

Git & Github

Version control



codi

Outline

— — —

- Problems
- What is Git
- Benefits of Git
- Installation of Git
- Git Practice
- Showcases

Problems

— — —



- Who did the changes
- What have been changed
- View the changes

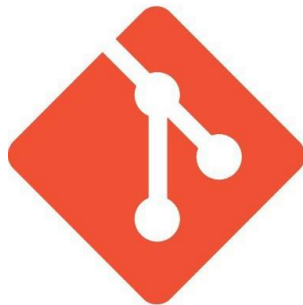


- Hard to collaborate
- Hard to share the code
- Inform all the team if there is any changes

What is Git?

Git is a open-source version control system that manages and tracks the changes made to the files. Also you can revert to specific versions you need.

It is a command-line program that is designed in the Unix tradition



git


Benefits of Git?

— — —

- Tracking and managing changes
- See the history of the changes
- Revert to specific versions
- Makes collaboration easier
- Able to work without blocking other's work
- Able to see and fix the conflicts
- Open Source
- Backup

Installation & Setup

— — —



```
//check if git is already installed
```

```
$ which git
```

```
//install it on linux
```

```
$ sudo apt install git-all
```


```
//configuration to identify your email & name
```

```
$ git config --global user.name your_name
```

```
$ git config --global user.email your_email@gmail.com
```

Initializing the repo

— — —



```
//make a directory
```

```
$ mkdir -p repos/website
```

```
//go to website directory
```

```
$ cd website
```


```
//create a new repository
```

```
//it will create a ./git hidden repo in it
```

```
$ git init
```

Making our first commit

— — —



```
//create a main page index.html
$ touch index.html

//check the status (modified/untracked files)
$ git status

//to tell git to add all untracked files
$ git add .

//after putting changes in the staging area
//we can make them part of local repo by committing
$ git commit -m "your commit"
```


Making our first commit

-- --



```
// see record/history of commits  
$ git log
```

Local vs Online

— — —

The **local repository** is a Git repository that is stored on **your computer**.

The **remote repository** is a Git repository that is stored on some **remote computer**. When it comes to sharing data with your teammates, a remote repo comes into play.

Gitlab & Github = both web-based repositories that help with code management and sharing local file changes with a remote repository using Git.

"repository or repo" = online folder/files storage. It is a central place to keep resources that users can pull from when necessary that holds everything your project contains e.g. : files, commits, *branches...

The main Git status sequence for changing a file

— — —



Essential Git Commands related to Online repo

CLONE : to get the files on your computer by downloading the online repo to your local machine (no need for git init).

