INDEX

**[GIT(Global Information Tracker)](#_Toc13803)**

**[1. GIT is a Version Control System also known as VCS](#_Toc21258)**

[1.1. Git config](#_Toc7227)

[1.2. Why GIT ?](#_Toc23926)

[1.3. USING GIT](#_Toc3902)

[1.4. Installation of Git](#_Toc14211)

[1.5. Terms](#_Toc28488)

**[2. Configuring Git](#_Toc21814)**

[2.1. Settings](#_Toc32253)

[2.2. Three different levels](#_Toc25503)

**[3. Git commands](#_Toc26134)**

[3.1. Help](#_Toc12433)

[3.2. Status](#_Toc32283)

[3.3. Git DIFF](#_Toc12168)

[3.3.1. How git stores the data ?](#_Toc27389)

[3.4. Rename and Restore files in GIT Repository using mv and the restore command](#_Toc19149)

[3.5. GIT Branching](#_Toc16822)

3.5.1. Checkout -b option used for creating a new branch and move that branch

3.5.2. By using checkout command change to the other branch

[3.5.3. Rename and Delete branch using -m and -d commands](#_Toc12031)

3.5.4. What are contents present inside the branch ?

[3.6. Git Merge](#_Toc19571)

[3.6.1. Merge Conflicts](#_Toc17658)

[3.7. Git Rebase](#_Toc2234)

[3.7.1. Interactive Rebasing](#_Toc8139)

[3.7.2. Options in git rebase](#_Toc9827)

[3.7.3. Modify the latest commit by using git rebase](#_Toc28317)

[3.7.4. Conflict](#_Toc18031)

[3.7.5. Examples](#_Toc18031)

[3.7.5.1. Case1](#_Toc18031)

[3.7.5.2. Case2](#_Toc18031)

[3.7.5.3. Case3](#_Toc18031)

[3.7.5.4. Case4](#_Toc18031)

[3.7.5.5. Case 5 (By using update-ref delete all commit history and root command )](#_Toc18031)

[3.7.5.6. Case6](#_Toc18031)

[3.7.5.7. Case7](#_Toc18031)

[3.7.5.8. Case 8( Edit the merge commit )](#_Toc18031)

[3.8. GIT Cherry Pick](#_Toc18031)

[3.9. GIT Head](#_Toc18031)

[3.10. GIT RESET](#_Toc9342)

[3.10.1. Reset has options](#_Toc12413)

3.10.2. Examples

3.10.2.1. Case1 (by using git hard command delete all the commits of that file)

3.10.2.2. Case2 (To remove the staging into working area by using mixed option)

3.10.2.3. To remove from staging area into working area

[3.11. GIT STASH](#_Toc12980)

3.11.1. To know previous commit name and stash id

3.11.2. Custom name also given to the stash

3.11.3. Stash pop

3.11.4. Stash apply

3.11.5. Handle multiple stash

3.11.6. To delete one stash

3.11.6.1. By using this drop automatically stash will be deleted

3.11.6.2. By using stash clear to clear the stash list

3.11.6.3. To create new branch with stashed data

[3.12. Git checkout](#_Toc23719)

[3.12.1. Detached Head](#_Toc22672)

3.12.2. List of all branches

**3.12.3. By using the restore command, we get changes at a particular commit.**

[3.13. Git revert](#_Toc10996)

**[4. GITHUB](#_Toc28915)**

[4.1. Git Vs GitHub](#_Toc15078)

[4.2. GIT Cloning](#_Toc7571)

[4.3. Github with SSH](#_Toc3011)

[4.3.1. How to setup the ssh ?](#_Toc8476)

[4.3.2. To implement SSH keys in the system](#_Toc8256)

[4.4. Create Repository in Github](#_Toc2167)

4.4.1. Adding Remote

4.4.2. Option 1 ( git push --force origin1 master)

4.4.3. Option 2 ( git pull origin master --allow-unrelated-histories )

4.4.4. Option 3 ( git remote prune origin )

4.4.4.1. Git branch -r option shows the remote branches

4.4.5. Option 4 ( git push --prune origin )

4.4.6. Option 5 (git push -u origin/b2)

4.4.7. Pushing the local changes from one branch to another remote branch

4.5. What git push -u means ?

4.5.1. Difference between the main and master branch in the git repository

4.6. Remote Tracking Branches

4.6.1. To know the remote branches by using branch -r

4.6.2. Checkout the Remote tracking branches in the local git repository

4.6.3. List of all branches in the remote repository

[4.7. Git Fetch](#_Toc31436)

[4.8. Git Pull](#_Toc4119)

[4.8.1. Implementation of the git pull Resolve merge conflicts for the change from remote repository](#_Toc382)

[4.9. Github Gists](#_Toc20930)

[4.10. Github Pages](#_Toc15783)

[4.10.1. Creating the github pages](#_Toc12591)

[4.11. Git Pull Requests](#_Toc1643)

[4.11.1. Pull Requests Workflow](#_Toc20642)

[4.11.2. Implement the pull requests](#_Toc331)

[4.11.3. Resolve conflicts for the feature branch when pull request is raised in Github repository](#_Toc2172)

[4.11.4. Add Collaborators and apply branch protection rules in Github for restricting commits](#_Toc31059)

[4.12. Git Forking](#_Toc3932)

[4.13. Git Tags](#_Toc5442)

[4.13.1. Viewing Tags](#_Toc8643)

[4.13.2. Creating lightweight tags](#_Toc30930)

[4.13.3. Create Annotated Tags](#_Toc25995)

[4.13.4. Pushing Tags to the GIthub Remote Repository](#_Toc3275)

[4.13.5. Moving the tags into two ways](#_Toc4036)

[4.14. Git Reflog](#_Toc9600)

GIT(Global Information Tracker)

1. GIT is a Version Control System also known as VCS

GIT Software is developed by the Linus Torvalds, who developed the Linux.

Basically VCS is the software designed to record changes made to the file over time.

GIT not only records the source code it also tracks the images , research papers or any type of files.

GitHub is a web-based version control and collaboration platform for software developers.

A version control system records the changes made to our code over time in a special database called **repository(storage space).** By using version control system we can track our project history and work together.

Version control systems fall into two categories

1. **Centralized** : All team members connect to a central server to get the latest copy of the code and to share their changes with others. For example : Subversion and Team foundation server.Problem: if the sever goes offline we cannot save the snapshots of our project. Until wait the sever come back.
2. **Distributed** : In distribute there is no problem every team member has a copy of the project with its history on their machine and save the snapshots of our local systems. If the central server is offline we can synchronize our work directly with others git.
   1. Git config : It is builtin Tools

* --global ( user info )
* --system ( system info )
* --local ( repo info )

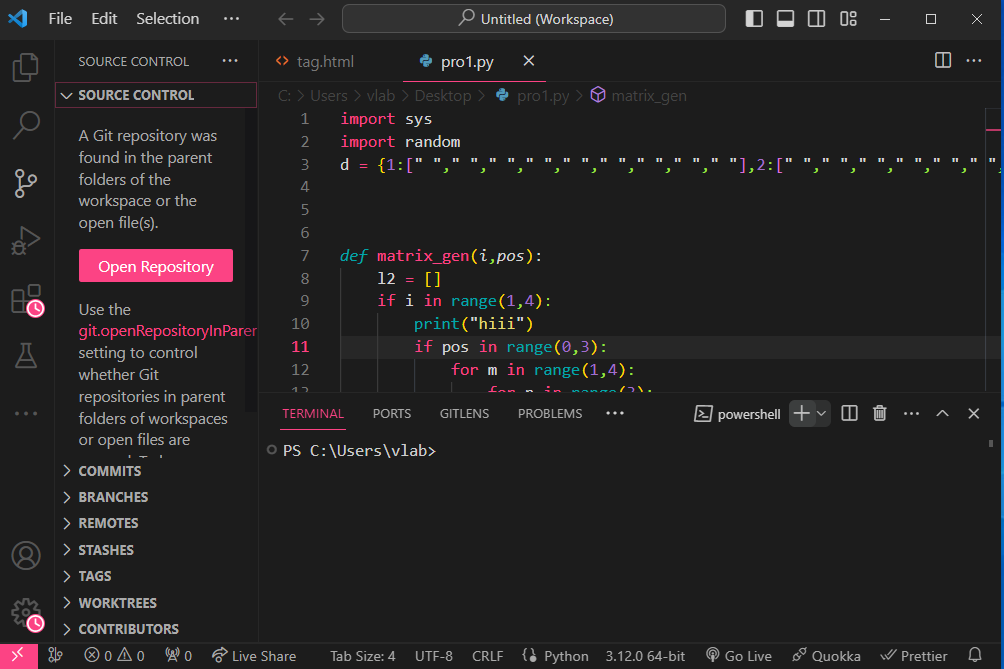
### Why GIT ?

* Free
* Distributed source control system
* Open Source
* Super Fast
* Scalable Operations
* Cheap branching and merging

### USING GIT

* The command line : Open a terminal or command prompt window to execute git commands.
* Code editors and IDEs (Integrated Development Environment):

For example : In vs code we have source control panel which gives us the essential git features.



1. Graphical user interfaces: Git comes with built-in GUI tools for committing (git-gui) like there are different platforms we have tools for windows , mac, linux, Android and iOS.

### Installation of Git:

Windows:

* Click the search icon and type cmd
* By knowing the version just type git --version
* Git version 2.44.0.windows.1 showing and install

### Terms:

* **Commit** : use , date and commit unique id
* **Clone**  : clone means Exact copy
* **Tracked/Untracked files :** .git folder is tracking the different files.
* **Branch**
* **Master(branch)**
* **Origin**

## Configuring Git:

### Settings:

* Name
* Email
* Default Editor
* Line Ending

### Three different levels:

* System : all users
* Global : All repositories of the current user
* Local : The current repository or the repository in the current folder

**To specify the level at which we are defining these settings:**

git config --global user.name "varalaxmi ganta"

git config --global user.email [varalaxmiganta066@gmail.com](mailto:varalaxmiganta066@gmail.com)

For example :

By using vs code when we type Code in cmd then vs code will be open but it will close automatically after some time.

git config --global core.editor "code --wait" by using this command we tell the terminal window to wait until we close the new vs code instance.

git config --global -e This will open default editor to edit all the global settings.In vs code : .gitconfig file will be open

For example

Git config --global core.editor “notepad”

Git config --get core.editor

[user]

Name = varalaxmi ganta

Email = [varalaxmiganta066@gmail.com](mailto:varalaxmiganta066@gmail.com)

[core]

Editor = code --wait

After close the Configuration file then terminal will be terminate the waiting command.

**To configure how git should handle end of lines:**

In windows end of lines are marked with two special characters

abc \r\n

\r -> Carriage Return

\n -> Line Feed

To configure one property called **core.autocrlf** which is short form carriage return

git config --global core.autocrlf input

vlab@VEDA2F118 MINGW32 ~/Projects/Moon (master)

$ git config --list

diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f

filter.lfs.smudge=git-lfs smudge -- %f

filter.lfs.process=git-lfs filter-process

filter.lfs.required=true

http.sslbackend=openssl

http.sslcainfo=C:/Users/vlab/AppData/Local/Programs/Git/mingw

32/etc/ssl/certs/ca-bundle.crt

core.autocrlf=true

core.fscache=true

core.symlinks=false

core.fsmonitor=true

pull.rebase=false

credential.helper=manager

credential.https://dev.azure.com.usehttppath=true

init.defaultbranch=master

pack.packsizelimit=2g

user.name=varalaxmi ganta

user.email=varalaxmiganta066@gmail.com

core.editor=code --wait

core.autocrlf=input

diff.tool=vscode

core.repositoryformatversion=0

core.filemode=false

core.bare=false

core.logallrefupdates=true

:

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir sampleproject/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd sampleproject/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ touch index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ ls

index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ ls -a

./ ../ index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/sample

project/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ ls -a

./ ../ .git/ index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config user.email

varalaxmiganta066@gmail.com

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config --local user.email varalaxmiganta77@gmail.com

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config user.email

varalaxmiganta77@gmail.com

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config user.name vara

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config user.name

vara

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config --unset user.name

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config user.name

varalaxmi ganta

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config user.email

varalaxmiganta77@gmail.com

**To remove either name or email by using --unset**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config --unset user.email

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config user.email

varalaxmiganta066@gmail.com

**To remove the complete section**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config --local --remove-section user

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config --local user.email varalaxmiganta77@gmail.com

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git config --local user.name vara

## Git commands

### Help :

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git help -a

Or

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git help --all

Or

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git help

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git help init

One new web page will be open

### Status:

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ ls

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ ls -a

./ ../ .git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ touch index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ ls

index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

index.html

nothing added to commit but untracked files present (use "git add" to track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git add index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: index.html

* + 1. **To Remove a file from Staging area by using --cached**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git rm --cached index.html

rm 'index.html'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

index.html

nothing added to commit but untracked files present (use "git add" to track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git add index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git commit -m "Initial commit"

[master (root-commit) d4b73c8] Initial commit

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ vi index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

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: index.html

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git add index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git commit -m " name add "

[master 5e712cd] name add

1 file changed, 2 insertions(+)

* + 1. **To know the history**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git log

commit 5e712cdb8192c85b9d49d46286bc8f9a4865f31c (HEAD -> master)

Author: vara <varalaxmiganta77@gmail.com>

Date: Wed Mar 6 15:27:35 2024 +0530

name add

commit d4b73c8da7418956ce99baf9febc96abb2d1d45e

Author: vara <varalaxmiganta77@gmail.com>

Date: Wed Mar 6 15:25:33 2024 +0530

Initial commit

### Git DIFF :

Diff command is used in git to track the difference between the changes made on a file.

Diff command takes two inputs and reflects the differences between them.

Git diff : Difference between the working and staging areas

Git diff --staged : Difference between the Staging area and Repository area

Git diff head : Difference between the working area and Repository area

**To check the changes in the working area and the staged area**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ vi contact.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

contact.txt

nothing added to commit but untracked files present (use "git add" to

track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git add contact.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: contact.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ vi contact.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: contact.txt

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: contact.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff

diff --git a/contact.txt b/contact.txt

index 4edc036..fe09b22 100644

--- a/contact.txt

+++ b/contact.txt

@@ -1 +1,2 @@

contact text

+changes made again

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff --staged

diff --git a/contact.txt b/contact.txt

new file mode 100644

index 0000000..4edc036

--- /dev/null

+++ b/contact.txt

@@ -0,0 +1 @@

+contact text

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff head

diff --git a/contact.txt b/contact.txt

new file mode 100644

index 0000000..fe09b22

--- /dev/null

+++ b/contact.txt

@@ -0,0 +1,2 @@

+contact text

+changes made again

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git commit -m "contact file added"

[master 48d405b] contact file added

1 file changed, 1 insertion(+)

create mode 100644 contact.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

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: contact.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff

diff --git a/contact.txt b/contact.txt

index 4edc036..fe09b22 100644

--- a/contact.txt

+++ b/contact.txt

@@ -1 +1,2 @@

contact text

+changes made again

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff --staged

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff head

diff --git a/contact.txt b/contact.txt

index 4edc036..fe09b22 100644

--- a/contact.txt

+++ b/contact.txt

@@ -1 +1,2 @@

contact text

+changes made again

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git status

On branch master

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: contact.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git add contact.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff --staged

diff --git a/contact.txt b/contact.txt

index 4edc036..fe09b22 100644

--- a/contact.txt

+++ b/contact.txt

@@ -1 +1,2 @@

contact text

+changes made again

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/sampleproject (master)

$ git diff head

diff --git a/contact.txt b/contact.txt

index 4edc036..fe09b22 100644

--- a/contact.txt

+++ b/contact.txt

@@ -1 +1,2 @@

contact text

+changes made again

### How git stores the data ?

Git stores the data in the form of keys and values.

Values is nothing but the contents of the file.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/chann

elproj/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ vi channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

channel.txt

nothing added to commit but untracked files present (use "git add" to

track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git log --oneline --all --graph

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git add channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git commit -m "Initial commit"

[master (root-commit) 2104add] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git log

commit 2104add0ab1048363997de04637283eab092a0ce (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 09:21:41 2024 +0530

Initial commit

commit 2104add0ab1048363997de04637283eab092a0ce

Commit is the key and SH1 value is 2104add0ab1048363997de04637283eab092a0ce (data)

21 represented key and 04add0ab1048363997de04637283eab092a0ce is value

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git cat-file 2104add0ab1048363997de04637283eab092a0ce -p

tree 7a8a37e1f0ab40e8547dfa6a84f0655fe27c2064

author varalaxmi ganta <varalaxmiganta066@gmail.com> 1709783501 +0530

committer varalaxmi ganta <varalaxmiganta066@gmail.com> 1709783501 +05

30

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

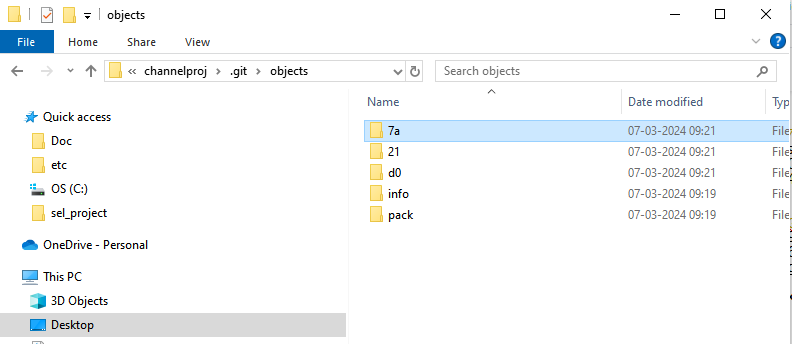
$ git cat-file 7a8a37e1f0ab40e8547dfa6a84f0655fe27c2064 -p

100644 blob d0c9f8634faba69dc02498f904e7fa30b2065726 channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git cat-file d0c9f8634faba69dc02498f904e7fa30b2065726 -p

hello veda world

21 file is linking the 7a tree file and 7a file is linking to the blob d0

d0 has the content of that file

#### Rename and Restore files in GIT Repository using mv and the restore command

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ ls

channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ mv channel.txt channel.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ ls

channel.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add/rm <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory

)

deleted: channel.txt

Untracked files:

(use "git add <file>..." to include in what will be committed)

channel.html

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git add channel.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: channel.html

Changes not staged for commit:

(use "git add/rm <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory

)

deleted: channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git add channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

renamed: channel.txt -> channel.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git commit -m "rename channel file"

[master b4aa2dd] rename channel file

1 file changed, 0 insertions(+), 0 deletions(-)

rename channel.txt => channel.html (100%)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ ls

channel.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git mv channel.html channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

renamed: channel.html -> channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git commit -m "channel rename"

[master 2d189df] channel rename

1 file changed, 0 insertions(+), 0 deletions(-)

rename channel.html => channel.txt (100%)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ vi channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

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: channel.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git add channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: channel.txt

**If we want to remove channel.txt file from staging area**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git restore --staged channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

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: channel.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git restore channel.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/channelproj (master)

$ vi channel.txt

**In that file modified content will be gone**

### GIT Branching :

Local Repository

HEAD : master

Head is current pointer when we commit then a new hash will formed and head will be moved

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/ecomm

erceproj/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

index.txt

nothing added to commit but untracked files present (use "git add" to

track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git commit -m "Initial commit"

[master (root-commit) c527478] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git log

commit c527478df7ddf80fa6411b1d4826432b825b46b1 (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 09:59:47 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git log --oneline

c527478 (HEAD -> master) Initial commit

* + 1. **Checkout -b option used for creating a new branch and move that branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git checkout -b login\_feature

Switched to a new branch 'login\_feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git log

commit c527478df7ddf80fa6411b1d4826432b825b46b1 (HEAD -> login\_feature

, master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 09:59:47 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ mkdir login

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ cd login

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj/login (login\_feature)

$ vi login.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj/login (login\_feature)

$ git status

On branch login\_feature

Untracked files:

(use "git add <file>..." to include in what will be committed)

./

nothing added to commit but untracked files present (use "git add" to

track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj/login (login\_feature)

$ cd ..

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git status

On branch login\_feature

Untracked files:

(use "git add <file>..." to include in what will be committed)

login/

nothing added to commit but untracked files present (use "git add" to

track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git status

On branch login\_feature

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: login/login.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git commit -m "Implement login feature ui"

[login\_feature 64f9dcf] Implement login feature ui

1 file changed, 2 insertions(+)

create mode 100644 login/login.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git log

commit 64f9dcf4ee0a2741360c6dc89e2ad45decdbabfb (HEAD -> login\_feature

)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 10:06:07 2024 +0530

Implement login feature ui

commit c527478df7ddf80fa6411b1d4826432b825b46b1 (master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 09:59:47 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git log --oneline

64f9dcf (HEAD -> login\_feature) Implement login feature ui

c527478 (master) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ ls

index.txt login/

* + 1. **By using checkout command change to the other branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ ls

index.txt

**Here we have only one commit in the master branch other commits are not visible**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git log

commit c527478df7ddf80fa6411b1d4826432b825b46b1 (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 09:59:47 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git log --oneline

c527478 (HEAD -> master) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git status

On branch master

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: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git commit -m "minor bug fixes"

[master 4daae3c] minor bug fixes

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

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git log --oneline

4daae3c (HEAD -> master) minor bug fixes

c527478 Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git checkout login\_feature

Switched to branch 'login\_feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git log --oneline

64f9dcf (HEAD -> login\_feature) Implement login feature ui

c527478 Initial commit

**Whatever the commits you are making in the master it will be there in the master branch only whatever the commits you are making in the branch login it will be there in the login branch only**

**List of all the branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

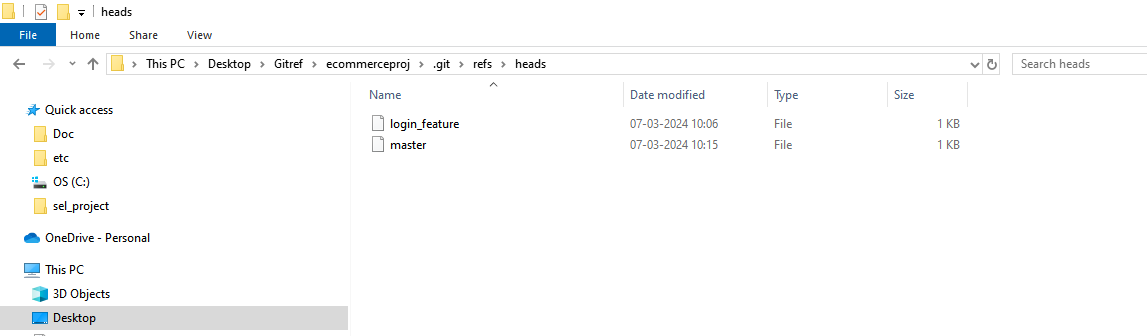
$ git branch

\* login\_feature -> current branch will show the green color

master

#### Rename and Delete branch using -m and -d commands

Branches are nothing but references(refs folder)



**Create one new branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git branch dummybranch

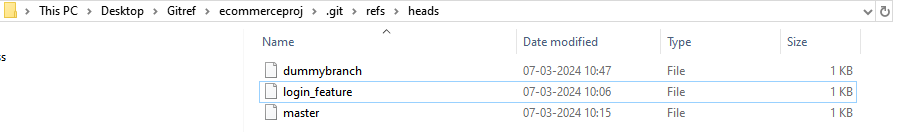
vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git branch

dummybranch

\* login\_feature

master



* + 1. **What are contents present inside the branch ?**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ cat .git/refs/heads/master

4daae3c6324f0ca554b635ad5d288c5d4e735fbc

**Here some hash will be there**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git log

commit 64f9dcf4ee0a2741360c6dc89e2ad45decdbabfb (HEAD -> login\_feature

, dummybranch)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 10:06:07 2024 +0530

Implement login feature ui

commit c527478df7ddf80fa6411b1d4826432b825b46b1

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 09:59:47 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ cat .git/refs/heads/login\_feature

64f9dcf4ee0a2741360c6dc89e2ad45decdbabfb

**See the above hash whatever the commit has it is the head in the login features so the same data is saved here**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ cat .git/refs/heads/dummybranch

64f9dcf4ee0a2741360c6dc89e2ad45decdbabfb

**To change the master branch then check the commit once**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (login\_feature)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git log

commit 4daae3c6324f0ca554b635ad5d288c5d4e735fbc (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 10:15:47 2024 +0530

minor bug fixes

commit c527478df7ddf80fa6411b1d4826432b825b46b1

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 09:59:47 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ cat .git/refs/heads/master

4daae3c6324f0ca554b635ad5d288c5d4e735fbc

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch

dummybranch

login\_feature

\* master

**Here master is shows green color that mean current branch but how come this git able to know that I am present in master branch ?**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ cat .git/HEAD

ref: refs/heads/master

**If I shift the branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git checkout dummybranch

Switched to branch 'dummybranch'

**Know if I check the head file it’s referencing to the dummy branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummybranch)

$ cat .git/HEAD

ref: refs/heads/dummybranch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummybranch)

$ git branch

\* dummybranch

login\_feature

master

**To rename this dummy branch :**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummybranch)

$ git branch -m new\_branch

Automatically the dummy branch renamed to the new\_branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_branch)

$ git branch

login\_feature

master

\* new\_branch

**To delete a branch :**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_branch)

$ git branch -d new\_branch

error: cannot delete branch 'new\_branch' used by worktree at 'C:/Users/v

lab/Desktop/Gitref/ecommerceproj'

What it is trying to tell it is telling that cannot delete the branch because we are already existing in that branch So

Change to other branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_branch)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch -d new\_branch

error: the branch 'new\_branch' is not fully merged

hint: If you are sure you want to delete it, run 'git branch -D new\_bran

ch'

hint: Disable this message with "git config advice.forceDeleteBranch fal

se"

**But it is not fully matched**

**For example**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch delete\_branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch -d delete\_branch

Deleted branch delete\_branch (was 4daae3c).

