

# GIT

(Distributed version control system)

Tables Contains.

- Version Control.
- What is GIT?
- GIT Workflow.
- GIT Installation.
- GIT File lifecycle.
- Starting or initialization of GIT
- Working on Local system.
- GIT Basic Command.
- GIT Branching.
- GIT Tags.
- Delete the Branch.
- Delete the directory.

Presented By :

***Kathir &***

***Satish Singh***

# GIT

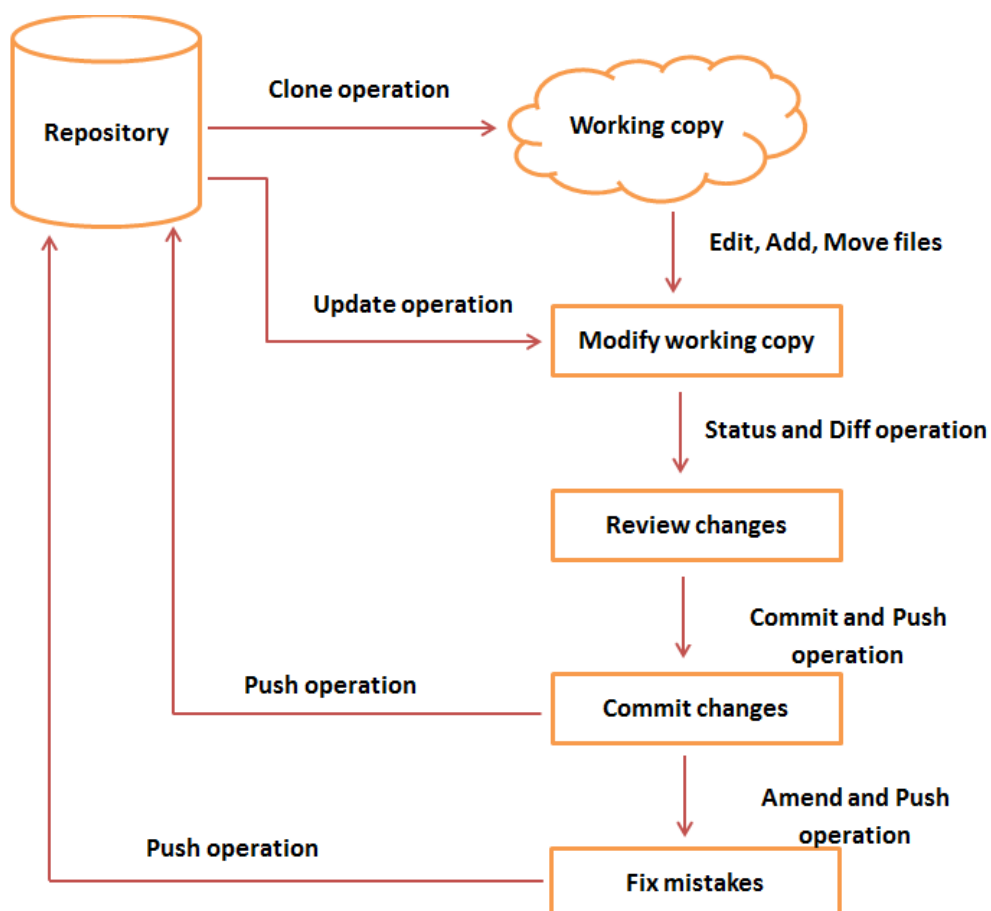
## What is Version Control?

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.

