Git

What is git? How is it different from svn or perforce?
Let us understand these 2 concepts.

1. Git is a distributed VCS(version control system);

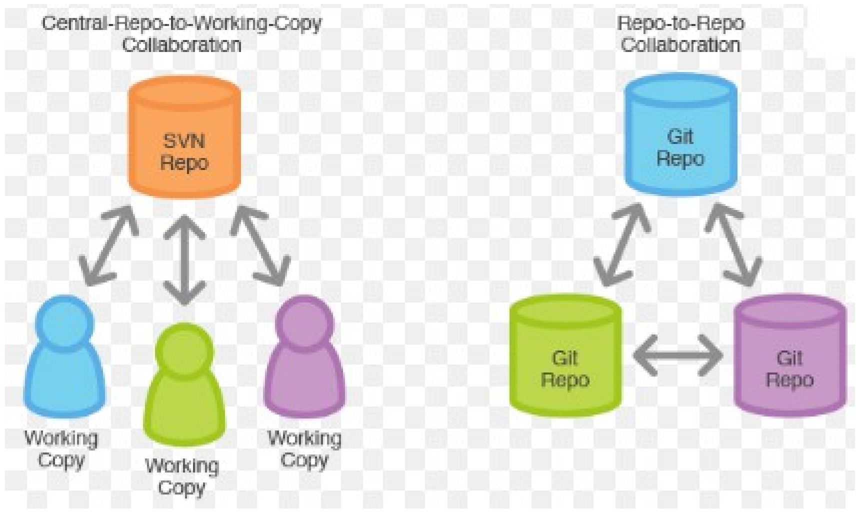
   SVN is a non-distributed VCS.
2. Git has a centralized server and repository;

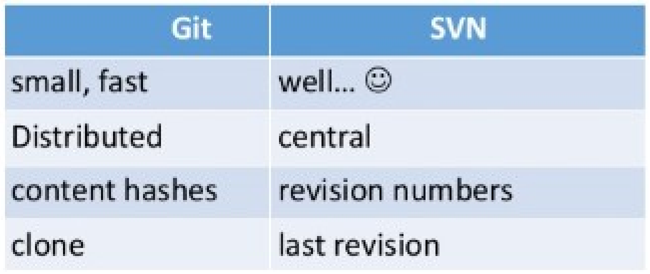
   SVN does not have a centralized server or repository.
3. The content in Git is stored as metadata;

   SVN stores files of content.
4. Git branches are easier to work with than SVN branches.
5. Git does not have the global revision number feature like SVN has.
6. Git has better content protection than SVN.
7. Git was developed for Linux kernel by Linus Torvalds;

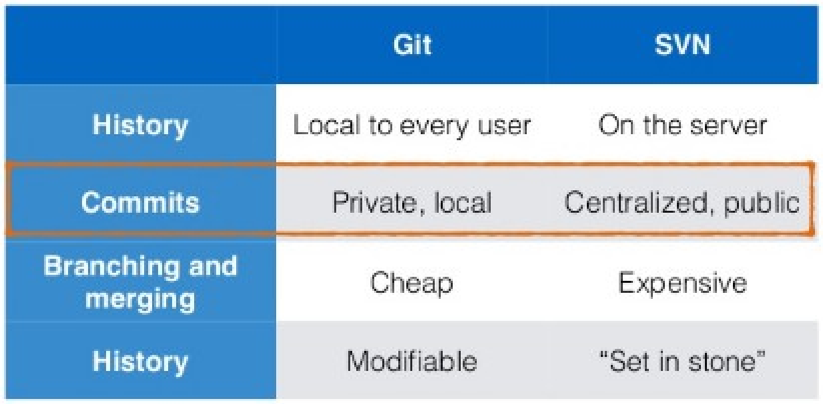
   SVN was developed by CollabNet, Inc.
8. Git is distributed under GNU, and its maintenance overseen by Junio Hamano; Apache Subversion, or SVN, is distributed under the open source license.