**It’s automatically deleted so know create the same branch and moved to the delete branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch delete\_branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git checkout delete\_branch

Switched to branch 'delete\_branch'

Modify one file and then commit it

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (delete\_branch)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (delete\_branch)

$ git status

On branch delete\_branch

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: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (delete\_branch)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (delete\_branch)

$ git commit -m "test commit"

[delete\_branch 2474f1b] test commit

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

**Know we are commited into the delete\_branch then move into the master branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (delete\_branch)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch -d delete\_branch

error: the branch 'delete\_branch' is not fully merged

hint: If you are sure you want to delete it, run 'git branch -D delete\_b

ranch'

hint: Disable this message with "git config advice.forceDeleteBranch fal

se"

**It shows same not fully matched that means some changes are done then it shows**

If you are sure you want to delete it, run 'git branch **-D** delete\_b

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch -D delete\_branch

Deleted branch delete\_branch (was 2474f1b).

### Git Merge :

Merging the two branches

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch

login\_feature

\* master

new\_branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git checkout -b new\_feature

Switched to a new branch 'new\_feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_feature)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_feature)

$ vi new.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_feature)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_feature)

$ git commit -m "new file added"

[new\_feature 772a43a] new file added

1 file changed, 1 insertion(+)

create mode 100644 new.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_feature)

$ git checkout master

Switched to branch 'master'

Here master and new\_feature will be merged

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git merge new\_feature

Updating 4daae3c..772a43a

Fast-forward

new.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 new.txt

**It shows to fast forward because there is no commits in the master then it called as fast forward**

**Create another branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ ls

index.txt new.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ vi index.txt

**-am is used to specified both add and message**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git commit -am "indexfile will be changed"

[master 7a8b73c] indexfile will be changed

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch

login\_feature

\* master

new\_branch

new\_feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ ls

dummy.txt index.txt new.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git checkout -b dummy\_branch

Switched to a new branch 'dummy\_branch'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummy\_branch)

$ vi dummy1.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummy\_branch)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummy\_branch)

$ git commit -m "modified one dummy file"

[dummy\_branch 2ca95e6] modified one dummy file

1 file changed, 1 insertion(+)

create mode 100644 dummy1.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummy\_branch)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git commit -m "modifed index.txt file"

[master 32bb04d] modifed index.txt file

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

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git merge dummy\_branch

Merge made by the 'ort' strategy.

dummy1.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 dummy1.txt

**If we commit changes in the dummy branch and master branch , merge it then it’s shows the ort strategy or recursive strategy**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git log

commit aec2d9d72d457c776748c59acb4c1fe12be1d6fa (HEAD -> master)

Merge: 32bb04d 2ca95e6

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 11:40:48 2024 +0530

Merge branch 'dummy\_branch' -> T**his is the new commit after two branches are merged**

commit 32bb04d943db5ed912cbd37d01bd5c78c5957b6d

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 11:40:30 2024 +0530

modifed index.txt file

commit 2ca95e6476f245361230c75d784041229085c638 (dummy\_branch)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 11:39:19 2024 +0530

modified one dummy file

commit 0c30fc7b82275be211ae3f85083c1fbf32464ad5

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 11:33:33 2024 +0530

**If change the same file in the master then conflicts can be occurred**

#### Merge Conflicts :

In the master branch if one file has been changed and also in the branch which you are going to merge into the master the same files will be changed with that branch then the git will not do the automatic merging then conflicts may raised.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir mergeConflicts

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd mergeConflicts

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/mergeC

onflicts/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git commit -m "index file add"

[master (root-commit) 3703683] index file add

1 file changed, 1 insertion(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git log

commit 3703683349bca8d9f76dad5e4cc85ae997b1c6d6 (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 11:54:18 2024 +0530

index file add

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git checkout -b feature/some\_feature

Switched to a new branch 'feature/some\_feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (feature/soature)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (feature/soature)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (feature/soature)

$ git commit -m "changes in index file feature"

[feature/some\_feature 6a09706] changes in index file feature

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (feature/soature)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ ls

index.txt

There is no content will be modified in the master branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ cat index.txt

created one new file

Modify the same file in master branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git commit -m "changes done in the master"

[master bb1e2f5] changes done in the master

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git log

commit bb1e2f553122f756534c152927fae9e9456bb88e (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 12:00:05 2024 +0530

changes done in the master

commit 3703683349bca8d9f76dad5e4cc85ae997b1c6d6

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 11:54:18 2024 +0530

index file add

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git merge feature/some\_feature

Auto-merging index.txt

CONFLICT (content): Merge conflict in index.txt

Automatic merge failed; fix conflicts and then commit the result.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master|MERGING)

$ git status

On branch master

You have unmerged paths.

(fix conflicts and run "git commit")

(use "git merge --abort" to abort the merge)

Unmerged paths:

(use "git add <file>..." to mark resolution)

both modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master|MERGING) -> Here master merging is going on

$ vi index.txt

created one new file

<<<<<<< HEAD

changes done in the master branch

=======

changes done in the feature branch

>>>>>>> feature/some\_feature

~

**Git wiill be automatically added the code in both master commit and branch commit**

**HEAD is currently pointing to the master branch**

**Delete some line whatever we don’t want**

**--abort is used to the stops the ongoing merge process and attempts to restore you per-merge state.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master|MERGING)

$ git merge --abort

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git merge feature/some\_feature

Auto-merging index.txt

CONFLICT (content): Merge conflict in index.txt

Automatic merge failed; fix conflicts and then commit the result.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master|MERGING)

$ vi index.txt

In this file contents have only master commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master|MERGING)

$ cat index.txt

created one new file

changes done in the master branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master|MERGING)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master|MERGING)

$ git commit

[master 6cb5f1c] Merge branch 'feature/some\_feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git log

commit 6cb5f1ce3196f4130d2ae9975640eadcc1b779ee (HEAD -> master)

Merge: bb1e2f5 6a09706

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 12:13:14 2024 +0530

Merge branch 'feature/some\_feature'

commit bb1e2f553122f756534c152927fae9e9456bb88e

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 12:00:05 2024 +0530

changes done in the master

commit 6a097068d60af8752b6d45827c8806ad8539aaf2 (feature/some\_feature)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 11:56:19 2024 +0530

changes in index file feature

commit 3703683349bca8d9f76dad5e4cc85ae997b1c6d6

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 7 11:54:18 2024 +0530

:q

### Git Rebase :

* Rebase is often used as an alternative to merging.
* Rebasing a branch updates one branch with another by applying the commits of one branch on top of the commits of another branch.
* Git rebase is used to clean up our local commit history.
* Merge preserve history
* Rebase doesn’t preserve history

**Do not use Rebase when :**

The branch is public when it is shared to all the developers.

Most of the teams prefer merge over rebase.

**Common places where we use rebases :**

Cleaning up your commits before sharing your branch.

Pulling changes from another branch without merge.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir rebseproj

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd rebseproj

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/rebse

proj/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ vi master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git commit -m "master file added in master"

[master (root-commit) cf71ba5] master file added in master

1 file changed, 1 insertion(+)

create mode 100644 master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git log --oneline

cf71ba5 (HEAD -> master) master file added in master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git branch feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git branch

feature

\* master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ vi master1.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git commit -m "master1 file added in master"

[master 1d32fd1] master1 file added in master

1 file changed, 1 insertion(+)

create mode 100644 master1.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ vi master2.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git commit -m "master 2 file added in master"

[master d852996] master 2 file added in master

1 file changed, 1 insertion(+)

create mode 100644 master2.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git log --oneline

d852996 (HEAD -> master) master 2 file added in master

1d32fd1 master1 file added in master

cf71ba5 (feature) master file added in master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git checkout feature

Switched to branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (feature)

$ vi feature1.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (feature)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (feature)

$ git commit -m "feature 1 added in feature"

[feature 939a275] feature 1 added in feature

1 file changed, 1 insertion(+)

create mode 100644 feature1.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (feature)

$ vi feature2.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (feature)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (feature)

$ git commit -m "feature 2 file addedin feature"

[feature 4174079] feature 2 file addedin feature

1 file changed, 1 insertion(+)

create mode 100644 feature2.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (feature)

$ git log --oneline

4174079 (HEAD -> feature) feature 2 file addedin feature

939a275 feature 1 added in feature

cf71ba5 master file added in master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (feature)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git log --oneline

d852996 (HEAD -> master) master 2 file added in master

1d32fd1 master1 file added in master

cf71ba5 master file added in master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git rebase feature

Successfully rebased and updated refs/heads/master.

Here all commits are shows in master branch only

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebseproj (master)

$ git log --oneline

a2f714d (HEAD -> master) master 2 file added in master

0ea6393 master1 file added in master

4174079 (feature) feature 2 file addedin feature

939a275 feature 1 added in feature

cf71ba5 master file added in master

#### Interactive Rebasing :

Rebase is used to combine the multiple commits to single commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir git\_irebasing

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git\_irebasing/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/git\_irebasing/.gi

t/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (master)

$ vi home.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (master)

$ git add .

i

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (master)

$ git commit -m "home page added"

[master (root-commit) 1da5c81] home page added

1 file changed, 1 insertion(+)

create mode 100644 home.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (master)

$ git checkout -b new\_feature

Switched to a new branch 'new\_feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ git add .

g

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ git commit -m "still working on the gfeature"

[new\_feature ed0c7d9] still working on the gfeature

1 file changed, 1 insertion(+)

create mode 100644 feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ git commit -m "fix logic issues"

[new\_feature 242faca] fix logic issues

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ git commit -m "completed new feature"

[new\_feature f2e5cc5] completed new feature

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_irebasing (new\_feature)

$ git log --oneline

f2e5cc5 (HEAD -> new\_feature) completed new feature

242faca fix logic issues

ed0c7d9 still working on the gfeature

1da5c81 (master) home page added

These three commit are converted into single commit or implemented a new feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature)

$ git rebase -i master

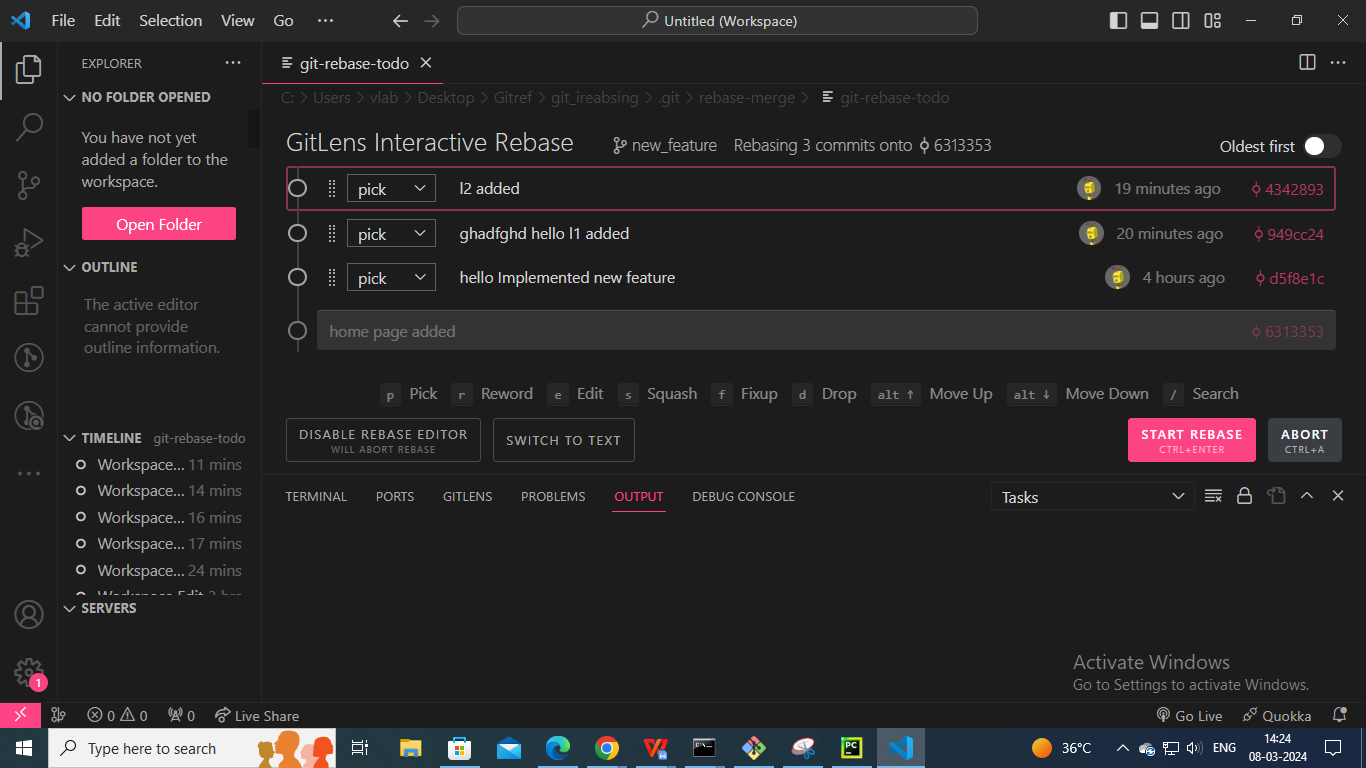
[detached HEAD 8b2d7b5] Implemented new feature

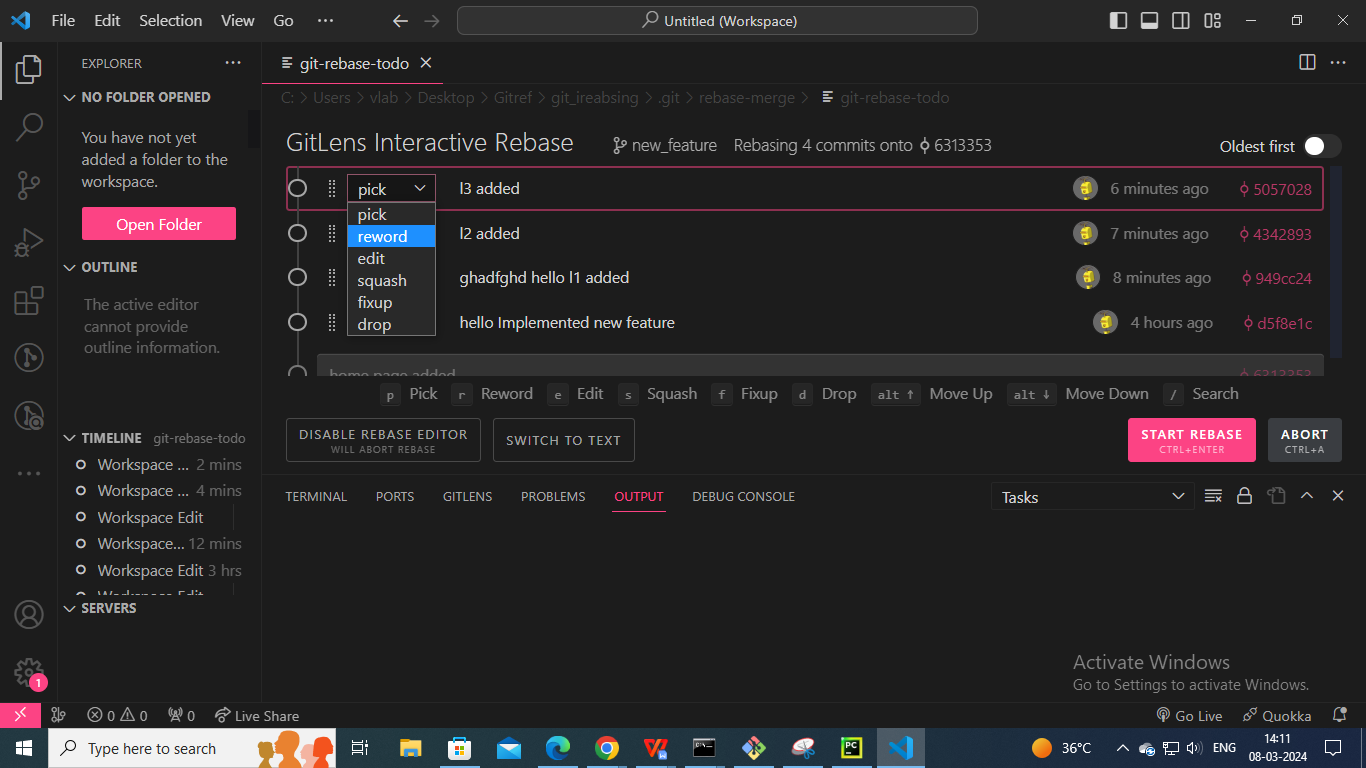
Date: Fri Mar 8 09:59:14 2024 +0530

1 file changed, 4 insertions(+)

create mode 100644 feature.txt

Successfully rebased and updated refs/heads/new\_feature.





**Note :**

* If we see all commits in the present branch , then give the first commit ID git rebase -i 1da5c81
* all the commits we have to delete, but the root commit isn't deleted(To select the drop down box).

#### Options in git rebase

In above figure shows pick drop-down box

* **Reward** : Edit particular commit and combine multiple commits to one commit in the master branch.
* **Drop** : Used to remove the particular commit and also Deleted that file in all stages (working area , staging area and repository).
* **Fixup** : Used to detach the commit in a particular files but these files are not Deleted available in the current branch

* **Edit :** same used as reward but when we close that file then it shows **you can amend the commit with some commands**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature)

$ git rebase -i master

Stopped at d5f8e1c... hello Implemented new feature

You can amend the commit now, with

git commit --amend

Once you are satisfied with your changes, run

git rebase --continue

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature|REBASE 1/2)

$ git commit --amend

[detached HEAD b33020a] hiii Implemented new feature

Date: Fri Mar 8 09:59:14 2024 +0530

1 file changed, 4 insertions(+)

create mode 100644 feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature|REBASE 1/2)

$ ls

feature.txt home.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature|REBASE 1/2)

$ git rebase --continue

Successfully rebased and updated refs/heads/new\_feature.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature)

$ ls

feature.txt home.txt l1.txt l2.txt

* **Squash** : Two combined multiple commits into single commits .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature)

$ ls

feature.txt home.txt l1.txt l2.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature)

$ git rebase -i master

[detached HEAD f1f50a2] l1 and l2 commited

Date: Fri Mar 8 14:03:48 2024 +0530

2 files changed, 4 insertions(+)

create mode 100644 l1.txt

create mode 100644 l2.txt

Successfully rebased and updated refs/heads/new\_feature.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature)

$ git log --oneline

f1f50a2 (HEAD -> new\_feature) l1 and l2 commited

d5f8e1c hello Implemented new feature

6313353 (master) home page added

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git\_ireabsing (new\_feature)

$ git log --oneline

8b2d7b5 (HEAD -> new\_feature) Implemented new feature

6313353 (master) home page added

#### Modify the latest commit by using git rebase

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir change

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd change

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/chang

e/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

index.txt

nothing added to commit but untracked files present (use "git add" to

track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git log

fatal: your current branch 'master' does not have any commits yet

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git commit -m "Initial commit"

[master (root-commit) a91dc3e] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ cat index.txt

hii veena

**If I want change the content in the index.txt file, we have to get another commit to modify so that we can use the initial commit.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git log

commit a91dc3edc629e0cbb13a9aab259e8b613585503e (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Fri Mar 8 10:47:32 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git status

On branch master

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: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ cat index.txt

hello veena

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git show a91dc

commit a91dc3edc629e0cbb13a9aab259e8b613585503e (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Fri Mar 8 10:47:32 2024 +0530

Initial commit

diff --git a/index.txt b/index.txt

new file mode 100644

index 0000000..4b4e84b

--- /dev/null

+++ b/index.txt

@@ -0,0 +1 @@

+hii veena

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git diff --cached

diff --git a/index.txt b/index.txt

index 4b4e84b..c469a19 100644

--- a/index.txt

+++ b/index.txt

@@ -1 +1 @@

-hii veena

+hello veena

If it is local only then use amend it went into the public branch it may confusion

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git commit --amend

[master 229f8a2] Initial commited

Date: Fri Mar 8 10:47:32 2024 +0530

1 file changed, 1 insertion(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git log --oneline

229f8a2 (HEAD -> master) Initial commited

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/change (master)

$ git show 229f8a

commit 229f8a2322ce641462c2e78cc90a1786b0d7d7a8

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Fri Mar 8 10:47:32 2024 +0530

Initial commited

diff --git a/index.txt b/index.txt

new file mode 100644

index 0000000..c469a19

--- /dev/null

+++ b/index.txt

@@ -0,0 +1 @@

+hello veena

#### Conflict :

#### If one file has three commits (a, b, and c), if you want to delete the b commit, then conflict may occur. to rectify that by using rebase --continue after adding and committing that file.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/rebase1/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -m "create one new file"

[master (root-commit) 8a6551d] create one new file

1 file changed, 1 insertion(+)

create mode 100644 f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

8a6551d (HEAD -> master) create one new file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git switch -c branch1

Switched to a new branch 'branch1'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ ls

f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ vi f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git commit -m "created one f2 file"

[branch1 94a59a0] created one f2 file

1 file changed, 1 insertion(+)

create mode 100644 f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ vi f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git commit -m "modify 1 f2"

[branch1 85847fc] modify 1 f2

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

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ vi f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git commit -m "modify 2 f2"

[branch1 444fe35] modify 2 f2

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

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git log --oneline

444fe35 (HEAD -> branch1) modify 2 f2

85847fc modify 1 f2

94a59a0 created one f2 file

8a6551d (master) create one new file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git rebase -i master

Auto-merging f2

CONFLICT (content): Merge conflict in f2

error: could not apply 444fe35... modify 2 f2

hint: Resolve all conflicts manually, mark them as resolved with

hint: "git add/rm <conflicted\_files>", then run "git rebase --continue".

hint: You can instead skip this commit: run "git rebase --skip".

hint: To abort and get back to the state before "git rebase", run "git rebase --

abort".

Could not apply 444fe35... modify 2 f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1|REBASE 3/3)

$ git rebase --continue

f2: needs merge

You must edit all merge conflicts and then

mark them as resolved using git add

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1|REBASE 3/3)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1|REBASE 3/3)

$ git commit -m "conflict fixed"

[detached HEAD b15bfd5] conflict fixed

1 file changed, 6 insertions(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1|REBASE 3/3)

$ git log --oneline

b15bfd5 (HEAD) conflict fixed

94a59a0 created one f2 file

8a6551d (master) create one new file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1|REBASE 3/3)

$ git rebase --abort

**--abort is used to terminate the re basing.**

* + 1. **Examples**

##### Case1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/rebase1/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -am "Created f1"

On branch master

Initial commit

Untracked files:

(use "git add <file>..." to include in what will be committed)

f1

nothing added to commit but untracked files present (use "git add" to track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -am "Created f1"

[master (root-commit) 3ae5c43] Created f1

1 file changed, 1 insertion(+)

create mode 100644 f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -m "modify1 f1"

[master 817f837] modify1 f1

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

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -m "modify2 f1"

[master 5860ec2] modify2 f1

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

5860ec2 (HEAD -> master) modify2 f1

817f837 modify1 f1

3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git rebase -i master

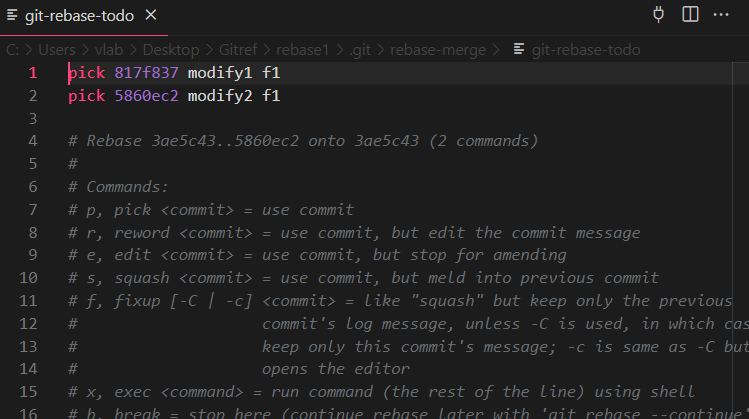
Successfully rebased and updated refs/heads/master.



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git rebase -i 3ae5c43

Successfully rebased and updated refs/heads/master.

