Today I'll Cover :

1. Introduction to GIT
2. Three stage architecture
3. Working with Git
4. Git Log and Git Diff
5. Branching and merging
6. Merge conflicts
7. Git ignore, Git Tag, Git Aliases and Git Stash
8. Working with Remote Repositories

Mohammad Altaf Hussain

Introduction

1. **Linus Torvalds** invented Git 16 years ago in order to continue development of the Linux kernel.
2. Git is a **Version Control System**

```
Client gave requirement to me to develop a project
client project
 |--100 files developed
 |- client suggested some changes
 |- I changed some files source code to meet client requirement
 |- I gave the demo and client suggested some more changes
 |- I changed some files source code to meet client requirement
 |- I gave demo 3rd time
 |- Client asked for first version only
 |- My Face with big ????
```

so what should we do?
we should not overwrite our code and should we maintain each and every version?

ummmmmm...
very complex!
right??

that’s why version control system comes into picture…

Mohammad Altaf Hussain

git allow us

maintain multiple version.

mutliple users to work together

to keep track of files

- who did the change
- when did the change
- what changes he/she did

Wow Git is really Awesome!!
we should start using git.

but wait! we don't have git.

no problem, lets install Git

Mohammad Altaf Hussain

for linux user:

```
sudo apt install git -y
```

for windows and Mac user:

go and find on google.

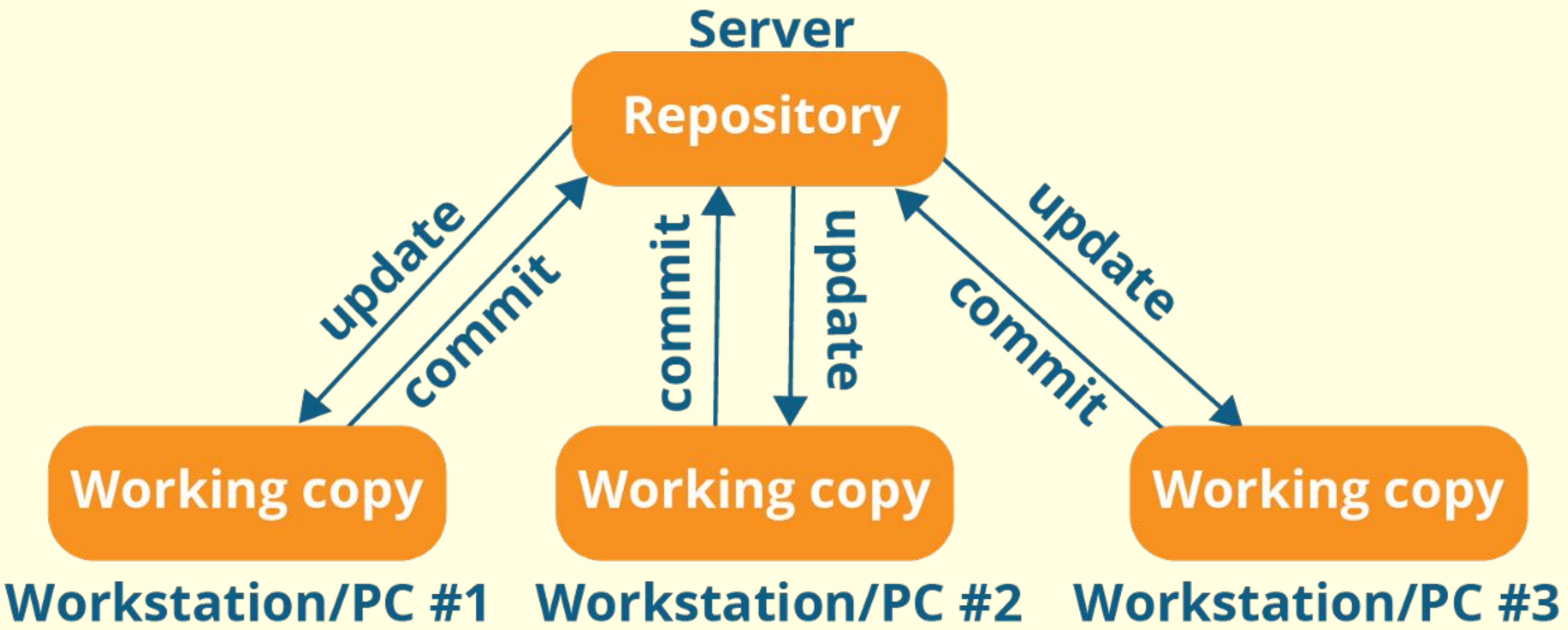
Once installed, check version of git

```
git --version
```

