

Module 2: Working with Git Repositories

Demo 2: Demo on difference between repositories.

Problem Statement:

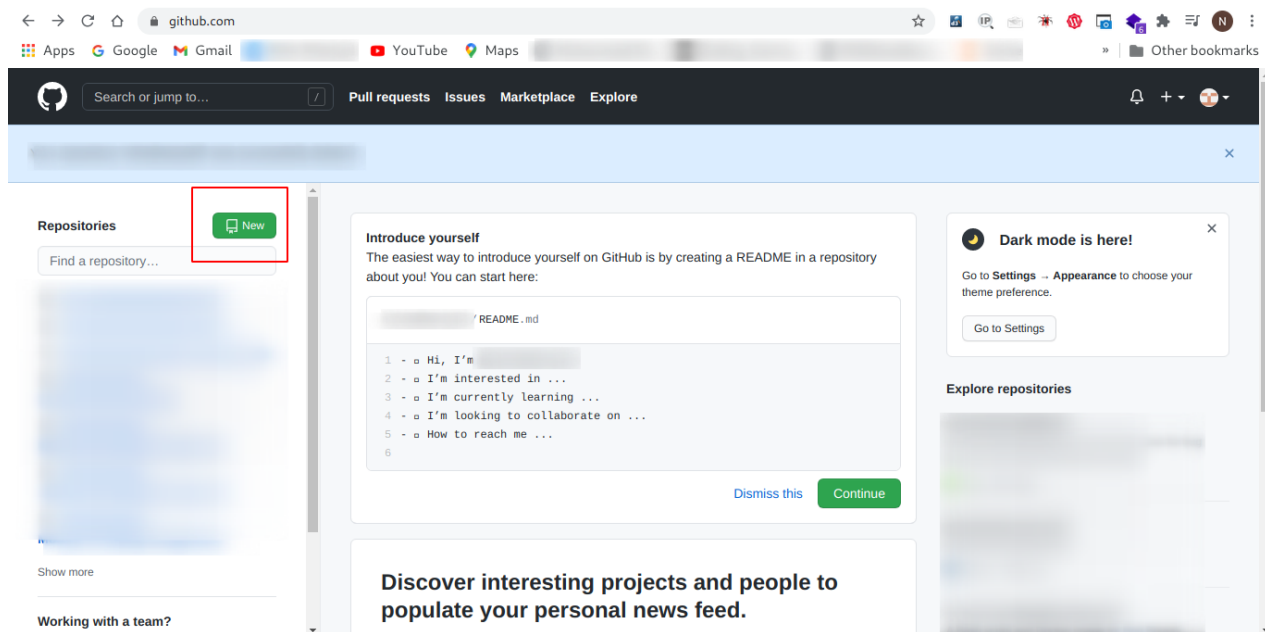
How we can use git push and git pull command for differentiating different repositories (local and remote).

Solution:

Step 1: In this demo, we are going to push our project into a remote git repository using the command ***git push***. Using the git push command, the commit is transferred or pushed to a remote repository like GitHub on a local branch on your computer. Below is the command used to push GitHub.

```
git push 'remote_name' 'branch_name'
```

Let us create a new repository first. Open your [Github](#) click on “new”.



You got a “**Create a new repository**”, after completing this page you successfully created your new repository.

Module 2 – Working With Git Repositories

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner * ntindharyel Repository name * demo ✓

Great repository names are short and memorable. Need inspiration? How about [solid-octo-funicular?](#)

Description (optional)

☒ Public
Anyone on the internet can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

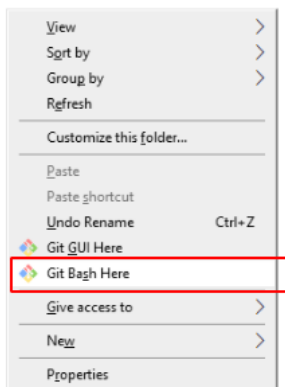
Initialize this repository with:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

☐ Choose a license
A license tells others what they can and can't do with your code. [Learn more.](#)

