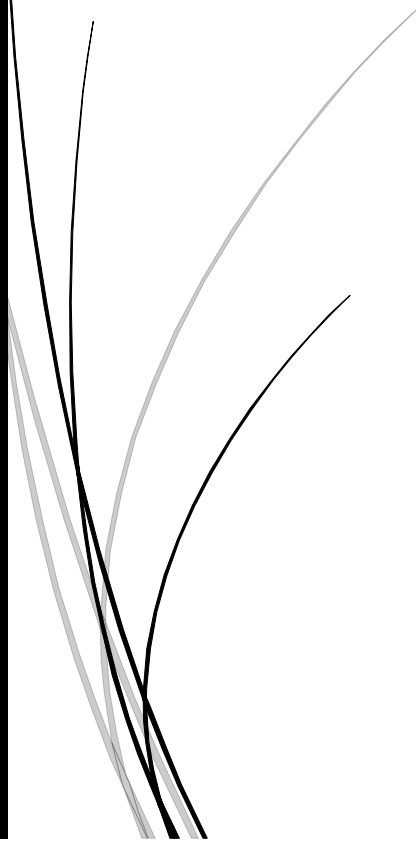
4/9/2022

SCM TASK 1



BY-

Aaryan Singh

ROLL NO. 2110990019

About Version Control:--

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. 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.

benefits of version control systems:-

- **Generate Backups:** Creating a backup of the current version of that repository is perhaps the most important benefit of using a version control system. Having numerous backups on various machines is beneficial because it protects data from being lost in the event of a server failure.
- **Experiment:** When a team works on a software project, they frequently use clones of the main project to build new features, test them, and ensure that they work properly before adding them to the main project. This could save time as different portions of the code can be created simultaneously.
- **Keep History:** Keeping track of the changes in a code file would assist you and new contributors understand how a certain section of the code was created. How did it begin and evolve over time to get at its current state.
- **Collaboration:** One of the most important advantages of version control systems, particularly DVCS, is that it allowed us to participate to projects we enjoyed despite the fact that we were in separate countries

Local Version Control Systems

Many people's version-control method of choice is to copy files into another directory. This approach is very common because it is so simple, but it is also incredibly error prone. It is easy to forget which directory you're in and accidentally write to the wrong file or copy over files you don't mean to.

Centralized Version Control Systems

Centralized version control systems are based on the idea that there is a single "central" copy of your project somewhere (probably on a server), and programmers will "commit" their changes to this central copy. Committing a change simply means recording the change in the central system.

Main benefits (CVCS):

- Centralized systems are typically easier to understand and use
- You can grant access level control on directory level
- performs better with binary files

Distributed Version Control Systems

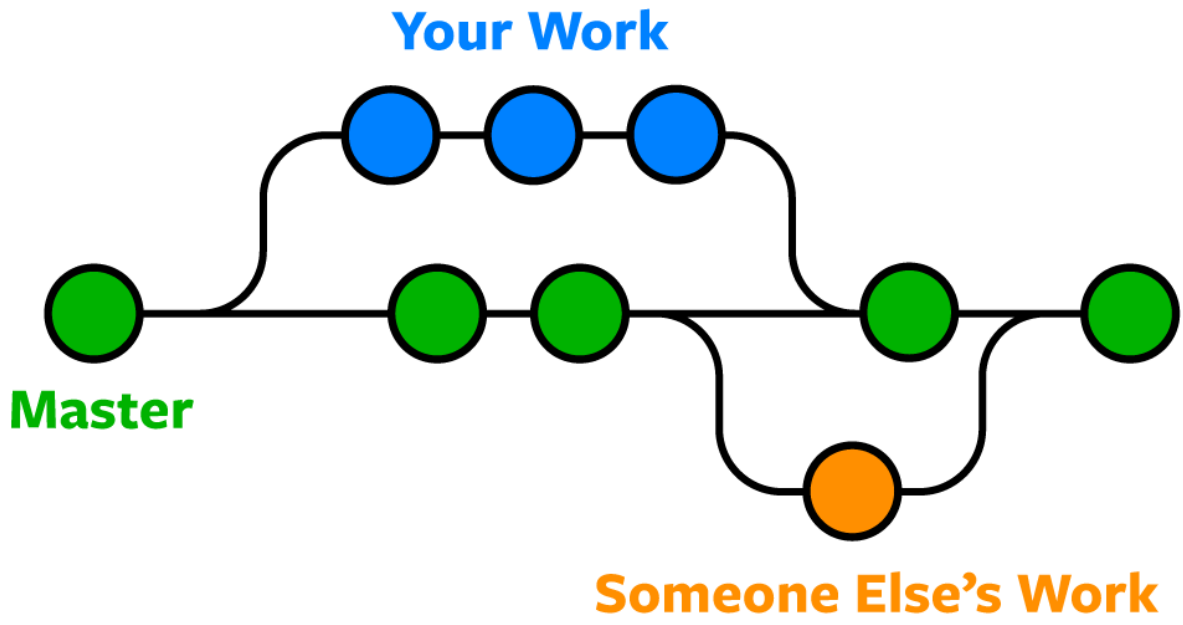
In distributed version control, every developer "clones" a copy of a repository and has the full history of the project on their own hard drive. This copy (or "clone") has all of the metadata of the original.

