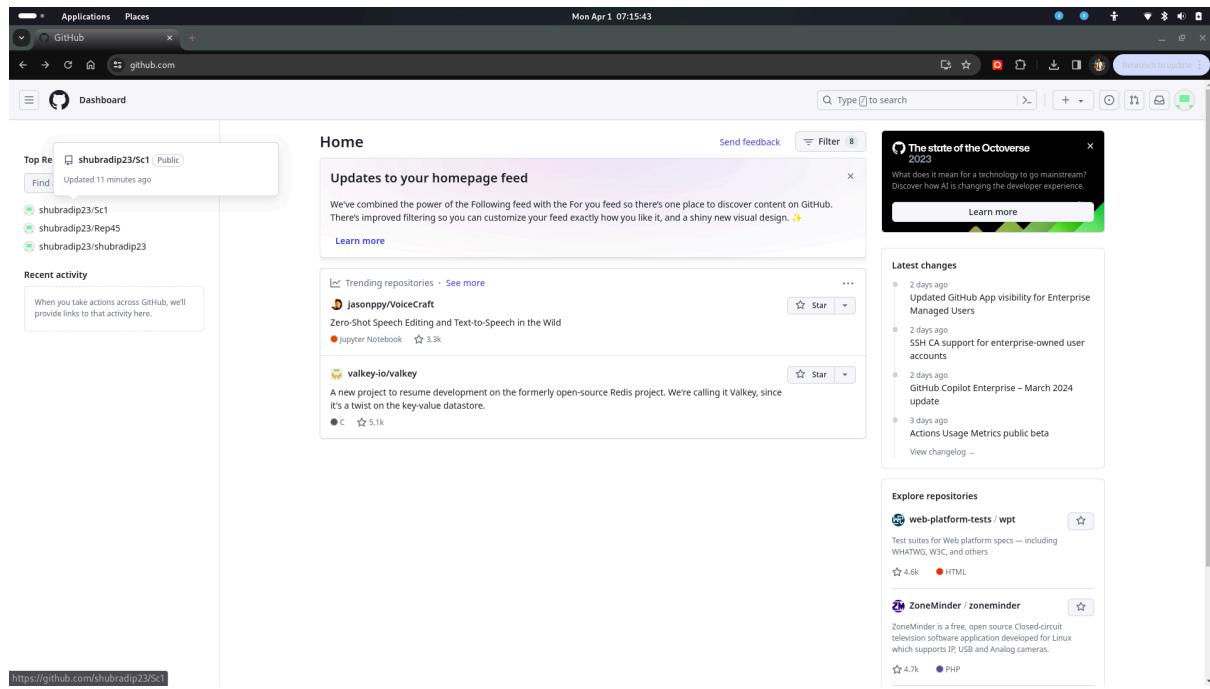
Scope	Description
user	Update ALL user data
read:user	Read ALL user profile data
user:email	Access user email addresses (read-only)
user:follow	Follow and unfollow users
delete_repo	Delete repositories
write:discussion	Read and write team discussions
read:discussion	Read team discussions
admin:enterprise	Full control of enterprises
manage_runners:enterprise	Manage enterprise runners and runner groups
manage_billing:enterprise	Read and write enterprise billing data
read:enterprise	Read enterprise profile data
audit_log	Full control of audit log
read:audit_log	Read access of audit log
codespace	Full Control of codespaces
codespace:secrets	Ability to create, read, update, and delete codespace secrets
copilot	Full Control of Github Copilot settings and seat assignments
manage_billing:copilot	View and edit Copilot Business seat assignments
project	Full control of projects
read:project	Read access of projects
admin:gpg_key	Full control of public user GPG keys
write:gpg_key	Write public user GPG keys
read:gpg_key	Read public user GPG keys
admin:ssh_signing_key	Full control of public user SSH signing keys
write:ssh_signing_key	Write public user SSH signing keys
read:ssh_signing_key	Read public user SSH signing keys