Mohammad Altaf Hussain

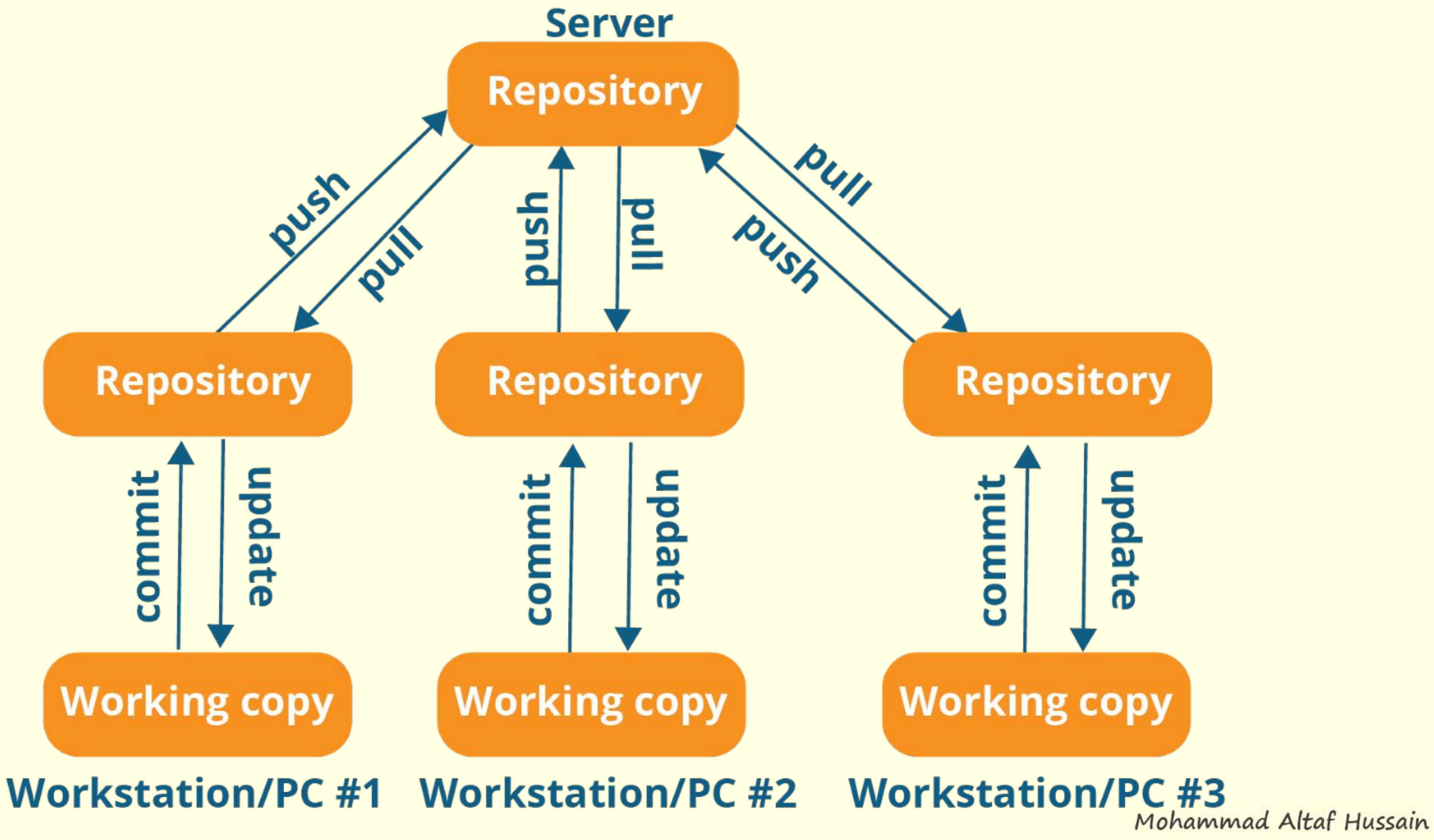
before we start using git, we should have a basic idea of how git or version control system internally work.