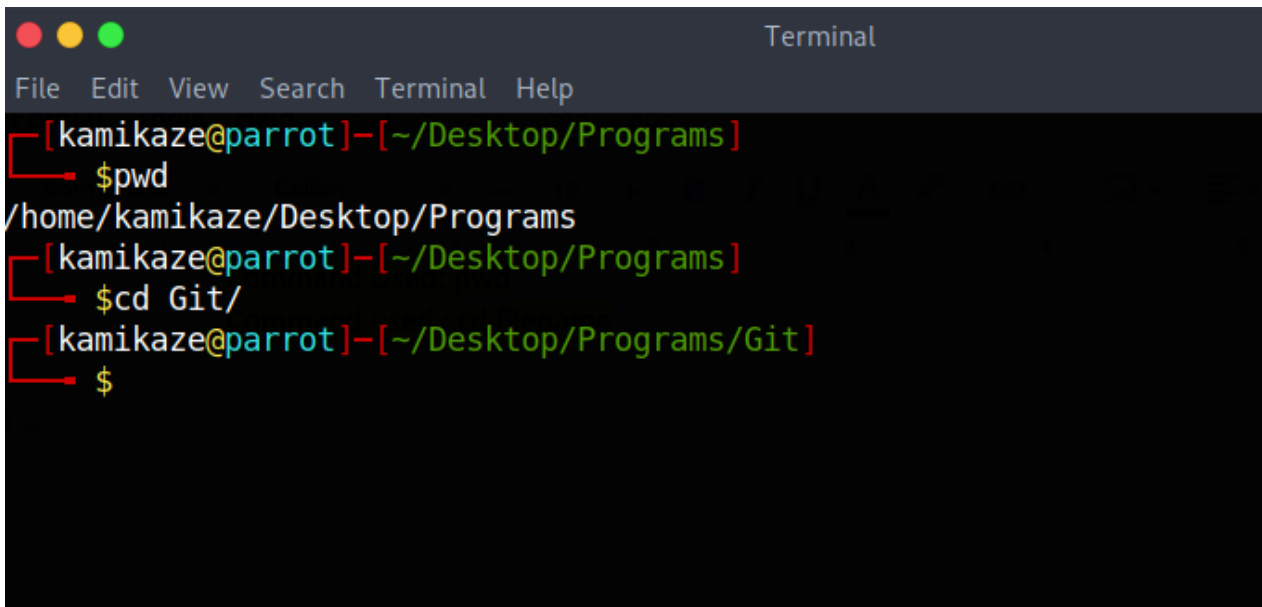
Step 2: **Open your Git Bash.** Git Bash can be downloaded [here](#), and it is a shell used to interface with the operating system which follows the UNIX command.



Step 3: Create your local project on your desktop, or open that project which you want to add into a remote git repository.

Command Used: `pad`

Command used: `cd filename`

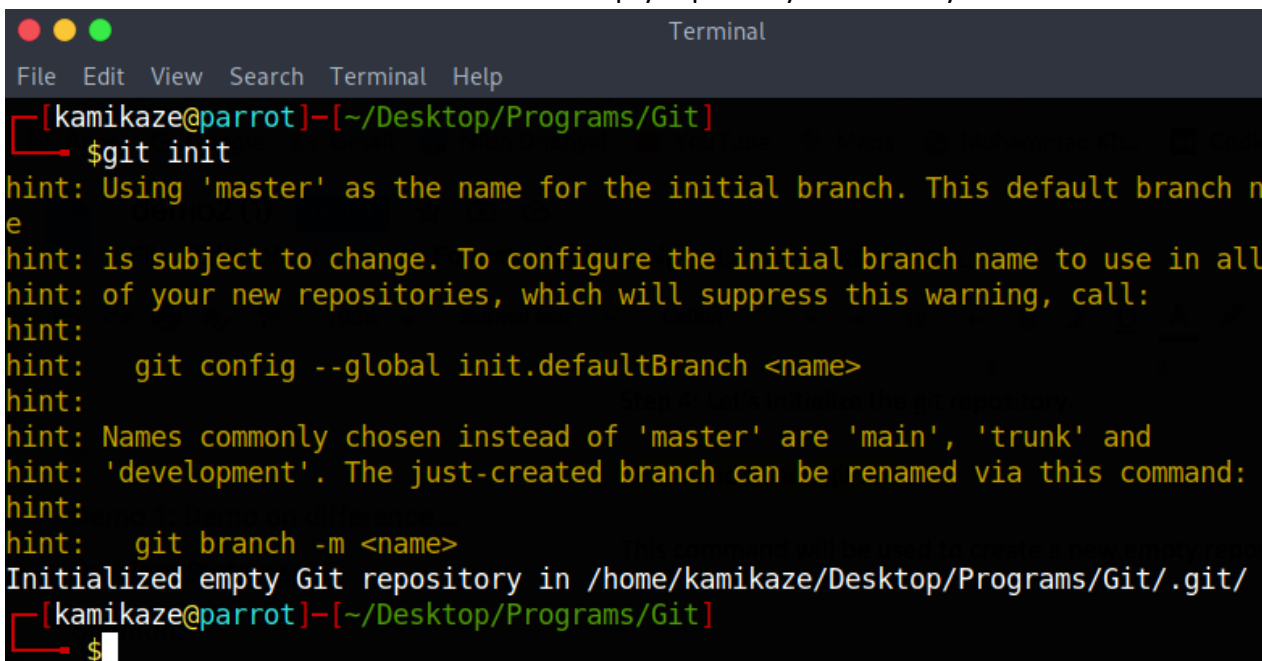
A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is [kamikaze@parrot]--[~/Desktop/Programs]. The user enters \$pwd, and the output is /home/kamikaze/Desktop/Programs. The user then enters \$cd Git/, and the prompt changes to [kamikaze@parrot]--[~/Desktop/Programs/Git]. The user enters \$, and the prompt remains the same.

```
[kamikaze@parrot]--[~/Desktop/Programs]
$pwd
/home/kamikaze/Desktop/Programs
[kamikaze@parrot]--[~/Desktop/Programs]
$cd Git/
[kamikaze@parrot]--[~/Desktop/Programs/Git]
$
```

Step 4: Let us Initialize the git repository.