Read more: Difference Between Git and SVN | Difference Between http://www.differencebetween.net/technology/software-technology/difference-between-git-and-svn/#ixzz5Wo7nwzVa

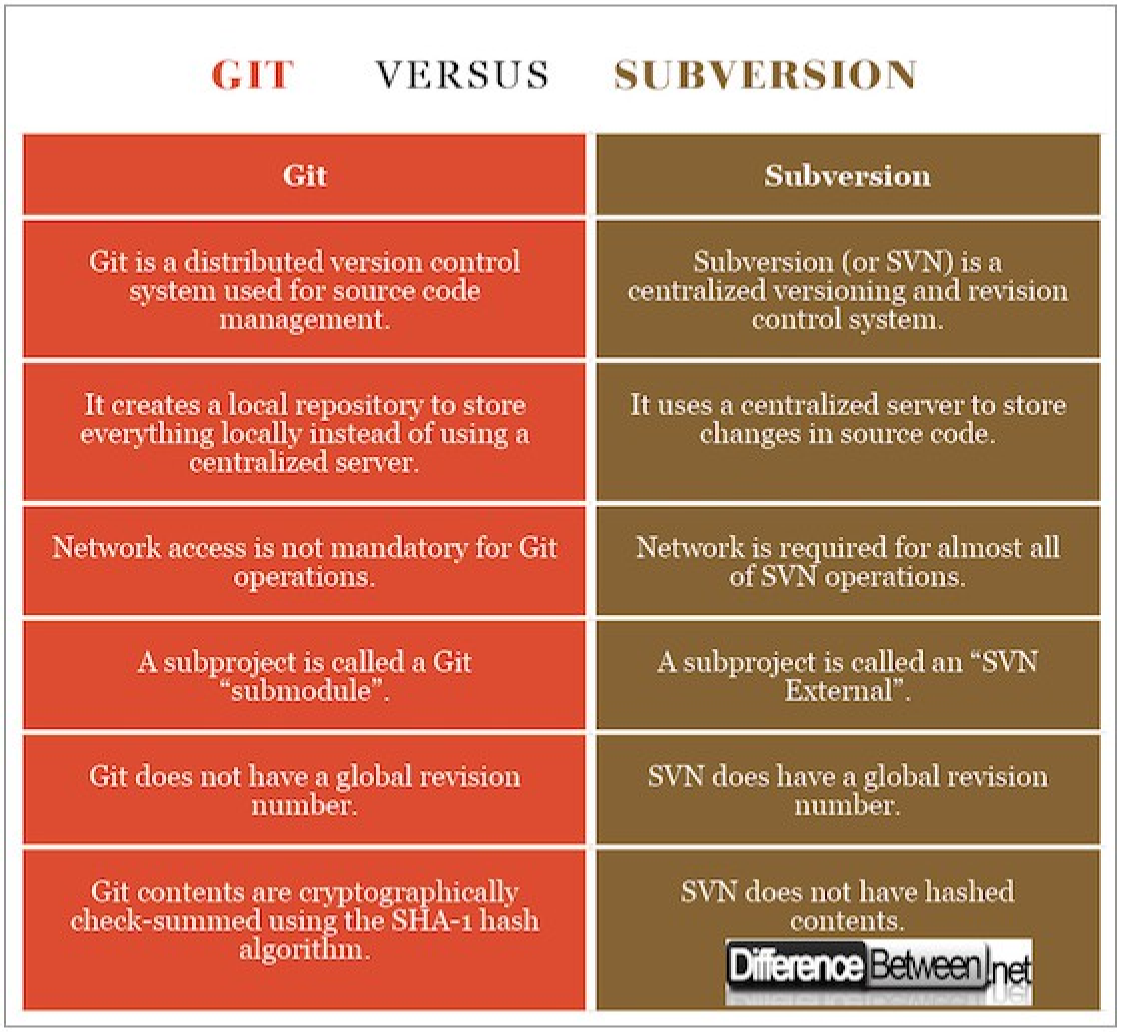

Central-Repo-to-Working-Copy Collaboration
SVN Repo
Working Copy
Working Copy
Working Copy
Repo-to-Repo Collaboration
Git Repo
Git Repo
Git Repo



Git	SVN
small, fast	well… ☺
Distributed	central
content hashes	revision numbers
clone	last revision



	Git	SVN
History	Local to every user	On the server
Commits	Private, local	Centralized, public
Branching and merging	Cheap	Expensive
History	Modifiable	"Set in stone"



GIT VERSUS SUBVERSION

Git	Subversion
Git is a distributed version control system used for source code management.	Subversion (or SVN) is a centralized versioning and revision control system.
It creates a local repository to store everything locally instead of using a centralized server.	It uses a centralized server to store changes in source code.
Network access is not mandatory for Git operations.	Network is required for almost all of SVN operations.
A subproject is called a Git "submodule".	A subproject is called an "SVN External".
Git does not have a global revision number.	SVN does have a global revision number.
Git contents are cryptographically check-summed using the SHA-1 hash algorithm.	SVN does not have hashed contents.