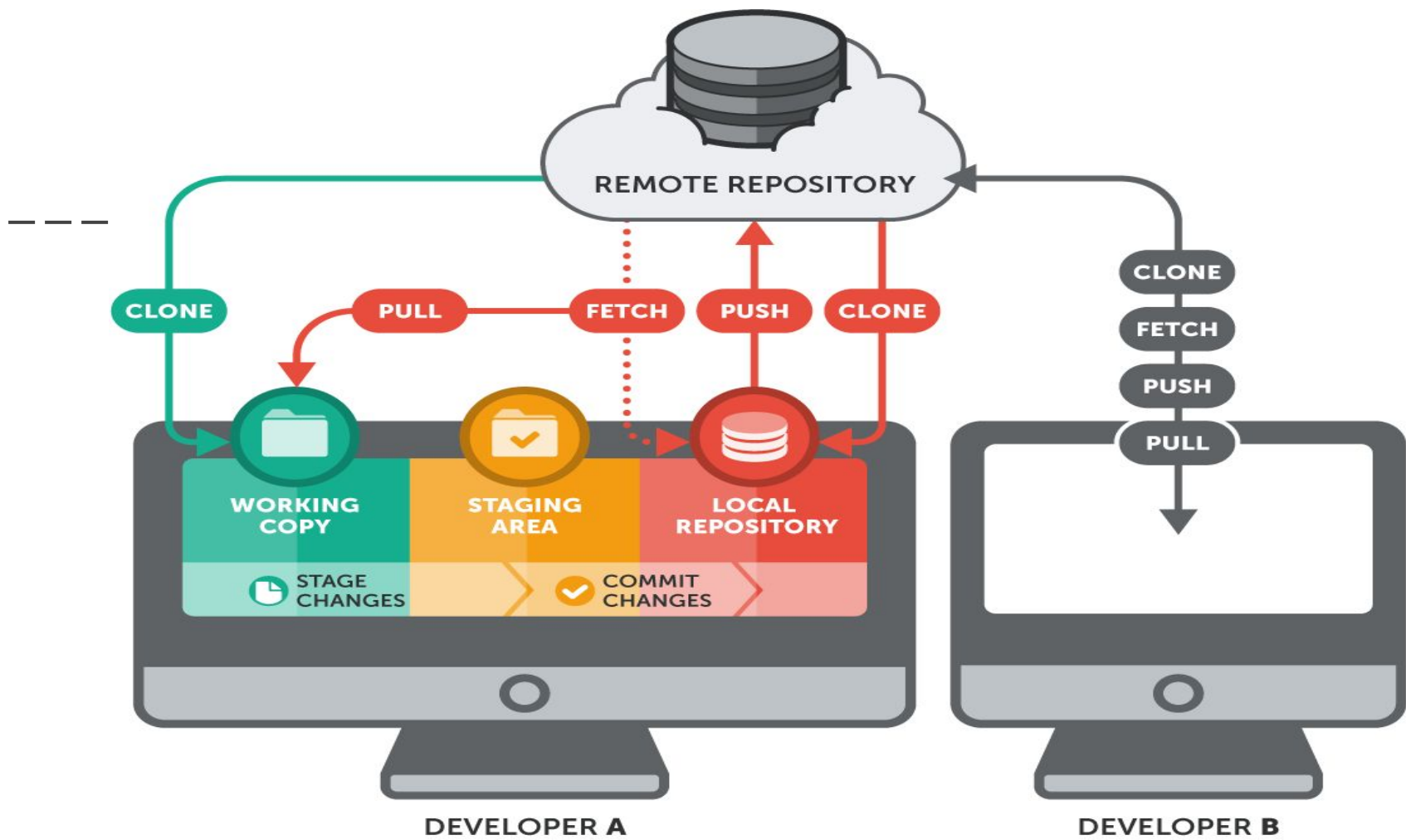
FORK : With any online version control manager you have the ability to 'copy' a repository to your own personal space we call this a 'fork'. So you can work on it on your own or with your team, without altering the original repository files.

Branch: A branch represents an independent line of development. Branches serve as an abstraction for the edit/stage/commit process. You can think of them as a way to request a brand new working directory, staging area, and project history.

PUSH : The git push command is used to upload local committed repository content to a remote repository, so the remote (online) repo will be updated with the changes you have made.

PULL : to fetch and download content from a remote repository and immediately update the local repository to match that content. You can't pull if there are active changes that have not been staged.

MERGE : The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.



Viewing the diff

Diffing is a function that takes two input data sets and outputs the changes between them. `git diff` is a multi-use Git command that when executed runs a diff function on Git data sources



```
//to view changes represented by the potential  
//commit before making it  
$ git diff
```

Show cases

1. I need to track my changes locally
2. I want to put my project on a remote repo in case my laptop got any problem
3. I need to work with a friend on the same project
4. I would love to contribute to an open-source project
5. I need to move a project from one repo to another
6. Each teammate will work on different features
7. Merging two branches

1. track my changes locally

Git commands needed


— — —



```
$ cd your_project  
$ git init  
$ git add .  
$ git commit -m "my first commit"
```

2. put my project on a remote repo

Git commands needed



```
$ git config --global user.name your_name
$ git config --global user.email your_email@gmail.com
$ cd your_project
$ git init
$ git add .
$ git commit -m "my first commit"
$ git remote add origin git_repo_link
$ git push --set-upstream origin master
```



Origin and Master (or main) are two different terminologies used when working and managing the git projects. **Origin is the name used for the remote repository. Master is the name of the branch.**

3. work with a friend on the same project

Git commands needed

— — —



```
$ git pull
```

```
$ git add .
```

```
$ git commit -m "my commit"
```

```
$ git push
```

4. contribute to an open-source project

Git commands needed

— — —

Fork the original one

5. move a project from one repo to another

Git commands needed



```
$ git clone repo_link  
$ rm -rf .git  
$ git init  
$ git commit -m "my first commit after cloning"  
$ git push --set-upstream origin master
```

Git commands needed - Another solution

Clone from repo 1

Delete ./git folder

git init

git add.


git commit -m "first-commit"

git push origin master -> my own repo

6. Each teammate will be working on a different feature or branch

Git commands needed

— — —



```
$ git branch feature1
$ git checkout feature1
// or git checkout -b feature1
$ git add .
$ git commit -m "done with feature 1"
$ git push
```

7. Merging two branches together

Git commands needed

--



```
$ git checkout branch1
```

```
$ git merge branch2
```