Command Used: git init

This command will be used to create a new empty repository or directory of hidden files.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is [kamikaze@parrot]--[~/Desktop/Programs/Git]. The user enters \$git init. The terminal displays several hints: "hint: Using 'master' as the name for the initial branch. This default branch name is subject to change. To configure the initial branch name to use in all of your new repositories, which will suppress this warning, call: hint: git config --global init.defaultBranch <name> hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and 'development'. The just-created branch can be renamed via this command: hint: git branch -m <name>". It then says "Initialized empty Git repository in /home/kamikaze/Desktop/Programs/Git/.git/". The prompt changes to [kamikaze@parrot]--[~/Desktop/Programs/Git]. The user enters \$, and the prompt remains the same.

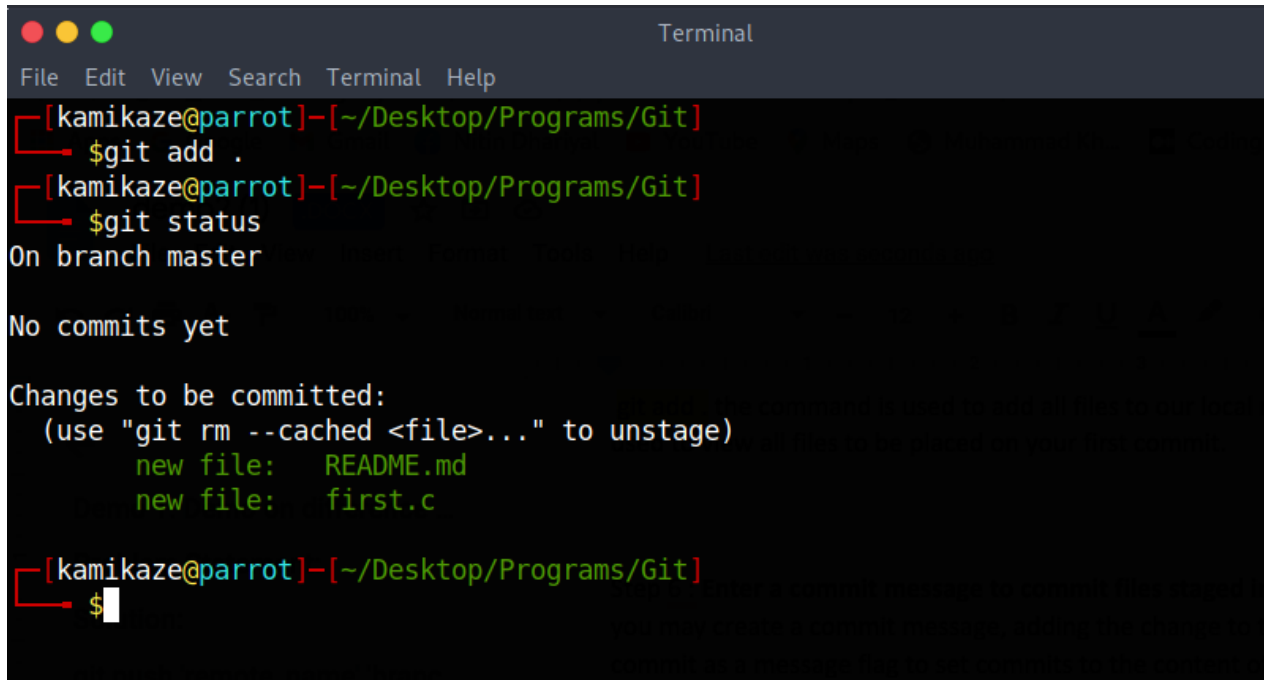
```
[kamikaze@parrot]--[~/Desktop/Programs/Git]
$git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint: git config --global init.defaultBranch <name>
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint: git branch -m <name>
Initialized empty Git repository in /home/kamikaze/Desktop/Programs/Git/.git/
[kamikaze@parrot]--[~/Desktop/Programs/Git]
$
```

Step 5: Let us add our files to the new local repository.

Command Used: git add.