DifferenceBetween.net

http://www.differencebetween.net/technology/difference-between-git-and-subversion/

Git Commands that will be covered in this tutorial:

1. git init
2. git add
3. git status
4. git config
5. git commit
6. git log
7. git diff
8. git branch
9. git remote
10. git push
11. git clone
12. git pull
13. git revert
14. git checkout
15. git fetch
16. git merge
17. .gitignore
18. git mv
19. git blame
20. git show
21. git rm
22. git format-patch
23. git am / git apply
24. git reset
25. git tag
26. git rebase
27. git clean
28. git replace
29. git whatchanged
30. git stash

Color conventions used in this document:

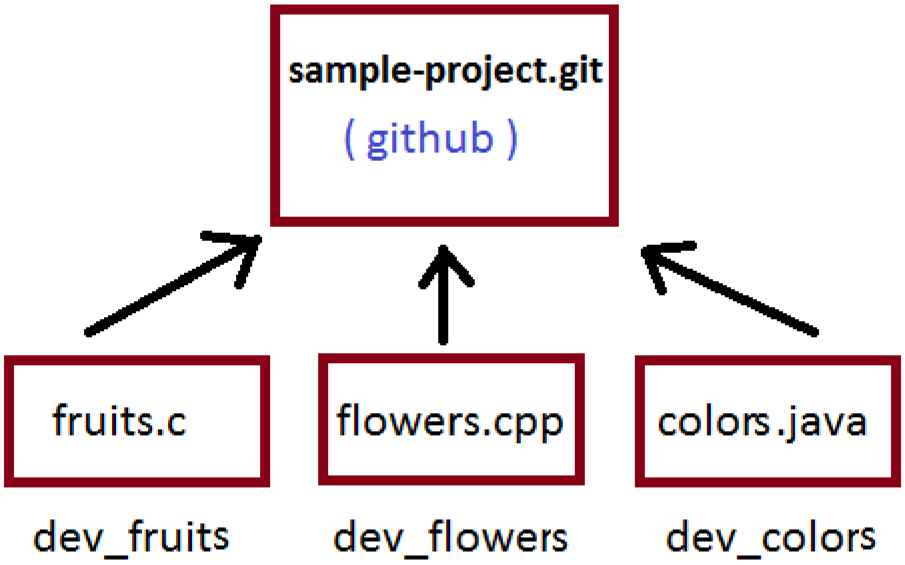
Purple:	Explanation
Bold:	Commands
Green:	Output of the command
Red:	Error messages

Note: All the Commands are executed in Ubuntu Machine:

How are we going to understand the usage of git commands?

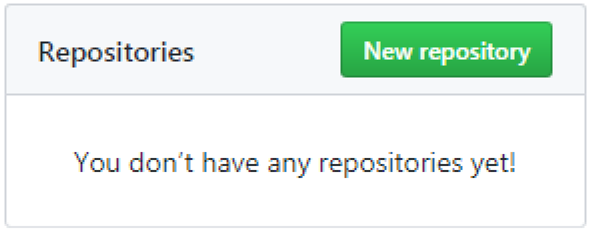
1. We are going to create an empty repository in **github** by name 'sample-project.git'.
2. We have three developers in the team contributing to this project.
3. We will create three directories and assume that each developer is working from one directory. This assumption is made to avoid creating multiple users and switching from one user to another.
4. dev_fruits, dev_flowers, dev_colors are our three developers. In this case it is three directories.
5. We will create few files like fruits.c, flowers.cpp, colors.java, sweets.py (**not** all at once) to understand the usage of git commands in this project.

These assumptions can be visualized as below.