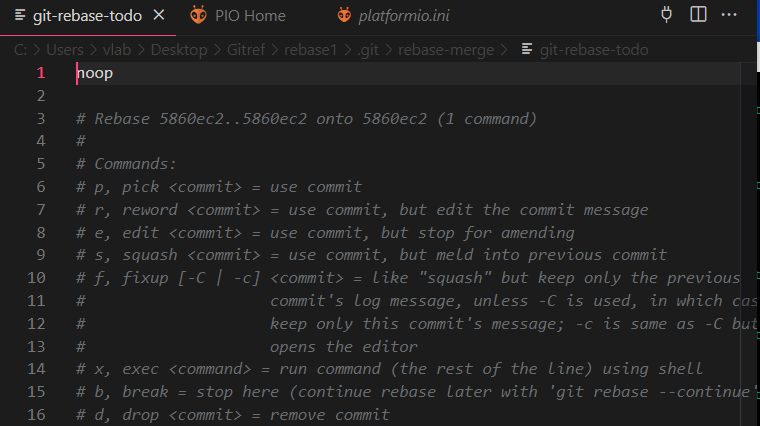


The above figure shows two commits because 3ae5c43 is the first commit. After that commit, there are two new commits and used any one  required pick-outs.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git rebase -i 5860ec2

Successfully rebased and updated refs/heads/master.



The above figure shows noop (no operation) because there are no commits after that commit, and the head will point to that commit only.

##### Case2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline --graph

\* 5860ec2 (HEAD -> master) modify2 f1

\* 817f837 modify1 f1

\* 3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git checkout -b branch1

Switched to a new branch 'branch1'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git log --oneline --graph

\* 5860ec2 (HEAD -> branch1, master) modify2 f1

\* 817f837 modify1 f1

\* 3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git rebase -i master

Successfully rebased and updated refs/heads/branch1.



The above figure shows noop because there are no changes in the currebt branch.

##### Case3

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ vi f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git commit -m "created f2"

[branch1 5faab4f] created f2

1 file changed, 1 insertion(+)

create mode 100644 f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ vi f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git commit -m "modify1 f2"

[branch1 7860294] modify1 f2

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ vi f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git commit -m "modify2 f2"

[branch1 ac2c589] modify2 f2

1 file changed, 1 insertion(+)

Here master will pointed to the modify f1 and branch1 has modify f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git log --oneline

ac2c589 (HEAD -> branch1) modify2 f2

7860294 modify1 f2

5faab4f created f2

5860ec2 (master) modify2 f1

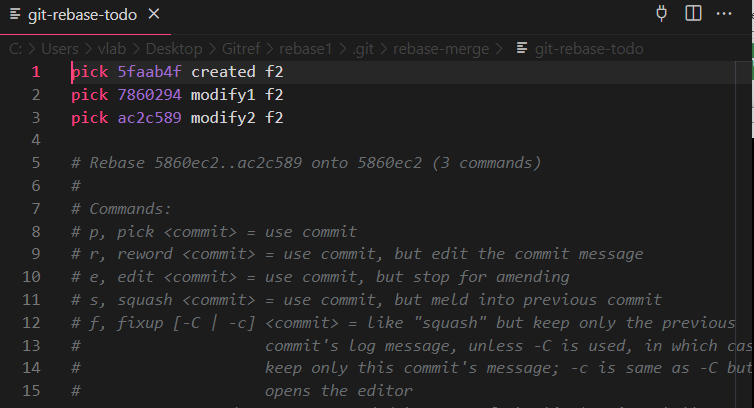
817f837 modify1 f1

3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git rebase -i master

Successfully rebased and updated refs/heads/branch1.



Here master has three commits and switches to branch 1, which has three commits. These three commits were created after master brach commits.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

5860ec2 (HEAD -> master) modify2 f1

817f837 modify1 f1

3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git rebase -i branch1

Successfully rebased and updated refs/heads/master.

In the above command shows the no commit because there is no new commit and also merge .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git log --oneline

ac2c589 (HEAD -> branch1, master) modify2 f2

7860294 modify1 f2

5faab4f created f2

5860ec2 modify2 f1

817f837 modify1 f1

3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ vi f2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git commit -m "again modify f2"

[branch1 5139907] again modify f2

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

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git rebase -i branch1

Successfully rebased and updated refs/heads/master.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

5139907 (HEAD -> master, branch1) again modify f2

ac2c589 modify2 f2

7860294 modify1 f2

5faab4f created f2

5860ec2 modify2 f1

817f837 modify1 f1

3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git log --oneline

5139907 (HEAD -> branch1, master) again modify f2

ac2c589 modify2 f2

7860294 modify1 f2

5faab4f created f2

5860ec2 modify2 f1

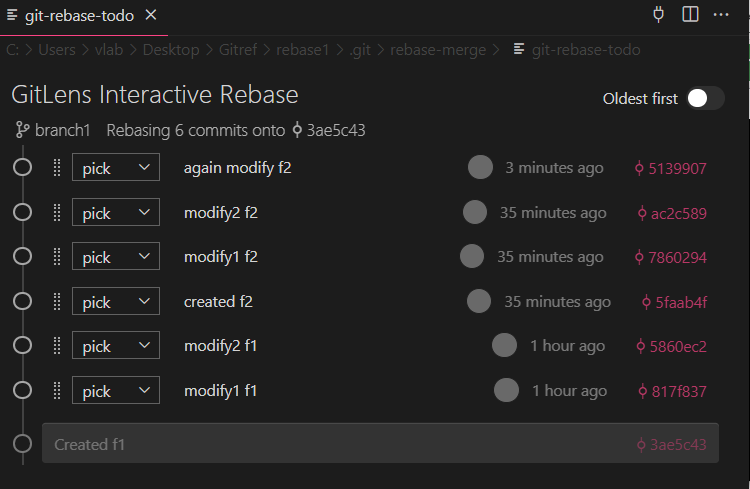
817f837 modify1 f1

3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git rebase -i 3ae5c43

Successfully rebased and updated refs/heads/branch1.



##### Case4

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ ls

f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

9f2b907 (HEAD -> master) add one file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -m "modify f1 file"

[master d3490fd] modify f1 file

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

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -m "modify f1 file"

[master d34c443] modify f1 file

1 file changed, 1 insertion(+), 1 deletion(-)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

d34c443 (HEAD -> master) modify f1 file

d3490fd modify f1 file

9f2b907 add one file

**If use this command all the commits are removed including root commit also**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git update-ref -d HEAD

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

fatal: your current branch 'master' does not have any commits yet

**master branch has no commit, but if we want to change the master branch to another branch, it shows errors. so we can commit the changes or stash them, then only move to another branch.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git switch branch1

error: Your local changes to the following files would be overwritten by

checkout:

f1

Please commit your changes or stash them before you switch branches.

Aborting

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -m "delete"

[master (root-commit) e30514e] delete

1 file changed, 2 insertions(+)

create mode 100644 f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git switch branch1

Switched to branch 'branch1'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git log --oneline

5139907 (HEAD -> branch1) again modify f2

ac2c589 modify2 f2

7860294 modify1 f2

5faab4f created f2

5860ec2 modify2 f1

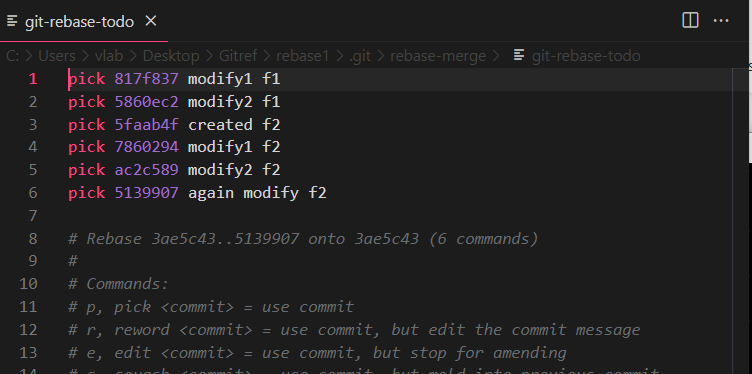
817f837 modify1 f1

3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git rebase -i 817f837^

Successfully rebased and updated refs/heads/branch1.

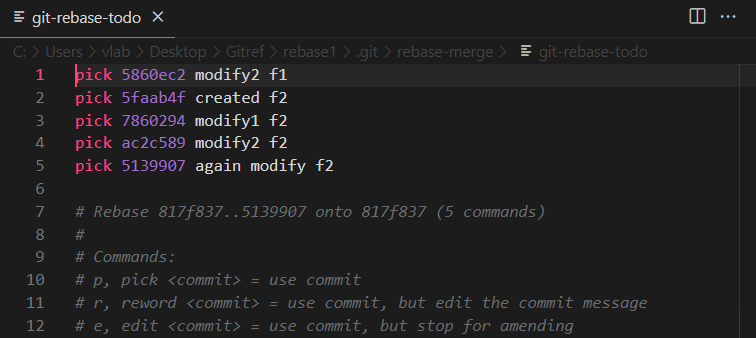


by using the 817f837^ symbol, including that commit also picks out.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git rebase -i 817f837

Successfully rebased and updated refs/heads/branch1.



Without using the ^ symbol, we can't inculcate that particular commit.

##### Case 5 (By using update-ref delete all commit history and root command )

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git log --oneline

5139907 (HEAD -> branch1) again modify f2

ac2c589 modify2 f2

7860294 modify1 f2

5faab4f created f2

5860ec2 modify2 f1

817f837 modify1 f1

3ae5c43 Created f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git update-ref -d HEAD

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (branch1)

$ git log --oneline

fatal: your current branch 'branch1' does not have any commits yet

##### Case6

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

7f3bfe5 (HEAD -> master) delete

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

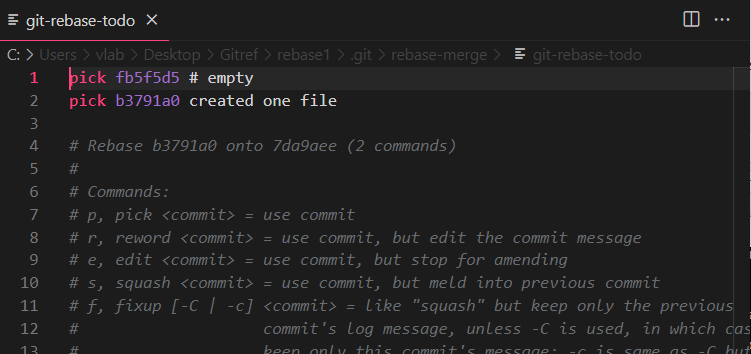
$ git rebase -i --root

Successfully rebased and updated refs/heads/master.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

fb5f5d5 (HEAD -> master)



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git commit -m "created one file"

[master b3791a0] created one file

1 file changed, 1 insertion(+)

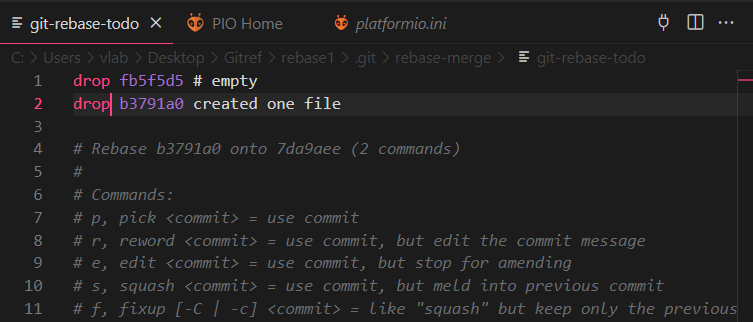
create mode 100644 f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

b3791a0 (HEAD -> master) created one file

fb5f5d5



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git rebase -i --root

Successfully rebased and updated refs/heads/master.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase1 (master)

$ git log --oneline

7da9aee (HEAD -> master)

This is another example. created one file, and that file has one commit that will be removed.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase2 (master)

$ git log --oneline

fba50e0 (HEAD -> master) other commit

30ac329 modify commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase2 (master)

$ git rebase -i --root

Successfully rebased and updated refs/heads/master.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase2 (master)

$ git log --oneline

81b8cc9 (HEAD -> master) other commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase2 (master)

$ git rebase -i --root

Successfully rebased and updated refs/heads/master.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase2 (master)

$ git log --oneline

d6e1b6c (HEAD -> master)

##### Case7

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir rebase

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd rebase

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/rebas

e/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ cd .git/objects/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase/.git/objects (GIT\_DIR!)

$ ls

98/ info/ pack/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase/.git/objects (GIT\_DIR!)

$ cd ../../

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ git commit -m "Initial commit"

[master (root-commit) 1e14086] Initial commit

1 file changed, 2 insertions(+)

create mode 100644 f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ cd .git/objects/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase/.git/objects (GIT\_DIR!)

$ ls

1e/ 2b/ 98/ info/ pack/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase/.git/objects (GIT\_DIR!)

$ cd ../../

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ git log --oneline

1e14086 (HEAD -> master) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ git rebase -i --root

Successfully rebased and updated refs/heads/master.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ git log --oneline

9db2210 (HEAD -> master)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ cd .git/objects/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase/.git/objects (GIT\_DIR!)

$ ls

1e/ 2b/ 98/ 9d/ info/ pack/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase/.git/objects (GIT\_DIR!)

$ cd ../../

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

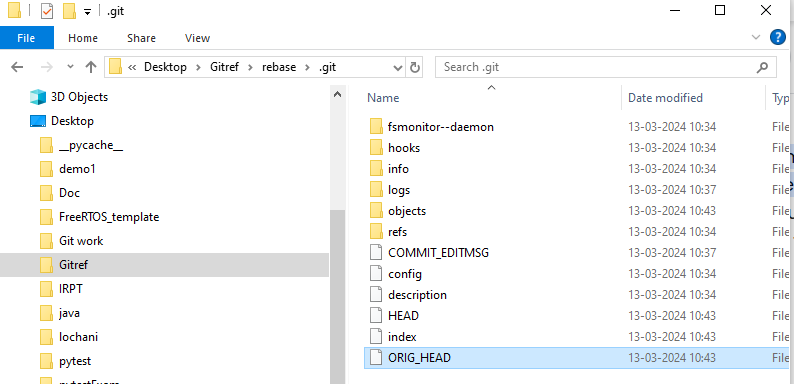
$ git rebase -i --root

Successfully rebased and updated refs/heads/master.

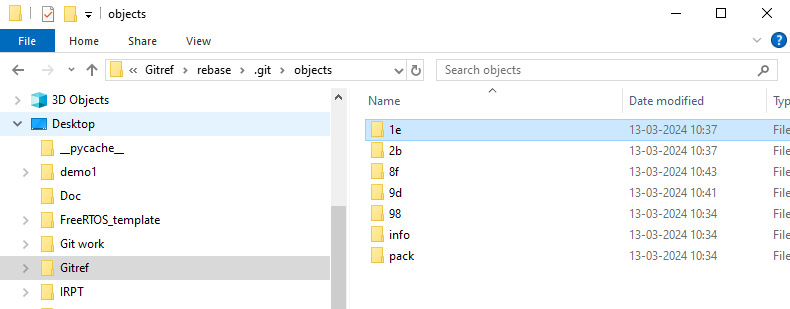
vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/rebase (master)

$ git log --oneline

8fff427 (HEAD -> master)



ORIG\_HEAD. is created by commands that move your HEAD in a drastic way ( git am , git merge , git rebase , git reset ), to record the position of the HEAD before their operation, so that you can easily change the tip of the branch back to the state before you ran them.



##### 3.7.5.8. Case 8( Edit the merge commit )

These are all commits

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/mergeConflicts (master)

$ git log --oneline

4d035e0 (HEAD -> master) added

fccbd0b (feature/some\_feature) modify1

a980527 an updated commit message

8d8a589 modified git

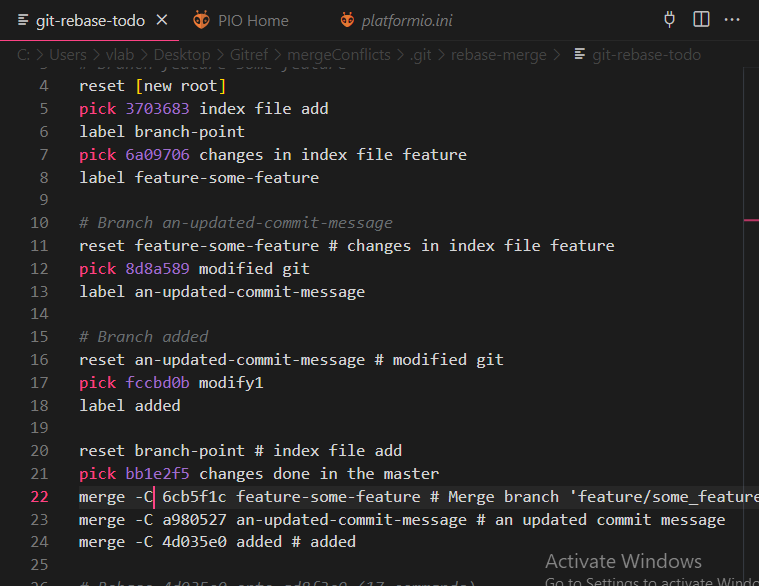
6cb5f1c Merge branch 'feature/some\_feature'

bb1e2f5 changes done in the master

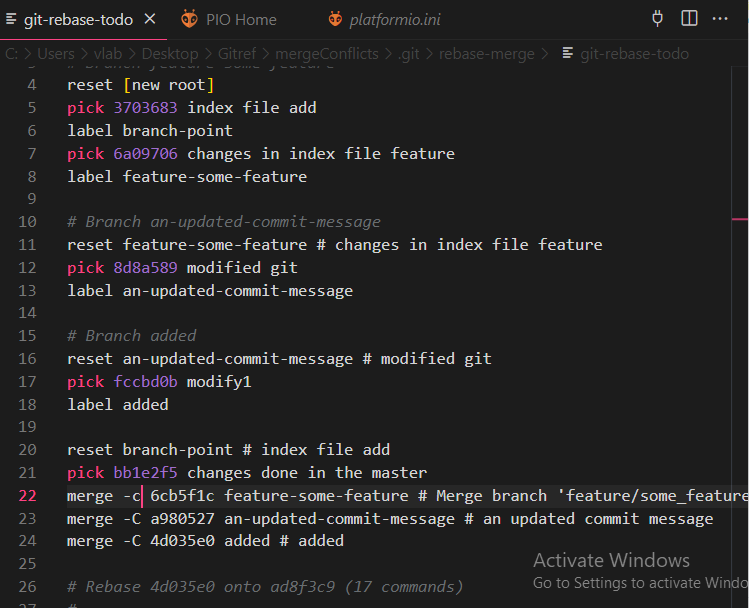
6a09706 changes in index file feature

3703683 index file add

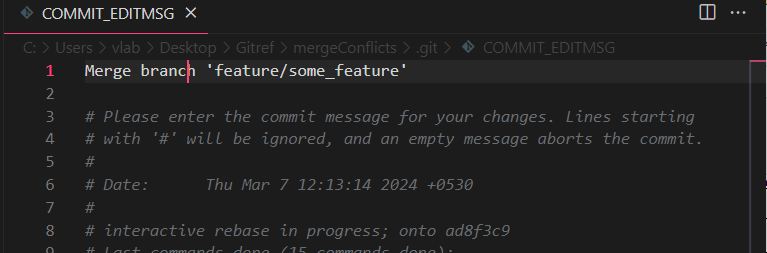
**I want to Edit this merge commit** 6cb5f1c Merge branch 'feature/some\_feature'



Here, whatever commit you want to edit, put a small c, then it becomes edit.



One pop up raised





### GIT Cherry Pick

Cherry Pick is used if more commits are available in one branch and you want to apply particular commit from one branch into another branch.

Cherry pick is mainly used if you don’t want to merge the whole branch and you want some of the commits.

Cherry pick just like rebase it is and advanced concept and also a powerful command.

Cherry pick is a useful tool, but always it is not a good option.

It can cause duplicate commits.

Mainly cherry pick is used for the bug fixes where you want to place that bugfix commit in all the version branches.

It is also used when we accidentally made a commit in wrong branch.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir git-cherrypick

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-cherrypick

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/git-cherrypic

k/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ ls

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git add .

g

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git commit -m "Initial commit"

[master (root-commit) 1c7cd88] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git log --oneline

1c7cd88 (HEAD -> master) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git branch 1.0

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git branch 2.0

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git branch 3.0

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git branch

1.0

2.0

3.0

\* master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git checkout 3.0

Switched to branch '3.0'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ git commit -m "working on the 3.0 feature"

[3.0 e660c81] working on the 3.0 feature

1 file changed, 1 insertion(+)

create mode 100644 feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ git log --oneline

e660c81 (HEAD -> 3.0) working on the 3.0 feature

1c7cd88 (master, 2.0, 1.0) Initial commit

**Here we find one bug in this 3.0 version and also having the same bug in the 1.0 and 2.0 versions.**

**For example :**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ vi bugfix.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ git commit -m "bug fix done"

[3.0 7af3a73] bug fix done

1 file changed, 1 insertion(+)

create mode 100644 bugfix.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ git log --oneline

7af3a73 (HEAD -> 3.0) bug fix done

e660c81 working on the 3.0 feature

1c7cd88 (master, 2.0, 1.0) Initial commit

**Here we fixed the bug and add it to the 1.0 and 2.0 but not merge the 1.0 , 2.0 and 3.0 because already still working in this feature**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (3.0)

$ git checkout 2.0

Switched to branch '2.0'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (2.0)

$ git log --oneline

1c7cd88 (HEAD -> 2.0, master, 1.0) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (2.0)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (2.0)

$ git cherry-pick 7af3a73

[2.0 10551e4] bug fix done

Date: Fri Mar 8 15:53:50 2024 +0530

1 file changed, 1 insertion(+)

create mode 100644 bugfix.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (2.0)

$ ls

bugfix.txt index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (2.0)

$ git log --oneline

10551e4 (HEAD -> 2.0) bug fix done

1c7cd88 (master, 1.0) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (2.0)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (master)

$ git checkout 1.0

Switched to branch '1.0'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (1.0)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (1.0)

$ git log --oneline

1c7cd88 (HEAD -> 1.0, master) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (1.0)

$ git cherry-pick 7af3a73

[1.0 d0ec93d] bug fix done

Date: Fri Mar 8 15:53:50 2024 +0530

1 file changed, 1 insertion(+)

create mode 100644 bugfix.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (1.0)

$ ls

bugfix.txt index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-cherrypick (1.0)

$ git log --oneline

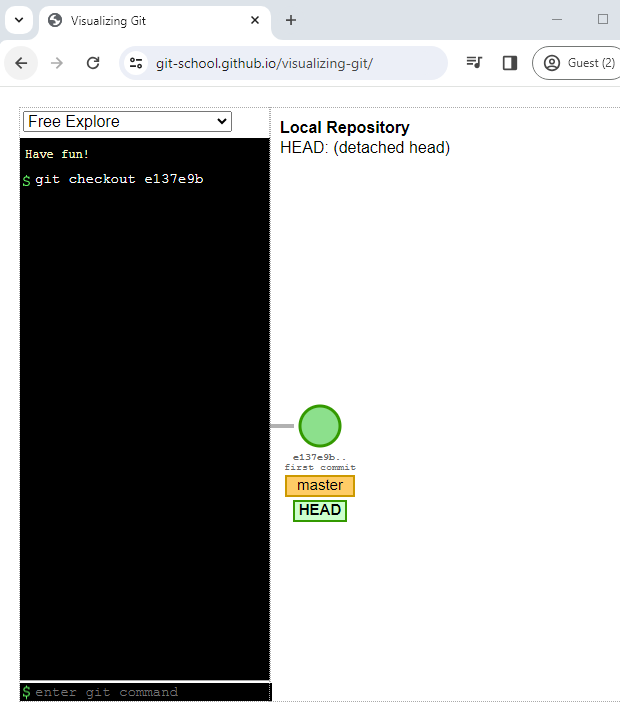
d0ec93d (HEAD -> 1.0) bug fix done

1c7cd88 (master) Initial commit

### CaptureGIT Head :

When we are working we only checkout one branch at a time.This is also called as Head branch.

Git makes note of this branch and store it in .**git/Head as the**

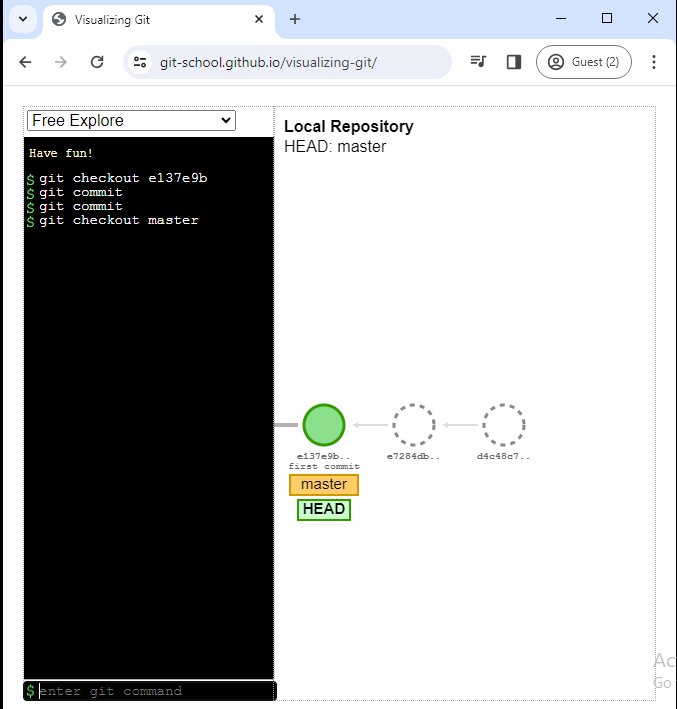
**reference for the path** of the branch.

