# GIT Orphan Branches

* In general, when you create a new branch off branch “master” you inherit its commit history.
* The exception is an orphan (or disconnected) branch.
* An orphan branch does not have a parent-child relationship to the master branches’ commits.
* Creating an orphan branch will retain the working tree of the branch it’s based on, but without an ancestor commit.

**Steps Followed:**

1. Clone a required GIT Repository

$ git clone <https://github.com/srikanth-josyula/sample.git>

$ cd sample/

1. The orphan branch is the kind of branch that git checkout --orphan makes: a branch that will have a new, disconnected root.

$ git checkout --orphan project-1

Switched to a new branch 'project-1'

1. If you want to start a disconnected history that records a set of paths that is totally different from the one of <start\_point>, then you should clear the index and the working tree right after creating the orphan branch by running git rm -rf . from the top level of the working tree. Afterwards you will be ready to prepare your new files

$ git rm -rf .

1. Place the projects under new orphan branch

$ git status

1. Add all the un staged files

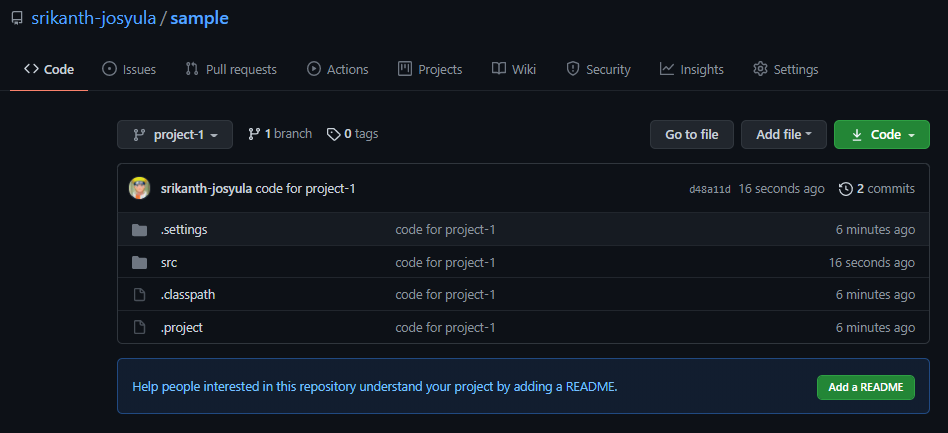
$ git add .

1. Commit the files to the branch

$ git commit -a -m "code for project-1"

1. Push the orphan branch git push origin <orphan branch name>

$ git push origin project-1



1. Create a another branch with new project-2
2. Check for the current branch

$ git branch

\* project-1

1. Repeat the steps for new orphan branch same as done for project-1

git checkout --orphan project-2

1. After above command you can still see the old files in the local folders, you can remove all the files you'll have in the staging area (so that they don't get committed)

$ git rm -rf .

1. If you do a git branch we can see the current branch we are in

$ git branch

project-1

\* project-2

1. Add the new code we want in our new branch, do a git add . to add all files to the new branch

$ git add .

