Open in app ↗ Sign up Sign In

Sanjana thamke Follow
Feb 19 · 4 min read · Listen

Save

Day 8 Task: Basic Git & GitHub for DevOps Engineers.

What is 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 back to earlier versions of the file if needed. Git 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.

What is 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 own 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.

What is Version Control? How many types of version controls we have?

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.

1. 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.

2. 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.

Why we use distributed version control over centralized version control?

1. 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.

2. 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.

3. 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.

4. 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.

Task:

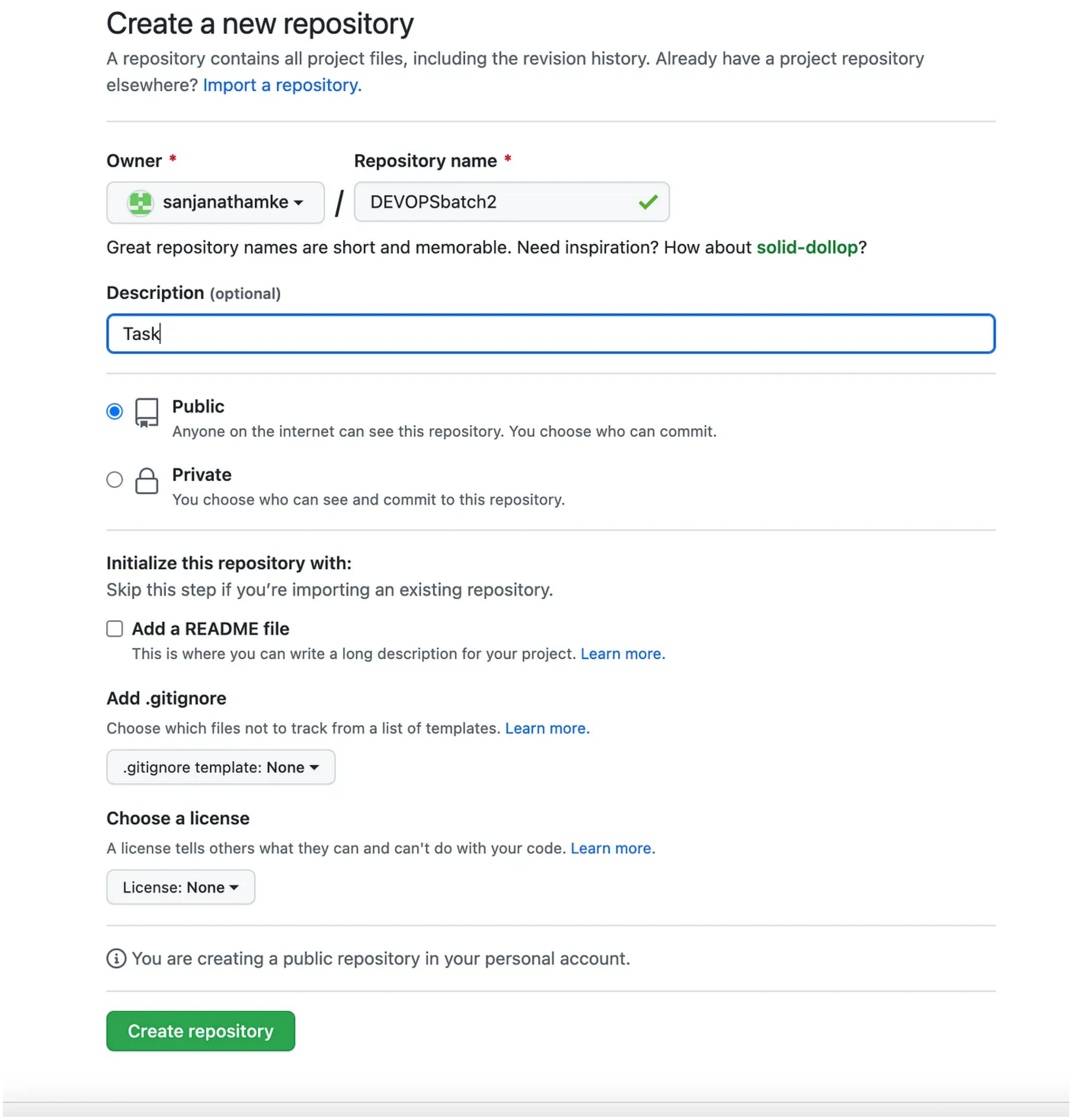
- Install Git on your computer (if it is not already installed). You can download it from the official website at https://git-scm.com/downloads

- Create a free account on GitHub (if you don't already have one). You can sign up at https://github.com/

Exercises:

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

a) create DEVOPSbatch2 repository 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.