Generate token Cancel

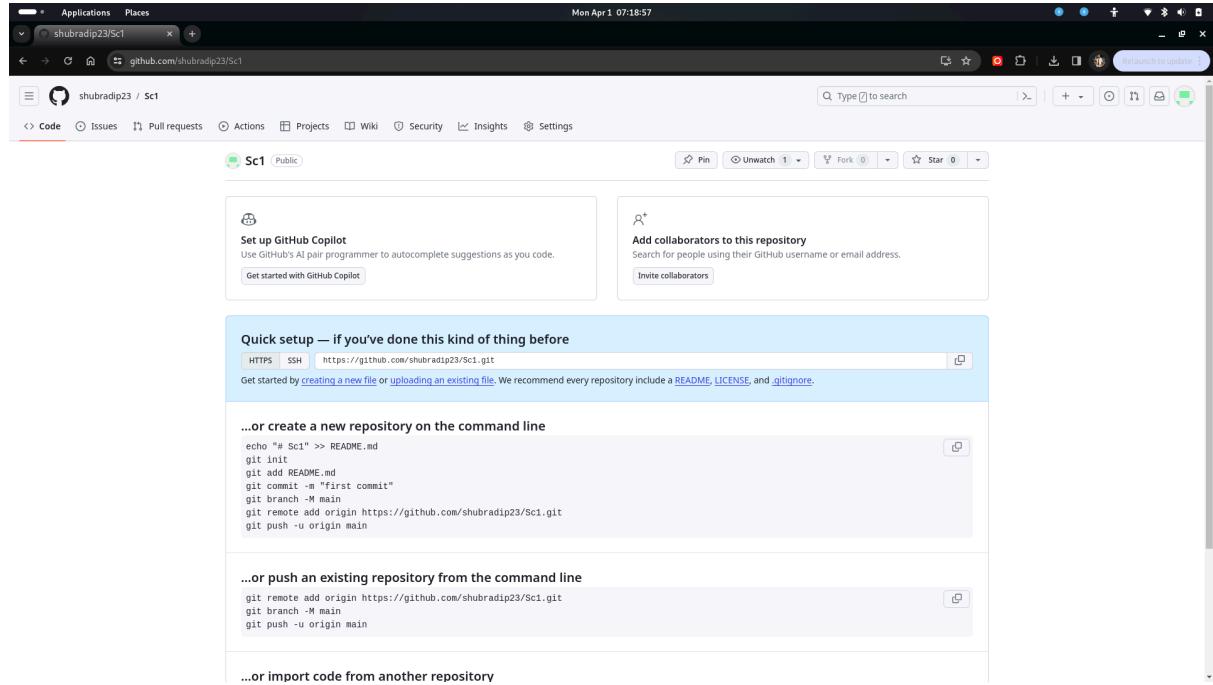
12. Now, copy the personal access token immediately as it won't be accessible again. It's advisable to save this token in a text document for easy access later on.



13. Return to the dashboard and select the repository, namely "**Repo1**."



14. In the subsequent window that opens up, choose "**HTTPS**" and copy the pathname for future reference.



15. Prior to proceeding, navigate to the "**Control Panel**". Then, select "**User Account**" and proceed to "**Manage Windows Credentials**". If there's any existing GitHub account opened, remove it.

16. Right-click on the "**HTML**" folder, which contains the project files to be uploaded to GitHub, and select "**Open with Git Bash here**".



17. In the terminal, enter the following commands:

->git init

(This command creates a new Git repository)

->dir

(Displays directories and files and directories stored on disk.)

```
(base) └─(shubradip㉿Shubradip)-[~/Desktop/html]
└─$ dir
About.html  Home.html  index.html
```

->git add .

(It adds the current content of existing paths as a whole.)

```
(base) └─(shubradip㉿Shubradip)-[~/Desktop/html]
└─$ git add .
```

->git status

(Displays the state of working directory.)

```
(base) └─(shubradip@Shubradip)-[~/Desktop/html]
└$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:  About.html
  new file:  Home.html
  new file:  index.html
```

->**git commit -m “done”**

(Take all of the changes that have been made locally and push them up to a remote repository.)

```
(base) └─(shubradip@Shubradip)-[~/Desktop/html]
└$ git commit -m "done"
[master (root-commit) a2e4426] done
  Committer: Shubradip Chowdhury <shubradip@Shubradip.net>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

  git config --global --edit
```

After doing this, you may fix the identity used for this commit with:

```
git commit --amend --reset-author

3 files changed, 37 insertions(+)
create mode 100644 About.html
create mode 100644 Home.html
create mode 100644 index.html
```

->**git remote add origin <the_repository_path_name>**

(**git remote add** is to add a short name, such as **origin**, which is like an alias to a URL. And **origin** is the usual path of where the remote repository points to.)

```
(base) └─(shubradip@Shubradip)-[~/Desktop/html]
└$ git remote add origin https://github.com/shubradip23/Sc1.git
```