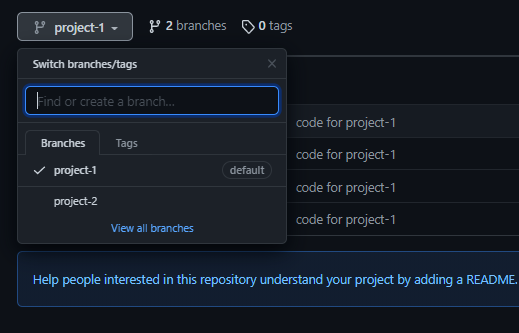
1. Commit the code of the project

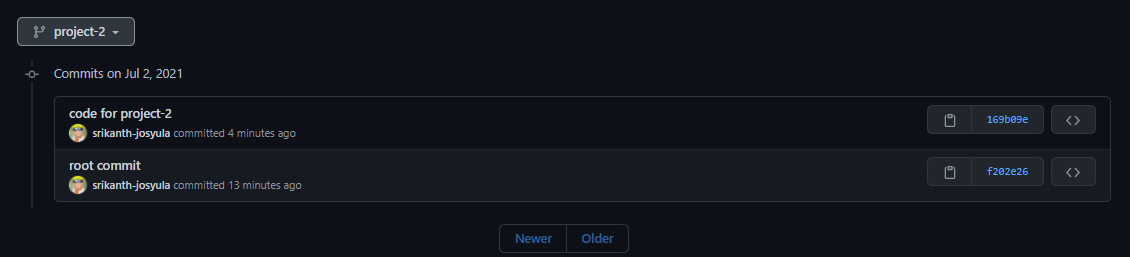
$ git commit -a -m "code for project-2"

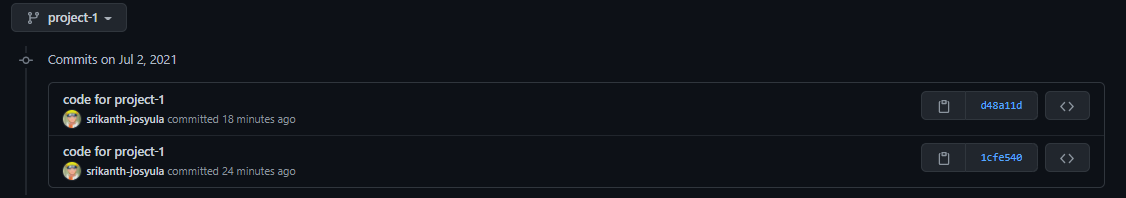
1. Push the orphan branch git push origin <orphan branch name>

$ git push origin project-2

1. Now in UI we can see two different branches handling their own history and code







1. If we need to switch between projects we can use

$ git checkout project-1

1. If we need to check the history the orphan branches provide their history

$ git log

commit d48a11d3bc3c2f17b1023f6d2740f069d1ef58a5 (HEAD -> project-1, origin/project-1)

Author: srikanth-josyula <josyulasrikanth94@gmail.com>

Date: Fri Jul 2 22:13:57 2021 +0530

code for project-1

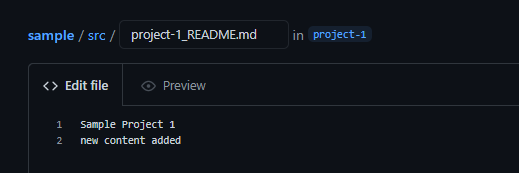
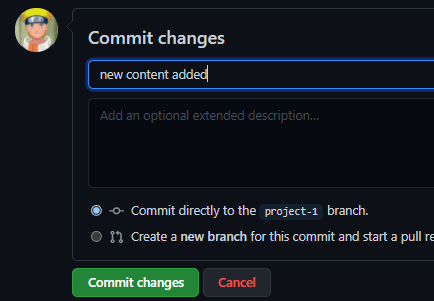
commit 1cfe54054ad923bb314597473f4720cb87c38082

Author: srikanth-josyula <josyulasrikanth94@gmail.com>

Date: Fri Jul 2 22:08:16 2021 +0530

code for project-1

1. git pull origin <branch\_name> will pull changes from the origin remote, branch and merge them to the local checked-out branch
2. for example there is a another push who changed the code, if we need to keep or local and remote in sync we can use this command to pull latest changes

**$ git pull origin project-1**

remote: Enumerating objects: 7, done.

remote: Counting objects: 100% (7/7), done.

remote: Compressing objects: 100% (2/2), done.

remote: Total 4 (delta 1), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (4/4), done.

From https://github.com/srikanth-josyula/sample

\* branch project-1 -> FETCH\_HEAD

d48a11d..978504a project-1 -> origin/project-1

Updating d48a11d..978504a

Fast-forward

src/project-1\_README.md | 3 ++-

1 file changed, 2 insertions(+), 1 deletion(-)

1. This way we can keep our local repo with the latest code