Owner * : sanjanathamke
Repository name * : DEVOPSbatch2

Great repository names are short and memorable. Need inspiration? How about solid-dollop?

Description (optional): Task

[✓ 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:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

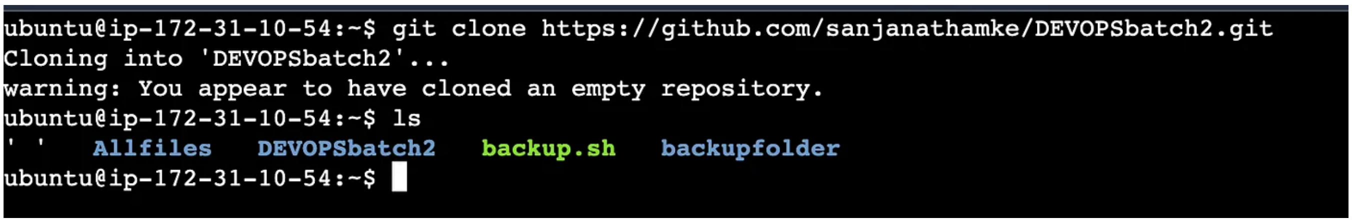
.gitignore template: None

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

License: None

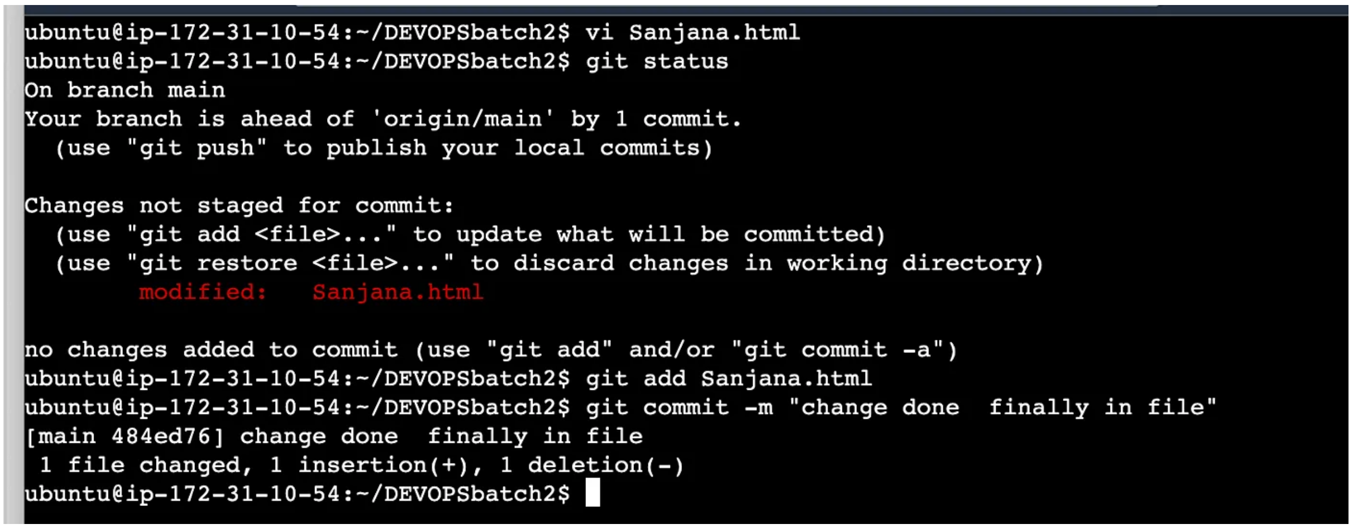
You are creating a public repository in your personal account.

b) clone it to your local machine



```
ubuntu@ip-172-31-10-54:~$ git clone https://github.com/sanjanathamke/DEVOPSbatch2.git
Cloning into 'DEVOPSbatch2'...
warning: You appear to have cloned an empty repository.
ubuntu@ip-172-31-10-54:~$ ls
' '   Allfiles   DEVOPSbatch2   backup.sh   backupfolder
ubuntu@ip-172-31-10-54:~$ 
```