->**git push -u origin master**

(Push the commits in the local branch named **master** to the remote named **origin**. Once this is executed, all the documents that is last synchronized with **origin** will be sent to the remote repository and other people will be able to see them there.) Next, a "Connect to GitHub" window will appear.

```
(base) └─(shubradip@Shubradip)-[~/Desktop/html]  
└$ git push -u origin master
```

Connect to GitHub X

GitHub

Sign in

[Browser/Device](#) [Token](#)

Sign in with your browser

Sign in with a code

Don't have an account? [Sign up](#)

18. Now, utilize the generated token and paste it into the designated field to sign in to GitHub.

Connect to GitHub X

GitHub

Sign in

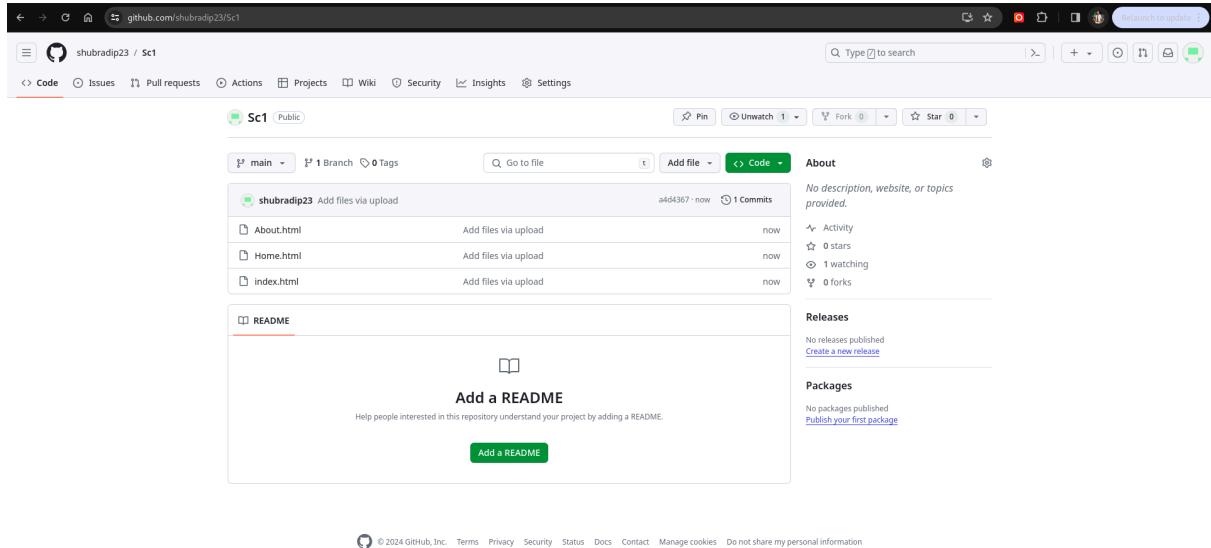
[Browser/Device](#) [Token](#)

.....

Sign in

Don't have an account? [Sign up](#)

19. GitHub will open up. Now, click on "**Code**" in the bar located below the repository name. Now, you can observe that the files have been successfully deployed.