Head is nothing but the reference to the branch.

Head not only reference a branch it also reference the commit sha1.

If the head points to a specific commit then it is called as detached head.

Now head is pointing to the particular commit that is called detached head.

Here e72 commit doesn’t belongs to any commit but head move that commit and add other commit .

Next checkout the master but how can we go back that commit ?

In project we are not go to these commits because there is no branch is there until unless these hash available then we can go.

So these commits are not belongs to any branch then git automatically delete these commits. These two commits are isolated.

### GIT RESET :

The term reset itself stands for Undoing changes.

Reset does different things in different contexts.

**If you want to move the branch by using these three commands**

1. Commit
2. Merge
3. Rebase

* If you observe, these commands are not explicitly used for moving the branch
* These are moved as a side effect of creating new commit.
* Do we have any command that is specially used only for moving a branch without creating any other commits.
* Reset does this. It is used to move the branch.

###### Why ?

Reset moves the current branch (change one commit to another commit) and optionally copies the data from the repositories to the working or staging areas.

#### Reset has options

* **--hard** : what are changes are done these are moves the files that present in repository area both to working area and staging area
* **--mixed** : moves the files that present in repository area only to staging area not working area(default option)
* **--soft** : does not move the files and files will not move to the working area and staging area.

##### Examples

###### Case1 (by using git hard command delete all the commits of that file)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir git-reset

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-reset

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/git-r

eset/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git commit -m "add index file"

[master (root-commit) 50da083] add index file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git log --oneline

50da083 (HEAD -> master) add index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git commit -m "create feature file"

[master f267781] create feature file

1 file changed, 1 insertion(+)

create mode 100644 feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git log --oneline

f267781 (HEAD -> master) create feature file

50da083 add index file

**Next day I want to working on the feature file only**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git commit -m "working on the feature"

[master b308602] working on the feature

1 file changed, 1 insertion(+), 1 deletion(-)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git log --oneline

b308602 (HEAD -> master) working on the feature

f267781 create feature file

50da083 add index file

**I can realize that this feature is not necessary ( that means not working on that feature or logic is wrong )**

**So that master will moved to the add index file commit.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ ls

feature.txt index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git reset 50da083

**Here two commits are removed but still feature file is available**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git log --oneline

50da083 (HEAD -> master) add index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

feature.txt

nothing added to commit but untracked files present (use "git add" to

track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ ls

feature.txt index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git log --oneline

50da083 (HEAD -> master) add index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git reset 50da083 --hard

HEAD is now at 50da083 add index file

**Still feature is there because already we moved**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ ls

feature.txt index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

feature.txt

nothing added to commit but untracked files present (use "git add" to

track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git commit -m "feature files added"

[master 022812d] feature files added

1 file changed, 1 insertion(+)

create mode 100644 feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git log --oneline

022812d (HEAD -> master) feature files added

50da083 add index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git reset 50da083 --hard

HEAD is now at 50da083 add index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

nothing to commit, working tree clean

**Here feature file will be removed**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ ls

index.txt

###### Case2 (To remove the staging into working area by using mixed option)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

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: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

* + - 1. **To remove from staging area into working area**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git reset head --mixed

Unstaged changes after reset:

M index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

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: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

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: index.txt

**Index.txt file is available both the working area and staging area**

**By using this reset head --hard command delete the file from both staging area and working area**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git reset head --hard

HEAD is now at 50da083 add index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git log --oneline

50da083 (HEAD -> master) add index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-reset (master)

$ cat index.txt

**Nothing is there in the index.txt file**

### GIT STASH :

Sometimes you want to switch the branches, but you are working on an incomplete part of your current project.

You don’t want to make a commit of half-done work.Git stashing allows you to do so.

Normally when you switch branch you will commit the code and switch to the new branch.

If you switch branch without committing.

Two things will happen :

* Switches to the branch carrying the changes.
* Git will not allow the branch and asks to commit or stash the changes.

The git stash command enables you to switch branches without committing the current branch.

The stash meaning is store something safely in a hidden place. The sense in Git is also the same for stash.Git temporarily saves your data safely without committing.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir git-stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/git

-stash/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git commit -m "Initial commit index.txt"

[master (root-commit) 53fd024] Initial commit index.txt

1 file changed, 1 insertion(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout -b feature

Switched to a new branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git status

On branch feature

Untracked files:

(use "git add <file>..." to include in what will be committed)

feature.txt

nothing added to commit but untracked files present (use "git add" t

o track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Untracked files:

(use "git add <file>..." to include in what will be committed)

feature.txt

nothing added to commit but untracked files present (use "git add" t

o track)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout feature

Switched to branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git commit -m "feature file added"

[feature 7b695b5] feature file added

1 file changed, 1 insertion(+)

create mode 100644 feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git log --oneline

7b695b5 (HEAD -> feature) feature file added

cda0e0b (master) Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git add .

g

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git commit -m "Index change1"

[master ff5f5f0] Index change1

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout -

Switched to branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git log --oneline

7b695b5 (HEAD -> feature) feature file added

cda0e0b Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat index.txt

created index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git checkout -

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

**Modify the index.txt file**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout -

Switched to branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git status

On branch feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat index.txt

created index file

Change 2 in feature file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git checkout -

error: Your local changes to the following files would be overwritte

n by checkout:

index.txt

Please commit your changes or stash them before you switch branches.

Aborting

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash

Saved working directory and index state WIP on feature: 7b695b5 feat

ure file added

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat index.txt

created index file

* + 1. **To know previous commit name and stash id**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: WIP on feature: 7b695b5 feature file added

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout feature

Switched to branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git status

On branch feature

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: WIP on feature: 7b695b5 feature file added

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash pop

On branch feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

Dropped refs/stash@{0} (e004d53187dd7e4951cbee806d1ee4eed12ed3df)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat index.txt

created index file

change 2 in feature

**Nothing is there in stash list**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

* + 1. **Custom name also given to the stash**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash save "modify index"

Saved working directory and index state On feature: modify index

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: modify index

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash pop

On branch feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

Dropped refs/stash@{0} (55167b1d9e2dbfdf61ddf0a94d4055ef747bcf29)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

**I don’t want to delete it from the stash list but I want to apply the feature branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash save "modify Index stash"

Saved working directory and index state On feature: modify Index stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: modify Index stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash apply

On branch feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: modify Index stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat index.txt

created index file

change 2 in feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat feature.txt

created feature file

**So that I will remove everything**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git reset head --hard

HEAD is now at 7f99c4b feature work WIP

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat index.txt

created index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat feature.txt

created feature file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git checkout -

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git stash apply

Auto-merging index.txt

CONFLICT (content): Merge conflict in index.txt

On branch master

Unmerged paths:

(use "git restore --staged <file>..." to unstage)

(use "git add <file>..." to mark resolution)

both modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

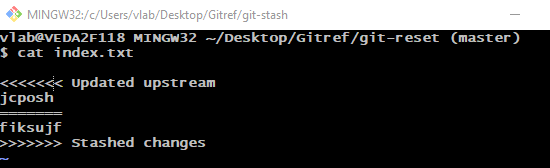
vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

****

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git stash list

stash@{0}: On feature: modify Index stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git stash

index.txt: needs merge

error: could not write index

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Unmerged paths:

(use "git restore --staged <file>..." to unstage)

(use "git add <file>..." to mark resolution)

both modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

* + 1. **Stash pop :**

the git stash pop command removes or throws away the latest or the topmost stash.

* + 1. **Stash apply :**

The git stash apply command leaves the topmost stash on the stash list so that we can later use it.

###### Handle multiple stash :

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git stash list

stash@{0}: On feature: modify Index stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout -

index.txt: needs merge

error: you need to resolve your current index first

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Unmerged paths:

(use "git restore --staged <file>..." to unstage)

(use "git add <file>..." to mark resolution)

both modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git commit -m "Index changes committed"

[master f8af027] Index changes committed

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout -

Switched to branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: modify Index stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git status

On branch feature

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ ls

feature.txt index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat index.txt

created index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat feature.txt

created feature file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat feature.txt

created feature file

change 1 in feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git status

On branch feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: feature.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash save "feature file modified"

Saved working directory and index state On feature: feature file mod

ified

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: feature file modified

stash@{1}: On feature: modify Index stash

**Here recent changes will apply but know I don’t want these changes**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash apply

On branch feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: feature.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git reset head --hard

HEAD is now at 7f99c4b feature work WIP

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat feature.txt

created feature file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git status

On branch feature

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: feature file modified

stash@{1}: On feature: modify Index stash

**How to get changes from the stash to the index.txt file**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash apply stash@{1}

On branch feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

To remove this all

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: feature file modified

stash@{1}: On feature: modify Index stash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git reset head --hard

HEAD is now at 7f99c4b feature work WIP

**Before apply stash I check the changes**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash show

feature.txt | 1 +

1 file changed, 1 insertion(+)

**To check the changes of the stashed data before pulling**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash show -p

diff --git a/feature.txt b/feature.txt

index 8ccff23..7145116 100644

--- a/feature.txt

+++ b/feature.txt

@@ -1 +1,2 @@

created feature file

+change 1 in feature

**To see the changes for the second one**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash show stash@{1} -p

diff --git a/index.txt b/index.txt

index 76ac964..a874a41 100644

--- a/index.txt

+++ b/index.txt

@@ -1 +1,2 @@

created index file

+change 2 in feature

**To see the changes for the first one**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash show stash@{0} -p

diff --git a/feature.txt b/feature.txt

index 8ccff23..7145116 100644

--- a/feature.txt

+++ b/feature.txt

@@ -1 +1,2 @@

created feature file

+change 1 in feature

* + 1. **To delete one stash**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: feature file modified

stash@{1}: On feature: modify stash

**Here pop is used to delete automatically and apply but apply it will add to the working area and not deleted.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash drop stash@{1}

Dropped stash@{1} (0cb5dfa979418724e73620fb5053b12f1135535e)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash show stash@{1} -p

fatal: log for 'stash' only has 1 entries

* + - 1. **By using this drop automatically stash will be deleted**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: feature file modified

**If we use only drop then deleted the recent commit**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash drop

Dropped refs/stash@{0} (5e763700d70676b3a4e4fb46f6258eb7ed7d1c0b)

* + - 1. **By using stash clear to clear the stash list**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash clear

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

**To modify the feature.txt file**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ vi feature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat feature.txt

created feature file

change 1 in feature1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash save "feature file changes"

Saved working directory and index state On feature: feature file cha

nges

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash list

stash@{0}: On feature: feature file changes

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat feature.txt

created feature file

* + - 1. **To create new branch with stashed data**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git stash branch new\_feature stash@{0}

Switched to a new branch 'new\_feature'

On branch new\_feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: feature.txt

no changes added to commit (use "git add" and/or "git commit -a")

Dropped stash@{0} (f0c6921233456d37d5cb42a105c5a51f6f0fe07c)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (new\_feature)

$ git status

On branch new\_feature

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: feature.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (new\_feature)

$ cat feature.txt

created feature file

change 1 in feature1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (new\_feature)

$ git branch

feature

master

\* new\_feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (new\_feature)

$ git checkout feature

M feature.txt

Switched to branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ cat feature.txt

created feature file

change 1 in feature1

### Git checkout

1. **Moves from one branch to other**
2. **Creates new branch if not existed and moves head to that branch**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-stash/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (new\_feature)

$ git branch

feature

master

\* new\_feature

**Here one branch will be created and head will pointed by using -b option**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (new\_feature)

$ git checkout -b feature1

Switched to a new branch 'feature1'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature1)

$ git branch

feature

\* feature1

master

new\_feature

1. **Also not only branch it also shifts to particular commit hash .**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature1)

$ git log --oneline

7b695b5 (HEAD -> feature1, new\_feature, feature) feature file added

cda0e0b Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature1)

$ git checkout cda0e0b

error: Your local changes to the following files would be overwritte

n by checkout:

feature.txt

Please commit your changes or stash them before you switch branches.

Aborting

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature1)

$ git status

On branch feature1

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: feature.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature1)

$ git commit -am "change1 feature file"

[feature1 0c990b5] change1 feature file

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

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature1)

$ git checkout cda0e0b

Note: switching to 'cda0e0b'.

You are in 'detached HEAD' state. You can look around, make experime

ntal

changes and commit them, and you can discard any commits you make in

this

state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you

may

do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead

to false

HEAD is now at cda0e0b Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash ((cda0e0b...))

$

#### Detached Head :

1. **You can leave and go back to the branch by re-attaching the head.**

**Git checkout master**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash ((cda0e0b...))

$ git checkout -

Previous HEAD position was cda0e0b Initial commit index.txt

Switched to branch 'feature1'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature1)

$ git checkout cda0e0b

Note: switching to 'cda0e0b'.

You are in 'detached HEAD' state. You can look around, make experime

ntal

changes and commit them, and you can discard any commits you make in

this

state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you

may

do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead

to false

HEAD is now at cda0e0b Initial commit index.txt

1. **Create a new branch and switch to it. You can now make and save changes since HEAD is no longer detached**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash ((cda0e0b...))

$ git checkout -b new\_branch

Switched to a new branch 'new\_branch'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (new\_branch)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git log --oneline

f8af027 (HEAD -> master) Index changes committed

ff5f5f0 Index change1

cda0e0b (new\_branch) Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git commit -m "created f1 file"

[master 03eb663] created f1 file

1 file changed, 1 insertion(+)

create mode 100644 f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git log --oneline

03eb663 (HEAD -> master) created f1 file

f8af027 Index changes committed

ff5f5f0 Index change1

cda0e0b (new\_branch) Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat f1

created f1 file

**Here head~2 2 is the list of second commit head will pointed**

**Means without knowing the hash we can move the commit one step back**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout Head~2

Note: switching to 'Head~2'.

You are in 'detached HEAD' state. You can look around, make experime

ntal

changes and commit them, and you can discard any commits you make in

this

state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you

may

do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead

to false

HEAD is now at ff5f5f0 Index change1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash ((ff5f5f0...))

$ git log --oneline

ff5f5f0 (HEAD) Index change1

cda0e0b (new\_branch) Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash ((ff5f5f0...))

$ git checkout master

Previous HEAD position was ff5f5f0 Index change1

Switched to branch 'master'

1. **Modify index.txt and master.txt both files are reset**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ ls

f1 index.txt master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

change 1 in master-checkout

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat master.txt

created master file

change 1 master-checkout

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

modified: index.txt

modified: master.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git reset head --hard

HEAD is now at ea5f956 master file added

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat master.txt

created master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

nothing to commit, working tree clean

1. **I need index.txt changes only but I don’t want master.txt changes**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

change 1 in master checkout

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat master.txt

created master file

change 1 in master checkout

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working direct

ry)

modified: index.txt

modified: master.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout head index.txt

Updated 1 path from 9578524

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: master.txt

no changes added to commit (use "git add" and/or "git commit -a")

**So that git checkout also used to discard changes present in the current stage and you can revert the changes of the particular file using**

**Git checkout Head <filename>**

**Or**

**Git checkout -- <filename>**

**Whereas reset is used to discard all changes in the available current stage files.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout -- master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

nothing to commit, working tree clean

* + 1. **List of all branches**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git branch

feature

feature1

\* master

new\_branch

new\_feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout feature

Switched to branch 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git switch feature

Already on 'feature'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (feature)

$ git switch master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git switch -c new\_switch

Switched to a new branch 'new\_switch'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (new\_switch)

$ git switch master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git log --oneline

ea5f956 (HEAD -> master, new\_switch) master file added

03eb663 created f1 file

f8af027 Index changes committed

ff5f5f0 Index change1

cda0e0b (new\_branch) Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout Head~2

Note: switching to 'Head~2'.

You are in 'detached HEAD' state. You can look around, make experim

ental

changes and commit them, and you can discard any commits you make i

n this

state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, yo

u may

do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead

to false

HEAD is now at f8af027 Index changes committed

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash ((f8af027...))

$ git checkout -

Previous HEAD position was f8af027 Index changes committed

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git switch f8af027

fatal: a branch is expected, got commit 'f8af027'

hint: If you want to detach HEAD at the commit, try again with the

--detach option.

**The difference between the checkout and switch is that by using the checkout to change the head point by commit hash, the switch doesn’t work.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

change 1 in master checkout

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

change 1 in master checkout

change 2 in master checkout

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directo

ry)

modified: index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git checkout head index.txt

Updated 1 path from 9578524

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

modified: index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git restore --staged index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working direct

ory)

modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

change 1 in master feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git restore index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

nothing to commit, working tree clean

**Checkout is used to remove all changes at once, and restore is used to remove them from staging and also from the working area by using two commands they are :**

**Git restore --staged <file-name>**

**Git restore <file-name>**

* + 1. **By using the restore command, we get changes at a particular commit.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git log --oneline

ea5f956 (HEAD -> master, new\_switch) master file added

03eb663 created f1 file

f8af027 Index changes committed

ff5f5f0 Index change1

cda0e0b (new\_branch) Initial commit index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git restore --source Head~3 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working direct

ory)

modified: index.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git restore index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ git status

On branch master

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-stash (master)

$ cat index.txt

created index file

change 1

change 2 in feature

### Git revert :

Difference between the revert and reset

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir git-revert

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-revert

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ ls

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ vi master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat index.txt

created index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

created master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/gi

t-revert/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git add .

gi

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git commit -m "create index and master file"

[master (root-commit) 4c79f69] create index and master file

2 files changed, 2 insertions(+)

create mode 100644 index.txt

create mode 100644 master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

4c79f69 (HEAD -> master) create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat index.txt

created index file

file changes 1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git commit -a -m "index file change 1"

[master a19e52a] index file change 1

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

a19e52a (HEAD -> master) index file change 1

4c79f69 create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ vi master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

created master file

master file change 1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git commit -a -m "master file change 1"

[master 3bca99c] master file change 1

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git commit -a -m "index file change 2"

[master 5e32ce0] index file change 2

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ vi master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git commit -a -m "master file change 2"

[master 372a7bd] master file change 2

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat index.txt

created index file

file changes 1

file changes 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

created master file

master file change 1

file changes 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

372a7bd (HEAD -> master) master file change 2

5e32ce0 index file change 2

3bca99c master file change 1

a19e52a index file change 1

4c79f69 create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git reset --hard Head~1

HEAD is now at 5e32ce0 index file change 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

5e32ce0 (HEAD -> master) index file change 2

3bca99c master file change 1

a19e52a index file change 1

4c79f69 create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

created master file

master file change 1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git reset --hard 372a7bd

HEAD is now at 372a7bd master file change 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

372a7bd (HEAD -> master) master file change 2

5e32ce0 index file change 2

3bca99c master file change 1

a19e52a index file change 1

4c79f69 create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

created master file

master file change 1

file changes 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git revert 372a7bd

[master d339579] Revert "master file change 2"

1 file changed, 1 deletion(-)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

created master file

master file change 1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

d339579 (HEAD -> master) Revert "master file change 2"

372a7bd master file change 2

5e32ce0 index file change 2

3bca99c master file change 1

a19e52a index file change 1

4c79f69 create index and master file

372a7bd master file change 2 these changes are removed and commit commit will be created.

Reset is used at when our commit are in local only then we use reset so when these all commits are went into the public remote other developers or collaborator are used and they pull there commits by using reset there will be a chance of losing the data.

If I try to reset the changes of master file changes 1 by using reset index file changes 2, then master file changes 2 go away.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git reset --hard Head~1

HEAD is now at 372a7bd master file change 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

372a7bd (HEAD -> master) master file change 2

5e32ce0 index file change 2

3bca99c master file change 1

a19e52a index file change 1

4c79f69 create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git reset --hard a19e52a

HEAD is now at a19e52a index file change 1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

a19e52a (HEAD -> master) index file change 1

4c79f69 create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat index.txt

created index file

file changes 1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

created master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git reset --hard 372a7bd

HEAD is now at 372a7bd master file change 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

372a7bd (HEAD -> master) master file change 2

5e32ce0 index file change 2

3bca99c master file change 1

a19e52a index file change 1

4c79f69 create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

created master file

master file change 1

file changes 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat index.txt

created index file

file changes 1

file changes 2

So that only particular commit, which is master file change 1, is changed by using revert.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

372a7bd (HEAD -> master) master file change 2

5e32ce0 index file change 2

3bca99c master file change 1

a19e52a index file change 1

4c79f69 create index and master file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git revert 3bca99c

Auto-merging master.txt

CONFLICT (content): Merge conflict in master.txt

error: could not revert 3bca99c... master file change 1

hint: After resolving the conflicts, mark them with

hint: "git add/rm <pathspec>", then run

hint: "git revert --continue".

hint: You can instead skip this commit with "git revert --skip".

hint: To abort and get back to the state before "git revert",

hint: run "git revert --abort".

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master|REVERTING)

$ vi master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master|REVERTING)

$ git commit -a -m "revert commit changes 1"

[master 6179436] revert commit changes 1

1 file changed, 2 deletions(-)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat master.txt

Created master file

file changes 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ cat index.txt

created index file

file changes 1

file changes 2

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-revert (master)

$ git log --oneline

6179436 (HEAD -> master) revert commit changes 1

372a7bd master file change 2

5e32ce0 index file change 2

3bca99c master file change 1

a19e52a index file change 1

4c79f69 create index and master file

## GITHUB

* Github is a hosting platform for the git repositories.
* Github allows us to host our git repository in the cloud.
* This provides that we can access the code from anywhere and also share the code with people around the world.

### Git Vs GitHub

* Git has huge difference with github
* Git is a version control software that runs locally on your machine.
* You don’t need to register for an account.
* You don’t need interenet to use it.
* Github is a service that hosts repositories in the cloud and makes it easier to collaborate with other people.
* You do need to sign up for an account to use githun
* Its and online place to share work that is done using git.

There are so many tools that provide similar hosting and collaboration features

Gitlab

Bitbucket

Gerrit

### Why we need to use Github ?

One main thing is collaboration.

If you ever plan on working on a project with at least on other person, github will make it easier

It is the world largest host of source code , github is free and the basic free tier allows for unlimited public and private repos and unlimited collaborators and more.

Open source project

Exposure

Your github profile showcases your projects ans also the contribution to other projects

It can acts as a sort of resume that many employers will consult in the hiring.

Additionally you can gain some friends and discuss by contributing to popular projects.

### GIT Cloning

* Upto now we had worked on the local git repositories.
* But to work with the remote repositories hosted in the github we need to get the local copy into the computer.
* For that we need is URL that we can tell git to clone the repository.
* If you got the documentation
* Git clone gets the repository that is not present in your machine based on the url we provide.
* To clone a repo

Git clone <url>

* Git will retrieve all the files associated with the repository and will copy them to your local machine
* In addition to that git initializes a new repository on your machine giving you access to the fill history of the cloned projects.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir cloning

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd cloning

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning (master)

$ ls

If you want the clone repository, then the present directory is not the repository.

Open the github account and go to one repository, and then click on the code to get the https and ssh link. Copy the https link.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning (master)

$ git clone https://github.com/vara221/sw.git

Cloning into 'sw'...

remote: Enumerating objects: 29, done.

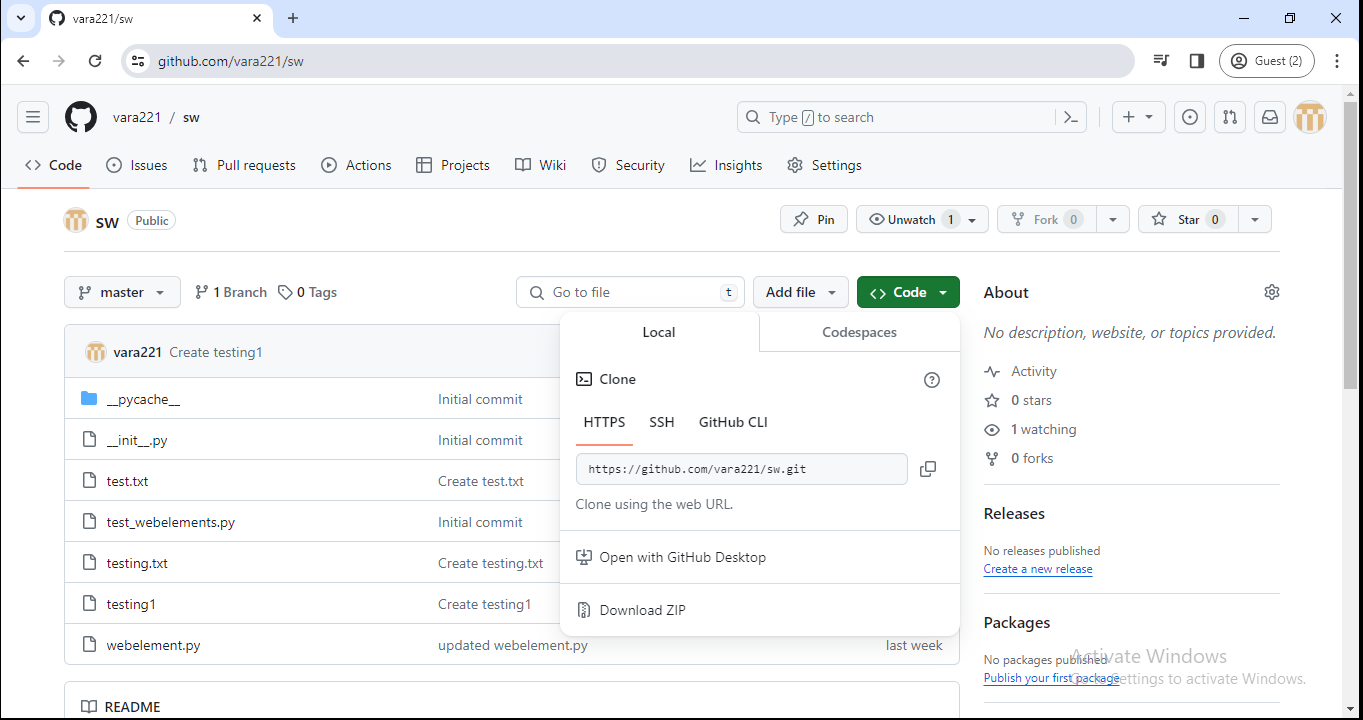
remote: Counting objects: 100% (29/29), done.

remote: Compressing objects: 100% (19/19), done.

remote: Total 29 (delta 9), reused 19 (delta 6), pack-reused 0

Receiving objects: 100% (29/29), 11.78 KiB | 753.00 KiB/s, done.