Main benefits (DVCS):

- Performance of distributed systems is better
- Branching and merging is much easier
- With a distributed system, you don't need to be connected to the network all the time (complete code repository is stored locally on PC)

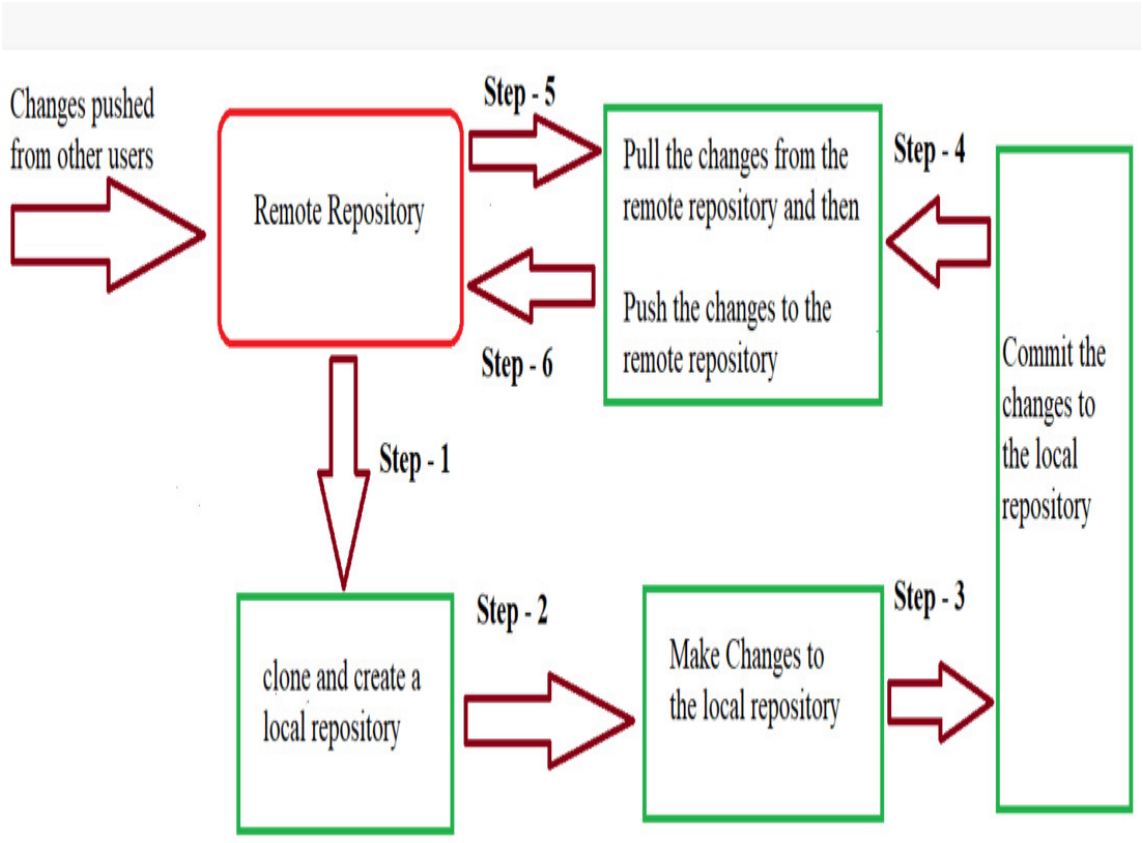
What is Git?

