Git & GitHub

What is Git?

- ***Git is a version control system that helps developers track changes, manage code, and collaborate on projects easily and securely.***
- **Version Control System:** Git tracks changes to files and code over time.
- **Distributed System:** Each developer has a full copy of the project’s history on their local machine, so there’s no single point of failure.
- **Branching and Merging:** Git allows you to create separate branches for features or fixes, test them independently, and then merge when ready.
- **Change History:** Every change is saved with details about who made it and when, making it easy to review and revert if needed.
- **Collaboration Tool:** Git supports teamwork by letting multiple people work on the same code without conflicts.
- **Open Source:** Git is free and widely used, with a strong community for support.

Why is Git Important?

- **Collaboration:** Git allows developers to work together smoothly without overwriting each other’s work.
- **Change Tracking:** Every edit is recorded, so you can review past changes or restore old versions if necessary.
- **Experimentation:** Branching allows for safe testing of new features or bug fixes without affecting the main codebase.
- **Backup:** By pushing code to a remote repository, Git keeps your work safe, even if your local machine fails.
- **Transparency:** Logs show who made each change, providing a clear history and accountability.
- **Industry Standard:** Git is widely used, making it a must-have skill for developers and essential in today’s software workflows.

Difference Between Main Branch and Master Branch

- **Master:** Traditionally used as the default branch name in Git repositories.
- **Main:** Adopted as the new default branch name in many platforms, including GitHub, to be more inclusive.

Difference Between Git and GitHub

- **Git**:
  - A version control system used to track changes in code and files.
  - Works locally on your machine.
- **GitHub**:
  - A cloud-based platform that hosts Git repositories.
  - Allows collaboration, sharing, and managing projects with others online.

How do you create a new repository on GitHub?

Here are the steps to create a new repository on GitHub:

1. Log in to your GitHub account at GitHub.com.
2. Click the '+' icon in the top right corner.
3. Select "New repository" from the dropdown menu.
4. Enter a name for your repository.
5. Optionally, add a description.
6. Choose the repository's visibility (Public or Private).
7. Optionally, check "Initialize this repository with a README."
8. Click the green "Create repository" button.

Difference Between Local and Remote Repository

- **Local Repository**:
    - Stored on your own computer.
    - Allows you to make changes and commit without internet access.
    - Contains the entire project history, including all versions of files.
    - Great for individual work and testing changes before pushing to a remote.
- **Remote Repository**:
    - Hosted on a server (like GitHub, GitLab, or Bitbucket).
    - Accessible over the internet, allowing collaboration with others.
    - Acts as a central backup for your project.
    - Used to share your code and work with a team.

How to Connect Local to Remote Repository

1. **Create a Remote Repository**:
   - Create a new repository on GitHub or another platform.
2. **Open Terminal**:
   - Navigate to your local project folder using the command line.
3. **Initialize Git (if not already done)**:

   ```
   git init
   ```
4. **Add Remote Repository**:
   - Use the command below, replacing <URL> with your remote repository's URL:

   ```
   git remote add origin <URL>
   ```
5. **Verify the Connection**:
   - To check if the remote was added correctly, run:

   ```
   git remote -v
   ```
6. **Push to Remote**:
   - Finally, push your local changes to the remote repository:

   ```
   git push -u origin main
   ```

(*Replace main with the appropriate branch name if necessary.)*

Now, your local repository is connected to the remote repository!

Task 1: Set Your User Name and Email Address in Git

To set your user name and email address in Git, follow these steps:

1. **Set Your User Name**:
   - Enter the following command, replacing "Your Name" with your actual name:

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

2. **Set Your Email Address**:
   - Enter the following command, replacing " email@example.com" with your actual email address:

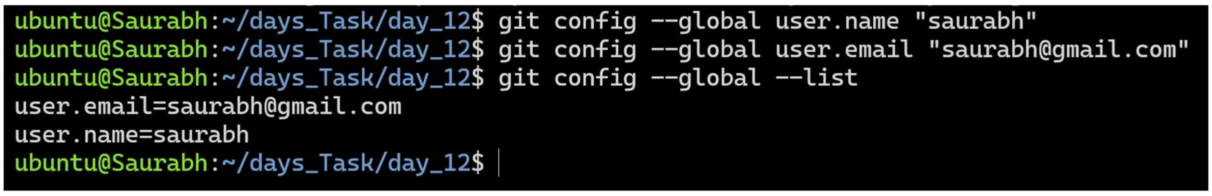
     git config --global user.email "your_email@example.com"

3. **Verify Your Settings**:
   - To check if your settings have been saved correctly, run:

     git config --global --list

   - This will display your user name and email address along with other configuration settings.