Resolving deltas: 100% (9/9), done.



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning (master)

$ ls

sw/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning (master)

$ cd sw

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ ls

\_\_init\_\_.py test.txt testing.txt webelement.py

\_\_pycache\_\_/ test\_webelements.py testing1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ git status

On branch master

Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ git log --oneline

d211afe (HEAD -> master, origin/master, origin/HEAD) Create testing

1

524debf Merge branch 'master' of https://github.com/vara221/sw

6904d96 Create test.txt

52d15f8 Merge remote-tracking branch 'origin/master'

ef5322d Create testing.txt

2910fb9 updated webelement.py

d000009 Initial commit

* Any one can clone a repository from github , provided the repo is public.
* You do not need to be owner or collaborator to clone the repo locally to your machine.
* You just need the url from github.
* Pushing up your changes to the github repo involves another process.you need permissions to do that.

### Github with SSH

We can connect to github using two methods

* Using HTTPS : Every time it will ask email and password
* Using SSH : It having ssh key

#### How to setup the ssh ?

When working with a Github repository , you’ll often need to identify yourself to Github using your username and password.

An SSH key is an alternate way to identify yourself that doesn’t require you to enter your username and password every time.

Using the SSH protocol , you can connect and authenticate to remote servers and services.

With SSH keys,you can connect to GitHub without supplying your username and personal access token at each visit.

#### To implement SSH keys in the system

We need to create the SSH key pair using

ssh-keygen -t ed25519 -C [“your\_email@example.com”](mailto:\“your_email@example.com\”)

Next add your SSH key to the ssh agent

Ensure the ssh-agent is running using command

eval `ssh-agent -s`

Next add the ssh key too the ssh agent

Ssh-add ~/.ssh/id\_ed25519

vlab@VEDA2F118 MINGW32 ~ (master)

$ cd Desktop/

vlab@VEDA2F118 MINGW32 ~/Desktop (master)

$ cd Gitref/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ ssh-keygen -t ed25519 -C "varalaxmiganta066@gmail.com"

Generating public/private ed25519 key pair.

Enter file in which to save the key (/c/Users/vlab/.ssh/id\_ed25519):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /c/Users/vlab/.ssh/id\_ed25519

Your public key has been saved in /c/Users/vlab/.ssh/id\_ed25519.pub

The key fingerprint is:

SHA256:mwfPe95GkpQPdzFU6wtBmijZQ/gZqWnuLggJE60vFSQ varalaxmiganta066@gmail.com

The key's randomart image is:

+--[ED25519 256]--+

|Eo. ... ...o|

|..o .+o. + o.|

| o . o++oo o .o|

|+ . +.o. + + .|

|.+. o S . \* o |

|.o. . \* o + .|

| .. . . o + o . |

| . . . . .... |

| o. .o... |

+----[SHA256]-----+

To know the while running the ssh key are not

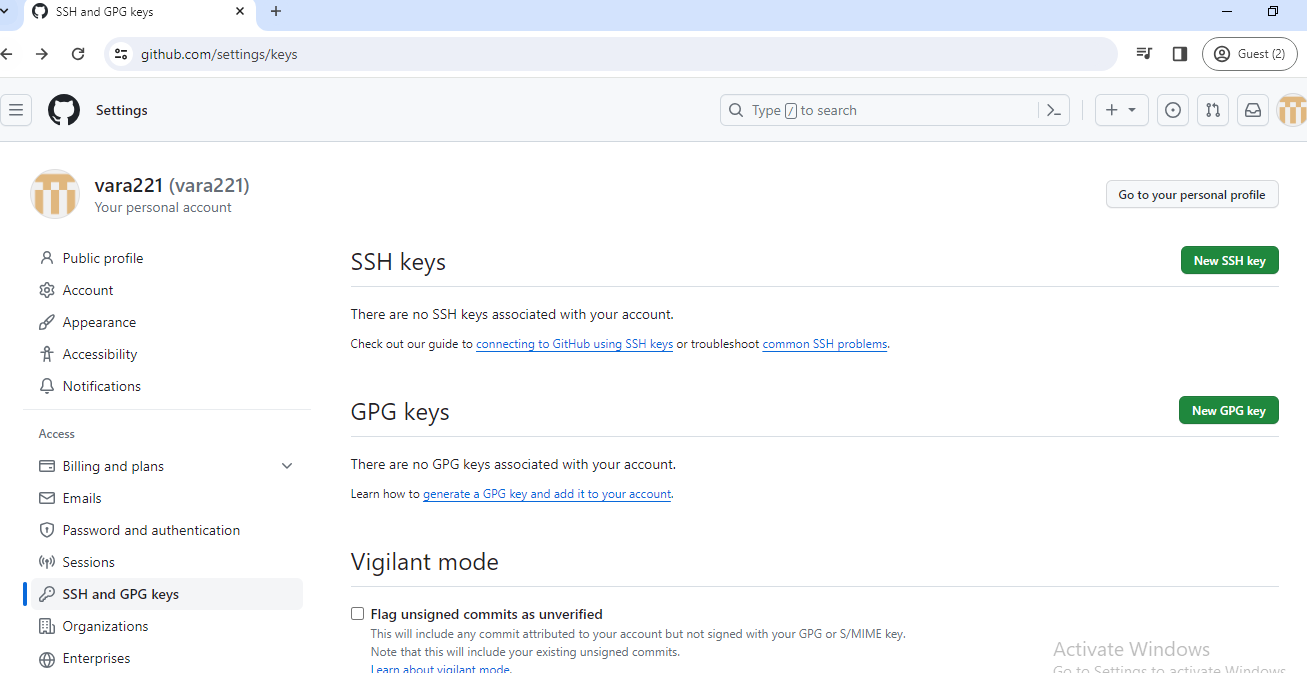
vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

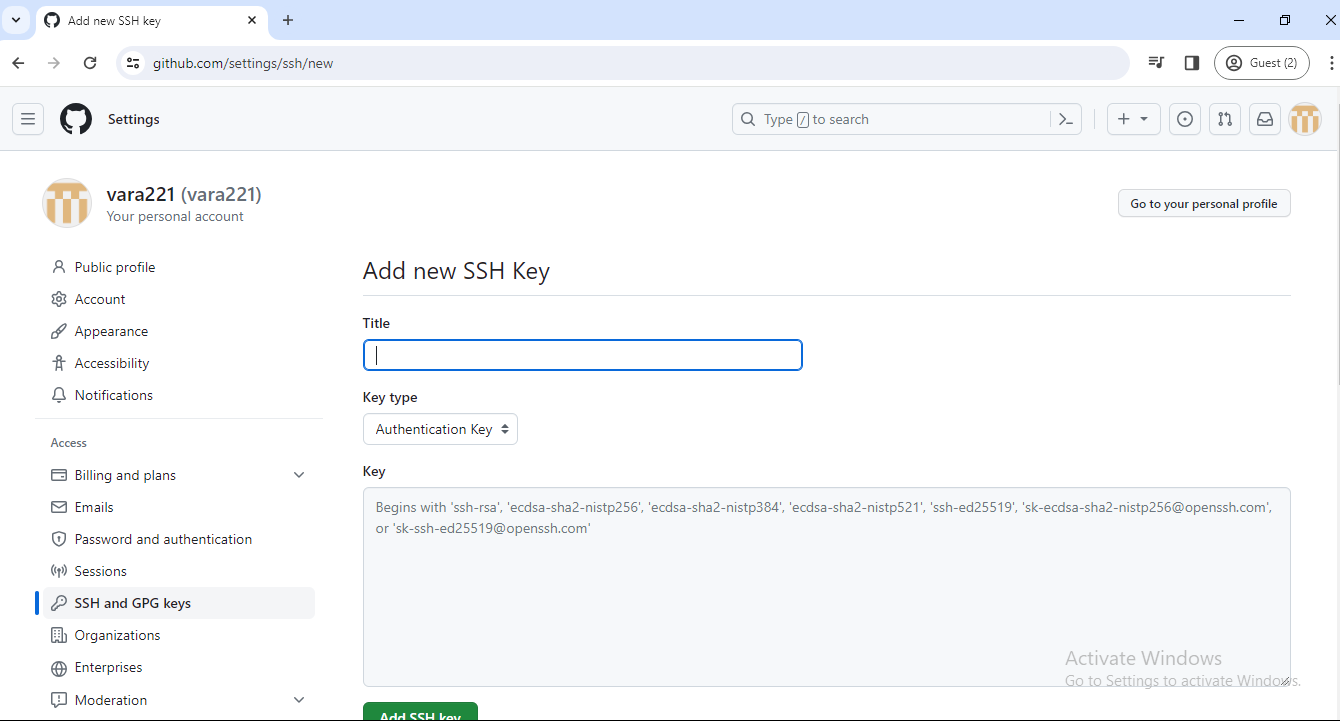
$ eval ssh-agent -s

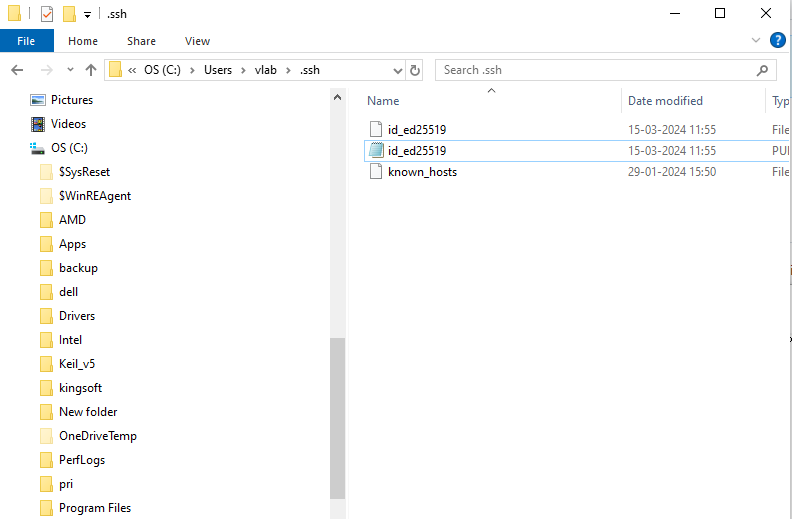
SSH\_AUTH\_SOCK=/tmp/ssh-C29jHtvGnm1Q/agent.7125; export SSH\_AUTH\_SOCK;

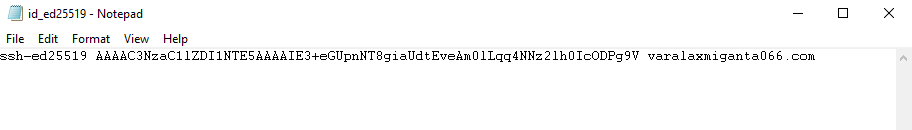
SSH\_AGENT\_PID=7126; export SSH\_AGENT\_PID;

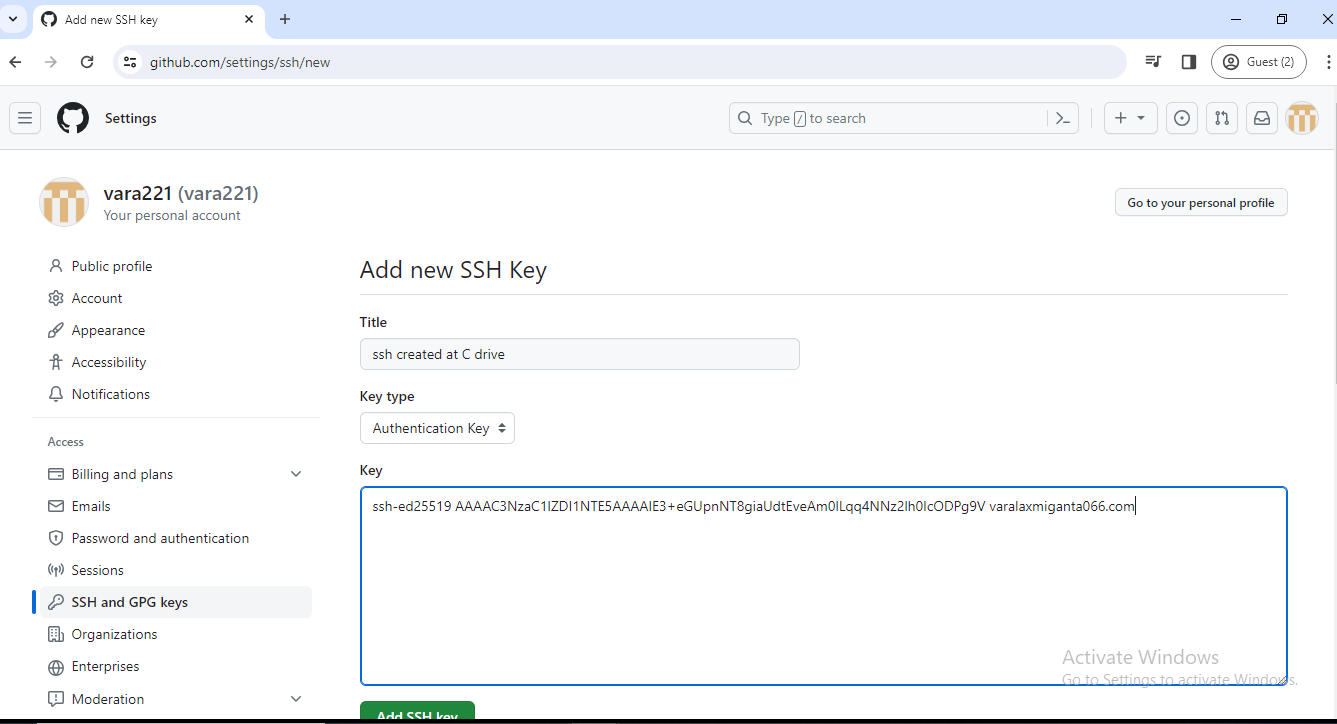
echo Agent pid 7126;













#### Create Repository in Github

Link remote repository to local repository

To check any remote repository are available or not ?

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch

dummy\_branch

login\_feature

\* master

new\_branch

new\_feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote -v

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ cd ../

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd cloning/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning (master)

$ ls

sw/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning (master)

$ cd sw/

To view any existing remotes for your repository ,we can run git remote or git remote -v

This command just displays a list of remotes.if you haven’t added any remote yet, you won’t see anything.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ git remote

origin

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ git remote -v

origin https://github.com/vara221/sw.git (fetch)

origin https://github.com/vara221/sw.git (push)

##### Adding Remote :

**Git remote add origin <url>**

Or

**Git remote add mygiturl <url>**

This means anytime you see the name “mygiturl”,we are referring to this particular github repo url.

Commonly used remote name is origin by all of them.

Origin is the short name for the url

That means when ever I use the name origin, I am referring to the particular github url like an alias name.

The name origin is the conventional git remote name, it is not at all special.it’s just a name for a URL.

When we clone a Github repo,the default remote name setup for us is called origin.you can change it.most people leave it as it is

How we have master as the default branch just like that origin is default, if needed we can change the name

Renaming the remotes :

**Git remote rename <old-name> <new-name>**

We can also remove the remote using

**Git remote remove <name>**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote add origin https://github.com/vara221/sw.git

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote

origin

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote -v

origin https://github.com/vara221/sw.git (fetch)

origin https://github.com/vara221/sw.git (push)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ ls

dummy.txt dummy1.txt index.txt new.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote rename origin githuburl

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote

githuburl

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote -v

githuburl https://github.com/vara221/sw.git (fetch)

githuburl https://github.com/vara221/sw.git (push)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote remove githuburl

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote -v

##### Push the local repository branch and commits :

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote add origin1 https://github.com/vara221/sw.git

**NOTE :**

If you push data from a local repository to a remote repository by using the push --force option, then remote repository data will arise and overwrite the local repository data.

So be careful while using the push-force option.

Before pushing the data from the remote repository to the local repository, you must pull the data from the remote repository.

If you have different histories in both the remote and local repositories, then you must pull the data after pulling the origin master. The allow-unrelated-histories command will be used.

###### Option 1 ( git push --force origin1 master)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git push --force origin1 master

Enumerating objects: 29, done.

Counting objects: 100% (29/29), done.

Delta compression using up to 4 threads

Compressing objects: 100% (22/22), done.

Writing objects: 100% (29/29), 2.65 KiB | 180.00 KiB/s, done.

Total 29 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (4/4), done.

To https://github.com/vara221/sw.git

+ d211afe...aec2d9d master -> master (forced update)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git log --oneline

aec2d9d (HEAD -> master, origin1/master) Merge branch 'dummy\_branch

'

32bb04d modifed index.txt file

2ca95e6 (dummy\_branch) modified one dummy file

0c30fc7 dummy file

1251119 Indexfile will be changed

00cd53d Dummy file

7a8b73c indexfile will be changed

772a43a (new\_feature) new file added

4daae3c minor bug fixes

c527478 Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ ls

\_\_init\_\_.py test.txt testing.txt webelement.py

\_\_pycache\_\_/ test\_webelements.py testing1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ git pull origin master

From https://github.com/vara221/sw

\* branch master -> FETCH\_HEAD

fatal: refusing to merge unrelated histories

###### Option 2 ( git pull origin master --allow-unrelated-histories )

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ git pull origin master --allow-unrelated-histories

From https://github.com/vara221/sw

\* branch master -> FETCH\_HEAD

Merge made by the 'ort' strategy.

dummy.txt | 1 +

dummy1.txt | 1 +

index.txt | 5 +++++

new.txt | 1 +

4 files changed, 8 insertions(+)

create mode 100644 dummy.txt

create mode 100644 dummy1.txt

create mode 100644 index.txt

create mode 100644 new.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ ls

\_\_init\_\_.py dummy1.txt test.txt testing1

\_\_pycache\_\_/ index.txt test\_webelements.py webelement.py

dummy.txt new.txt testing.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ git log --oneline

7efe791 (HEAD -> master) retrieving contents of remote repo

aec2d9d (origin/master, branch1) Merge branch 'dummy\_branch'

32bb04d modifed index.txt file

2ca95e6 modified one dummy file

0c30fc7 dummy file

1251119 Indexfile will be changed

00cd53d Dummy file

7a8b73c indexfile will be changed

772a43a new file added

4daae3c minor bug fixes

c527478 Initial commit

d211afe Create testing1

524debf Merge branch 'master' of https://github.com/vara221/sw

6904d96 Create test.txt

52d15f8 Merge remote-tracking branch 'origin/master'

ef5322d Create testing.txt

2910fb9 updated webelement.py

d000009 Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/cloning/sw (master)

$ git push origin master

Enumerating objects: 32, done.

Counting objects: 100% (32/32), done.

Delta compression using up to 4 threads

Compressing objects: 100% (18/18), done.

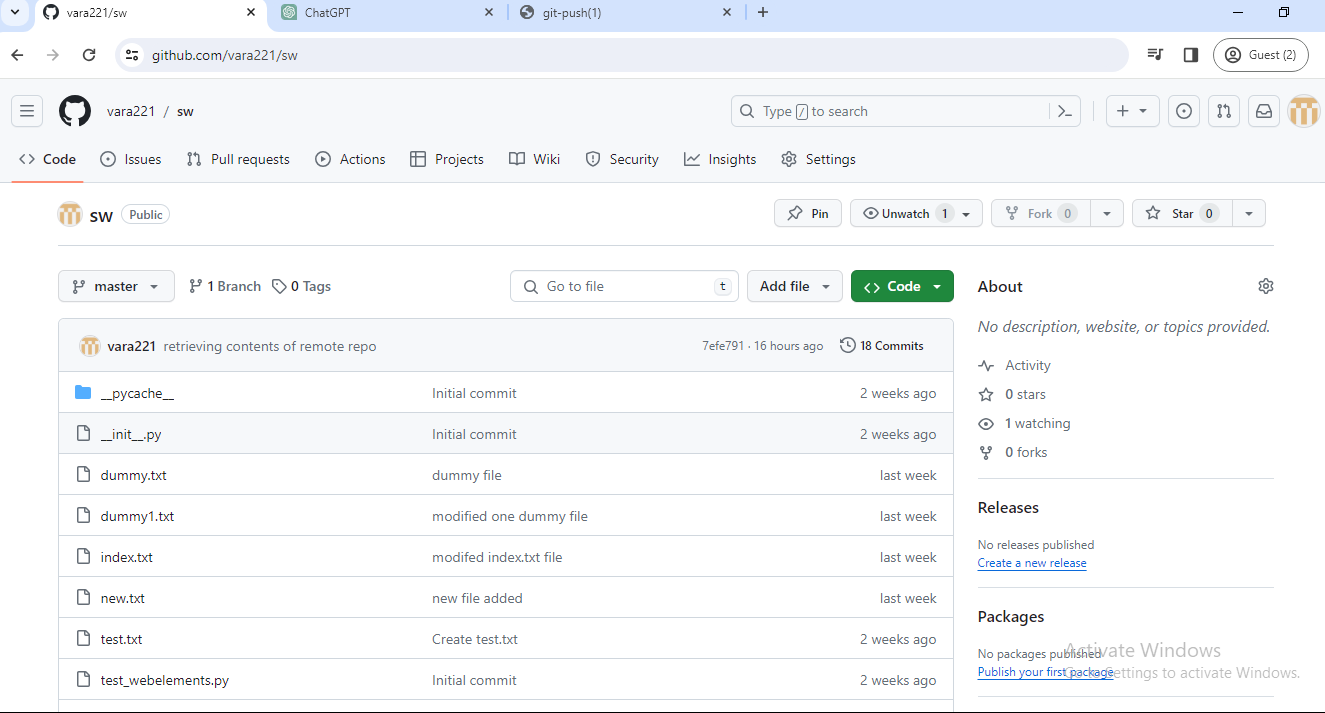
Writing objects: 100% (31/31), 12.10 KiB | 3.02 MiB/s, done.

Total 31 (delta 10), reused 28 (delta 9), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (10/10), done.

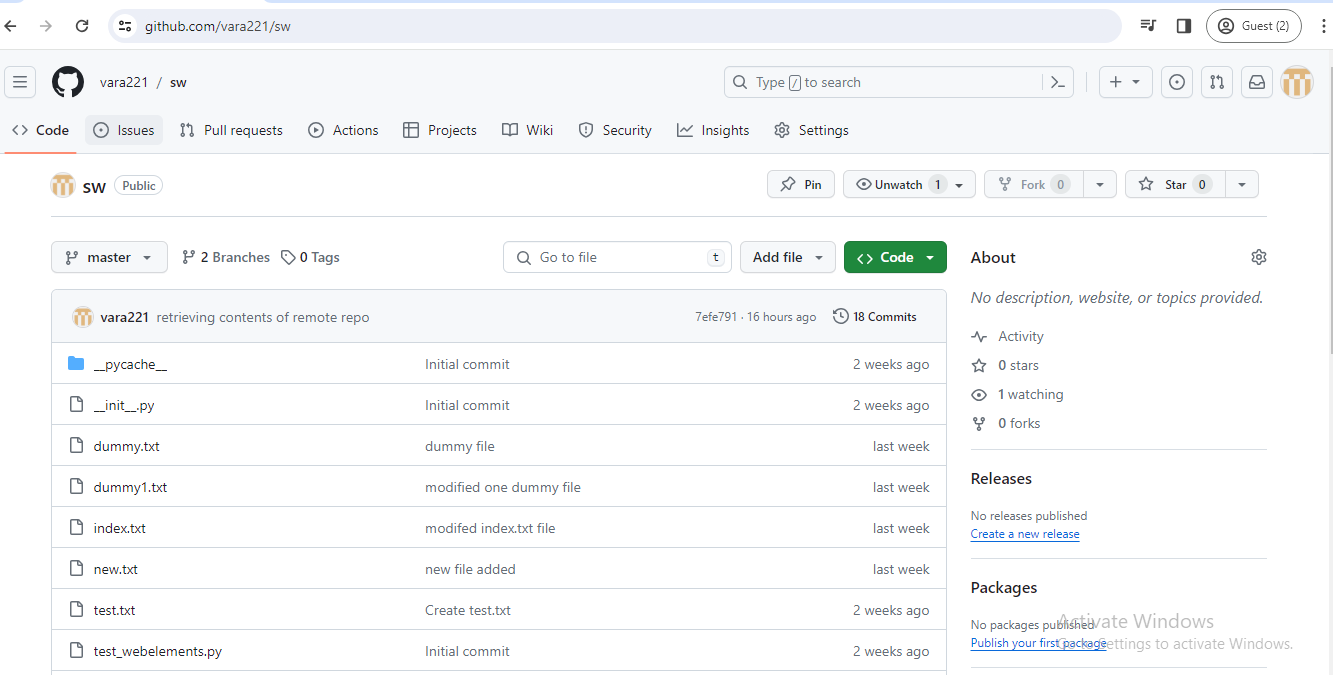
To https://github.com/vara221/sw.git

aec2d9d..7efe791 master -> master



Here we only have one master branch.

