

Git Complete Course

...

By **Kanzal Qalandri**

Install Git

sudo apt install git

For Ubuntu and Debian

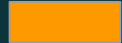
sudo yum install git

For RedHat

sudo dnf install git

For Fedora

Command:



Text:



Setup Starship.rs (Optional)

1: `curl -sS https://starship.rs/install.sh | sh`

2: `vi ~/.bashrc`

3: `eval "$(starship init bash)"` # Add this line at the end of .bashrc file

4: Copy File content from [here](#).

5: `vi ~/.config/starship.toml`

6: Paste the file that you copied from github here and save.

Git Init

- 1: First Create a folder for practicing git (**mkdir Your-Folder**)
- 2: And cd into the newly created folder.
- 3: **git init** # Initializes a new Git repository in the current directory

The **git init** command is the first step in creating a new Git repository. It creates a **.git** directory in the current directory, which contains all of the necessary files for Git to track changes to your code.

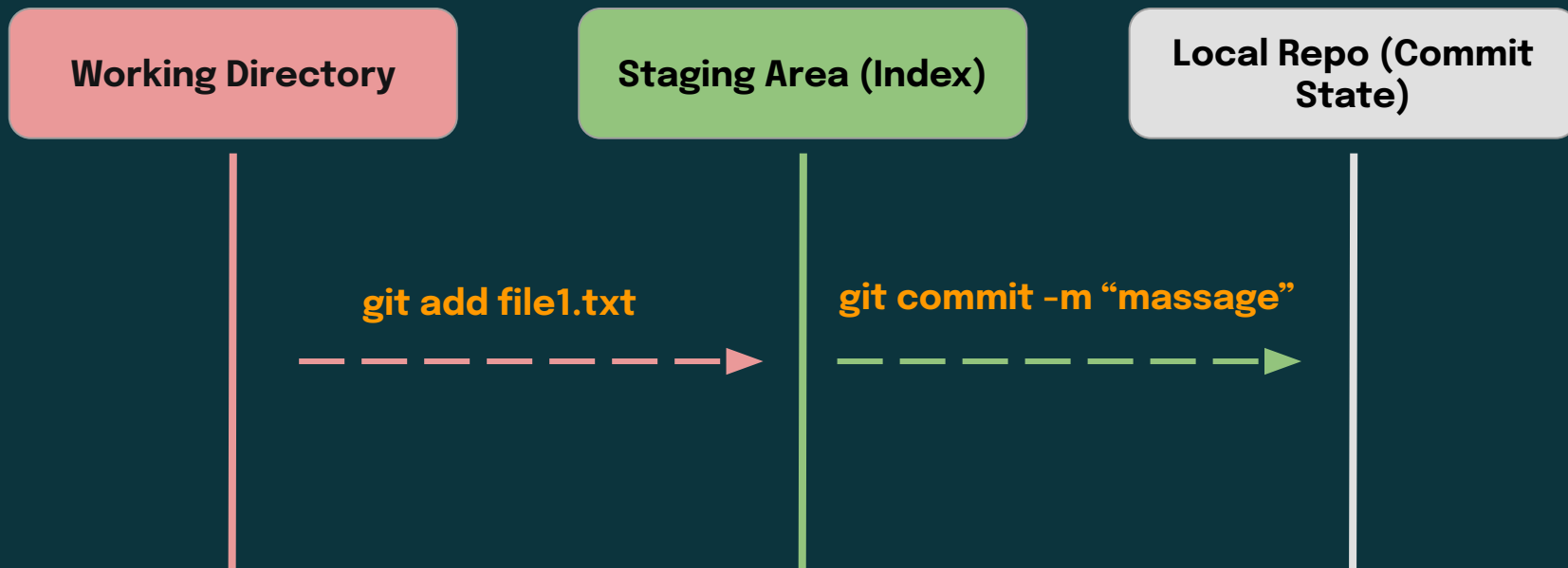
Configure Git

- 1: `git config --global user.name "Your Name"`
- 2: `git config --global user.email "your-email@gmail.com"`
- 3: `git config --global -e` # Check Configuration

First Commit

- 1: Create a file `file1.txt` with some content.
- 2: `git add file1.txt` # Add file to the staging area
- 3: `git commit -m "a good message"` # Created first commit

Git 3 Tier Architecture Workflow



git add / Staging Area

The **git add** command is used to add changes from working directory to the staging area.

The staging area is a temporary holding area for changes that you want to include in your next commit. Once you have added changes to the staging area, you can commit them to the repository with the **git commit** command.

- 1: **git add <filename>** # Add single file to the staging area
- 2: **git add .** # Add all changed files in the current directory to Index
- 3: **git ls-files** # List all files present in Index (Staging Area).

Git Commit

The **git commit** command is used to create a permanent snapshot of the changes that have been added to the staging area. Once you have committed changes, they cannot be undone without explicitly reverting the commit.

- 1: **git commit -m "commit message"** # Create a Commit
- 2: **git log** # List all previous commits
- 3: **git ls-tree <commit-id>** # List all files present in particular commit

The commit message should be a brief description of the changes that you have made.

Git Show

1: **git ls-tree <commit-id>**

Above command will show files present in a particular commit.

2: **git show <file-id>**

Using **git ls-tree <commit-id>** you will get files list with their file id on that particular point of time. So, use **git show <file-id>** to view the content of any file at that point of time.

Git Diff

1: **git diff** # diff b/w working directory and Index.

2: **git diff --staged** # diff b/w Index and last commit.

Setup a Better Diff tool (VSCode):

git config --global diff.tool vscode

git config --global difftool.vscode.cmd 'code --wait --diff \$LOCAL \$REMOTE'

3: **git difftool** # To view diff on VSCode

Remove Files

git rm <file-name>

git rm is a command in Git used to remove files from both the working directory and the staging area, effectively deleting them from the repository while also staging the removal for the next commit.

Renaming Files

git mv <file-name>

git mv is a Git command used to move or rename files and directories within a repository. It automatically stages the changes, combining the actions of renaming the file and staging it for the next commit.

Ignoring Files

In Git, the **.gitignore** file is a powerful tool that allows developers to specify certain files and directories to be ignored by Git. This means these files will not be tracked or staged, making it essential for excluding temporary files, build artifacts, and sensitive information from version control.

git rm <file-name> --cached #Remove file just from Index

Status & Short Status

git status is a Git command that provides a detailed view of the current state of your repository. It shows which files are modified, which files are staged for the next commit, and which files are not being tracked.

git status -s or **git status --short** is a more concise version of git status. It provides a compact summary of the repository's status, displaying only essential information. It uses a two-letter status code to represent the changes:

?? for untracked files

A for files staged and waiting to be committed

M for modified files

D for deleted files