Command Used: git status

`git add .` the command is used to add all files to our local repository. whereas `git status` command is used to view all files to be placed on your first commit.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is [kamikaze@parrot]~/. The user enters \$git add . and then \$git status. The output shows "On branch master" and "No commits yet". Under "Changes to be committed:", it lists "new file: README.md" and "new file: first.c". The prompt returns to [kamikaze@parrot]~/.

```
[kamikaze@parrot]~[~/Desktop/Programs/Git]
$git add .
[kamikaze@parrot]~[~/Desktop/Programs/Git]
$git status
On branch master

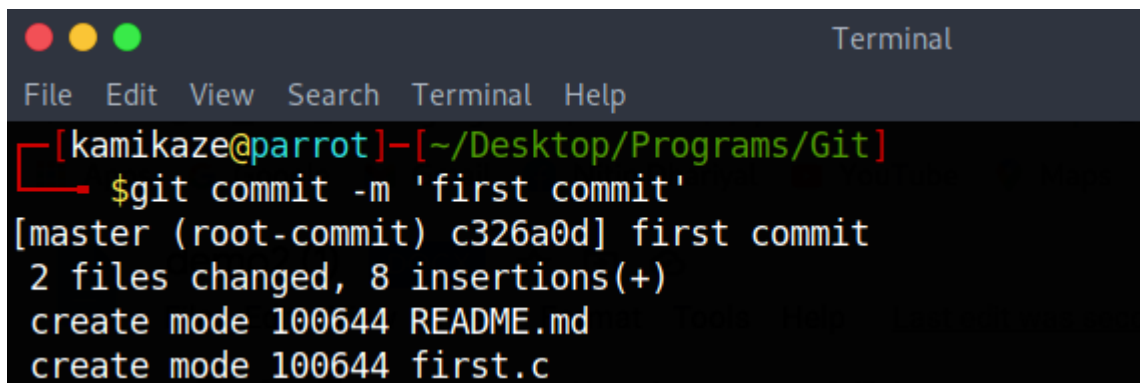
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   first.c

[kamikaze@parrot]~[~/Desktop/Programs/Git]
$
```

Step 6: Let us make a commit message to commit files staged in your local repository.

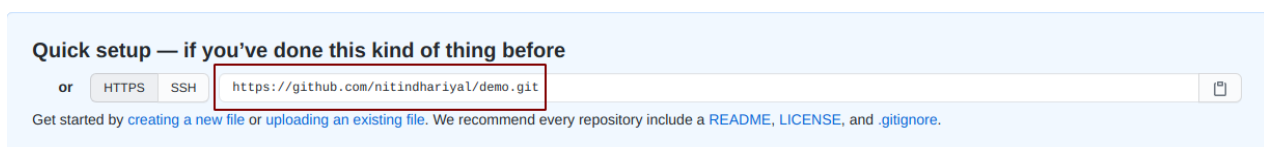
Command Used: `git commit -m 'your commit here'`

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The prompt is [kamikaze@parrot]~/. The user enters \$git commit -m 'first commit'. The output shows "[master (root-commit) c326a0d] first commit", "2 files changed, 8 insertions(+)", and file creation details for README.md and first.c. The prompt returns to [kamikaze@parrot]~/.

```
[kamikaze@parrot]~[~/Desktop/Programs/Git]
$git commit -m 'first commit'
[master (root-commit) c326a0d] first commit
2 files changed, 8 insertions(+)
create mode 100644 README.md
create mode 100644 first.c

[kamikaze@parrot]~[~/Desktop/Programs/Git]
$
```

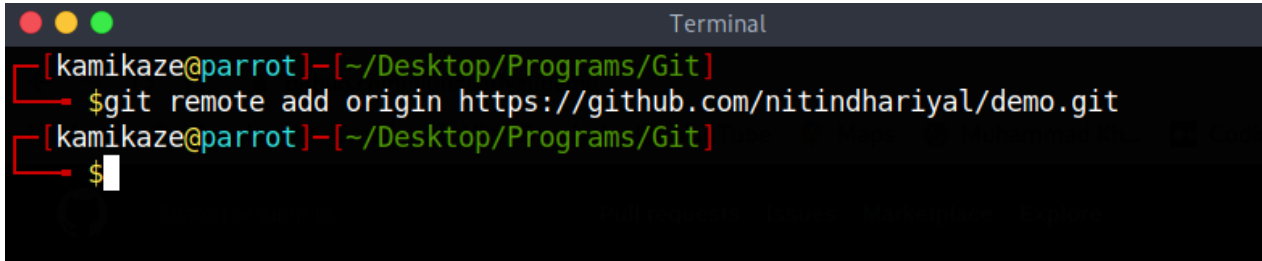
Step 7: Let us upload these repositories into our remote repository, for that we must copy our remote repository's URL from GitHub. The HTTPS or URL of the remote repository is copied from a GitHub account.



Step 8: For adding our local content to our remote repository.

Command Used: `git remote add origin 'your_url_name'`

here in the above code, 'origin' is the remote name, the remote URL is "<https://github.com/nitindhariyal/Demo.git>".

A terminal window titled "Terminal" with a dark background. The prompt is [kamikaze@parrot]--[~/Desktop/Programs/Git]. The command entered is \$git remote add origin https://github.com/nitindhariyal/demo.git. The prompt changes to [kamikaze@parrot]--[~/Desktop/Programs/Git] and then to \$ with a cursor.