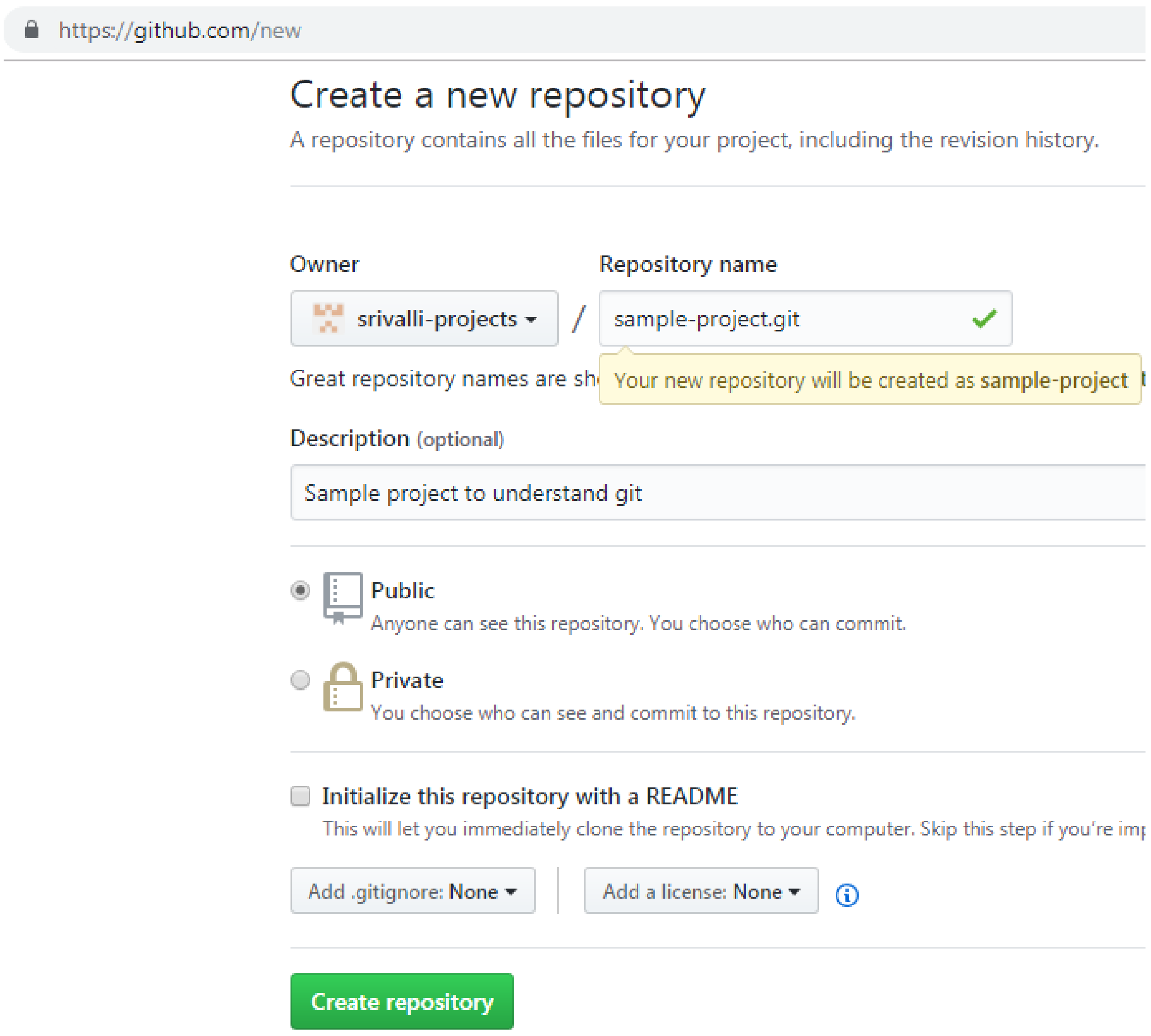


Now, create an account in https://github.com and verify your email.

Create a new repository in that.



You don't have any repositories yet!


https://github.com/new

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: srivalli-projects / Repository name: sample-project.git