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.



Git Life Cycle:-

Git is used in our day-to-day work, we use git for keeping a track of our files, working in a collaboration with our team, to go back to our previous code versions if we face some error. Git helps us in many ways. Let us look at the Life Cycle that git has and understand more about its life cycle. Let us see some of the basic steps that we follow while working with Git



- ***In Step – 1****, We first clone any of the code residing in the remote repository to make our won local repository.*
- ***In Step-2*** *we edit the files that we have cloned in our local repository and make the necessary changes in it.*
- ***In Step-3*** *we commit our changes by first adding them to our staging area and committing them with a commit message.*

- ***In Step – 4 and Step-5*** *we first check whether there are any of the changes done in the remote repository by some other users and we first pull that changes.*
- *If there are no changes we directly proceed with* ***Step – 6*** *in which we push our changes to the remote repository and we are done with our work.*

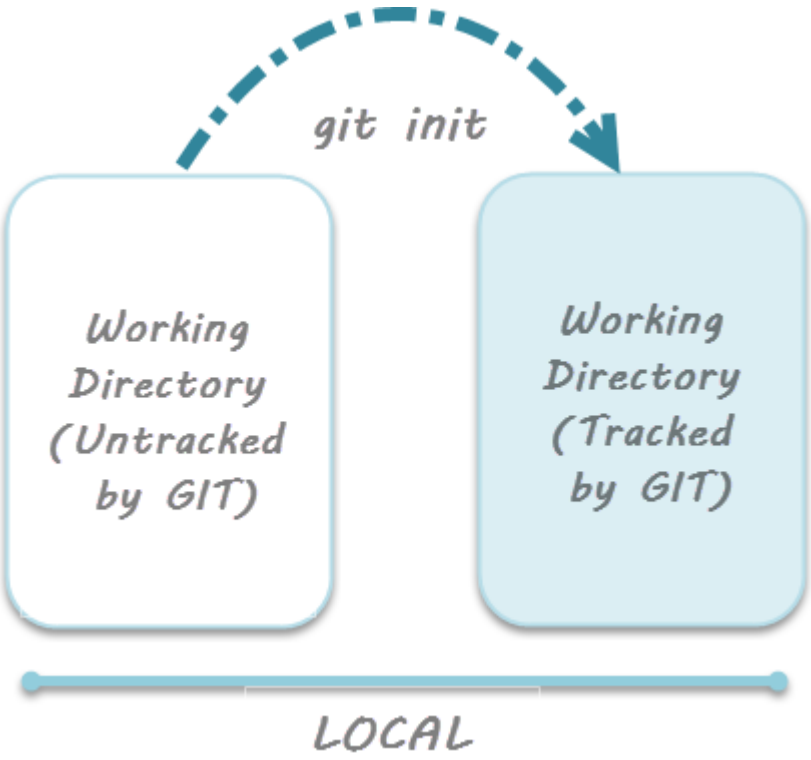
When a directory is made a git repository, there are mainly 3 states which make the essence of Git Version Control System. The three states are-

- Working directory
- Staging area
- Git directory

Working Directory

Whenever we want to initialize our local project directory to make it a git repository, we use the ***git init*** command. After this command, git becomes aware of the files in the project although it doesn't track the files yet. The files are further tracked in the staging area.