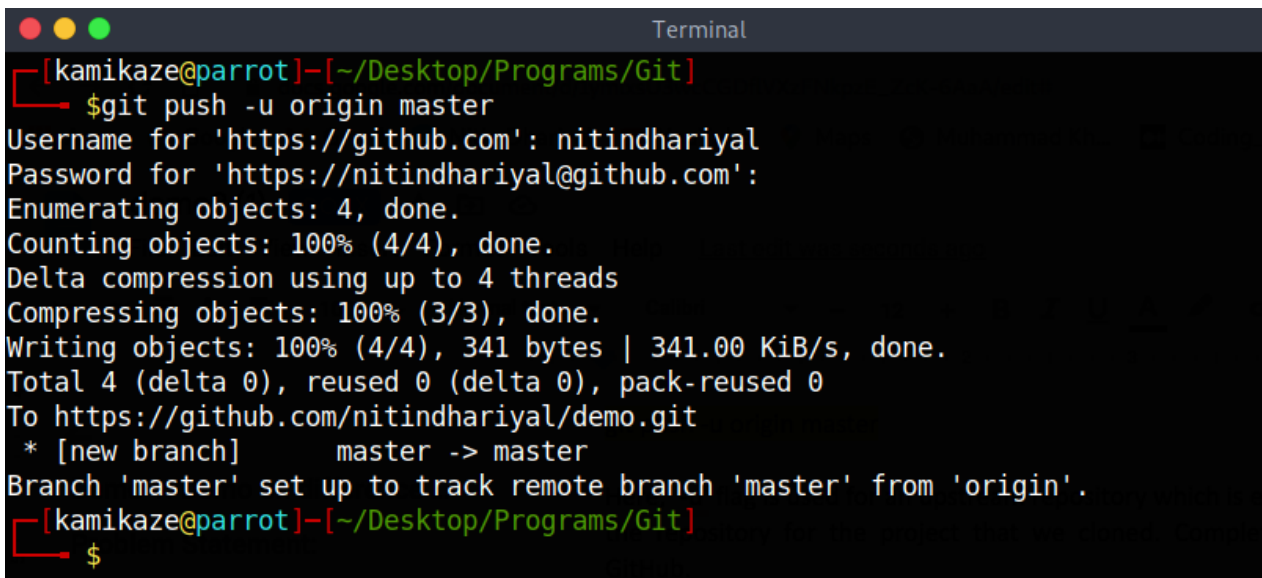
```
[kamikaze@parrot]--[~/Desktop/Programs/Git]
$git remote add origin https://github.com/nitindhariyal/demo.git
[kamikaze@parrot]--[~/Desktop/Programs/Git]
$
```

Step 9: Let us Push your local GitHub repository's code.

`git push -u origin master`

Here, '-u' flag is used for an upstream repository which is equivalent to '-set-upstream,'. Upstream is the repository for the project that we cloned. Complete the username and password of your GitHub.

Command used: `git push -u origin master`

A terminal window titled "Terminal" with a dark background. The prompt is [kamikaze@parrot]--[~/Desktop/Programs/Git]. The command entered is \$git push -u origin master. The output shows the push process: Username for 'https://github.com': nitindhariyal, Password for 'https://nitindhariyal@github.com':, Enumerating objects: 4, done., Counting objects: 100% (4/4), done., Delta compression using up to 4 threads, Compressing objects: 100% (3/3), done., Writing objects: 100% (4/4), 341 bytes | 341.00 KiB/s, done., Total 4 (delta 0), reused 0 (delta 0), pack-reused 0, To https://github.com/nitindhariyal/demo.git, * [new branch] master -> master, Branch 'master' set up to track remote branch 'master' from 'origin'. The prompt changes to [kamikaze@parrot]--[~/Desktop/Programs/Git] and then to \$ with a cursor.

```
[kamikaze@parrot]--[~/Desktop/Programs/Git]
$git push -u origin master
Username for 'https://github.com': nitindhariyal
Password for 'https://nitindhariyal@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 341 bytes | 341.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/nitindhariyal/demo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
[kamikaze@parrot]--[~/Desktop/Programs/Git]
$
```

Step 10: Now let us check our GitHub host files in our repository. The files hosting on GitHub you can finally see.

Module 2 – Working With Git Repositories

This screenshot shows the GitHub interface for a repository named 'demo' by user 'nitindhariyal'. The repository has 1 commit, 1 branch (master), and 0 tags. The commit history shows a single commit 'first commit' by 'nitindhariyal' 11 minutes ago, with files 'README.md' and 'first.c'. The README content is 'Hello world'. The right sidebar shows 'About' (demo repository), 'Releases' (No releases published), and 'Packages' (No packages published).

Step 11: GIT PULL allows others to view changes if you change the repository. It is used to recognise the changes you have made to your repository. Or also called a target repository. Below is the simple command to PULL from a branch:

Command Used: `git pull 'remote_name' 'branch_name'`

The 'git pull' command is a combination of 'git fetch' and 'git merge'. 'remote name' is the name of repository and 'branch name' is the name of a specific branch. Copy of a repository is the "Fork." Forking a repository allows you to experiment freely with modifications without affecting the original project."

This screenshot shows the GitHub interface for a repository named 'Adds quiet mode' by user 'f212493'. The repository has 6 commits, 1 branch (master), and 0 tags. The commit history shows a single commit 'Initial' by 'f212493' 2 years ago, with files '.gitignore', 'LICENSE', 'README.mkd', and 'main.go'. The right sidebar shows 'About' (A tool for adding new lines to files, skipping duplicates), 'Releases' (No releases published), and 'Packages' (No packages published). The 'Fork' button in the top right is highlighted with a red box.