Centralized version control system



Mohammad Altaf Hussain

Distributed version control system

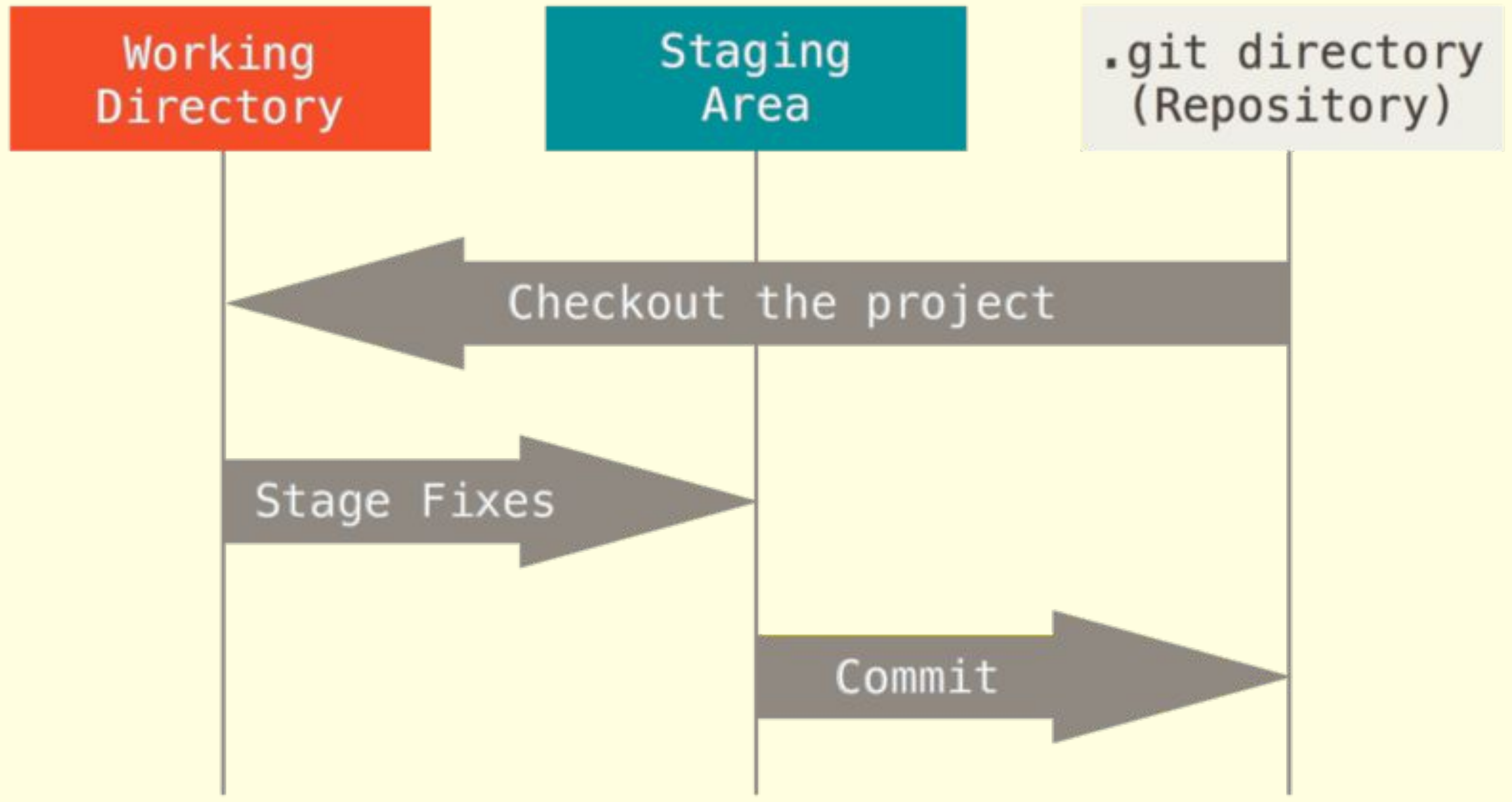


Git is Distributed

Version Control System

Mohammad Altaf Hussain

Git - 3 Stage Architecture



Mohammad Altaf Hussain

working directory:

- The place where we can create new files or modify existing files

Staging Area

- it is the place where we send those files that we want to commit.
- This staging area is helpful to cross-check our changes before commit.

you can relate with

- you have added some product in your cart before you buy.