git init



Staging Area

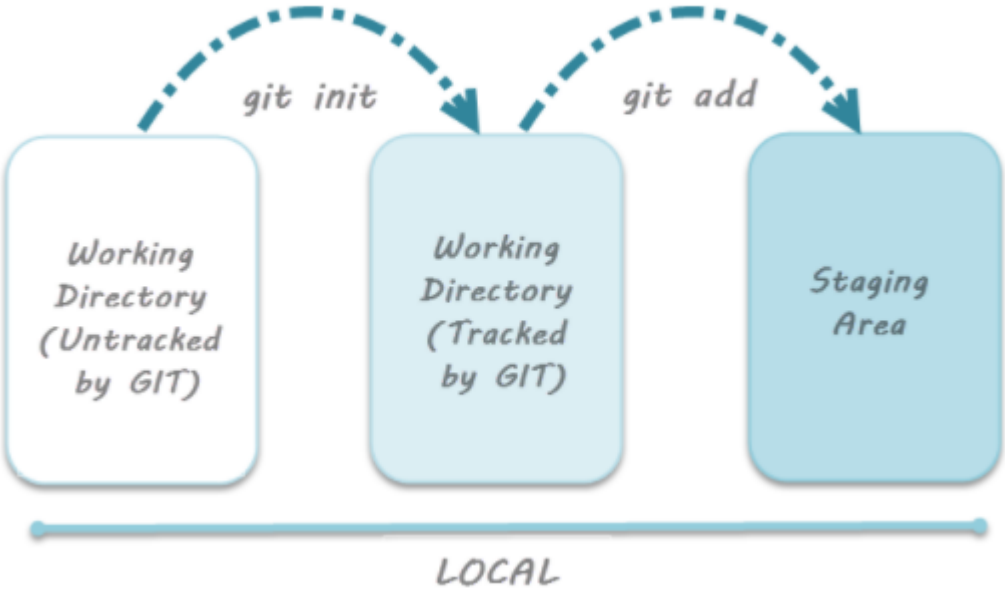
Now, to track the different versions of our files we use the command ***git add***. We can term a staging area as a place where different versions of our files are stored. ***git add*** command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don't want to get tracked, examples include node modules, env files, temporary files, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the ***.git*** folder inside the ***index*** file.

// to specify which file to add to the staging area

git add <filename>

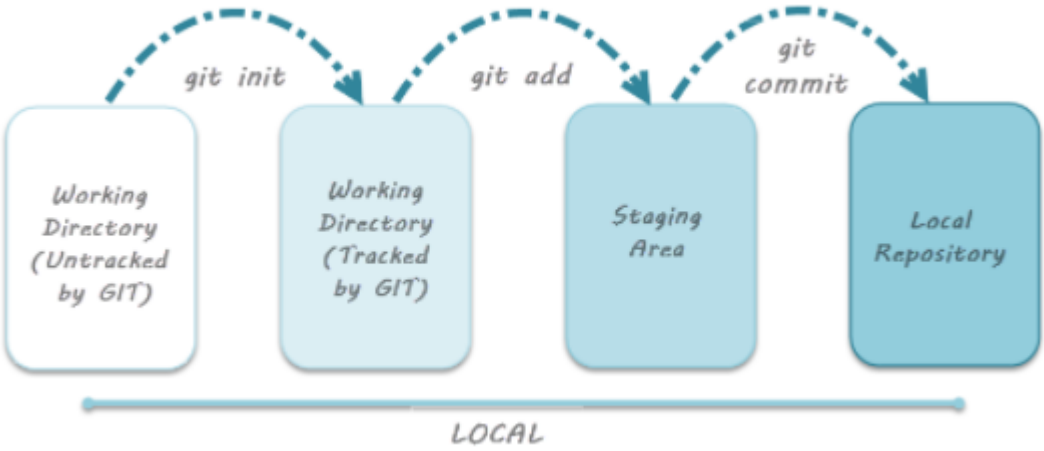
// to add all files of the working directory to the staging area

git add .