### 1.What is GIT?

- *Git* is currently the most popular implementation of a distributed version control system.
1. Git originates from the Linux kernel development and was founded in 2005 by Linus Torvalds

### 1.1 GIT Workflow



## 1.2 Installation of GIT in LINUX

Use the `apt-get install git` command

## 1.3 Running our first Git command

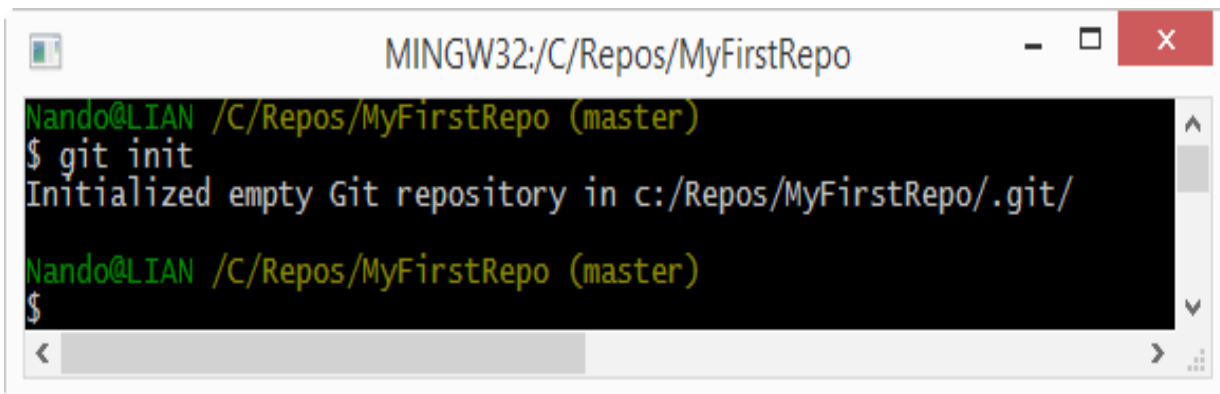
Open a prompt and simply type `git` (or the equivalent, `git --help` )

## 1.4 Setting up a new repository

```
$mkdir git_prac
$cd git_prac
$ls -al
$git init
```

The first step is to set up a new repository (or repo, for short).

A repo is a container for your entire project; every file or subfolder within it belongs to that repository, in a consistent manner.

A screenshot of a Windows command prompt window titled "MINGW32:/C/Repos/MyFirstRepo". The prompt shows the user "Nando@LIAN" in the directory "/C/Repos/MyFirstRepo (master)". The user enters the command "\$ git init", and the output is "Initialized empty Git repository in c:/Repos/MyFirstRepo/.git/". The prompt then returns to "\$".

```
MINGW32:/C/Repos/MyFirstRepo
Nando@LIAN /C/Repos/MyFirstRepo (master)
$ git init
Initialized empty Git repository in c:/Repos/MyFirstRepo/.git/
Nando@LIAN /C/Repos/MyFirstRepo (master)
$
```