Repository

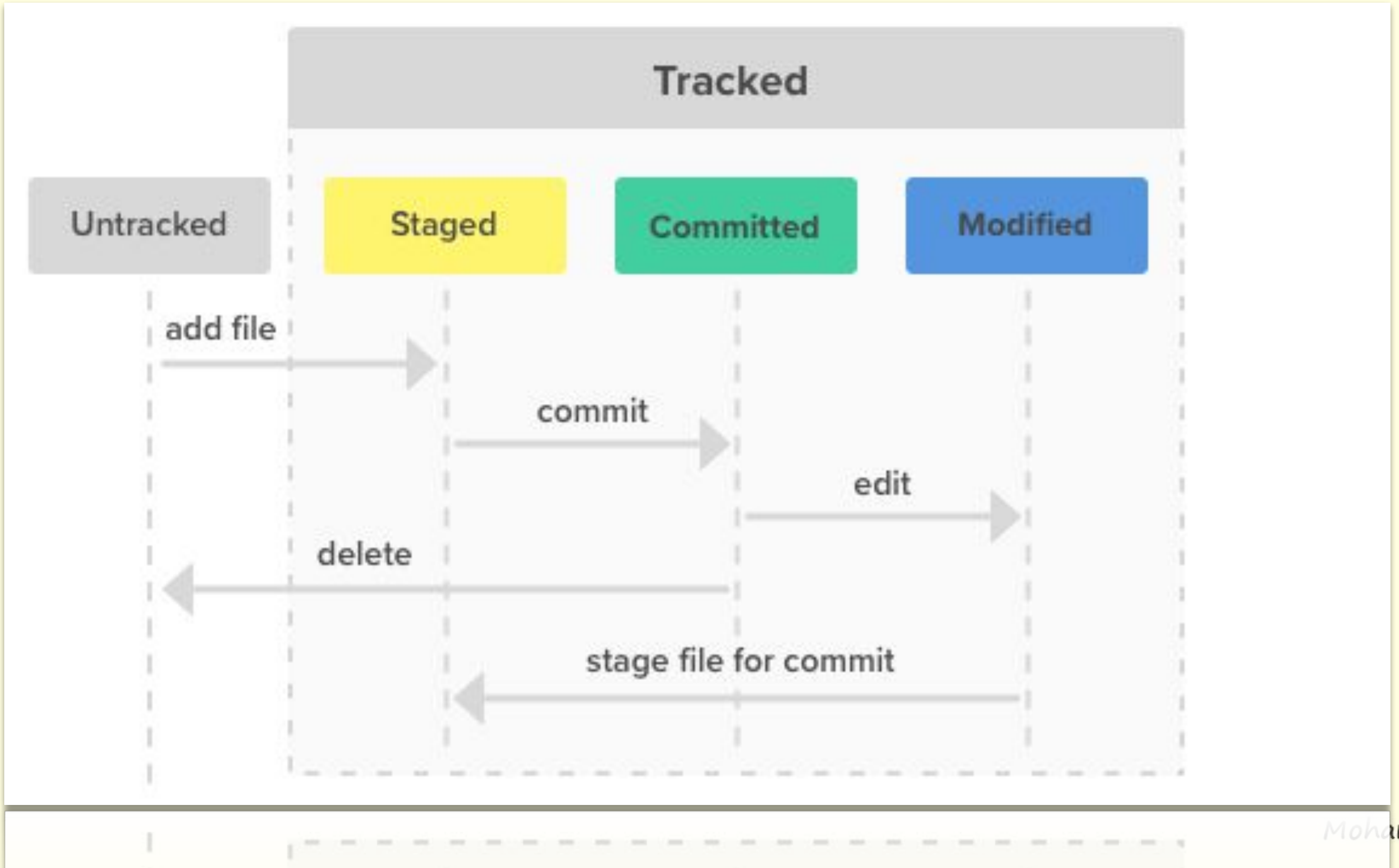
- A Git repository is a virtual storage of your project.
- It allows you to save versions of your code, which you can access when needed.
- It tracks and saves the history of all changes made to the files

