Git Introduction

What is Version Control System?

- Version control, also known as source control, is the practice of tracking and managing changes to software code.
- Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter.
- Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.
- There are two types of version control system:

1. **Centralized Version Control System:** In a centralized system all team members connect to a central sever to get the latest copy of the code, and to share their changes with others. The problem with a Centralized VCS is the single point of failure, if the server goes offline, it is impossible to collaborate, or continue to take snapshots of the code. Examples are subversion and Microsoft team foundation server.

2. **Distributed Version Control System:** In a distributed system each team member as a full copy of the code and repository in its machine, if the server goes offline, they can keep on working. Examples are Git and Mercurial.

Benefits of VCS:-

- Allows developers to work simultaneously
- Does not allow overwriting each other’s changes.
- Maintains a history of every version.
- Branching and merging.
- Traceability

What is Git?

- Git is the most popular **Version Control System (VCS)** in the world that records the changed made to our code in a special database called repository.
- It enables us to look into our project history, and see what changes were made, by whom, when and why. And if we mess something it we can easily revert our project back to an earlier state.

Advantages of Git:-

1. Free and open source
2. Fast and small
3. Implicit backup
4. Security
5. Scalable
6. Easier branching

Installing Git:-

- If you are using Debian base GNU/Linux distribution, then **apt-get** command will do the needful.

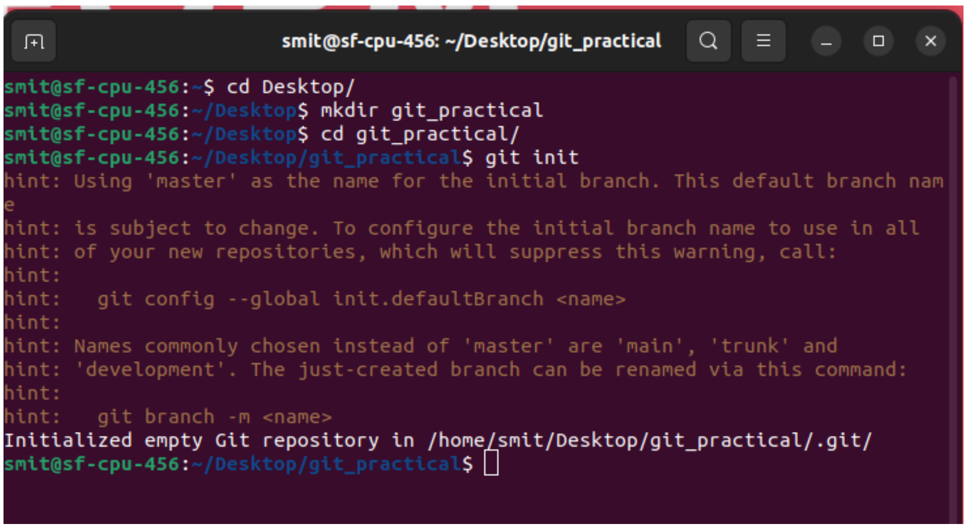
```
[ubuntu ~]$ sudo apt-get install git-core
        [sudo] password for ubuntu:


[ubuntu ~]$ git –version

git version 1.8.1.2
```

- And if you are using RPM based GNU/Linux distribution, then use **yum** command as given.

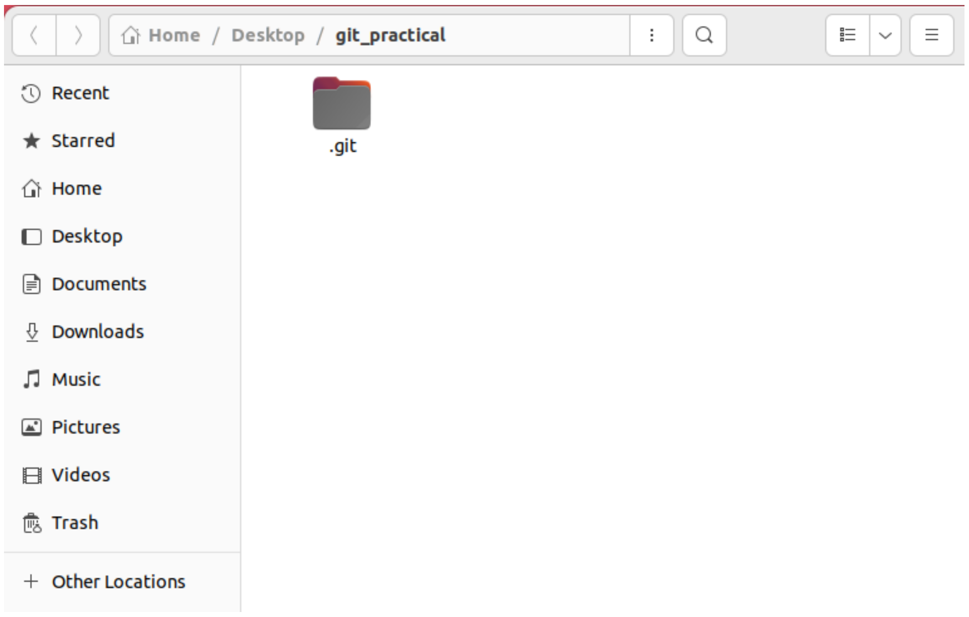
```
[CentOS ~]# yum -y install git-core

[CentOS ~]# git --version
git version 1.7.1
```

Initializing repository in Git:-

- A Git repository is a virtual storage of your project. It allows us to save versions of your code, which we can access when needed.
- To create a new repo, we need to use the **git init** command. git init is a one-time command we use during the initial setup of a new repo.
- Executing this command will create a new .git subdirectory in your current working directory. This will also create a new main branch.



```
smit@sf-cpu-456:~$ cd Desktop/
smit@sf-cpu-456:~/Desktop$ mkdir git_practical
smit@sf-cpu-456:~/Desktop$ cd git_practical/
smit@sf-cpu-456:~/Desktop/git_practical$ git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
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/smit/Desktop/git_practical/.git/
smit@sf-cpu-456:~/Desktop/git_practical$ 
```



Home / Desktop / git_practical
Recent
Starred
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
Trash
Other Locations
.git