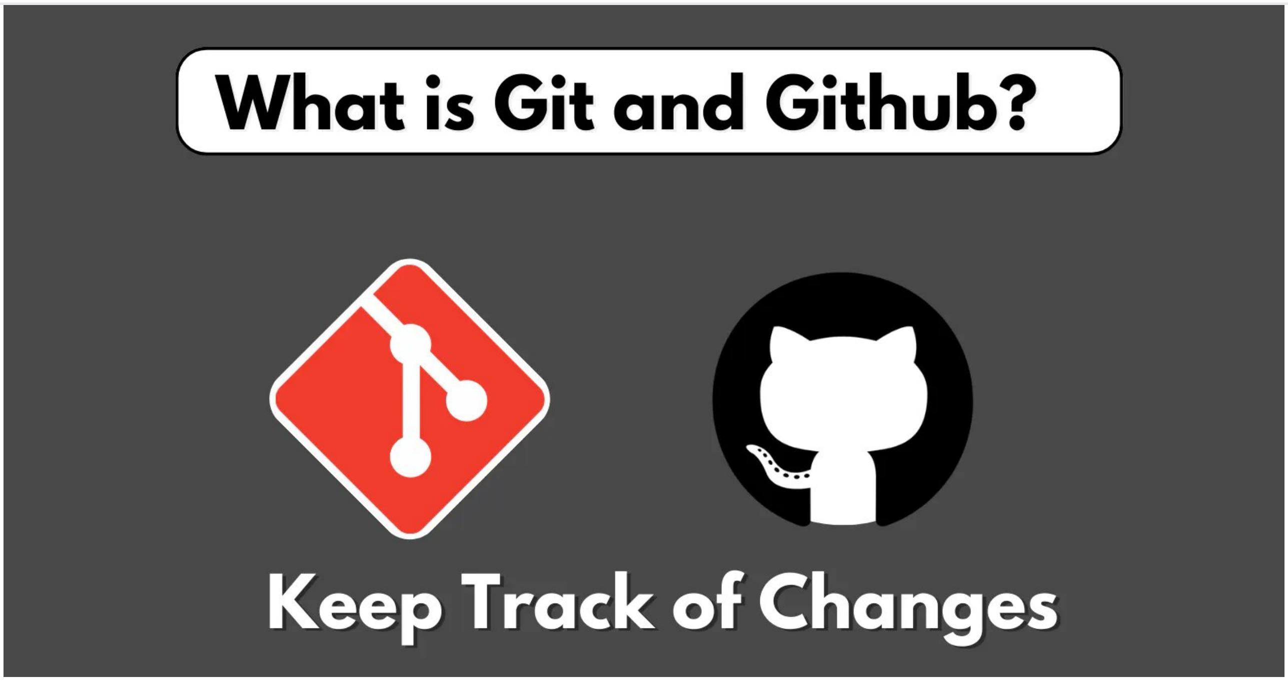
Kunal Maurya's Blog



Day 8 : Basic Git & GitHub for DevOps Engineers.

Kunal Maurya

Mar 1, 2023 · 4 min read

Git and GitHub are two closely related technologies that have revolutionized the way software development is done. In this blog, we'll take a closer look at what Git and GitHub are, what they do, and how they work together.

What is Git?

Git is a version control system that allows developers to keep track of changes to their code over time. With Git, developers can make changes to their code, save those changes, and then revert to previous versions if needed.

This is incredibly useful when multiple people are working on the same codebase, as it allows developers to collaborate without overwriting each other's changes.

What is GitHub?

GitHub is a web-based platform that allows developers to host their Git repositories online. With GitHub, developers can collaborate on code, share their work with others, and contribute to open-source projects.

GitHub offers a wide range of features, including pull requests, code reviews, issue tracking, and more. These features make it easy for developers to collaborate on code and ensure that everyone is on the same page.

What is Version Control?

Version control is a system that allows developers to manage changes to their code over time. It is an essential tool for software development, as it allows multiple developers to work on the same codebase without interfering with each other's work. Version control systems keep track of changes to code, who made the changes, and when they were made.

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

Centralized Version Control: A centralized version control system (CVCS) uses a central repository to store the codebase and manage changes. Developers check out files from the central repository, make changes, and then check the files back in. Examples of CVCS include Subversion and CVS.