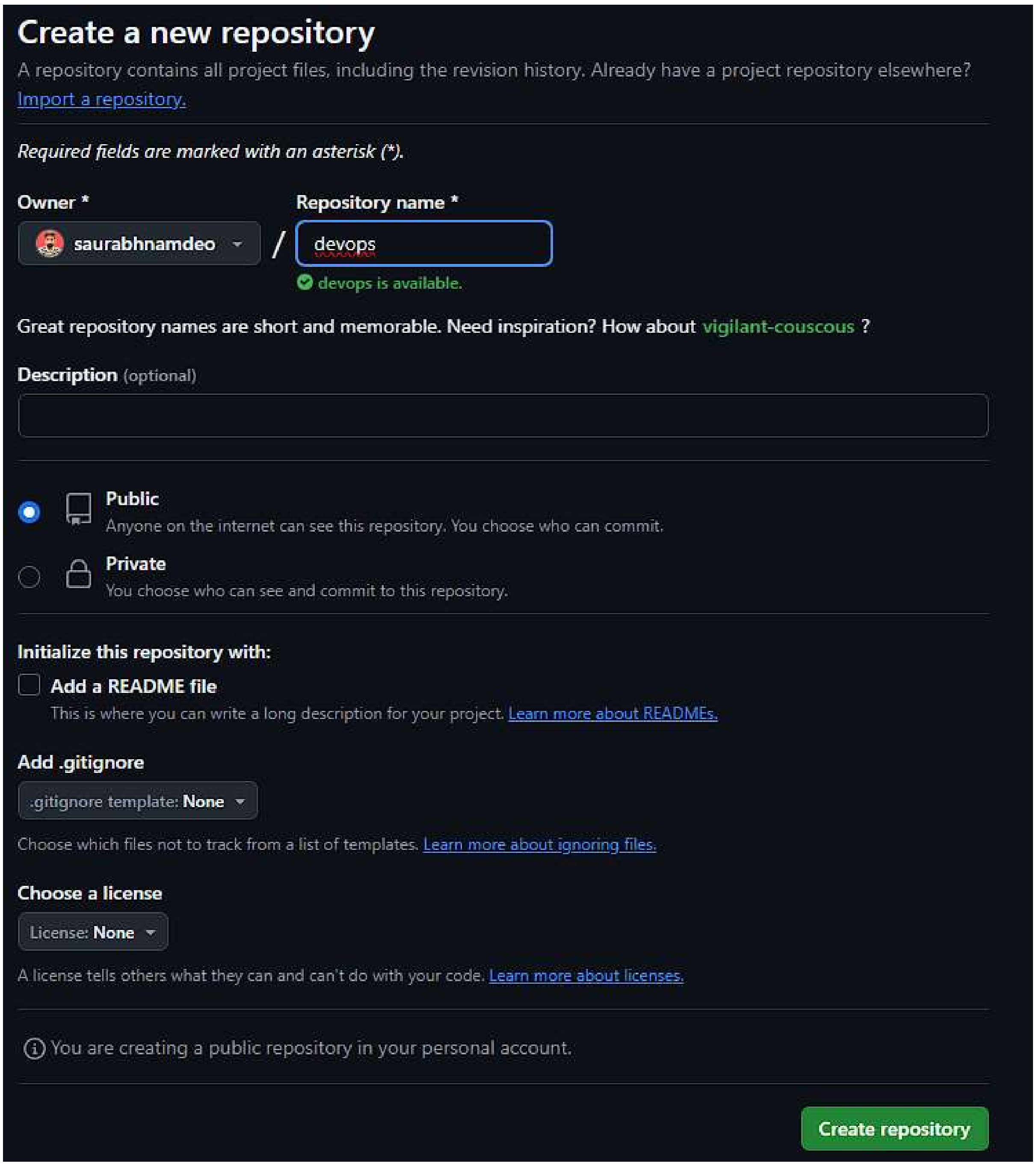
Now your user name and email address are set up in Git, and they will be associated with your commits!



```
ubuntu@Saurabh:~/days_Task/day_12$ git config --global user.name "saurabh"
ubuntu@Saurabh:~/days_Task/day_12$ git config --global user.email "saurabh@gmail.com"
ubuntu@Saurabh:~/days_Task/day_12$ git config --global --list
user.email=saurabh@gmail.com
user.name=saurabh
ubuntu@Saurabh:~/days_Task/day_12$
```

