



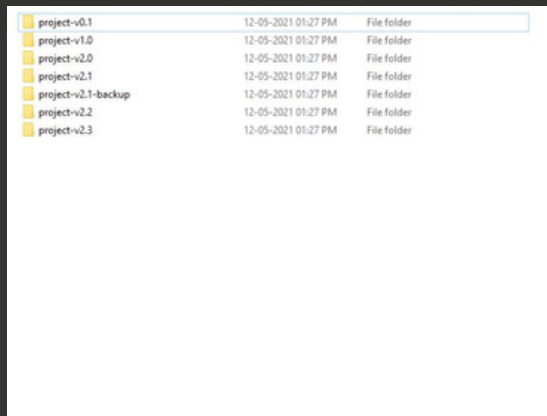
Git

EVERYTHING YOU NEED TO KNOW AS A BEGINNER

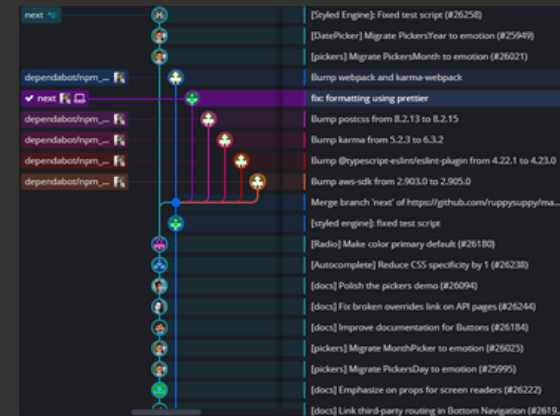
What is Git?

Git helps to keep track of different versions of a **single code base** by **tracking all changes** and making it very easy to **collaborate** of humongous projects with ease

Before



After



Installing Git

If you are on using Window, visit:

<https://git-scm.com/download/win>

If you are using a Linux-based system, run the following command:

```
> sudo apt-get install git
```

To verify if git was properly installed, use:

```
> git --version
```

Initializing a Repo

Before using Git in your project, you need to initialize a Repository.

To initialize one, use the following command:

```
> git init
```

Git creates a hidden directory called `.git`, which stores all of the objects and refs that Git uses and creates as a part of your project's history.



Staging

To **commit**, you need to specify the files whose changes you want to save. This is done by **staging** the changes. It is **NOT** required to stage all files you modified, you can stage only the files whose changes you want to commit

To stage changes, use:

```
> git add <file 01 path> <file 02 path> <...>
```

or,

```
> git add .
```

Committing

Finally, we come to committing changes.

To save the changes you have staged, use:

```
> git commit -m "<message>"
```

The commit command captures and **saves a snapshot** of the project's currently staged changes

Logs

Git log is a **utility tool** to review and read a **history** of everything that happens to a repository.

```
> git log
```

```
$ git log_
commit b06d634e334947fd9d890184e9986ee0988633e1 (HEAD -> master)
Author: Tapajyoti Bose <[REDACTED]>
Date: Thu May 13 09:28:36 2021 +0530

    modified test.txt

commit 8a11c5095f2dcd70b0bc8c66061a1368558a3abf
Author: Tapajyoti Bose <[REDACTED]>
Date: Thu May 13 09:24:45 2021 +0530

    added test.txt
```

Undoing Changes

To err is man

It is quite possible that you might make some mistake while working on a project. Wondering how to fix them?