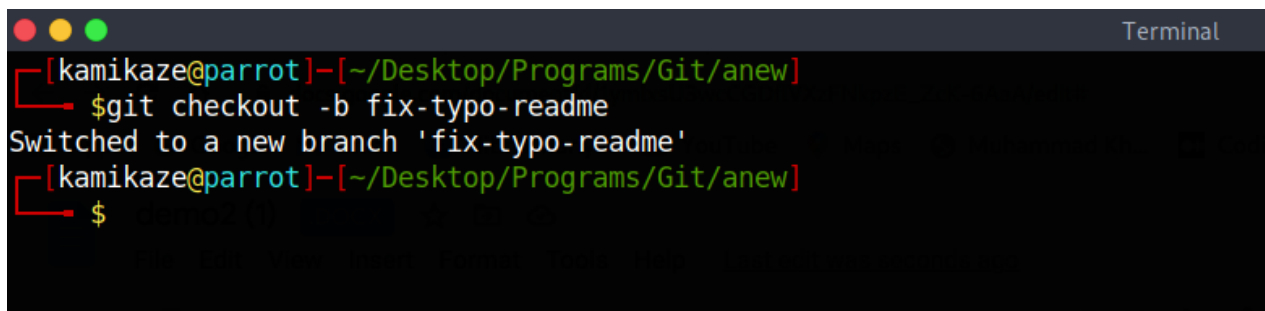
Step 12: open file using command cd and ls is used for listing the content in that file.

A terminal window titled "Terminal" with a dark background. The prompt is [kamikaze@parrot]~/. The user enters 'cd anew/' and the prompt changes to [kamikaze@parrot]~/Desktop/Programs/Git/anew/. Then the user enters 'ls' and the output is LICENSE main.go README.mkd. The prompt returns to [kamikaze@parrot]~/Desktop/Programs/Git/anew/ and the user enters '\$' at the end of the line.

```
[kamikaze@parrot]~[~/Desktop/Programs/Git]
$cd anew/
[kamikaze@parrot]~/Desktop/Programs/Git/anew
$ls
LICENSE main.go README.mkd
[kamikaze@parrot]~/Desktop/Programs/Git/anew
$
```

Step 13: Let us build a new branch.

Command Used: git checkout -b 'branch_name'

A terminal window titled "Terminal" with a dark background. The prompt is [kamikaze@parrot]~/Desktop/Programs/Git/anew/. The user enters 'git checkout -b fix-typo-readme' and the output is 'Switched to a new branch 'fix-typo-readme''. The prompt returns to [kamikaze@parrot]~/Desktop/Programs/Git/anew/ and the user enters '\$' at the end of the line.

```
[kamikaze@parrot]~/Desktop/Programs/Git/anew
$git checkout -b fix-typo-readme
Switched to a new branch 'fix-typo-readme'
[kamikaze@parrot]~/Desktop/Programs/Git/anew
$
```

Step 14: Let us make a change by vim from bash or by substituting the original README file directly.

Command used: nano filename

add something into that file and press ctrl+x then "Y".


```

Terminal
[kamikaze@parrot]--[~/Desktop/Programs/Git/aneu]
$git status
On branch fix-typo-readme
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.mkd

no changes added to commit (use "git add" and/or "git commit -a")
[kamikaze@parrot]--[~/Desktop/Programs/Git/aneu]
$git diff
diff --git a/README.mkd b/README.mkd
index 54203f0..b736de2 100644
--- a/README.mkd
+++ b/README.mkd
@@ -4,7 +4,7 @@ Append lines from stdin to a file, but only if they don't already appear in the
Outputs new lines to `stdout` too, making it a bit like a `tee -a` that removes duplicates.

## Usage Example
-
+## edited file
Here, a file called `things.txt` contains a list of numbers. `newthings.txt` contains a second
list of numbers, some of which appear in `things.txt` and some of which do not. `aneu` is used
to append the latter to `things.txt`.
[kamikaze@parrot]--[~/Desktop/Programs/Git/aneu]
$

```

Step 15: Let us add and commit the file to the repository. Using the following commands, you need to add and commit.