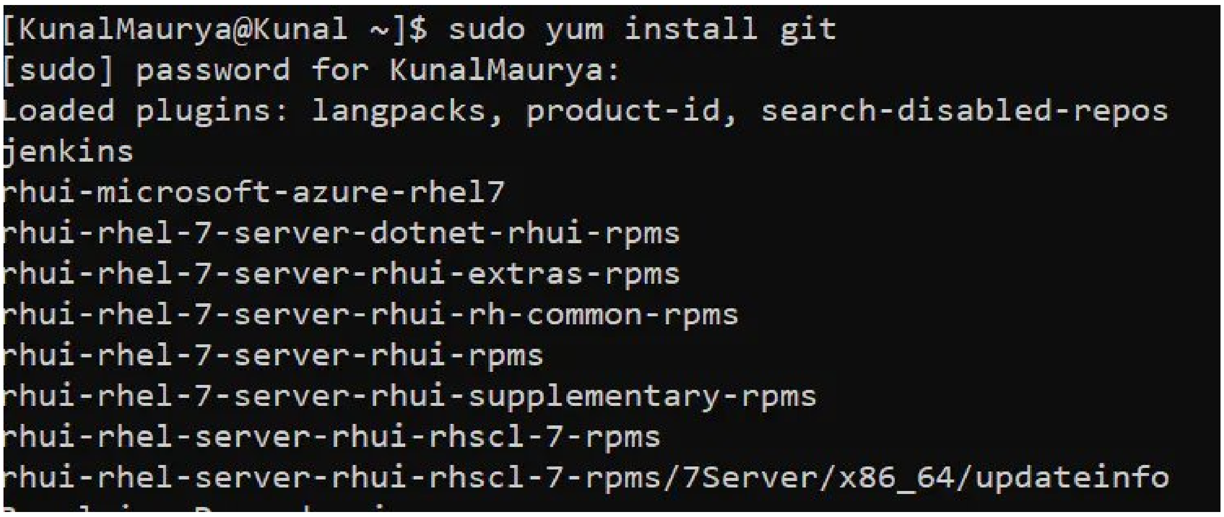
Distributed Version Control: A distributed version control system (DVCS) uses a distributed model, where each developer has their own

copy of the codebase and can make changes independently. Changes can then be merged back into the main codebase as needed. Examples of DVCS include Git and Mercurial.
Why do we use distributed version control over centralized version control?
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.
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.
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.
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.
How to install git?
Use the following command to update your system:

```
sudo yum update
```

Use the following command to install Git:

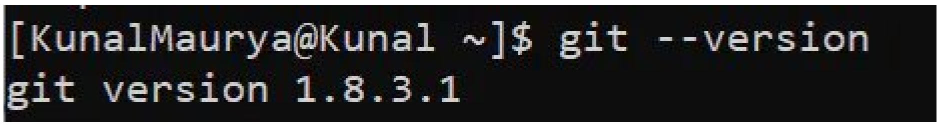
```
sudo yum install git
```



```
[KunalMaurya@Kunal ~]$ sudo yum install git
[sudo] password for KunalMaurya:
Loaded plugins: langpacks, product-id, search-disabled-repos
jenkins
rhui-microsoft-azure-rhel7
rhui-rhel-7-server-dotnet-rhui-rpms
rhui-rhel-7-server-rhui-extras-rpms
rhui-rhel-7-server-rhui-rh-common-rpms
rhui-rhel-7-server-rhui-rpms
rhui-rhel-7-server-rhui-supplementary-rpms
rhui-rhel-server-rhui-rhscl-7-rpms
rhui-rhel-server-rhui-rhscl-7-rpms/7Server/x86_64/updateinfo
```

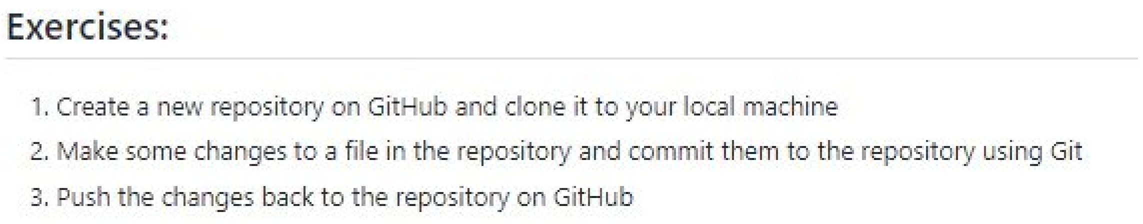
After Git has been installed, you can check the version of Git that was installed using the following command:

```
git --version
```



```
[KunalMaurya@Kunal ~]$ git --version
git version 1.8.3.1
```