2 Types of Repository

1. Local Repository
2. Remote Repository

Mohammad Altaf Hussain

Life Cycle of file in Git



Working with git

git init

- to convert an existing, unversioned project to a Git repository or initialize a new, empty repository.
- it will create a hidden directory **.git**
- we can use git command within that directory.
- if we delete **.git** folder the folder will no longer be under versioning control.

git configurations

we have to configure username and email id, so that git can use this information in the commit records.

git config --<local/global> user.email <"email">

git config --<local/global> user.name <"username">

if we use **--local** then the configuration will be applied for current repository only, but **--global** will apply the configurations for all the repository.

git config user.<name/email>

to check the configurations

git status

To check the status of files like which file is modified, untracked, tracked files etc..

-s flag is useful for shorter info.

git add <file>

It will all the file into staging area and if the file was untracked then git will also start tracking that file.

- to add all file into staging area

  git add .

Mohammad Altaf Hussain

git restore <file>

To discard changes in **working directory.** (file should not be in staging area)

git restore --staged <file>

it will remove the file staging area but keep the file and changes in the working directory

Mohammad Altaf Hussain

git commit -m <message>

It is used to create a snapshot of the staged changes along a timeline of a Git projects history.

-a flag is useful when you have to do add and commit in one time.

git checkout <commit id/branch name>

It helps you to navigate between the versions/branches of the project/repository

Mohammad Altaf Hussain

git log

It shows a list of all the commits made to a repository.

it also show hash,author info, and message of each commit.

--oneline flag is useful for one one liner logs

git log <filename> log related to particular file.

--grep=”search string” search in log message

within specific time range

--until <date yyyy-mm-dd>

--before <date>

Mohammad Altaf Hussain

git diff

- It is used in git to track the difference between the changes made on file.
- By default it shows difference between last commit and working directory of all files. We can also specify file name.
- **git diff HEAD** difference between working directory and last commit
- **git diff --staged HEAD** difference between staging area and last commit
- **git diff <commit1> <commit2>** difference between 2 commit id

Mohammad Altaf Hussain

branching in git

While working on real time projects code base, branching is one of mandatory and unavoidable concept.

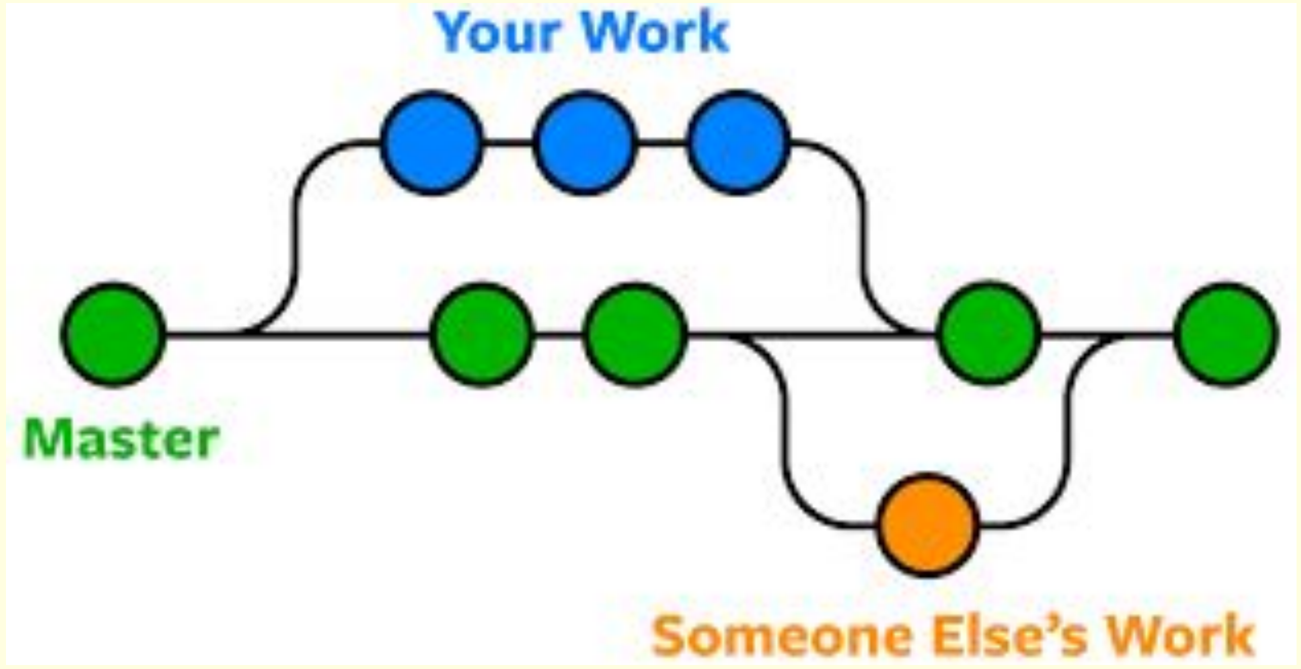
Till now whatever files created and whatever commits we did, all these happened in **master branch**.