Git has two commands to undo changes you made

1. **Reset**
2. **Revert**



Reset

Reset enables you to reset recent changes you made. The command is:

```
> git reset --soft HEAD~1
```

Let's break down the command

```
git reset <reset type> HEAD~<number of commits to undo>
```

The types field allows the following:

1. **soft**: uncommit and keep (staged) changes
2. **hard**: uncommit and delete changes

Revert

Every commit is associated with a **hash**.

```
$ git log_
commit b06d634e334947fd9d890184e9986ee0988633e1 (HEAD -> master)
Author: Tapajyoti Bose <[redacted]>
Date: Thu May 13 09:28:36 2021 +0530

    modified test.txt

commit 8a11c5095f2dcd70b0bc8c66061a1368558a3abf
Author: Tapajyoti Bose <[redacted]>
Date: Thu May 13 09:24:45 2021 +0530

    added test.txt
```



You can undo a specific commit using the revert command and its **hash**

```
> git revert 8a11c5095f2dcd70b0bc8c66061a1368558a3abf
```

NOTE: An additional commit is added on reverting modifications

Branch

Git branches are effectively a **pointer** to a **snapshot of your changes**. When you want to make some modifications, like **feature additions**, **bug fixes**, or **documentation**, no matter how big or how small, you spawn a new branch to encapsulate your changes.

The former convention was to call the base branch **master**, but recently the name has been changed to **main**. You can change the name or the base branch as per your requirement though.



Branch (continued)

To create a new branch use:

```
> git checkout -b <new branch name>
```

To switch to an existing branch use:

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

Merge

After working on a branch, you may need to updating another branch with the code from the current branch.

To **merge** changes from another branch, first, move to the branch you want to update and use:

```
> git merge <update source branch name>
```

```
$ git merge test
Updating e1c0ddf..38ea0ae
Fast-forward
 test.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Conflict

After merging branches if in both of the branches, the **same part of the same file** was updated, git doesn't know which change to keep and which to discard.

So git creates a **conflict** message for manual review.

```
$ git merge master
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Conflict (continued)

The conflict message outlines where the conflict occurred as well as the **current** (available in the branch) and **incoming changes** (merging from another branch).

```
<<<<<<< HEAD
Some text!!!!
=====|
Some text!
>>>>>>> master
```

After resolving the conflict, you need to commit to save the **resolved merge**.



Fork

If you are contributing to a repository you don't have **write access** to, you must **Fork** the repository as the first step.

A screenshot of a GitHub repository page for 'ruppysupply/SmartsApp'. The 'Fork' button is highlighted with a red box. The repository has 4 Unwatch, 37 Stars, and 13 Forks. The main content shows a commit history table and a file browser for 'README.md'.

| Commit | Author | Message | Time |
|---------|-------------|---------------------------------|-------------|
| b637342 | ruppysupply | fix(web-app): fixed theme color | on Apr 25 |
| | | | 156 commits |