To check the `git init` repository type the cmd

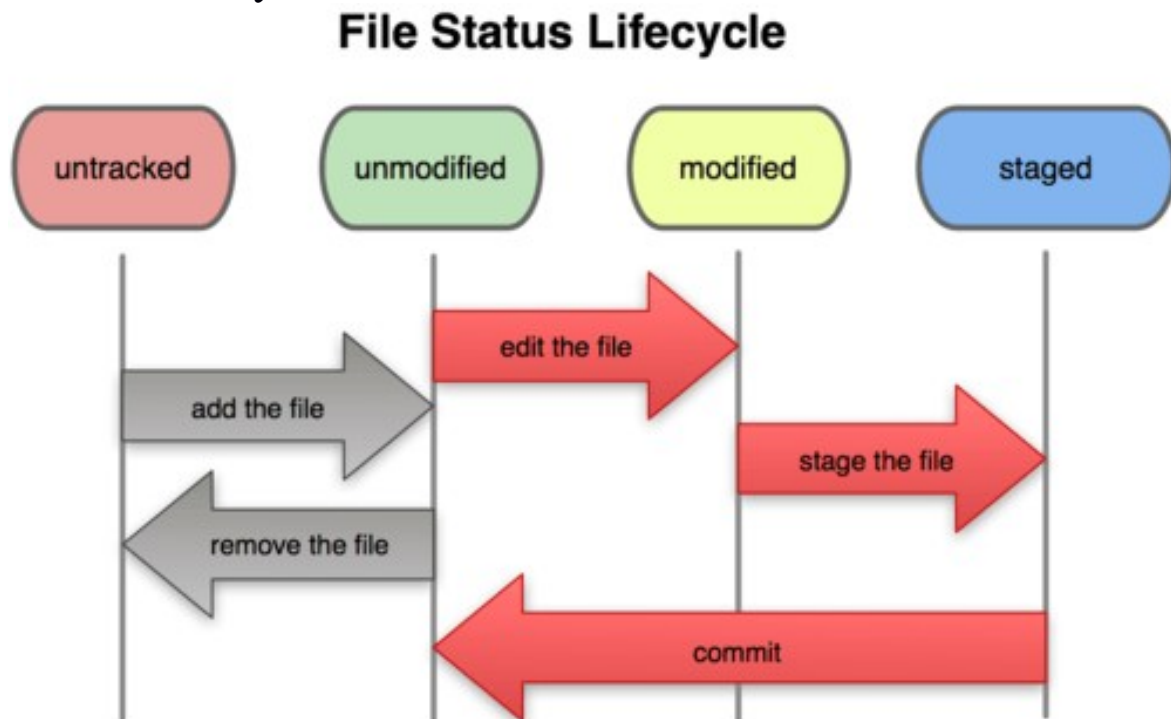
- `ls -al`

Git created a `.git` subfolder. The subfolder (normally hidden in Windows) contains some files and folders.

## 1.5 Config user name and user password

```
git config --global user.name "Your Name"
git config --global user.email you@example.com
```

## 1.6 GIT FileLifecycle



## 1.7 Adding a file

We have to tell Git to put this file in your repo, explicitly

I want MyFile.c under the control of Git, so let's add it

Create **main\_prog.c** file using vi editor. It contains **#include<stdio.h> only**

Type the cmd

**\$git status**

It will show **main\_prog.c** in **untracked files**. (Red marked)

**\$git add main\_prog.c**

A terminal window titled 'MINGW32:/C/Repos/MyFirstRepo' with a yellow border. The prompt is 'Nando@LIAN /C/Repos/MyFirstRepo (master)'. The command '\$ git add MyFile.txt' has been entered and executed. The prompt '\$' is shown again on the next line.

```
MINGW32:/C/Repos/MyFirstRepo
Nando@LIAN /C/Repos/MyFirstRepo (master)
$ git add MyFile.txt
Nando@LIAN /C/Repos/MyFirstRepo (master)
$
```

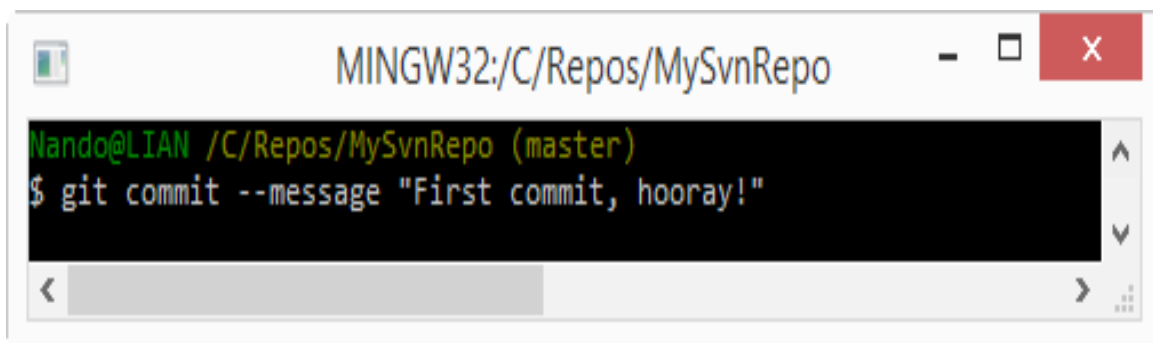
Now Type the cmd

**\$git status**

It will show **main\_prog.c** in **tracked files**.(Green Marked)

## 1.8 Commit the added file

Git knows about **main\_prog.c** , but we have to perform another step ,We have to commit it using the appropriate **git commit** command. This time, we will add some flavor to our command ,using the **--message** (or **-m** ) subcommand, as shown here