Git Directory

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the ***git commit*** command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message.
git commit -m <Commit Message>



COMMAND

```
ls
```

The ls command lists the current directory contents and by default will not show hidden files. If you pass it the -a flag, it will display hidden files. You can navigate into the . git

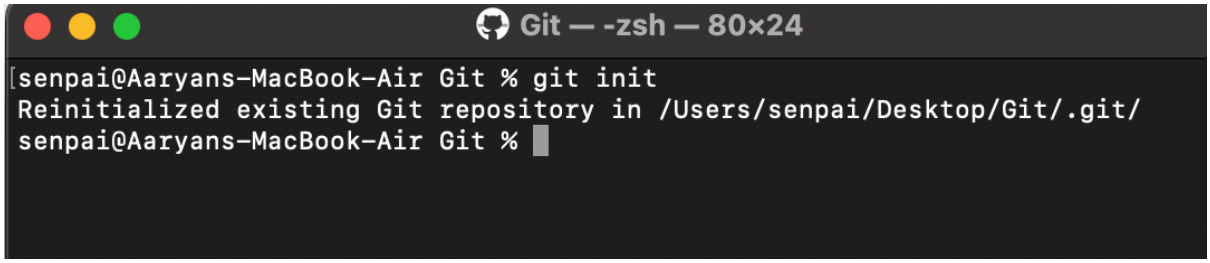
directory like any other normal directory.


```
pwd
```

The Bash command pwd is used to print the 'present working directory'

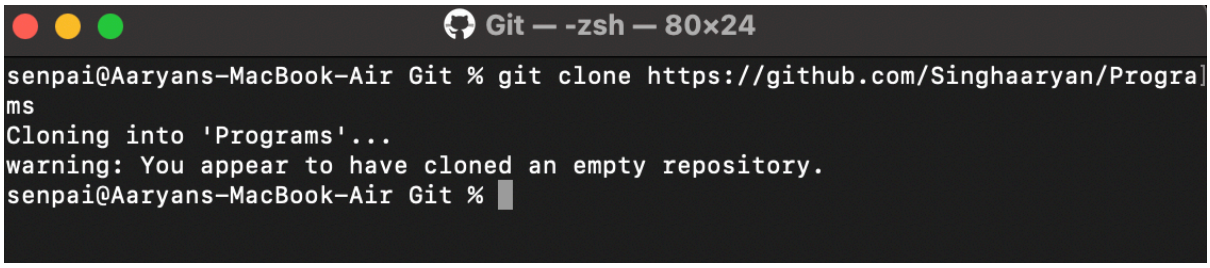

```
git init
```

The `git init` command is used to generate a new, empty Git repository or to reinitialize an existing one. With the help of this command, a .git subdirectory is created, which includes the metadata, like subdirectories for objects and template files, needed for generating a new Git repository.