Task 2:

- Create a repository named "DevOps" on GitHub.



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 *: saurabhnamdeo / Repository name *: devops

devops is available.

Great repository names are short and memorable. Need inspiration? How about vigilant-couscous ?

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.

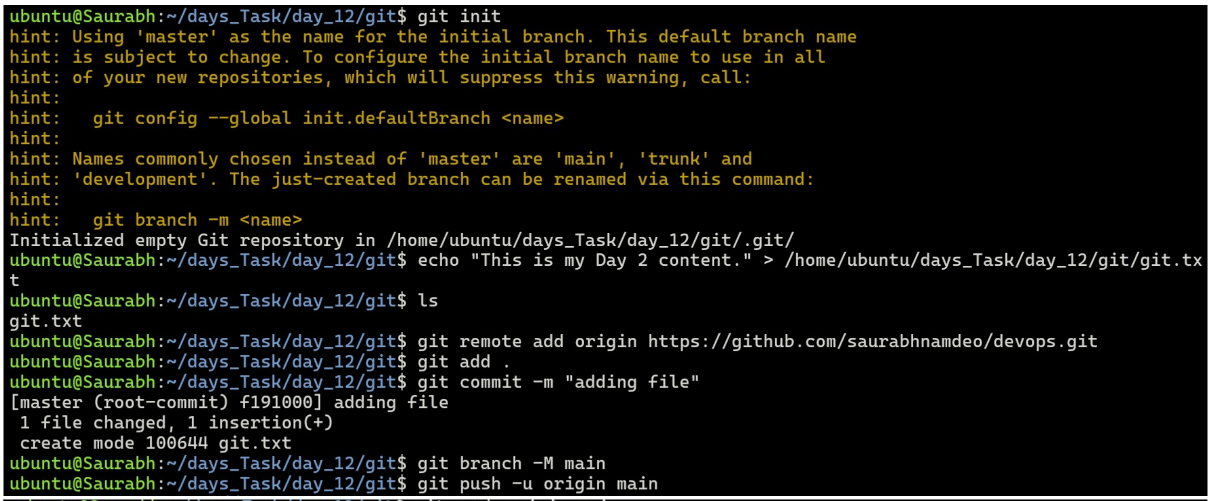
Choose a license

License: None

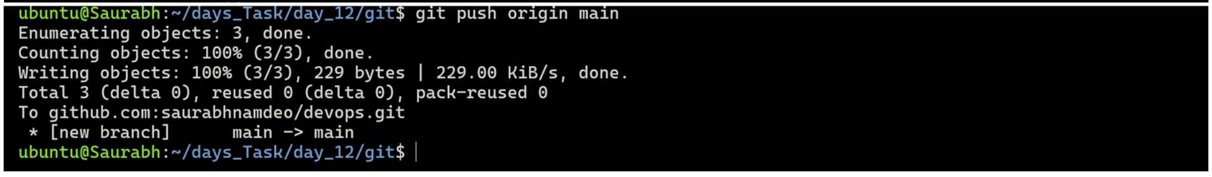
A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