Command Used: git add filename

Command Used: git commit -m 'commit here'

```

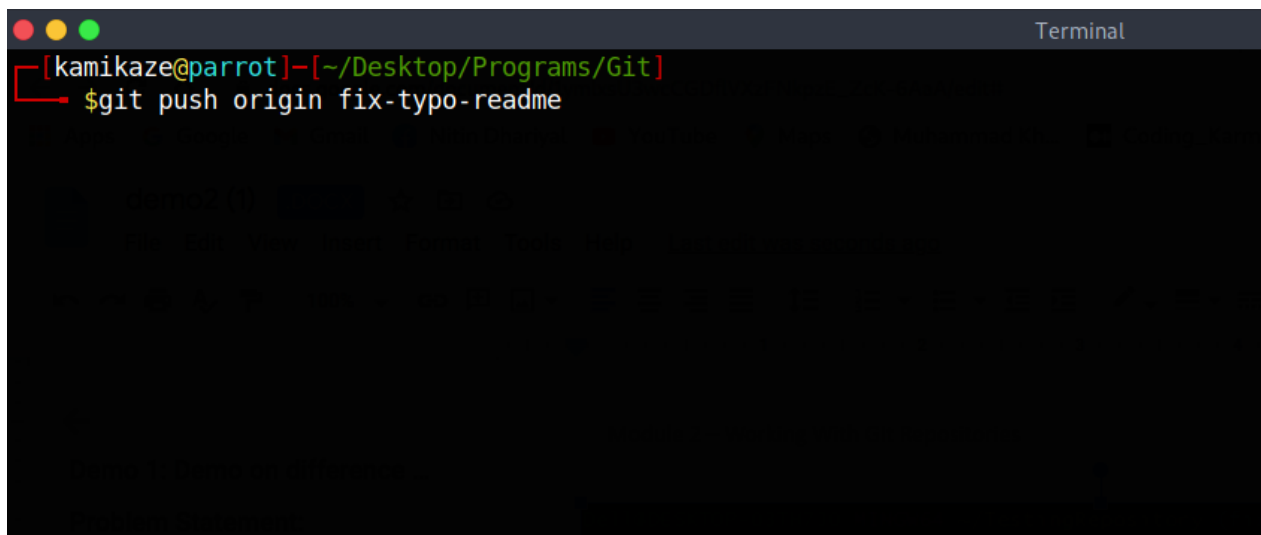
Terminal
[kamikaze@parrot]--[~/Desktop/Programs/Git/aneu]
$git commit -m 'Modified in README'
[fix-typo-readme c0c2688] Modified in README
1 file changed, 1 insertion(+), 1 deletion(-)
[kamikaze@parrot]--[~/Desktop/Programs/Git/aneu]
$

```

Step 16: Let us push the GitHub repository. we must push the content through,

Command Used: git push origin 'branch_name'.

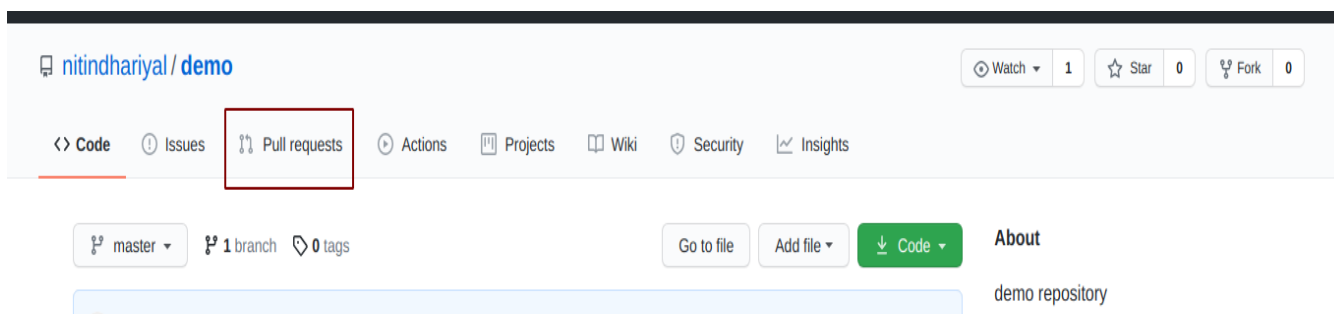
In the above code, the remote repository is the origin, and 'branch_name' is the branch that we want to upload.



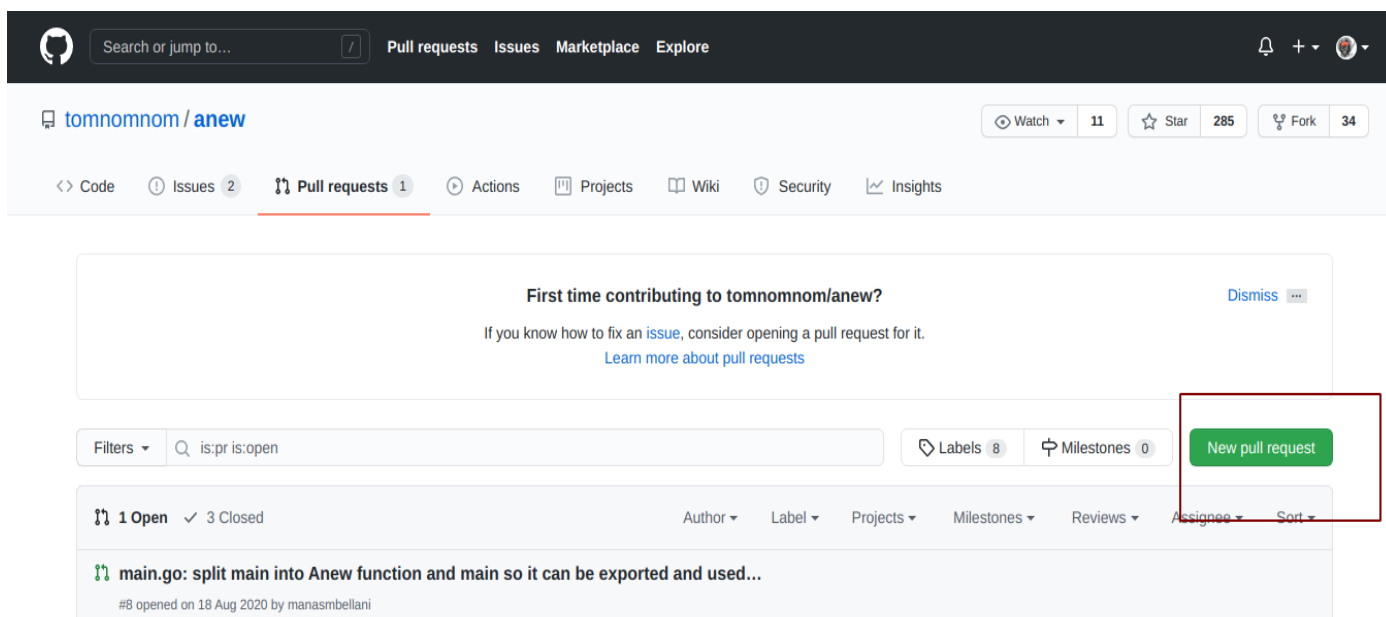
```
[kamikaze@parrot]-[~/Desktop/Programs/Git]
$git push origin fix-typo-readme
```