A terminal window titled 'MINGW32:/C/Repos/MySvnRepo' with a light gray border. The prompt is 'Nando@LIAN /C/Repos/MySvnRepo (master)'. The command '\$ git commit --message "First commit, hooray!"' has been entered and executed.

```
MINGW32:/C/Repos/MySvnRepo
Nando@LIAN /C/Repos/MySvnRepo (master)
$ git commit --message "First commit, hooray!"
```

Now Type the command

**\$git commit -m “ header file added”**

## 1.9 To check commit is added or not

- Type the command

**\$git log**

**commit b336e9645f0d8af9d3f2f0ba95a5bdcb6d1b4d78**

**Author: Training Sessions <sessitra@vta025l.votarytech.com>**

**Date: Wed Jan 18 09:43:59 2017 +0530**

**header file added**

Type the command

**\$git branch**

**\* master**

Master branch is created once the commit is added

## **1.10 Modify a committed file**

Now, we can try to make some modifications to the file

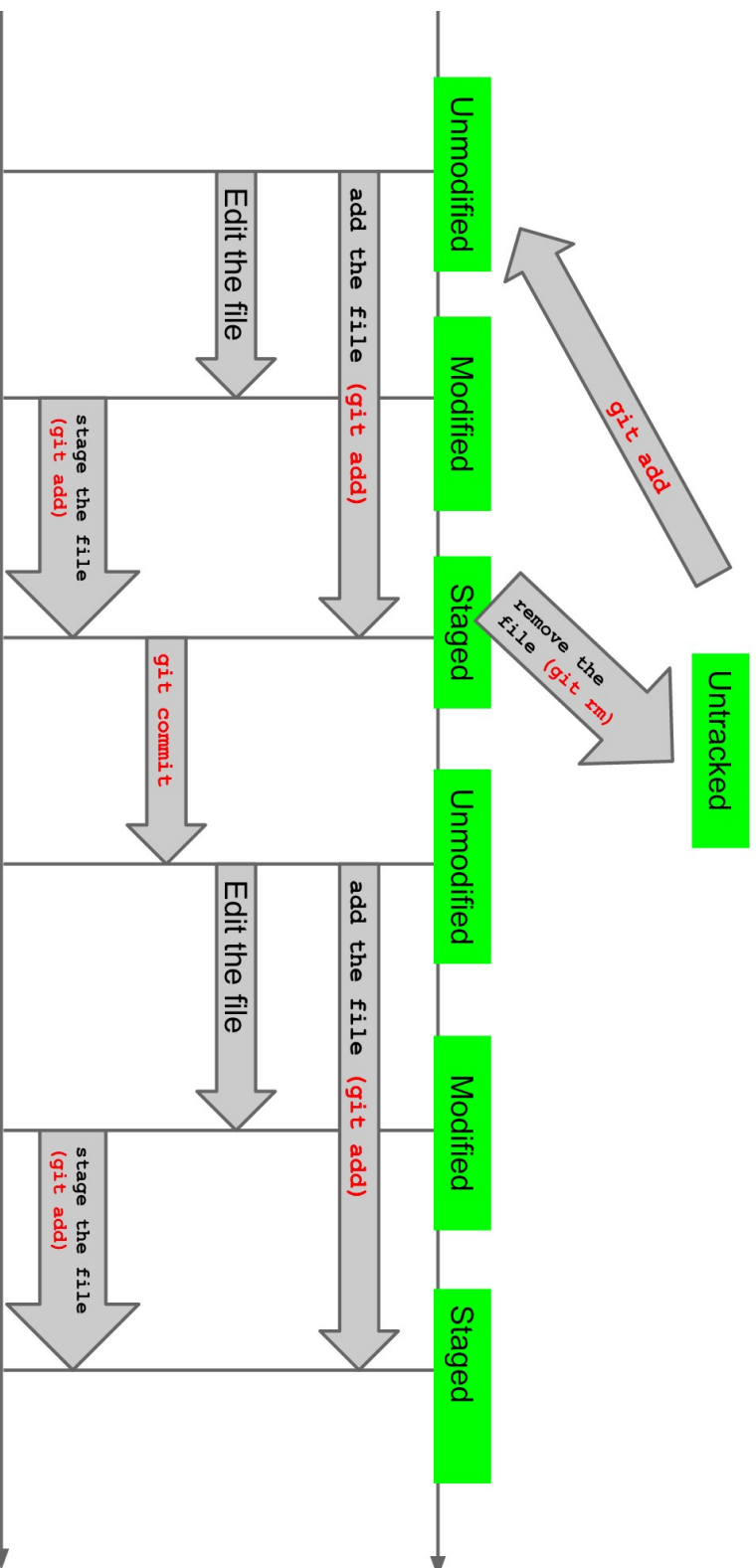
Open **main\_prog.c**

Edit **int main(void)**

```
{  
}
```

Now Type the cmds:

- **\$git status**  
It will show **main\_prog.c** in **untracked files**. (Red Marked)
- **\$git add main\_prog.c**
- **\$git status**  
It will show **main\_prog.c** in **tracked files**. (Green Marked)
- **\$git commit -m "main function added"**
- **\$git log**  
It will display all **commit ids** created
- **\$git diff previous old commit id new commit id**



*The lifecycle of the status of your files*

## 2.0 To Create a Branch

- **\$git branch branch\_name**  
git branch newbranch

## 2.1 To view the branch created or not

- **\$git branch**  
\*master  
newbranch

## 2.2 To move to new branch

**\$git checkout newbranch**  
Switched to branch 'newbranch'

## 2.3 Open main\_prog.c

```
vi main_prog.c
#include<stdio.h>
int main()
{
    int val1 , val2 ;
}
```

Now Type the cmds:

- **\$git status**  
It will show main\_prog.c in untracked files.(Red Marked)
- **\$git add main\_prog.c**
- **\$git status**  
