Basic Git & GitHub

Git

Git is a **version control system** that allows you to track changes to files and coordinate work on those files among multiple people. It is commonly used for software development, but it can be used to track changes to any set of files.

With Git, you can keep a record of who made changes to what part of a file, and you can revert to earlier versions of the file if needed. It also makes it easy to collaborate with others, as you can share changes and merge the changes made by different people into a single version of a file. Being a software developer it is essential to keep track of changes in source code during Development Process. Thus Git serves to be distributed version control system.

Github

GitHub is a **web-based platform** that provides hosting for version control using Git. It is a subsidiary of Microsoft, and it offers all of the distributed version control and source code management (SCM) functionality of Git as well as adding its features. GitHub is a very popular platform for developers to share and collaborate on projects, and it is also used for hosting open-source projects.

Version Control

Version control is a system that tracks changes to a file or set of files over time so that you can recall specific versions later. It allows you to revert files back to a previous state, revert the entire project back to a previous state, compare changes over time, see who last modified something that might be causing a problem, who introduced an issue and when, and more.

There are two main types of version control systems: centralized version control systems and distributed version control systems.

i) A **centralized version control system** (CVCS) uses a central server to store all the versions of a project's files. Developers "check out" files from the central server, make changes, and then "check-in" the updated files. Examples of CVCS include Subversion and Perforce.

ii) A **distributed version control system** (DVCS) allows developers to "clone" an entire repository, including the entire version history of the project. This means that they have a complete local copy of the repository, including all branches and past versions. Developers can work independently and then later merge their changes back into the main repository. Examples of DVCS include Git, Mercurial, and Darcs.

Distributed version control over Centralized version control

i) **Better collaboration**: In a DVCS, every developer has a full copy of the repository, including the entire history of all changes. This makes it easier for developers to work together, as they don't have to constantly communicate with a central server to commit their changes or to see the changes made by others.

ii) **Improved speed**: Because developers have a local copy of the repository, they can commit their changes and perform other version control actions faster, as they don't have to communicate with a central server.

iii) **Greater flexibility**: With a DVCS, developers can work offline and commit their changes later when they do have an internet connection. They can also choose to share their changes with only a subset of the team, rather than pushing all of their changes to a central server.

iv) **Enhanced security**: In a DVCS, the repository history is stored on multiple servers and computers, which makes it more resistant to data loss. If the central server in a CVCS goes down or the repository becomes corrupted, it can be difficult to recover the lost data. Overall, the decentralized nature of a DVCS allows for greater collaboration, flexibility, and security, making it a popular choice for many teams.

Install Git on local machine

You can download it from the official website at https://git-scm.com/downloads

sudo apt-get install git

Create a free account on GitHub

You can sign up from: https://github.com

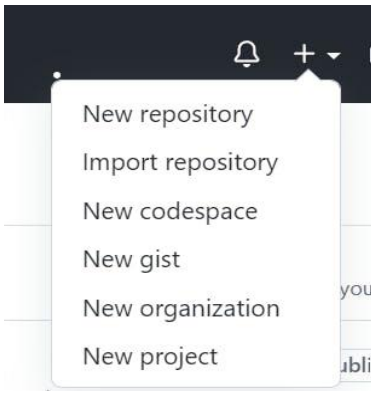
```
Welcome to GitHub!
Let's begin the adventure

Enter your email
→
```

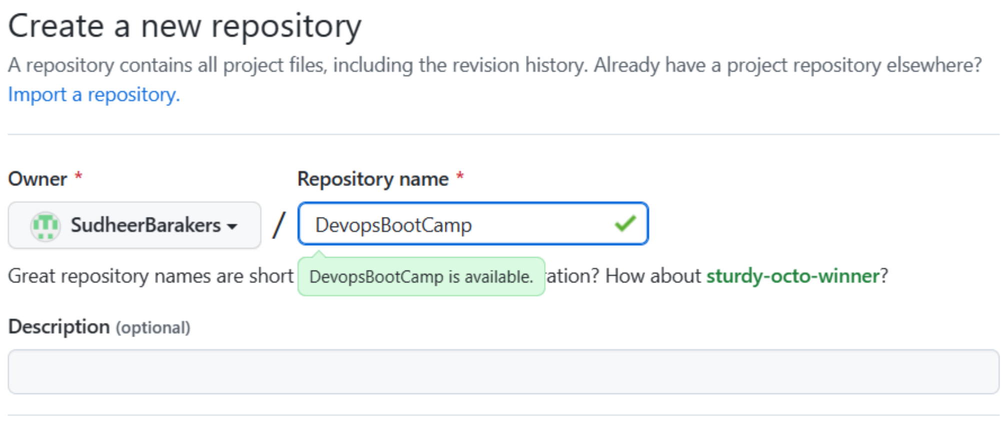
Create a new repository on GitHub and clone it to your local machine

A Repository is a place where you have all your codes or kind of folder on a server.

1. Go to github.com and log in to your account.
2. Click the "New repository" button



3. Enter a name for your repository and a brief description.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

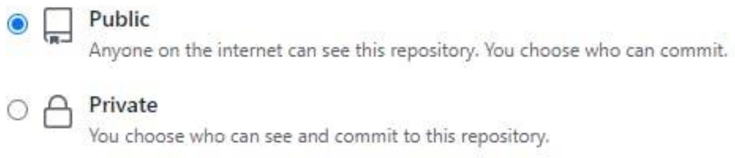
Owner * SudheerBarakers / Repository name * DevopsBootCamp

Great repository names are short DevopsBootCamp is available. ation? How about sturdy-octo-winner?

Description (optional)

4. Choose whether you want the repository to be public or private.

   Choose repository visibility. Click on the public if you want your repository to be cloned by anyone.