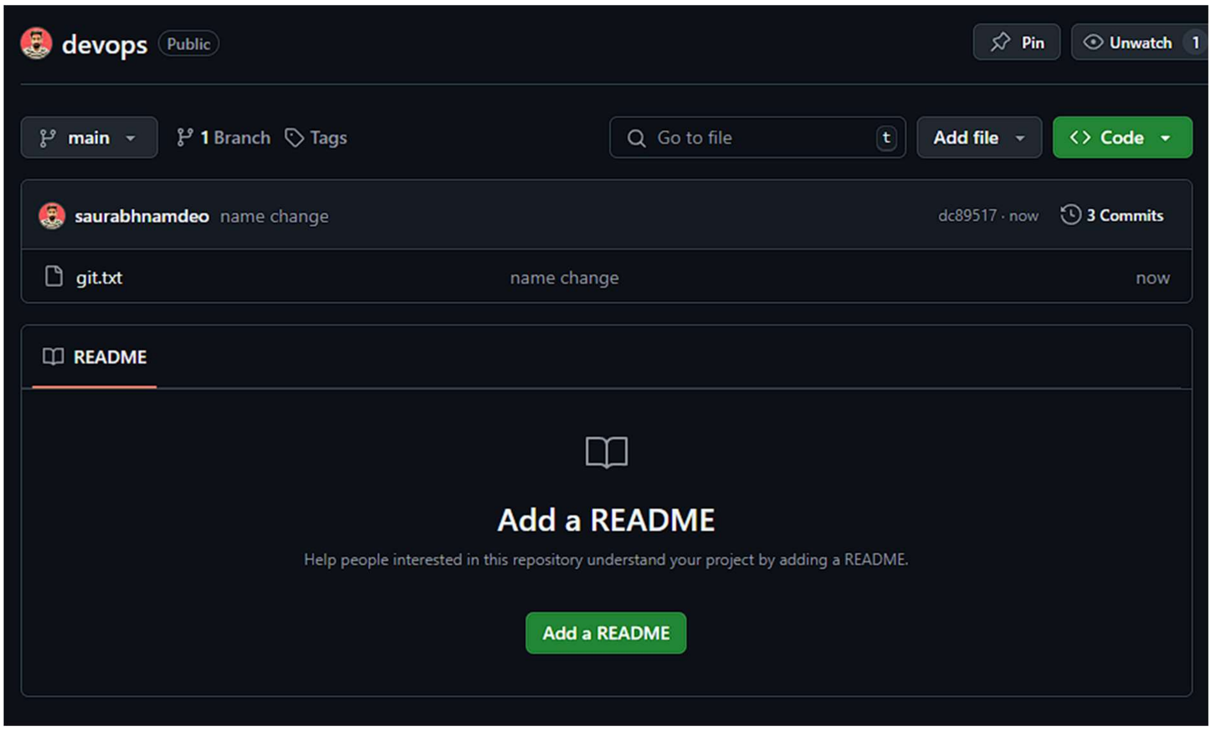
- Connect your local repository to the repository on GitHub.
- Create a new file in git.txt & add some content to it.
- Push your local commits to the repository on GitHub.



```
ubuntu@Saurabh:~/days_Task/day_12/git$ 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/ubuntu/days_Task/day_12/git/.git/
ubuntu@Saurabh:~/days_Task/day_12/git$ echo "This is my Day 2 content." > /home/ubuntu/days_Task/day_12/git/git.tx
t
ubuntu@Saurabh:~/days_Task/day_12/git$ ls
git.txt
ubuntu@Saurabh:~/days_Task/day_12/git$ git remote add origin https://github.com/saurabhnamdeo/devops.git
ubuntu@Saurabh:~/days_Task/day_12/git$ git add .
ubuntu@Saurabh:~/days_Task/day_12/git$ git commit -m "adding file"
[master (root-commit) f191000] adding file
 1 file changed, 1 insertion(+)
 create mode 100644 git.txt
ubuntu@Saurabh:~/days_Task/day_12/git$ git branch -M main
ubuntu@Saurabh:~/days_Task/day_12/git$ git push -u origin main
```



```
ubuntu@Saurabh:~/days_Task/day_12/git$ git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 229 bytes | 229.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:saurabhnamdeo/devops.git
 * [new branch]      main -> main
ubuntu@Saurabh:~/days_Task/day_12/git$
```



saurabhnamdeo name change dc89517 · now 3 Commits

git.txt name change now

Add a README

Help people interested in this repository understand your project by adding a README.

Happy Learning