It will show main\_prog.c in tracked files.(Green Marked)
- **\$git commit -m “variables defined in newbranch”**



- **\$git log**  
It will display all **commit ids created**
- **\$git diff previous old commit id new commit id**

## 2.4 To Create a SubBranch

- **\$git branch branch\_name**  
git branch newbranch

## 2.5 To view the branch created or not

- **\$git branch**  
master  
\*newbranch  
sub\_branch

## 2.6 To move to newly created branch

**\$git checkout sub\_branch**  
Switched to branch 'sub\_branch'

## 2.7 Open main\_prog.c

```
vi main_prog.c
#include<stdio.h>
int main()
{
    int val1 , val2 ;
    val1 = 5, val2 = 10;
    printf("val1 is %d val2 is %d\n,val1,val2);
}
```

## 2.8 Now Type the cmds:

- **\$git status**  
It will show main\_prog.c in untracked files.(Red Marked)
- **\$git add main\_prog.c**

- **\$git status**  
It will show **main\_prog.c** in **tracked files**. (Green Tracked)
- **\$git commit -m "variables initialized and printed in sub\_branch"**
- **\$git log**  
It will display all **commit ids** created
- **\$git diff previous old commit id new commit id**

## 2.9 To create patch file

**\$git format-patch previous\_commit\_id**

**0001-variables-initialized-and-printed-in-sub\_branch.patch**

It will **create a patch file**

**Patch file contains the modified contents**

## 3.0 To merge the created patch file to the previous branch

- **\$git checkout newbranch**
- **\$git am patch\_file name**

## 3.1 To merge the branch

- **\$git checkout master**
- **\$git merge new\_branch**

## 3.2 To Make Tags

- **\$git tag v1.1**  
create the **v1.1** for the patch file
- **\$git show v1.1**  
display the modified contents
- **\$git tag**  
display the versions

## 3.3 To Delete branch

- `$git branch -D branch name`

### 3.4 To delete git repository

Type the command

- `ls -al`
- `rm -rf .git -->to remove .git dir`
- `rm git_prac -->to remove git installed dir`