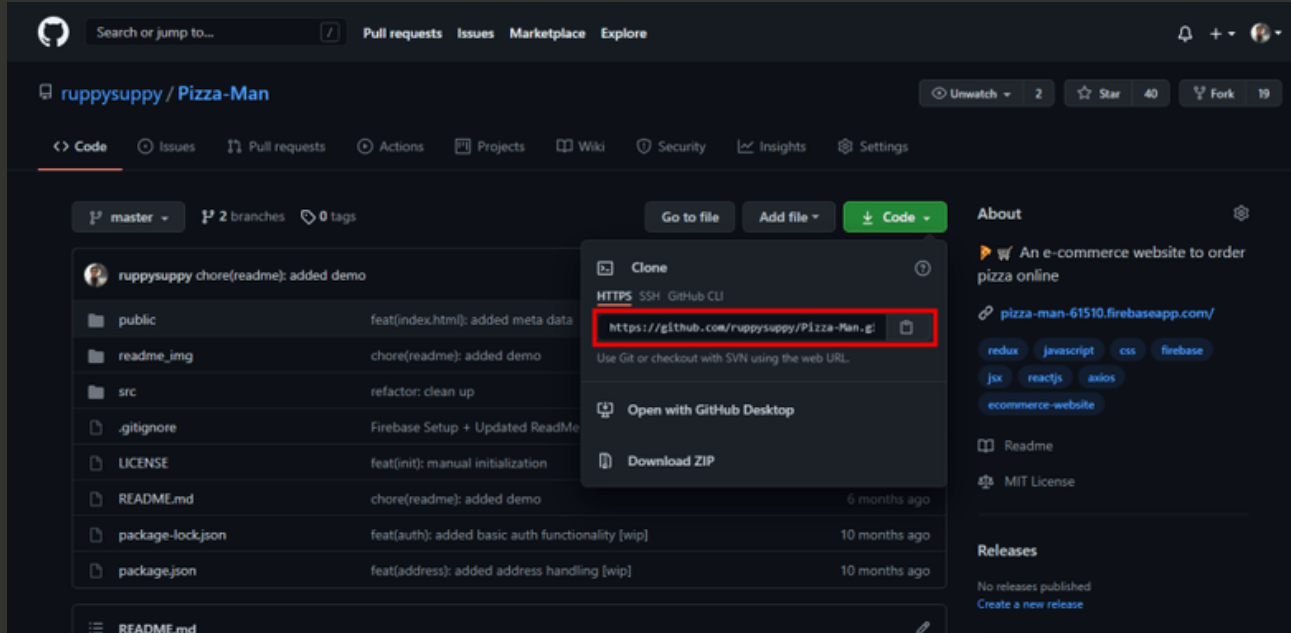
| File | Commit | Message | Time |
|---------------------|--------|--|--------------|
| desktop_app | | feat(desktop-app): implemented dark mode for splash screen | 6 months ago |
| e2ee_key_gen_server | | chore(nit): fixed a few not important though (nit) issues | 6 months ago |
| phone_app | | chore(readme): added app packaging section | 6 months ago |
| readme_img | | chore(readme): added demo | 6 months ago |
| web_app | | fix(web-app): fixed theme color | 2 months ago |
| LICENSE | | chore(web-app): initialized web-app | 7 months ago |
| README.md | | chore(readme): added demo | 6 months ago |

Clone

Now you have a personal copy of the repository.

Clone the repository using the command:

```
> git clone <clone link>
```



The screenshot shows the GitHub interface for the repository 'rappysupply / Pizza-Man'. The 'Code' button is clicked, opening a dropdown menu with the following options:

- Clone (selected)
- Open with GitHub Desktop
- Download ZIP

The 'Clone' option is expanded, showing the following options:

- HTTPS (selected)
- SSH
- GitHub CLI

The HTTPS URL `https://github.com/rappysupply/Pizza-Man.git` is highlighted with a red box. Below the URL, it says 'Use Git or checkout with SVN using the web URL.'

The repository page also shows a file tree on the left with files like 'public', 'readme_img', 'src', '.gitignore', 'LICENSE', 'README.md', 'package-lock.json', and 'package.json'. The right sidebar contains an 'About' section with a description: 'An e-commerce website to order pizza online', a link to the live site, and a 'Releases' section.

Push Directly

After making the required changes, you would like to push the changes to the **remote repository** for others to access it.

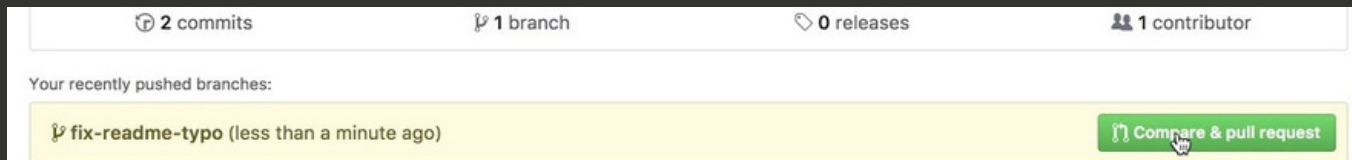
To push the changes to the remote repository use:

```
> git push <remote> <branch name>
```

The **remote** is just a fancy term for the **repository** alias. For the repo you clone from, it is set to **origin**

Pull Request

If you don't have **write access**, you will need to create a **Pull Request** in the source repository where your changes will be **reviewed and merged** by the owners or collaborators of the repository.





That's all folks!

HAPPY DEVELOPING!

Connect to me on:

 Medium

 LinkedIn

 GitHub