###### Option 3 ( git remote prune origin ) If I want to delete one branch at github, but in that situation, after deleting Git Lab, we also have that branch, so that we remove the git lab branch, we also use the git remote prune origin option.



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_branch)

$ git branch

dummy\_branch

login\_feature

master

\* new\_branch

new\_feature

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (new\_branch)

$ git checkout master

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git pull origin b1

From https://github.com/vara221/sw

\* branch b1 -> FETCH\_HEAD

\* [new branch] b1 -> origin/b1

Already up to date.

* + - 1. **Git branch -r option shows the remote branches**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch -r

origin/b1

origin/master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch -r

origin/b1

origin/master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote prune origin

Pruning origin

URL: https://github.com/vara221/sw.git

\* [pruned] origin/b1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git branch -r

origin/master

###### Option 4 ( git push --prune origin )

###### Option 5 (git push -u origin/b2)

###### Optional :

**By default, main to main or master to master will be connected by using a git branch. You create a branch set up to track the upstream with a remote repository, and you must pull the changes without specifying the branch name. All commits are available in the current branch at gitlab.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummy\_branch)

$ git branch -u origin/b2

branch 'dummy\_branch' set up to track 'origin/b2'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummy\_branch)

$ git log --oneline

2ca95e6 (HEAD -> dummy\_branch) modified one dummy file

0c30fc7 dummy file

1251119 Indexfile will be changed

00cd53d Dummy file

7a8b73c indexfile will be changed

772a43a (new\_feature) new file added

4daae3c minor bug fixes

c527478 Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummy\_branch)

$ git pull

Updating 2ca95e6..6633aa5

Fast-forward

dummy1.txt | 1 -

f1 | 1 +

index.txt | 3 ++-

3 files changed, 3 insertions(+), 2 deletions(-)

delete mode 100644 dummy1.txt

create mode 100644 f1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (dummy\_branch)

$ git log --oneline

6633aa5 (HEAD -> dummy\_branch, origin/master, origin/b2, origin/b1,

master, b1) deleted all files

7efe791 retrieving contents of remote repo

aec2d9d Merge branch 'dummy\_branch'

32bb04d modifed index.txt file

2ca95e6 modified one dummy file

0c30fc7 dummy file

1251119 Indexfile will be changed

00cd53d Dummy file

7a8b73c indexfile will be changed

772a43a (new\_feature) new file added

4daae3c minor bug fixes

c527478 Initial commit

d211afe Create testing1

524debf Merge branch 'master' of https://github.com/vara221/sw

6904d96 Create test.txt

52d15f8 Merge remote-tracking branch 'origin/master'

ef5322d Create testing.txt

2910fb9 updated webelement.py

d000009 Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ ls

dummy.txt f1 index.txt new.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ touch master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git commit -m "Add master file"

[master 8c4d042] Add master file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git remote -v

origin https://github.com/vara221/sw.git (fetch)

origin https://github.com/vara221/sw.git (push)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ git push origin master

Enumerating objects: 4, done.

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

Delta compression using up to 4 threads

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

Writing objects: 100% (3/3), 273 bytes | 273.00 KiB/s, done.

Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object

To https://github.com/vara221/sw.git

6633aa5..8c4d042 master -> master

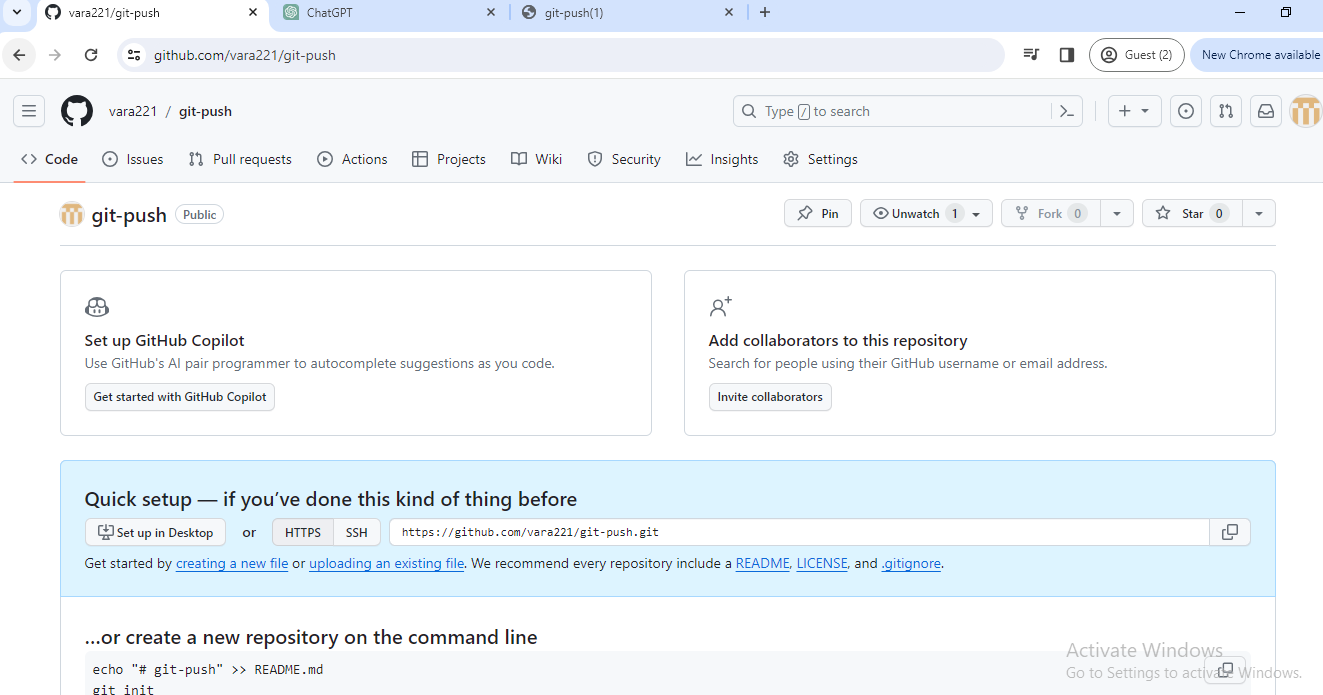
vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/ecommerceproj (master)

$ ls

dummy.txt f1 index.txt master.txt new.txt

##### Pushing the local changes from one branch to another remote branch

Especially we will see the difference between the local branch and the branch on GitHub.



Here, there is no branch master or main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir gitpush

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd gitpush

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/git

push/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ touch pushfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git commit -m "create push file"

[master (root-commit) 350bce0] create push file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 pushfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git remote

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git remote add origin https://github.com/vara221/git-push.git

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git log --oneline

350bce0 (HEAD -> master) create push file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git push origin master

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

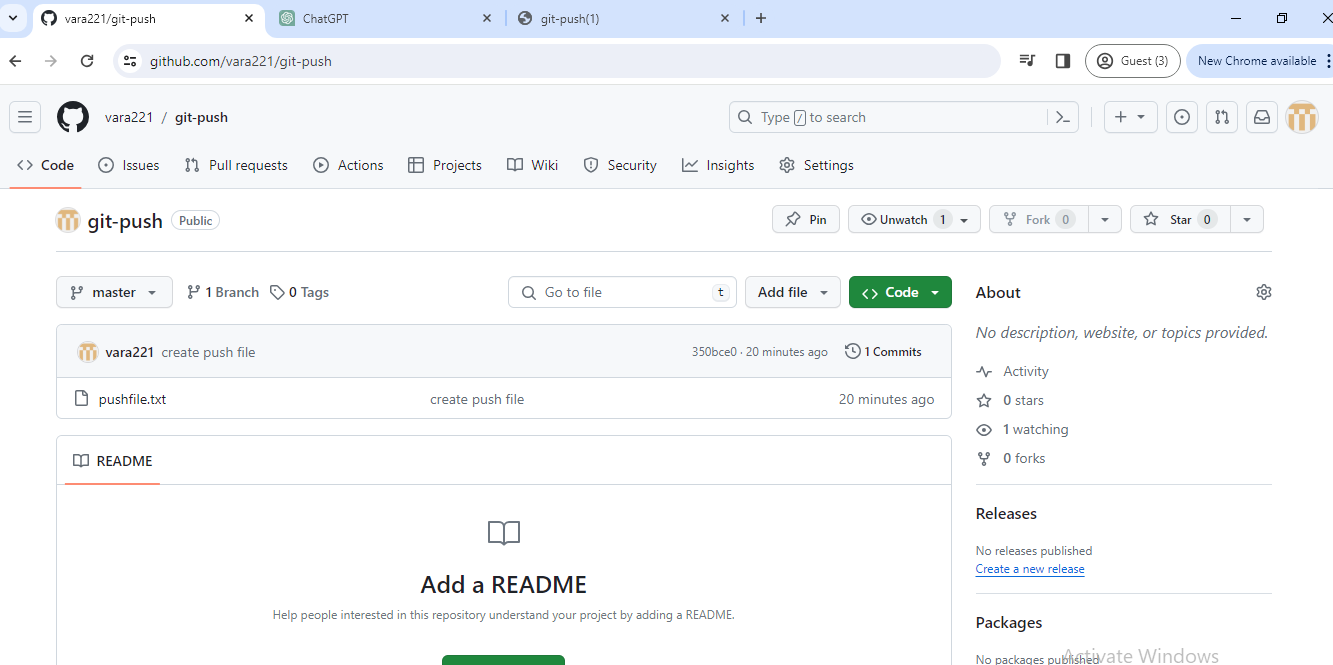
Writing objects: 100% (3/3), 222 bytes | 222.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-push.git

\* [new branch] master -> master

Here one master branch is created in the remote repository



Create a new file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ touch index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git commit -m "create index file"

[master 5b70483] create index file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.txt

When we push origin master, if the master branch is not present in the remote repository, we create the master branch or otherwise add that file to the remote repository.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git push origin master

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

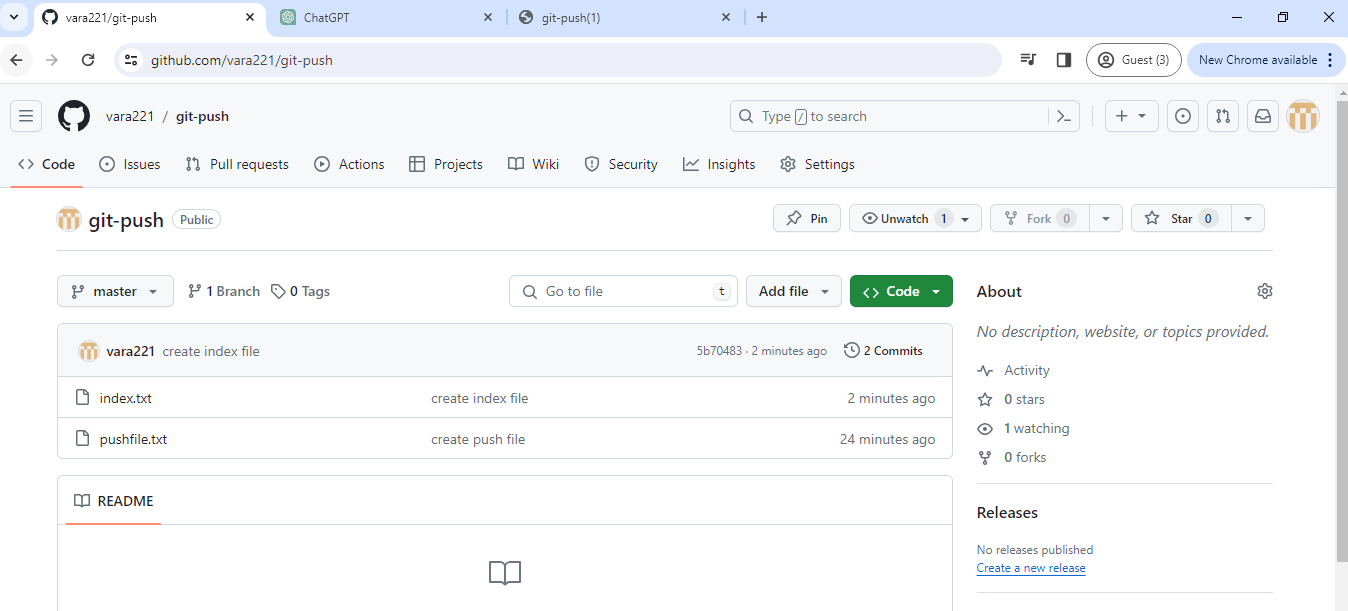
Compressing objects: 100% (2/2), done.

Writing objects: 100% (2/2), 254 bytes | 254.00 KiB/s, done.

Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-push.git

350bce0..5b70483 master -> master



Not only to master we can also push the changes in the mster or any branch to the different branch in github

Git push <remote> <local-branch>:<remote-branch>

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (master)

$ git switch -c new\_branch

Switched to a new branch 'new\_branch'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ touch branchfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ git commit -m "create branch file"

[new\_branch efdfd1b] create branch file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 branchfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ git push origin new\_branch

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

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

Writing objects: 100% (2/2), 260 bytes | 130.00 KiB/s, done.

Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

remote:

remote: Create a pull request for 'new\_branch' on GitHub by visiting:

remote: https://github.com/vara221/git-push/pull/new/new\_branch

remote:

To https://github.com/vara221/git-push.git

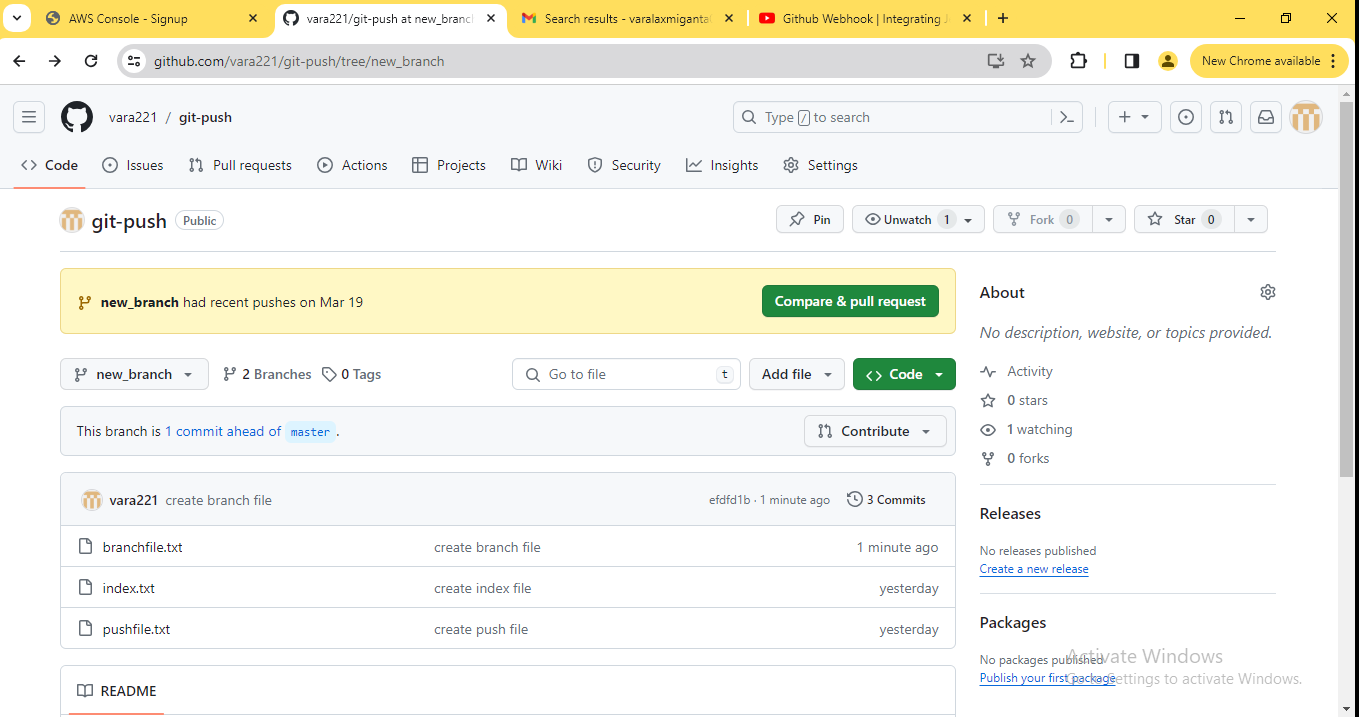
\* [new branch] new\_branch -> new\_branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ git remote -v

origin https://github.com/vara221/git-push.git (fetch)

origin https://github.com/vara221/git-push.git (push)



Here having two branches

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ ls

branchfile.txt index.txt pushfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ touch vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ git commit -m "create vara file"

[new\_branch b8b45d9] create vara file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/gitpush (new\_branch)

$ git push origin new\_branch:master

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

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

Writing objects: 100% (2/2), 234 bytes | 16.00 KiB/s, done.

Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)

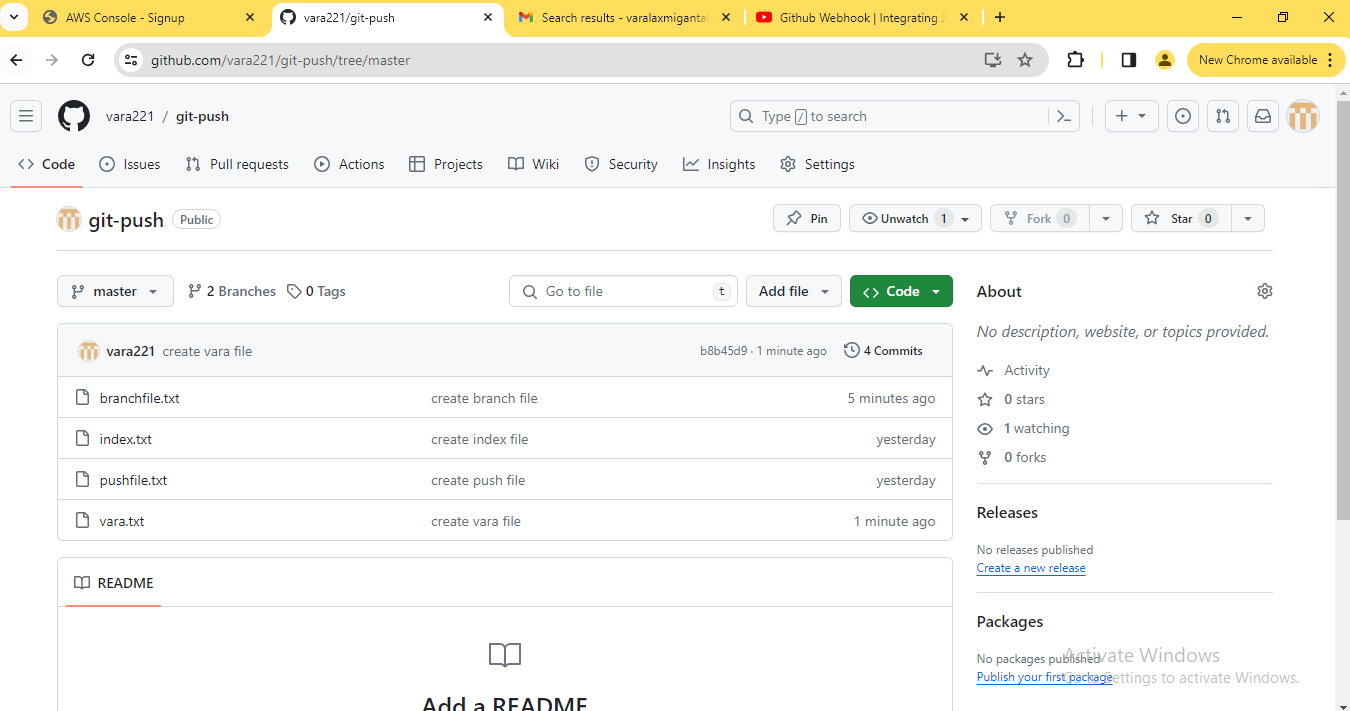
remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/vara221/git-push.git

5b70483..b8b45d9 new\_branch -> master

**new\_branch -> master newbranch will pointing to the master**

**Newbranch has all commit are moved to master**



This type of pushing is not at all common but you can do it.

You need to remember that we can also push the changes to the different remote branch.

##### What git push -u means ?

The -u option allows us to set the upstream of the branch we are pushing.

You can think of this as a link between the local branch to a branch in the github.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir git-upstream

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-upstream

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/git-u

/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git remote add origin https://github.com/vara221/git-upstream.git

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ touch index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git add .

g

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git commit -m "create index file"

[master (root-commit) 5c7d988] create index file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git push origin master

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 219 bytes | 219.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-upstream.git

\* [new branch] master -> master

Git push -u origin master

Running this command set the upstream of the local master branch so that it tracks the master branch on the origin repository

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ touch vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git commit -m "create vara file"

[master a1568b8] create vara file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git push -u origin master

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

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

Writing objects: 100% (2/2), 245 bytes | 245.00 KiB/s, done.

Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-upstream.git

5c7d988..a1568b8 master -> master

branch 'master' set up to track 'origin/master'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ ls

index.txt vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ touch master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git commit -m "create master file"

[master d117195] create master file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 master.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git push

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

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

Writing objects: 100% (2/2), 260 bytes | 260.00 KiB/s, done.

Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-upstream.git

a1568b8..d117195 master -> master



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (master)

$ git switch -c new\_branch

Switched to a new branch 'new\_branch'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ touch branchfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ git commit -m "create branch file"

[new\_branch 0520a27] create branch file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 branchfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ git push

fatal: The current branch new\_branch has no upstream branch.

To push the current branch and set the remote as upstream, use

git push --set-upstream origin new\_branch

To have this happen automatically for branches without a tracking

upstream, see 'push.autoSetupRemote' in 'git help config'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ git push -u origin new\_branch

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

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

Writing objects: 100% (2/2), 249 bytes | 249.00 KiB/s, done.

Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

remote:

remote: Create a pull request for 'new\_branch' on GitHub by visiting:

remote: https://github.com/vara221/git-upstream/pull/new/new\_bran

ch

remote:

To https://github.com/vara221/git-upstream.git

\* [new branch] new\_branch -> new\_branch

branch 'new\_branch' set up to track 'origin/new\_branch'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ touch newfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ git add .

g

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ git commit -m "create new file"

[new\_branch f984268] create new file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 newfile.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-upstream (new\_branch)

$ git push

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

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

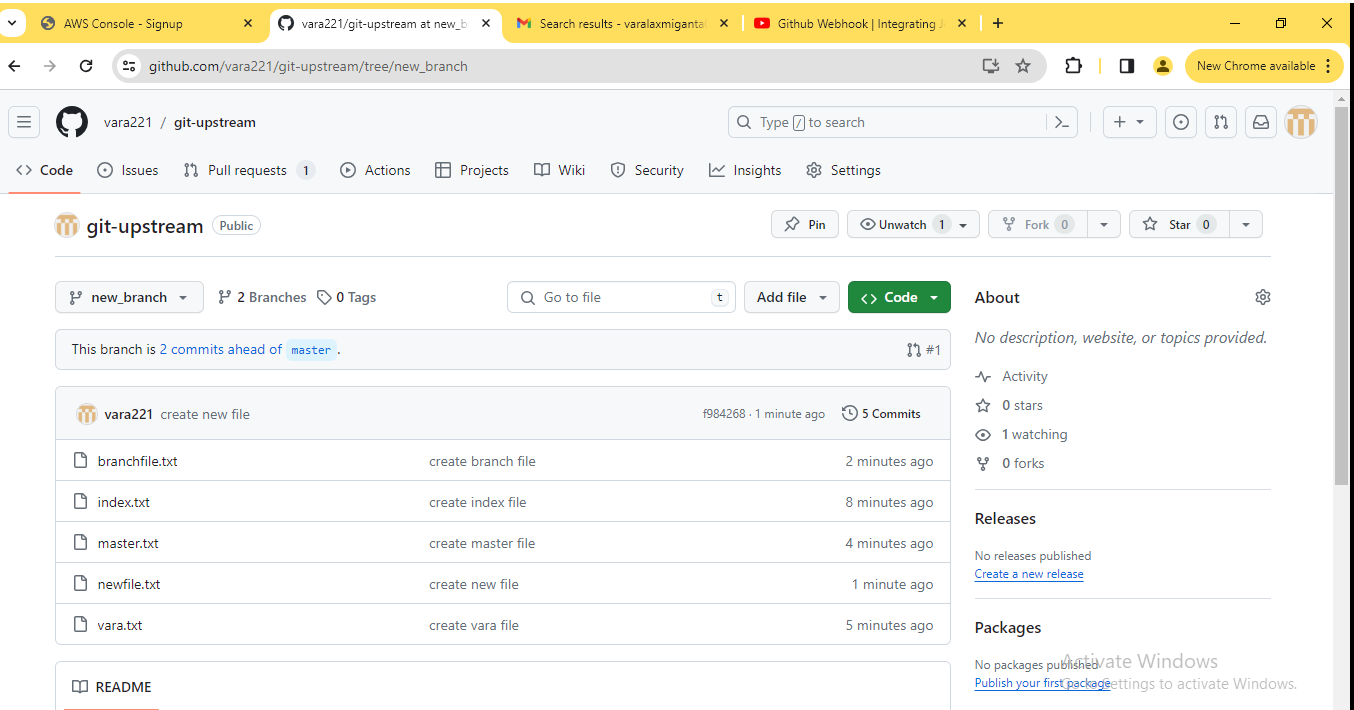
Writing objects: 100% (2/2), 236 bytes | 236.00 KiB/s, done.

Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/vara221/git-upstream.git

0520a27..f984268 new\_branch -> new\_branch



##### Difference between the main and master branch in the git repository

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/vara221/git-clonning.git

Cloning into 'git-clonning'...

warning: You appear to have cloned an empty repository.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-clonning/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (main)

$ ls

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (main)

$ git remote -v

origin https://github.com/vara221/git-clonning.git (fetch)

origin https://github.com/vara221/git-clonning.git (push)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (main)

$ touch index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (main)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (main)

$ git commit -m "create index file"

[main (root-commit) 3f995e4] create index file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (main)

$ git push origin main

Enumerating objects: 3, done.

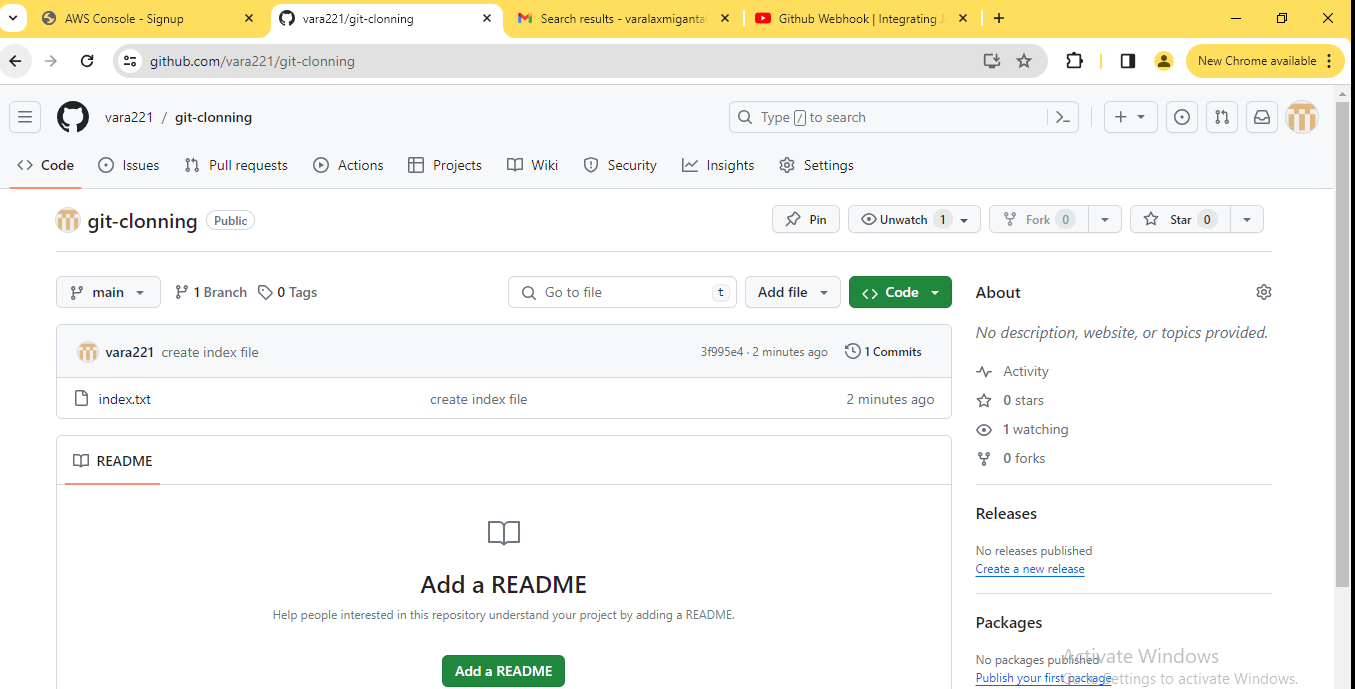
Counting objects: 100% (3/3), done.

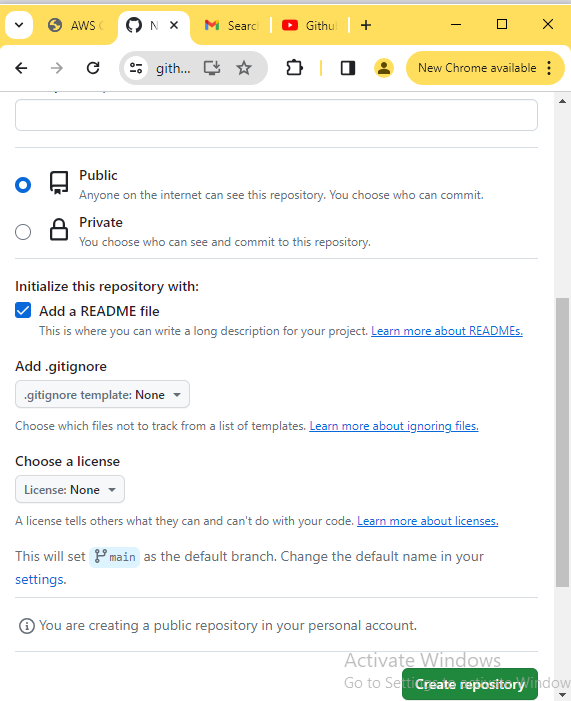
Writing objects: 100% (3/3), 219 bytes | 109.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

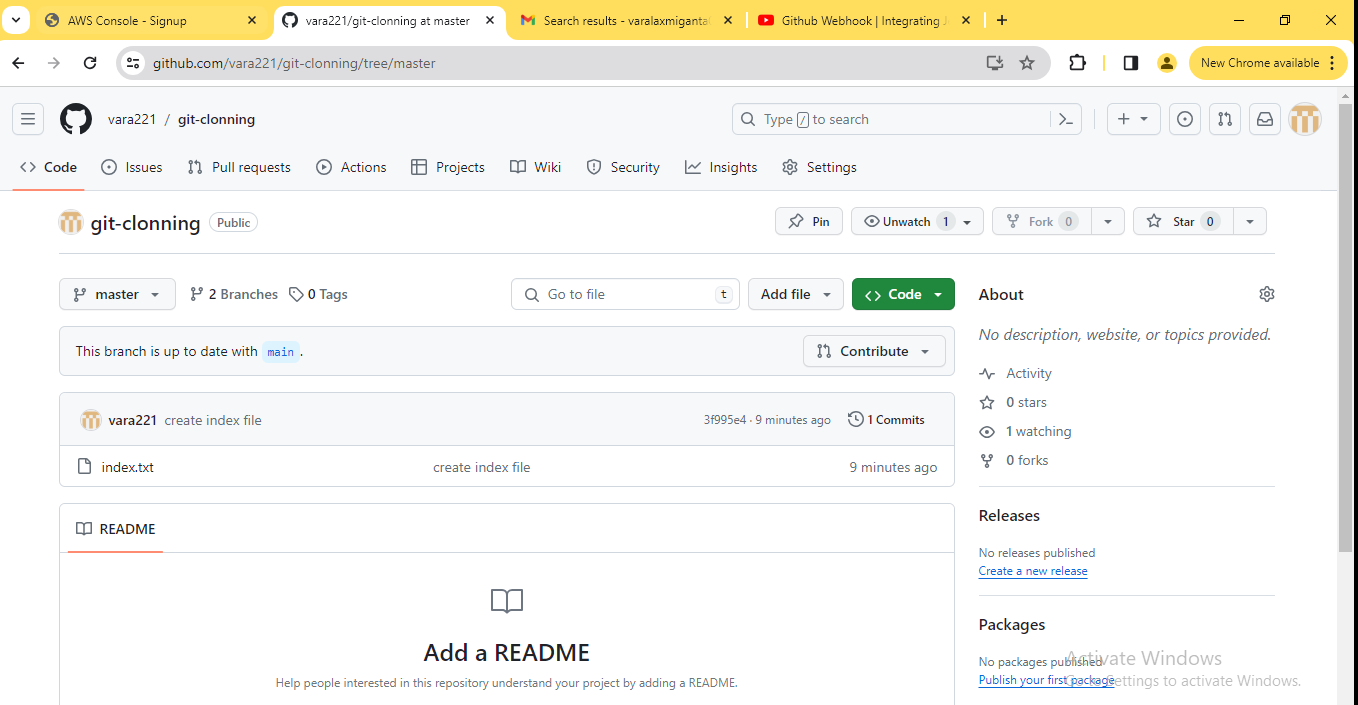
To https://github.com/vara221/git-clonning.git

\* [new branch] main -> main





By default to click the add a readme file then create a main branch



Here we having two branches one is main and master other one

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (main)

$ git branch

\* main

To change the branch name main to master

But by default main is available

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (main)

$ git branch -M master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (master)

$ git push origin master

Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

remote:

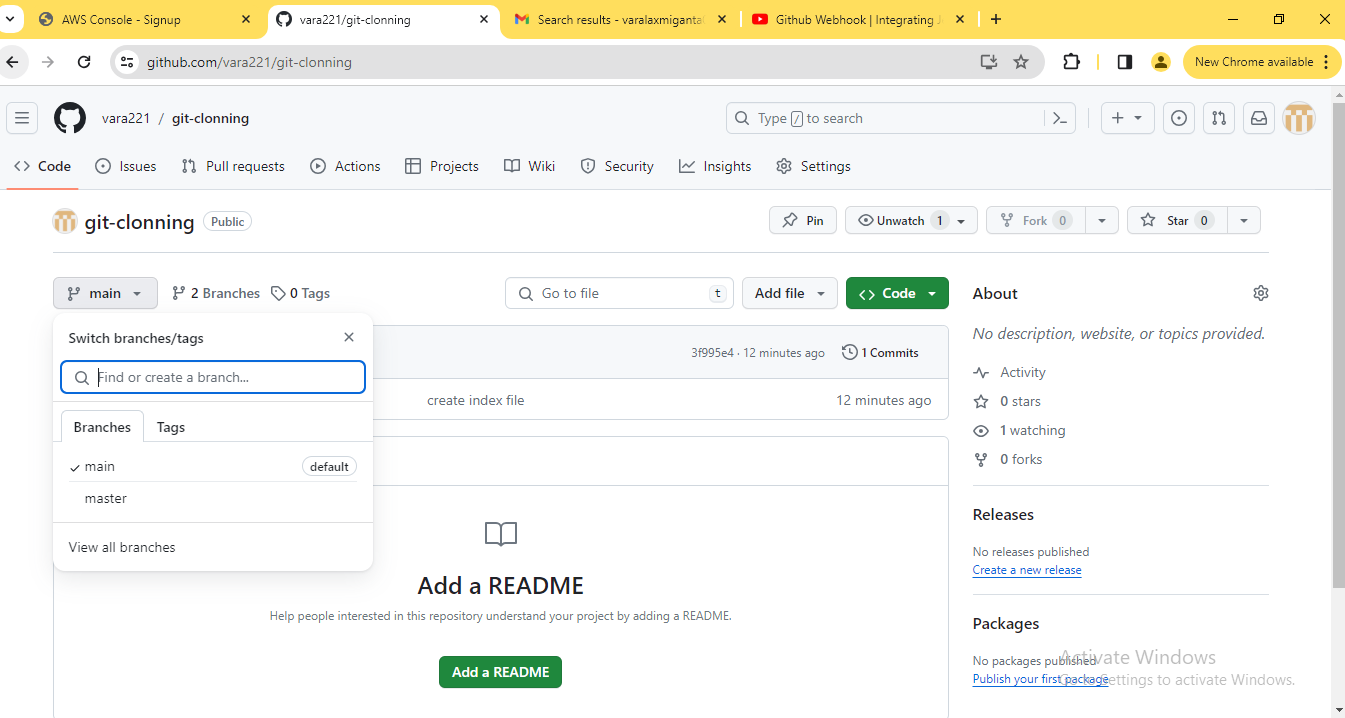
remote: Create a pull request for 'master' on GitHub by visiting:

remote: https://github.com/vara221/git-clonning/pull/new/master

remote:

To https://github.com/vara221/git-clonning.git

\* [new branch] master -> master

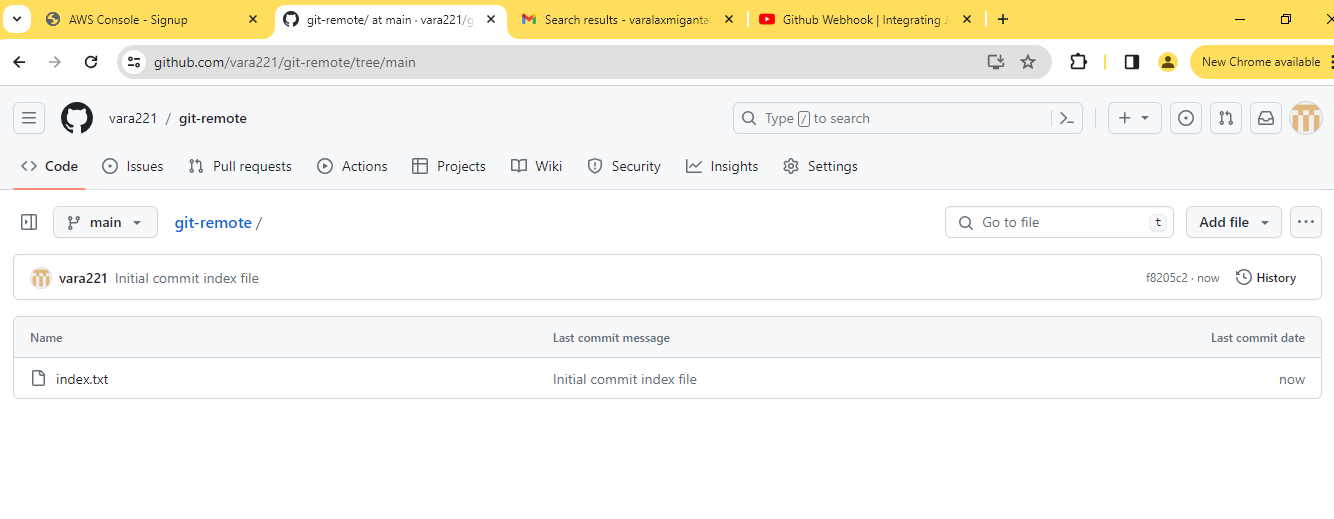


Main is the default branch

If we want to change the master has default then main will be delete

Settings -> Branches -> Change default branch

##### Remote Tracking Branches



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/vara221/git-remote.git

Cloning into 'git-remote'...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), done.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-remote/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git log --oneline

f8205c2 (HEAD -> main, origin/main, origin/HEAD) Initial commit index

file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ ls

index.txt

**This head -> main is the local branch and origin/main is the remote branch**

**These both pointing to the same commit**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ touch vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git add .

i

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git commit -m "create vara file"

[main 7589fb4] create vara file

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 vara.txt

g

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git log --oneline

7589fb4 (HEAD -> main) create vara file

f8205c2 (origin/main, origin/HEAD) Initial commit index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git status

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git branch

\* main

* + 1. **To know the remote branches by using branch -r**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git branch -r

origin/HEAD -> origin/main

origin/main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git push origin main

Enumerating objects: 4, done.

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

Delta compression using up to 4 threads

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

Writing objects: 100% (3/3), 274 bytes | 137.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-remote.git

f8205c2..7589fb4 main -> main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git status

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git log --oneline

7589fb4 (HEAD -> main, origin/main, origin/HEAD) create vara file

f8205c2 Initial commit index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ touch index1.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git commit -m "create index1.txt"

[main de52ef6] create index1.txt

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index1.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git log --oneline

de52ef6 (HEAD -> main) create index1.txt

7589fb4 (origin/main, origin/HEAD) create vara file

f8205c2 Initial commit index file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git status

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git checkout origin/main

Note: switching to 'origin/main'.

You are in 'detached HEAD' state. You can look around, make experiment

al

changes and commit them, and you can discard any commits you make in t

his

state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you m

ay

do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead to

false

HEAD is now at 7589fb4 create vara file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote ((7589fb4...))

$ git switch -

Previous HEAD position was 7589fb4 create vara file

Switched to branch 'main'

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git push origin main

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

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

Writing objects: 100% (2/2), 276 bytes | 276.00 KiB/s, done.

Total 2 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-remote.git

7589fb4..de52ef6 main -> main

##### Checkout the Remote tracking branches in the local git repository

If remote having multiple branch so that not contain in the local repository

There four branches available including main also

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/vara221/git-branches.git

Cloning into 'git-branches'...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), done.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-branches/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (main)

$ git branch

\* main

**4.6.3. List of all branches in the remote repository**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (main)

$ git branch -r

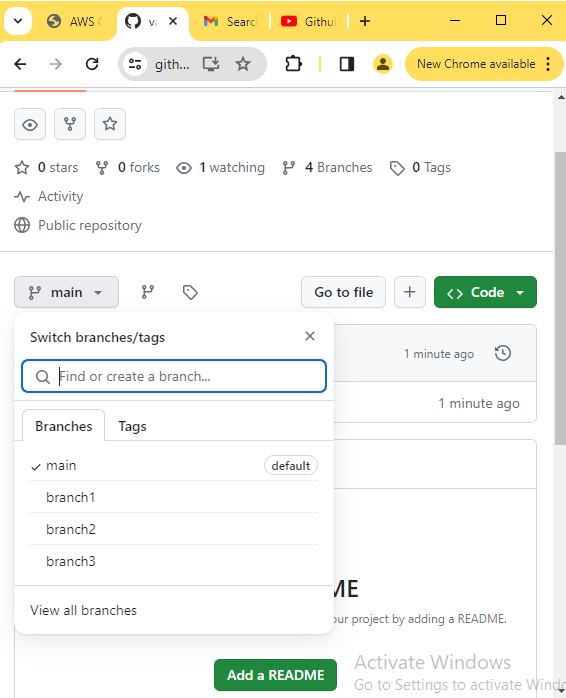
origin/HEAD -> origin/main

origin/branch1

origin/branch2

origin/branch3

origin/main



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (main)

$ git checkout origin/branch1

Note: switching to 'origin/branch1'.

You are in 'detached HEAD' state. You can look around, make experiment

al

changes and commit them, and you can discard any commits you make in t

his

state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you m

ay

do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead to

false

HEAD is now at 4845c34 Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches ((4845c34...))

$ git switch -

Switched to branch 'main'

Your branch is up to date with 'origin/main'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (main)

$ git status

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (main)

$ git switch branch4

fatal: invalid reference: branch4

**When entering the switch branch1 command, then git lab, first check the local branch; if it is not available, then go to check the remote branches.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (branch2)

$ git switch branch1

branch 'branch1' set up to track 'origin/branch1'.

Switched to a new branch 'branch1'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (branch1)

$ git status

On branch branch1

Your branch is up to date with 'origin/branch1'.

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (branch1)

$ git branch

\* branch1

branch2

main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (branch1)

$ git branch -r

origin/HEAD -> origin/main

origin/branch1

origin/branch2

origin/branch3

origin/main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (branch1)

$ git switch branch3

branch 'branch3' set up to track 'origin/branch3'.

Switched to a new branch 'branch3'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (branch3)

$ git branch

branch1

branch2

\* branch3

main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-branches (branch3)

$ git branch -r

origin/HEAD -> origin/main

origin/branch1

origin/branch2

origin/branch3

origin/main

### Git Fetch

When you are working on with other collaborators on a github repository.

One of your teammate has pushed up the changes to the master branch,but my local repository doesn’t know!

Then how do I get those changes?

Git fetch and pull get those changes from the github repository to your local repository .

But these two commands has some differences.

Fetching allows us to download changes from remote repository.

But those changes will not be automatically integrated to our working files

It just lets you see what others have been working in, without merging those changes into your local repository.

**Git fetch <remote>**

This command fetches branches and history from a specific remote repository. It only updated the remote tracking branches.

Git fetch origin would fetch all changes from the origin remote repository

If not specified remote it defaults to origin

We can also fetch a specific branch from a remote using the command

**Git fetch <remote> <branch>**

For example git fetch origin master would retrieve the latest information from the master branch on the origin remote repository.

After fetching the changes.I will have those changes on my machine, but if I want to see then I have to do the checkout to origin/master

Your local master branch will be untouched

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/vara221/git-fetch.git

Cloning into 'git-fetch'...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), done.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-fetch/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ ls

t1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ git branch

\* main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ git branch -r

origin/HEAD -> origin/main

origin/login

origin/main

origin/registration

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ git status

On branch main

Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ ls

t1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ git fetch origin

remote: Enumerating objects: 6, done.

remote: Counting objects: 100% (6/6), done.

remote: Compressing objects: 100% (4/4), done.

remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (5/5), 1.77 KiB | 9.00 KiB/s, done.

From https://github.com/vara221/git-fetch

5858d99..fc3e780 main -> origin/main

5858d99..b1e8c22 login -> origin/login

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ git status

On branch main

Your branch is behind 'origin/main' by 1 commit, and can be fast-forwa

rded.

(use "git pull" to update your local branch)

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ git log --oneline

5858d99 (HEAD -> main, origin/registration) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ ls

t1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ git checkout origin/main

Note: switching to 'origin/main'.

You are in 'detached HEAD' state. You can look around, make experiment

al

changes and commit them, and you can discard any commits you make in t

his

state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you m

ay

do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHead to

false

HEAD is now at fc3e780 create main file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch ((fc3e780...))

$ ls

main.txt t1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch ((fc3e780...))

$ git log --oneline

fc3e780 (HEAD, origin/main, origin/HEAD) create main file

5858d99 (origin/registration, main) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch ((fc3e780...))

$ git switch main

Previous HEAD position was fc3e780 create main file

Switched to branch 'main'

Your branch is behind 'origin/main' by 1 commit, and can be fast-forwa

rded.

(use "git pull" to update your local branch)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (main)

$ git switch login

branch 'login' set up to track 'origin/login'.

Switched to a new branch 'login'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (login)

$ git status

On branch login

Your branch is up to date with 'origin/login'.

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (login)

$ ls

login.txt t1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (login)

$ git log --oneline

b1e8c22 (HEAD -> login, origin/login) Create login.txt

5858d99 (origin/registration, main) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (login)

$ git branch

\* login

main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (login)

$ git switch registration

branch 'registration' set up to track 'origin/registration'.

Switched to a new branch 'registration'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (registration)

$ git branch

login

main

\* registration

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (registration)

$ git fetch

remote: Enumerating objects: 4, done.

remote: Counting objects: 100% (4/4), done.

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

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), 913 bytes | 10.00 KiB/s, done.

From https://github.com/vara221/git-fetch

5858d99..3263d03 registration -> origin/registration

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (registration)

$ git status

On branch registration

Your branch is behind 'origin/registration' by 1 commit, and can be fa

st-forwarded.

(use "git pull" to update your local branch)

nothing to commit, working tree clean

### Git Pull :

Git push is the command we can use to retrieve changes from the remote repository.

Unlike Fetch,pull actually updated our HEAD branch with whatever changes are retrieved from the remote.

Git pull = git fetch + git merge

Git fetch - update the remote tracking branch with the latest changes from the remote repository

Git merge - update my current branch with whatever changes are on the remote tracking branch.

Git pull <remote> <branch>

Git pull origin master would fetch the latest information from the origins master branc

Pull can result in the merge conflicts

#### Implementation of the git pull Resolve merge conflicts for the change from remote repository

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git branch

login

\* main

registration

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git branch -r

origin/HEAD -> origin/main

origin/login

origin/main

origin/registration

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git status

On branch main

Your branch is up to date with 'origin/main'.

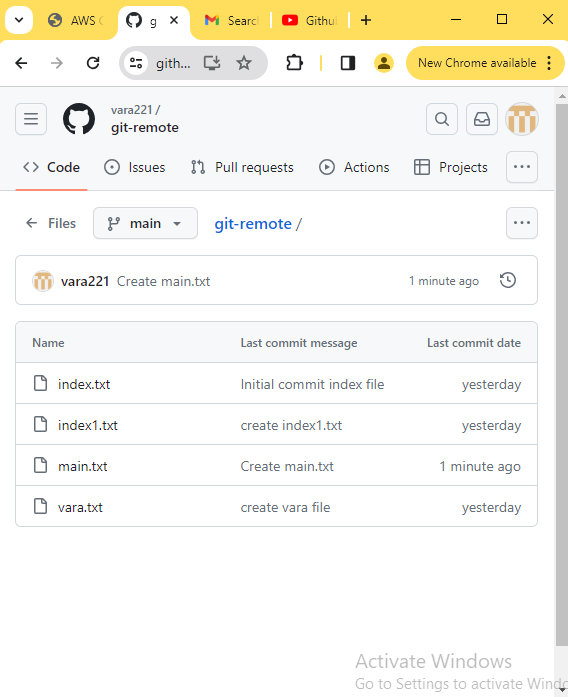
nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ ls

index.txt index1.txt vara.txt

**Open github create one file that is main.txt**



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git fetch

remote: Enumerating objects: 4, done.

remote: Counting objects: 100% (4/4), done.