Great repository names are sh

Your new repository will be created as **sample-project**

Description (optional): Sample project to understand git

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

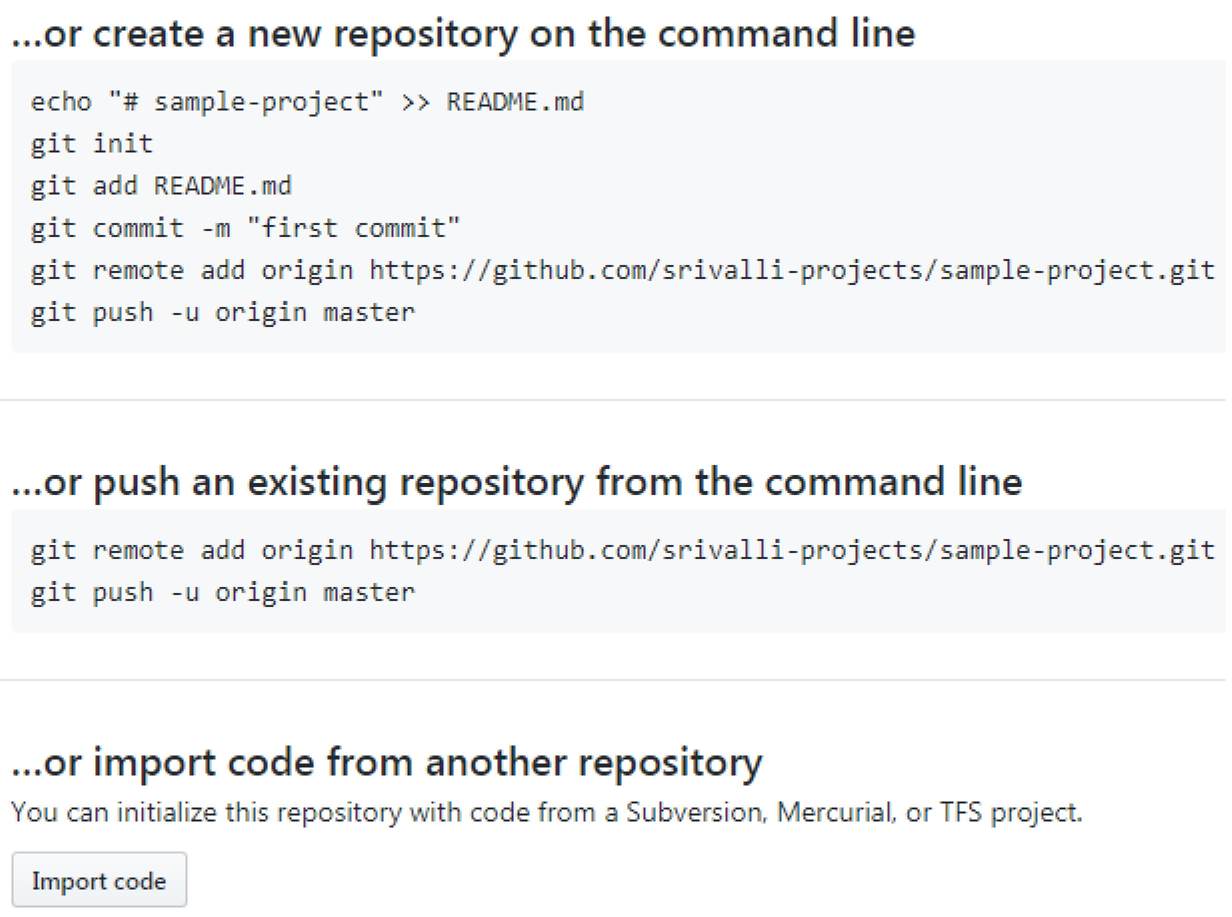
Private — You choose who can see and commit to this repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're imp

Please select the options as shown in the picture.

We are provided with below commands to get started with this project.



...or create a new repository on the command line

```
echo "# sample-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/srivalli-projects/sample-project.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/srivalli-projects/sample-project.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

But before we start with git init, we will understand the need of this command.

Let us create a directory by name dev_fruits.

```
fms@git_practice:~$ mkdir dev_fruits
fms@git_practice:~$ cd dev_fruits
fms@git_practice:~/dev_fruits$
```

Create a file by name fruits.c

```
fms@git_practice:~/dev_fruits$ vi fruits.c
fms@git_practice:~/dev_fruits$
```

cat command displays contents in the file.

```
fms@git_practice:~/dev_fruits$ cat fruits.c
Apple
Pine Apple
fms@git_practice:~/dev_fruits$
```

Edit the file again.

```
fms@git_practice:~/dev_fruits$ vi fruits.c
fms@git_practice:~/dev_fruits$
```

```
fms@git_practice:~/dev_fruits$ cat fruits.c
Apple
Custard Apple
Pine Apple
fms@git_practice:~/dev_fruits$
```

We have changed the content in the file.

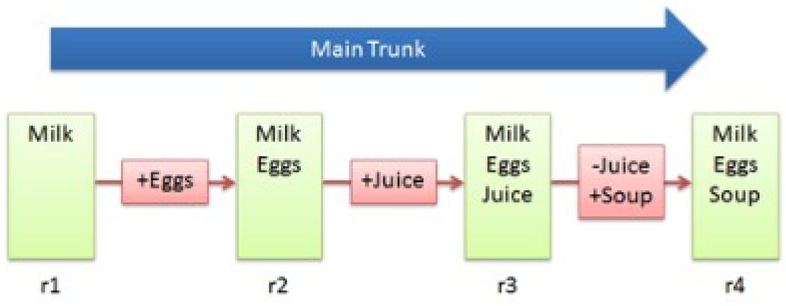
Is there a way to go back to the previous version of this file to get only those 2 lines in the file? NO.

We have only one file in the directory with only one revision.