The terminal window shows a user named kamikaze@parrot in the directory ~/Desktop/Programs/Git. They have executed the command `$git push origin fix-typo-readme`. The output shows a successful push to the 'fix-typo-readme' branch on the 'origin' remote.

Step 17: let us PULL GitHub application for a particular branch. go to Github and just click on Pull requests.



after clicking into this, click on “New Pull request”.



Step 18: Let us Open a Pull request. To complete the action, you must click on "Create pull request."

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: TheAlgorithms/Java ▾


base: master ▾

←

head repository: nitindhariyal/Java ▾

compare: fix-README-typo ▾

✓ **Able to merge.** These branches can be automatically merged.



adding a typo in README

Write

Preview

H B I ≡ <> 🔗 ≡ ≡ ☑ @ 📎 ↶ ▾

Adding a typo in README.

Attach files by dragging & dropping, selecting or pasting them.

☒ Allow edits by maintainers ⓘ

Create pull request ▾

ⓘ Remember contributions to this repository should follow its [contribution guidelines](#) and [code of conduct](#)

Step 19: Now add some pull requests into this repository which we are edited before.

Module 2 – Working With Git Repositories

This screenshot shows a GitHub pull request titled "adding a typo in README #2199". The pull request is from the branch "nitindhariyal:fix-README-typo" to "TheAlgorithms:master". A red box highlights the "Files changed" tab, which shows a commit titled "adding in readme" with a commit hash of "5f035c0". The commit message is "Adding a typo in README." The right sidebar shows the "Reviewers" section with "No reviews" and the "Assignees" section with "No one assigned".

Step 20: Let us see these changes, click on the 'File changed'.

This screenshot shows the same GitHub pull request page, but with the "Files changed" tab highlighted by a red box. The "Files changed" tab shows the same commit as the previous screenshot. The "Reviewers" section now shows "No reviews" and the "Assignees" section shows "No one assigned".

Step 21: Finally, we have pulled this, you can see into the screenshot below,

Module 2 – Working With Git Repositories

<> Code

Issues 120

Pull requests 606

Actions

Projects

Wiki

Security

Insights

adding a typo in README #2199

nitindhariyal wants to merge 1 commit into TheAlgorithms:master from nitindhariyal:fix-README-typo

EditOpen with

Conversation 1

Commits 1

Checks 0

Files changed 1

+1 -1

Changes from all commitsFile filterJump to0 / 1 files viewedReview changes

2 README.md

<>Viewed

... @@ -1,5 +1,5 @@

1	[![Gitpod ready-to-code](https://img.shields.io/badge/Gitpod-ready--to--code-blue?logo=gitpod)](https://gitpod.io/#https://github.com/TheAlgorithms/Java)	1	[![Gitpod ready-to-code](https://img.shields.io/badge/Gitpod-ready--to--code-blue?logo=gitpod)](https://gitpod.io/#https://github.com/TheAlgorithms/Java)
2	-	2	+ #edited one
3	# The Algorithms - Java	3	# The Algorithms - Java
4	[![Build Status](https://api.travis-ci.com/TheAlgorithms/Java.svg?branch=master)](https://travis-ci.com/TheAlgorithms/Java) 	4	[![Build Status](https://api.travis-ci.com/TheAlgorithms/Java.svg?branch=master)](https://travis-ci.com/TheAlgorithms/Java)
5	[![Donate](https://img.shields.io/badge/Donate-PayPal-green.svg)](https://www.paypal.me/TheAlgorithms/100)	5	[![Donate](https://img.shields.io/badge/Donate-PayPal-green.svg)](https://www.paypal.me/TheAlgorithms/100)