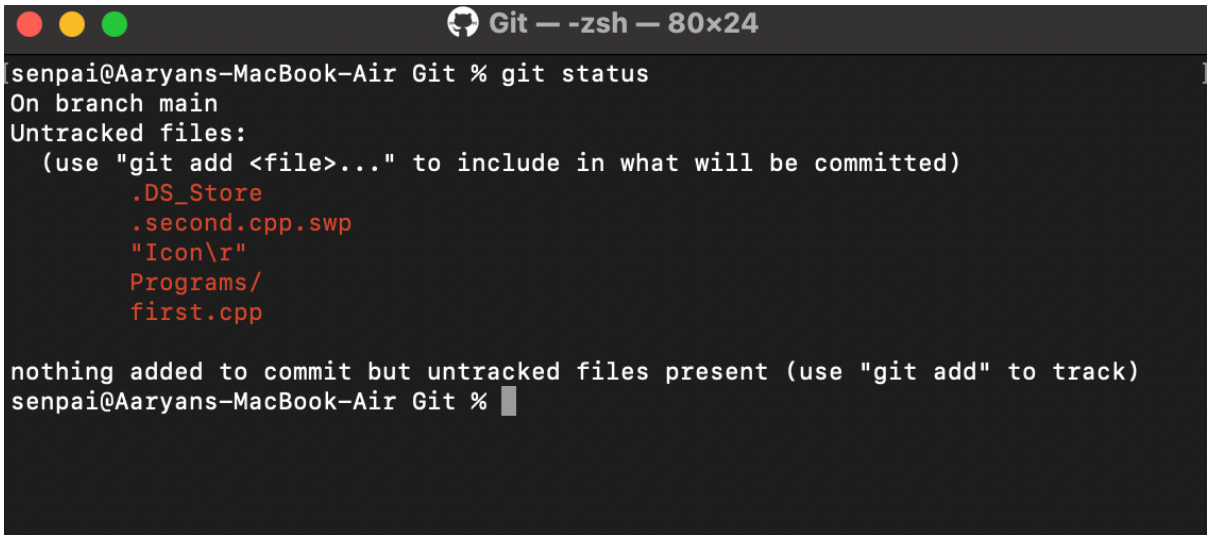
```
git clone <url>
```

The git clone command is used to target an existing repository and create a clone, or copy of the target repository.



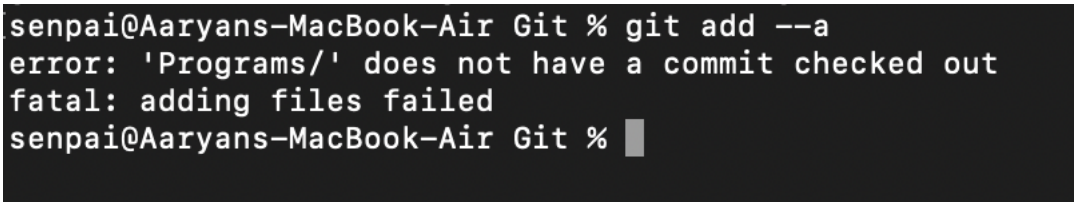

```
git status
```

The git status command displays the state of the working directory and the staging area




```
git add --a
```

The git add --a command is used to add file contents to the Index (Staging Area). This command updates the current content of the working tree to the staging area. It also prepares the staged content for the next commit.



```
git commit -m "message"
```

The git commit -m command captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as "safe" versions of a project

```
senpai@Aaryans-MacBook-Air Git % git commit -m "message"
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .DS_Store
        .second.cpp.swp
        "Icon\r"
        Programs/
        first.cpp

nothing added to commit but untracked files present (use "git add" to track)
senpai@Aaryans-MacBook-Air Git % git add first.cpp
senpai@Aaryans-MacBook-Air Git % git commit -m "message"
[main 1f81e6d] message
 Committer: Aaryan Singh <senpai@Aaryans-MacBook-Air.local>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit
```

```
git branch
```

The git branch command is used to List all of the branches in your repository.

```
senpai@Aaryans-MacBook-Air Git % git branch
* main
senpai@Aaryans-MacBook-Air Git %
```

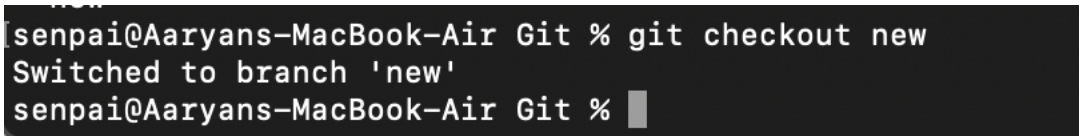
```
git branch <branch>
```

The git branch <branch> command is used to Create a new branch called ＜branch＞.

```
senpai@Aaryans-MacBook-Air Git % git branch new
senpai@Aaryans-MacBook-Air Git % git branch
* main
  new
```

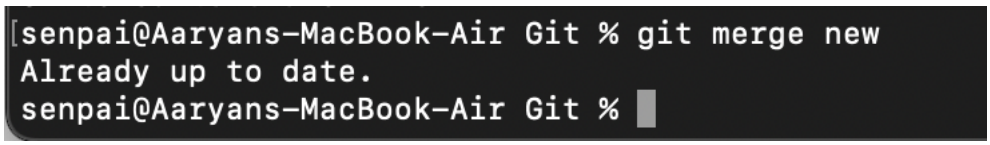
```
git checkout <branch name>
```

The git checkout command is used to switch the currently active branch.




```
git merge <branch name>
```

The git merge command is used to merge the branches.




```
git log
```

The git log command shows a list of all the commits made to a repository.