```
fms@git_practice:~/dev_fruits$ ls -al
total 12
drwxr-xr-x  2 fms fms 4096 Nov  5 11:18 .
drwxr-xr-x 27 fms fms 4096 Nov  5 11:18 ..
-rw-r--r--  1 fms fms   31 Nov  5 11:18 fruits.c
fms@git_practice:~/dev_fruits$
```

But this is not a good practice to continue developing our code without maintaining the revisions of the file.

Ideal way of developing the code is as below.



Since we cannot keep track of the changes under this directory, we will take the help of git.
Initialize git in this directory as below.

```
fms@git_practice:~/dev_fruits$ git init
Command 'git' not found, but can be installed with:
sudo apt install git
```

Oops, git is not installed in this Ubuntu machine
Let’s install it.

```
fms@git_practice:~/dev_fruits$ sudo apt install git
[sudo] password for fms:
Reading package lists... Done
```

Now init again

```
fms@git_practice:~/dev_fruits$ git init
Initialized empty Git repository in /home/fms/dev_fruits/.git/
fms@git_practice:~/dev_fruits$
```

We get a new folder by name .git

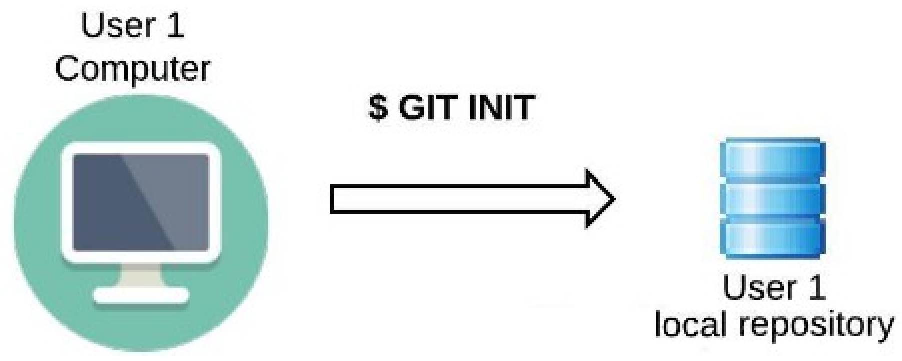
```
fms@git_practice:~/dev_fruits$ ls -al
total 16
drwxr-xr-x  3 fms fms 4096 Nov  5 11:19 .
drwxr-xr-x 27 fms fms 4096 Nov  5 11:18 ..
-rw-r--r--  1 fms fms   31 Nov  5 11:18 fruits.c
drwxr-xr-x  7 fms fms 4096 Nov  5 11:19 .git
fms@git_practice:~/dev_fruits$
```

Move to .git folder to check the files and folders that are maintained by .git

```
fms@git_practice:~/dev_fruits$ cd .git
fms@git_practice:~/dev_fruits/.git$

fms@git_practice:~/dev_fruits/.git$ ls -al
total 40
drwxr-xr-x 7 fms fms 4096 Nov  5 11:19 .
drwxr-xr-x 3 fms fms 4096 Nov  5 11:19 ..
drwxr-xr-x 2 fms fms 4096 Nov  5 11:19 branches
-rw-r--r-- 1 fms fms   92 Nov  5 11:19 config
-rw-r--r-- 1 fms fms   73 Nov  5 11:19 description
-rw-r--r-- 1 fms fms   23 Nov  5 11:19 HEAD
drwxr-xr-x 2 fms fms 4096 Nov  5 11:19 hooks
drwxr-xr-x 2 fms fms 4096 Nov  5 11:19 info
drwxr-xr-x 4 fms fms 4096 Nov  5 11:19 objects
drwxr-xr-x 4 fms fms 4096 Nov  5 11:19 refs
fms@git_practice:~/dev_fruits/.git$
```

We will see what git stores revisions/history of our workspace files in these .git directories/files. In short, .git is the directory where the metadata of your repository is stored.



Git commands covered in this doc:

1. git init