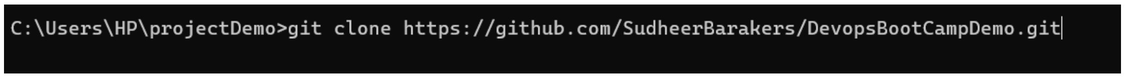
[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private You choose who can see and commit to this repository.

5. Click the “Create repository” button.

You are creating a public repository in your personal account.
Create repository

6. To clone the repository to your local machine, follow these steps:
    1. On the repository page on GitHub, click the “Clone or download” button.
    2. Click the “Copy to clipboard” icon to copy the clone URL.
    3. Open a terminal window and navigate to the directory where you want to clone the repository.
    4. Run the following command: “git clone <clone-url>”



```
C:\Users\HP\projectDemo>git clone https://github.com/SudheerBarakers/DevopsBootCampDemo.git
```

Make some changes to a file in the local repository and commit them to the main repository using Git.

Open the Git bash from the folder /repo you wish to add to the version control system (git)

Opens a text editor called and create a new file called "file.txt".

With following commands, We do

git status: The files listed in **red show that need to be added to the repository.**

git add . : adds the file.

Check the status

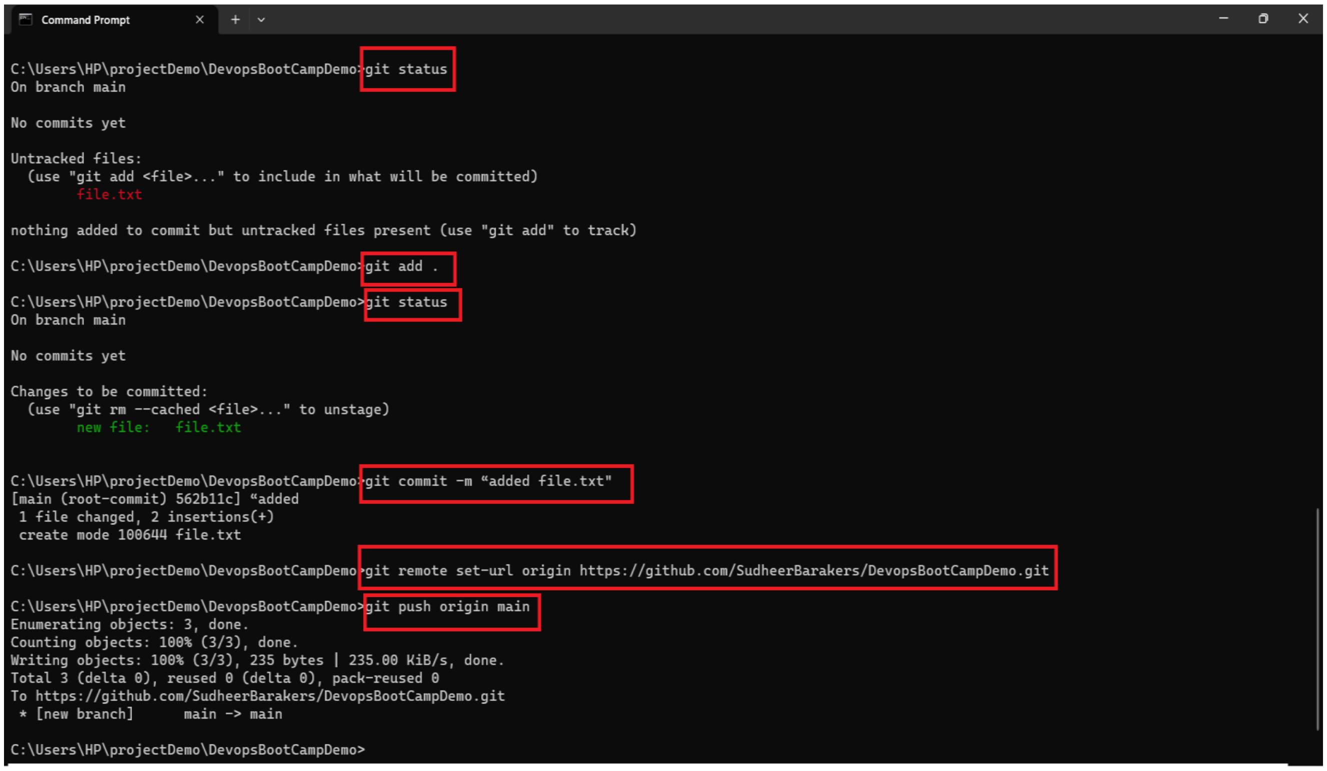
git status

git commit: commit message **, git commit -m “added file.txt"**

git remote set-url origin: used to change the remote repository url.

Ex. git remote set-url origin https://github.com/SudheerBarakers/DevopsBootCampDemo.git

git push origin main: used to push the changes made locally and committed to the remote repository specified.



Command Prompt

```
C:\Users\HP\projectDemo\DevopsBootCampDemo>git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\HP\projectDemo\DevopsBootCampDemo>git add .

C:\Users\HP\projectDemo\DevopsBootCampDemo>git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file.txt


C:\Users\HP\projectDemo\DevopsBootCampDemo>git commit -m “added file.txt"
[main (root-commit) 562b11c] “added
 1 file changed, 2 insertions(+)
 create mode 100644 file.txt

C:\Users\HP\projectDemo\DevopsBootCampDemo>git remote set-url origin https://github.com/SudheerBarakers/DevopsBootCampDemo.git

C:\Users\HP\projectDemo\DevopsBootCampDemo>git push origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 235 bytes | 235.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/SudheerBarakers/DevopsBootCampDemo.git
 * [new branch]      main -> main

C:\Users\HP\projectDemo\DevopsBootCampDemo>
```