Exercises:



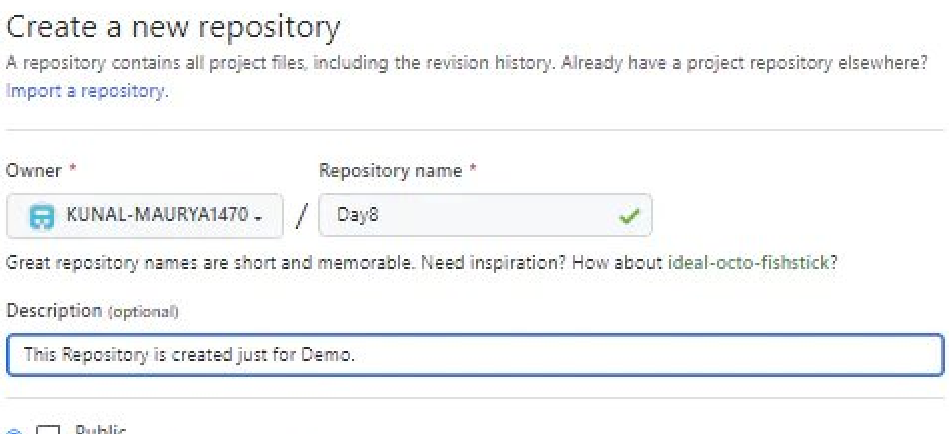
Exercises:

1. Create a new repository on GitHub and clone it to your local machine
2. Make some changes to a file in the repository and commit them to the repository using Git
3. Push the changes back to the repository on GitHub

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

→ Got to https://github.com/

→ Give the name to the repository and click on create repository.



Create a new repository

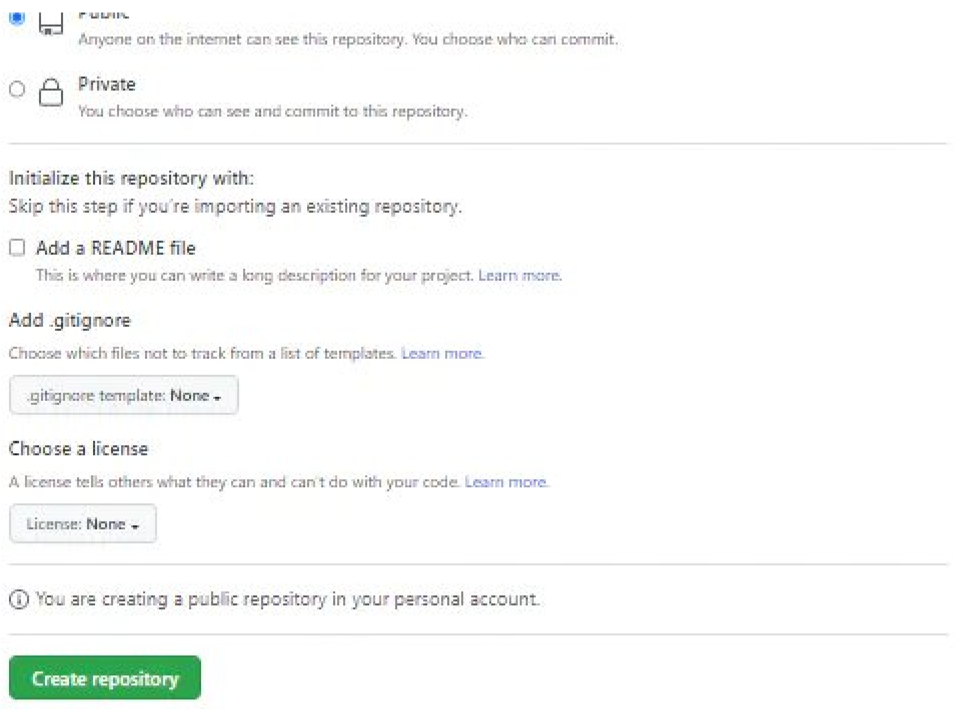
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * KUNAL-MAURYA1470 / Repository name * Day8

Great repository names are short and memorable. Need inspiration? How about ideal-octo-fishstick?