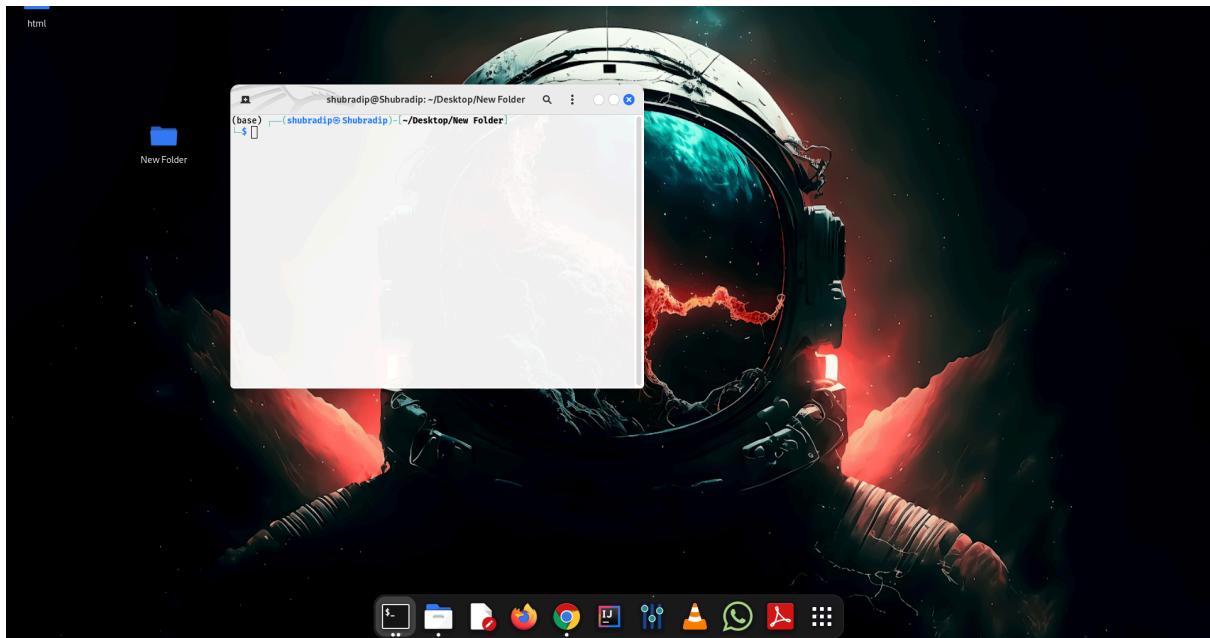


Now, deploying the project from GitHub to local machine, the steps are as follows:-

1. Create a new folder on your desktop.



2. Right-click on the folder and select "**Open with Git Bash**".



3. In the terminal start with the command,

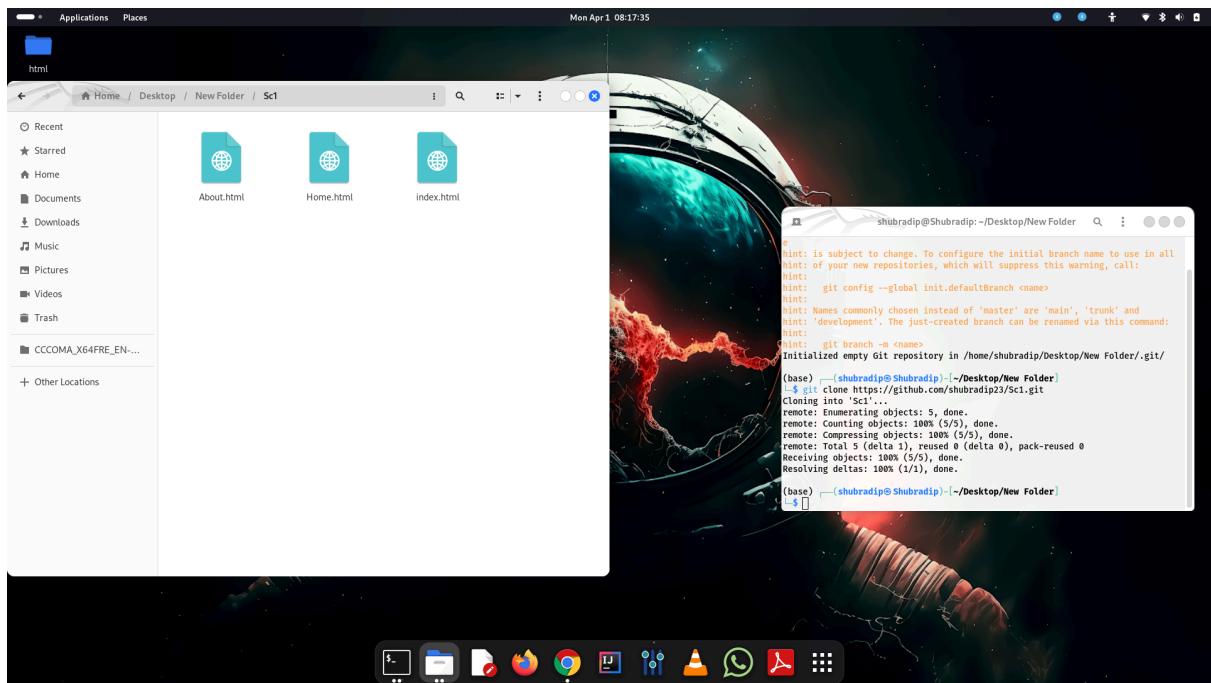
->git init

```
(base) [shubradip@Shubradip ~] ~/Desktop/New Folder
$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/shubradip/Desktop/New Folder/.git/
```

->git clone <Repository_Path>

```
(base) [shubradip@Shubradip] ~/Desktop/New Folder
└$ git clone https://github.com/shubradip23/Sc1.git
Cloning into 'Sc1'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
Resolving deltas: 100% (1/1), done.
```

4. The files have been successfully cloned. Hence, the deployment of the project from GitHub to the local machine has been completed.



[OBJ]