master branch is the **default branch**/ main branch in git.

Generally main source code will be placed in master branch.

Need of creating branch

Assume we required to work on new requirements independently, then instead of working in the master branch, we can create a separate branch and we can work in that branch, related to that new requirement without affecting main branch.



Mohammad Altaf Hussain

To View list of all available branches

```
git branch
```

Create new branch

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

delete branch

```
git branch -d <branch name>
```

-D flag is useful in order to delete unmerged branch forcefully

Mohammad Altaf Hussain

to create and navigate a branch

```
git checkout -b <branch name>

git switch -c <branch name>
```

Important Conclusions:

1. All branches are isolated to each other.
2. In GIT branching, logical duplication of files will be happend.

Advantages of Branching

1. We can enable Parallel development.
2. We can work on multiple flows in isolated way.
3. We can organize source code in clean way.
4. Implementing new features will become easy
5. Bug fixing will become easy.
6. Testing new ideas or new technologies will become easy.

Merging of Branch

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

Merge Conflict

It is an event that takes place when Git is unable to automatically resolve differences in code between two commits.

Rebase

1. **Rebase keeps history linear.**
2. **Clear workflow (Linear) will there. Hence easy to understand for the developers.**
3. **No extra commit like merge commit.**

- It is 2 step process

```
git checkout <branch name>
git rebase master
git checkout master
git merge <branch name>
```

Mohammad Altaf Hussain

.gitignore

It is a text file in the root folder of project that tells Git which files or folders (or patterns) to ignore in a project.

for eg:

```
*.txt (ignore text file)
abc.txt  (ignore abc.txt only)
logs/   (ignore logs directory)
.*  (ignore hidden file)
```

Mohammad Altaf Hussain

git tag

- Tag is nothing but a label or mark to a particular commit in our repository.
- Sometimes, we have to mark significant events or milestones with some label in our repository commits. We can do this labeling by using **git tag** command.

syntax: **git tag <tag name>**

- list all tag : **git tag -l**
- delete tag : **git tag -d <tag name>**
- checkout using tag : **git checkout <tag>**

Mohammad Altaf Hussain

git aliases

- Alias means nickname or short name or other alternative name.
- If any git command is lengthy and repeatedly required, then for that command we can give our own convenient alias name and we can use that alias name every time.

<u>Create Alias:</u>

```
git config --global alias.<aliasname> "<original command without git>"
```

<u>Delete Alias:</u>

```
git config --global --unset alias.<alias name>
```

<u>List Alias:</u>

```
git config --get-regexp alias
```

Mohammad Altaf Hussain

git stash

It takes our uncommitted changes (both staged and unstaged), saves in some temporary location.

- To list out stash

  git stash list

- To view a stash

  git show stash@{0}

- To load

  git stash apply stash@{0}

- To clean a stash

  git stash clear

Working with Remote Repository

- To list out remote repository

  git remote -v

- To add remote repository

  git remote add <any name> <repository url>

- To inspect remote repository

  git remote show <name>

- To upload on remote repository

  git push <remote name> <branch name>

- To download from remote repository

  git pull <name> <branch name>

git clone

- To clone repository

  git clone <url>

Mohammad Altaf Hussain