Description (optional)

This Repository is created just for Demo.


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.

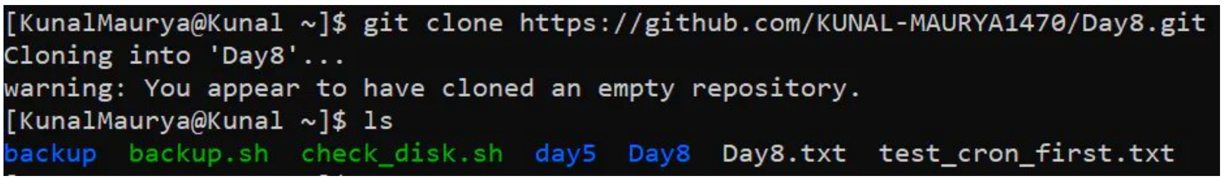
.gitignore template: None

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

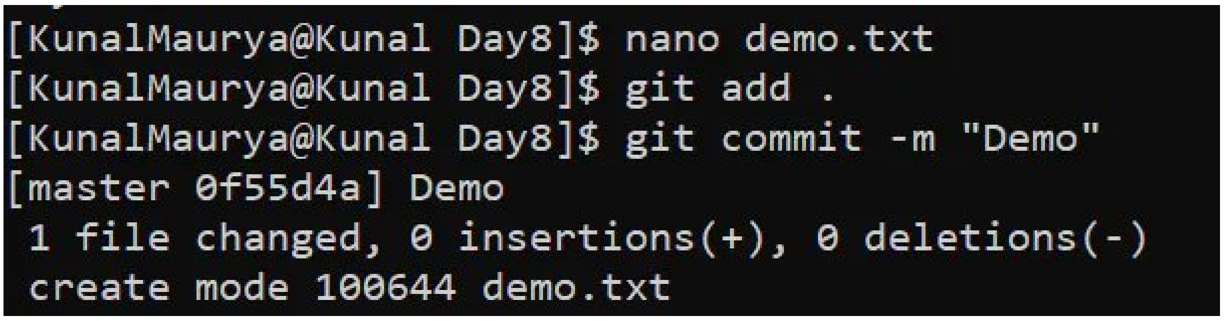
License: None

You are creating a public repository in your personal account.

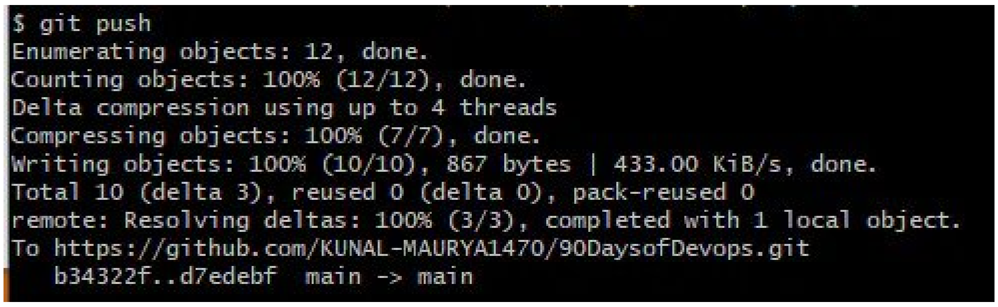
Create repository


```
[KunalMaurya@Kunal ~]$ git clone https://github.com/KUNAL-MAURYA1470/Day8.git
Cloning into 'Day8'...
warning: You appear to have cloned an empty repository.
[KunalMaurya@Kunal ~]$ ls
backup  backup.sh  check_disk.sh  day5  Day8  Day8.txt  test_cron_first.txt
```

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


```
[KunalMaurya@Kunal Day8]$ nano demo.txt
[KunalMaurya@Kunal Day8]$ git add .
[KunalMaurya@Kunal Day8]$ git commit -m "Demo"
[master 0f55d4a] Demo
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 demo.txt
```

Push the changes back to the repository on GitHub.


```
$ git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (10/10), 867 bytes | 433.00 KiB/s, done.
Total 10 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/KUNAL-MAURYA1470/90DaysofDevops.git
   b34322f..d7edebf  main -> main
```

Thank you for reading! Hope you find this article helpful.

~Kunal