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

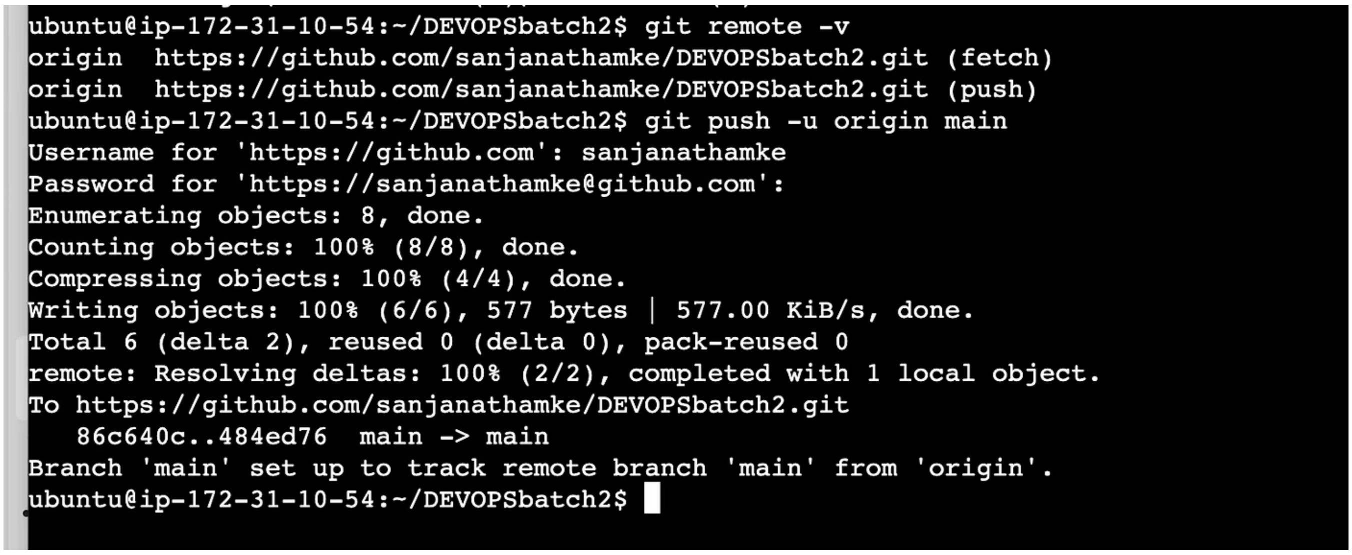


```
ubuntu@ip-172-31-10-54:~/DEVOPSbatch2$ vi Sanjana.html
ubuntu@ip-172-31-10-54:~/DEVOPSbatch2$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Sanjana.html

no changes added to commit (use "git add" and/or "git commit -a")
ubuntu@ip-172-31-10-54:~/DEVOPSbatch2$ git add Sanjana.html
ubuntu@ip-172-31-10-54:~/DEVOPSbatch2$ git commit -m "change done  finally in file"
[main 484ed76] change done  finally in file
 1 file changed, 1 insertion(+), 1 deletion(-)
ubuntu@ip-172-31-10-54:~/DEVOPSbatch2$
```

3.Push the changes back to the repository on GitHub



```
ubuntu@ip-172-31-10-54:~/DEVOPSbatch2$ git remote -v
origin  https://github.com/sanjanathamke/DEVOPSbatch2.git (fetch)
origin  https://github.com/sanjanathamke/DEVOPSbatch2.git (push)
ubuntu@ip-172-31-10-54:~/DEVOPSbatch2$ git push -u origin main
Username for 'https://github.com': sanjanathamke
Password for 'https://sanjanathamke@github.com':
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 577 bytes | 577.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/sanjanathamke/DEVOPSbatch2.git
   86c640c..484ed76  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
ubuntu@ip-172-31-10-54:~/DEVOPSbatch2$
```

[Note: When enter #git push command it asks for username and password so enter username of github account and for password enter personal access token of account

Steps for generate token are :

1. Go to Settings of github account
2. Click on last option Developer setting on left panel
3. Click on Personal access token on left panel
4. Select option Token classic
5. Click on generate token and select generate new token (classic)
6. Enter name , expiry date ,description and other information

7. click on generate token

8. copy the token and use as password]

_Thank you

_Sanjana ✌🏼

Github DevOps Learning 90daysofdevops