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

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), 973 bytes | 5.00 KiB/s, done.

From https://github.com/vara221/git-remote

de52ef6..77ef86d main -> origin/main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git status

On branch main

Your branch is behind 'origin/main' by 1 commit, and can be fast-forwa

rded.

(use "git pull" to update your local branch)

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ ls

index.txt index1.txt vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git pull origin main

From https://github.com/vara221/git-remote

\* branch main -> FETCH\_HEAD

Updating de52ef6..77ef86d

Fast-forward

main.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 main.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ ls

index.txt index1.txt main.txt vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git log --oneline

77ef86d (HEAD -> main, origin/main, origin/HEAD) Create main.txt

de52ef6 (origin/registration, origin/login, registration, login) create index1.txt

7589fb4 create vara file

f8205c2 Initial commit index file

**Modify the main.txt file from local**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ vi main.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ cat main.txt

created one file

changes to local

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git commit -m "updated main from local"

[main e2d8f8b] updated main from local

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git status

On branch main

Your branch is ahead of 'origin/main' by 1 commit.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

**Modify main.txt file from remote**



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git fetch

remote: Enumerating objects: 5, done.

remote: Counting objects: 100% (5/5), done.

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

remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), 948 bytes | 5.00 KiB/s, done.

From https://github.com/vara221/git-remote

77ef86d..62b6c1c main -> origin/main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git pull origin main

From https://github.com/vara221/git-remote

\* branch main -> FETCH\_HEAD

Auto-merging main.txt

CONFLICT (content): Merge conflict in main.txt

Automatic merge failed; fix conflicts and then commit the result.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main|MERGING)

$ git status

On branch main

Your branch and 'origin/main' have diverged,

and have 1 and 1 different commits each, respectively.

(use "git pull" if you want to integrate the remote branch with your

s)

You have unmerged paths.

(fix conflicts and run "git commit")

(use "git merge --abort" to abort the merge)

Unmerged paths:

(use "git add <file>..." to mark resolution)

both modified: main.txt

no changes added to commit (use "git add" and/or "git commit -a")

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main|MERGING)

$ vi main.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main|MERGING)

$ cat main.txt

created one file

changes from local

changes from remote

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main|MERGING)

$ git add .

g

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main|MERGING)

$ git commit -m "merge conflicts reslove"

[main 6f5f455] merge conflicts reslove

i

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git status

On branch main

Your branch is ahead of 'origin/main' by 2 commits.

(use "git push" to publish your local commits)

nothing to commit, working tree clean

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git log --oneline

6f5f455 (HEAD -> main) merge conflicts reslove

62b6c1c (origin/main, origin/HEAD) Update main.txt from remote

e2d8f8b updated main from local

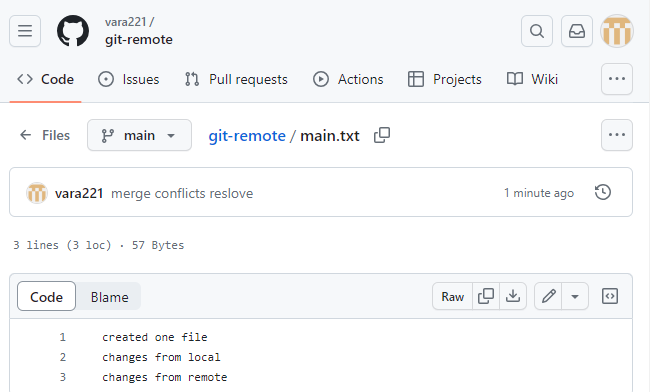
77ef86d Create main.txt

de52ef6 (origin/registration, origin/login, registration, login) creat

e index1.txt

7589fb4 create vara file

f8205c2 Initial commit index file



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git pull origin main

From https://github.com/vara221/git-remote

\* branch main -> FETCH\_HEAD

Already up to date.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git push origin main

Enumerating objects: 10, done.

Counting objects: 100% (10/10), done.

Delta compression using up to 4 threads

Compressing objects: 100% (5/5), done.

Writing objects: 100% (6/6), 608 bytes | 304.00 KiB/s, done.

Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (2/2), completed with 1 local obje

To https://github.com/vara221/git-remote.git

62b6c1c..6f5f455 main -> main

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ cat main.txt

created one file

changes from local

changes from remote

### Github Gists

Github gists are the simple way to store the code snippets and useful fragments to others.

Gists are much easier to create , but offers few features compared to normal Git repository.

Every gist is a Git repository, means it can be cloned.

If you are signed in to Github when you create a gist,the gist will be associated with your account and you will see it in your list of gists when you navigate to your gist home page.

<https://gist.github.com/>

Gists can be public or secret.

Public gists show up in Discover,where people can browse new gists as they’re created.They are also searchable, so you can use them if you wold like other people to find and see your work.

Secret gists don’t show up in Discover and are not searchable. Secret gists aren’t private.

If you send the URL of a secret gist to a friend , they will be able to see it.

However, if someone you don’t know discovers the URL, they will also be able to see your gist.

### Github Pages

Github pages are the public web pages that are hosted and published by Github.

They allow you to create a website by simply pushing your code to github.

Github pages is a hosting service for serving static web pages.

It does not support service side code like PHP,Python, Ruby or Node.

Github pages supports only HTML, CSS and js code only.

You get unlimited project sites in the github

Each github repo can have a corresponding hosted website.

It is as simple as just telling the github which branch to take for hosting the repository.

The default url in github pages follow this pattern

<https://username>.github.com/repo-name

Remember that you can get only one website for account or organization

But unlimited repository project sites.

#### Creating the github pages

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/vara221/varaganta.github.io.git

Cloning into 'varaganta.github.io'...

warning: You appear to have cloned an empty repository.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd varaganta.github.io/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/varaganta.github.io (main)

$ vi index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/varaganta.github.io (main)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/varaganta.github.io (main)

$ git commit -m "add index file"

[main (root-commit) db2c615] add index file

1 file changed, 5 insertions(+)

create mode 100644 index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/varaganta.github.io (main)

$ git push -u origin main

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Delta compression using up to 4 threads

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

Writing objects: 100% (3/3), 273 bytes | 136.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/varaganta.github.io.git

\* [new branch] main -> main

branch 'main' set up to track 'origin/main'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/varaganta.github.io (main)

$ cat index.html

<html>

<body>

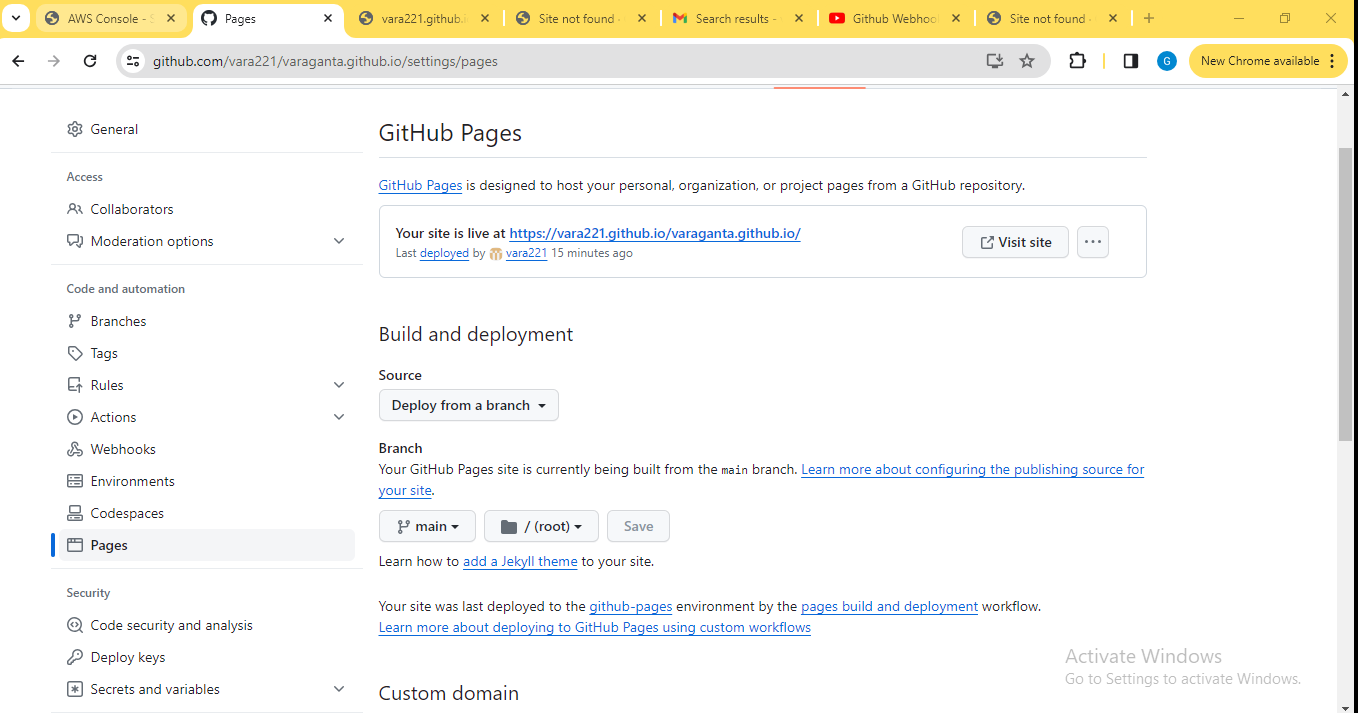
<h1>Welcome to veda world</h1>

</body>

</html>

Open varaganta.github.io repository after

Go to settings --> Pages -->



Source : select Deploy from branch

Branch : select main and save it

[https://varaganta.github.io](https://varalamxi.github.io)

It can takes 20 minutes to open web page

After create a

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-fetch/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (registration)

$ git switch -c gh-pages

Switched to a new branch 'gh-pages'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (gh-pages)

$ vi index.html

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (gh-pages)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (gh-pages)

$ git commit -m "add index.html"

[gh-pages 697e2e8] add index.html

1 file changed, 4 insertions(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fetch (gh-pages)

$ git push -u origin gh-pages

Enumerating objects: 4, done.

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

Delta compression using up to 4 threads

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

Writing objects: 100% (3/3), 313 bytes | 313.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

remote:

remote: Create a pull request for 'gh-pages' on GitHub by visitin

g:

remote: https://github.com/vara221/git-fetch/pull/new/gh-pag

es

remote:

To https://github.com/vara221/git-fetch.git

\* [new branch] gh-pages -> gh-pages

branch 'gh-pages' set up to track 'origin/gh-pages'.

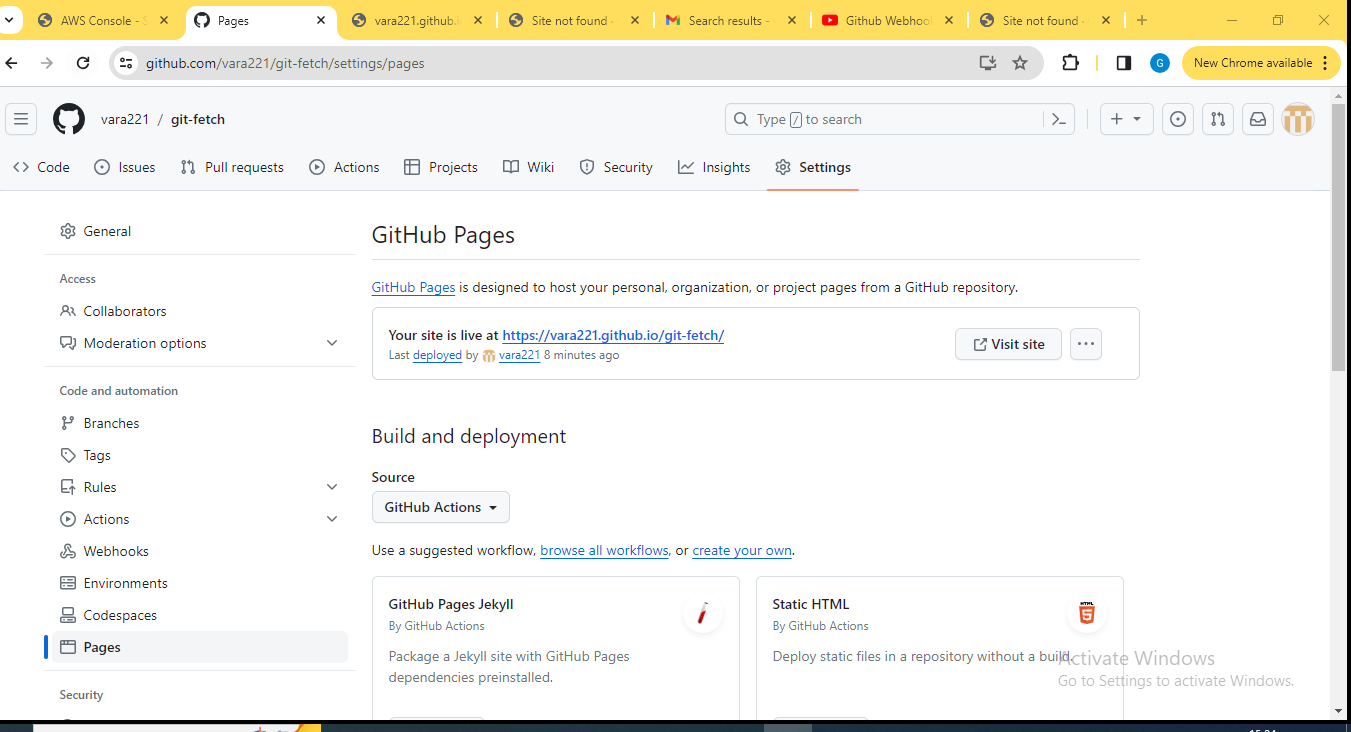
For this main branch we have separate branch that is gh-pages

Know I want add a project repository for this website.

So,

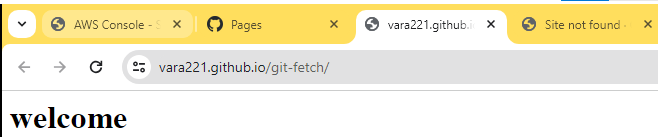
Setting --> Pages --> Source select GitHub Actions

Then visit site page



Open

<https://vara221.github.io/git-fetch/>



### Git Pull Requests

Pull Requests are a feature built in to products like Github and Bitbuket.

They are not native to Git itself.

Mainly used to allow

They allow developers to alert team members to work that needs to be reviewed.

They provide a mechanism to approve or reject the work on a given branch.

They also help facilitate discussion and feedback on the specified commits.

Pull requests is nothing but merging in feature branches

At some point the work we did on the feature branch need to be merge into the master branch

**There are couple of options for how to do this**

1. Merge at will, without any sort of discussion with teammates.just do it whatever you want.
2. Send an email or chat message or something to your team to discuss if the changes should be merged in
3. Pull Requests

#### Pull Requests Workflow

1. Do some work locally on a feature branch
2. Push up the feature branch to Github
3. Open a pull request using the feature branch just pushed up to Github.
4. Wait for the PR to be approved and merged . Start a discussion on the PR.This part depends on the team structure.

#### Implement the pull requests

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-remote/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (registration)

$ ls

index.txt index1.txt vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (registration)

$ git switch main

Switched to branch 'main'

Your branch is up to date with 'origin/main'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ ls

index.txt index1.txt main.txt vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git switch registration

Switched to branch 'registration'

Your branch is up to date with 'origin/registration'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (registration)

$ ls

index.txt index1.txt vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (registration)

$ vi registationnew.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (registration)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (registration)

$ git commit -m "register done"

[registration a2c10ca] register done

1 file changed, 1 insertion(+)

create mode 100644 registationnew.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (registration)

$ git push

Enumerating objects: 4, done.

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

Delta compression using up to 4 threads

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

Writing objects: 100% (3/3), 340 bytes | 68.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

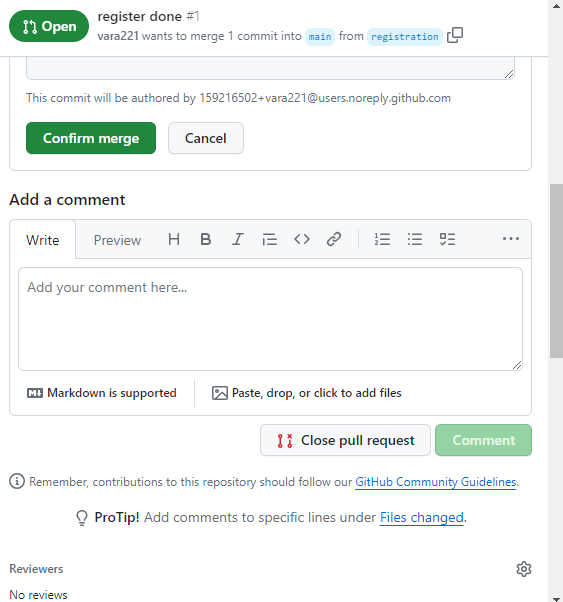
To https://github.com/vara221/git-remote.git

de52ef6..a2c10ca registration -> registration

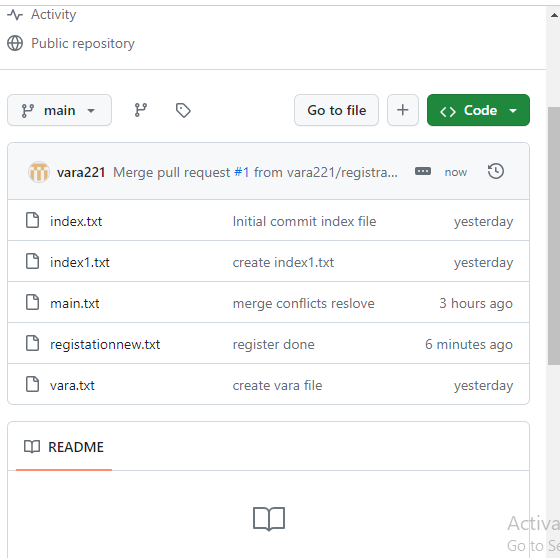


To see the all commit and Files changed contains also

And Merge the pull request to the main



Give the new commit and commit the changes and click on the conform merge



Main also having registrationnew.txt file after merging

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (registration)

$ git switch -

Switched to branch 'main'

Your branch is up to date with 'origin/main'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ ls

index.txt index1.txt main.txt vara.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ git pull

remote: Enumerating objects: 4, done.

remote: Counting objects: 100% (4/4), done.

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

remote: Total 2 (delta 1), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (2/2), 976 bytes | 4.00 KiB/s, done.

From https://github.com/vara221/git-remote

6f5f455..eaaf865 main -> origin/main

Updating 6f5f455..eaaf865

Fast-forward

registationnew.txt | 1 +

1 file changed, 1 insertion(+)

create mode 100644 registationnew.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-remote (main)

$ ls

index.txt index1.txt main.txt registationnew.txt vara.txt

#### Resolve conflicts for the feature branch when pull request is raised in Github repository

Open github goto the git-clonning repository and create one new branch

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-clonning/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (master)

$ git fetch

From https://github.com/vara221/git-clonning

\* [new branch] bugfix -> origin/bugfix

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (master)

$ git branch -r

origin/bugfix

origin/main

origin/master

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (master)

$ ls

index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (master)

$ git switch bugfix

branch 'bugfix' set up to track 'origin/bugfix'.

Switched to a new branch 'bugfix'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (bugfix)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (bugfix)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (bugfix)

$ git commit -m "changes done"

[bugfix d04f4dd] changes done

1 file changed, 1 insertion(+)

Then goto github edit the index.txt file

Come to gitbash

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (bugfix)

$ git push

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

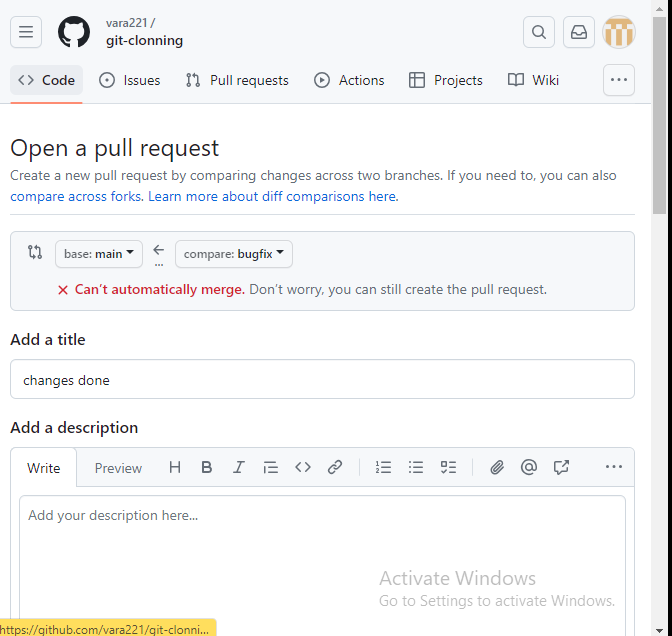
Writing objects: 100% (3/3), 266 bytes | 133.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-clonning.git

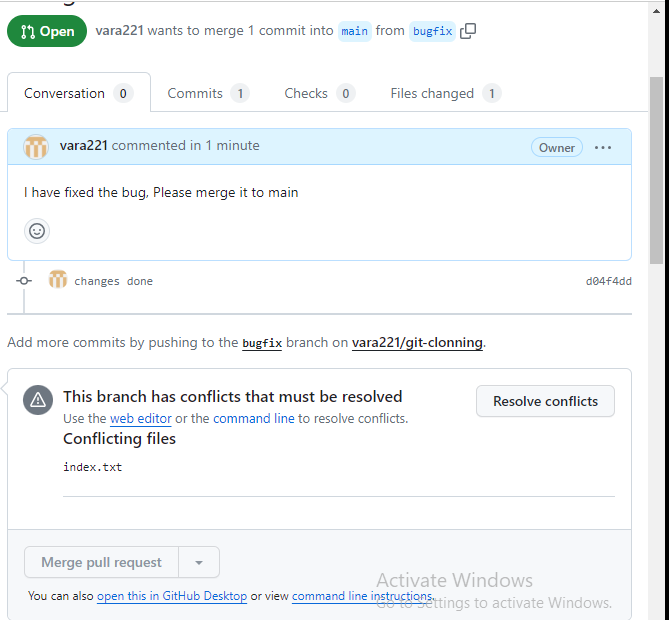
3f995e4..d04f4dd bugfix -> bugfix

Reload the github and go to the pull request it shows **Can’t automatically merge**



But still create pull request

Add description and click on the create pull request



Here conflict may be resolved

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-clonning (bugfix)

$ git merge master

Already up to date.

#### Add Collaborators and apply branch protection rules in Github for restricting commits

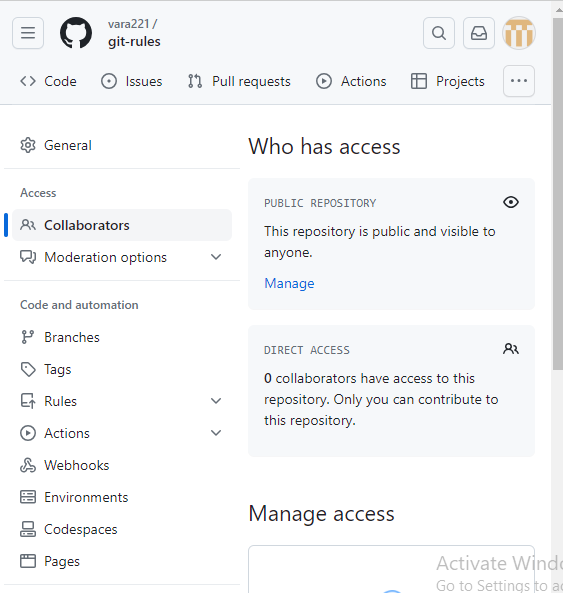
Create one repository name git-rules and go to

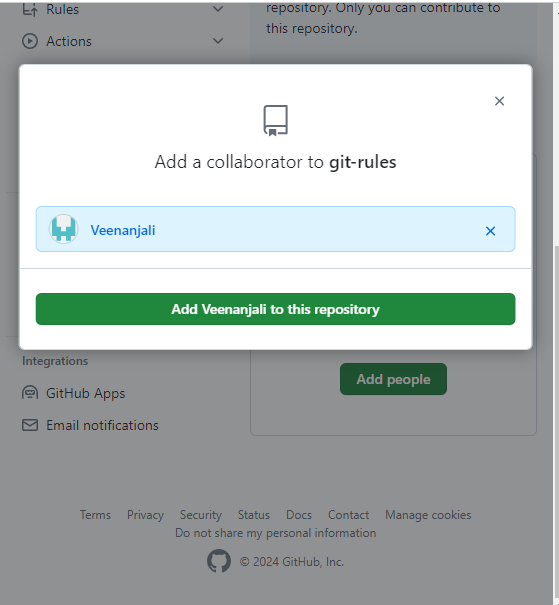
Settings --> Access --> Collaborators

It shows 0 collaborators

So I am invite a collaborator to git-rules

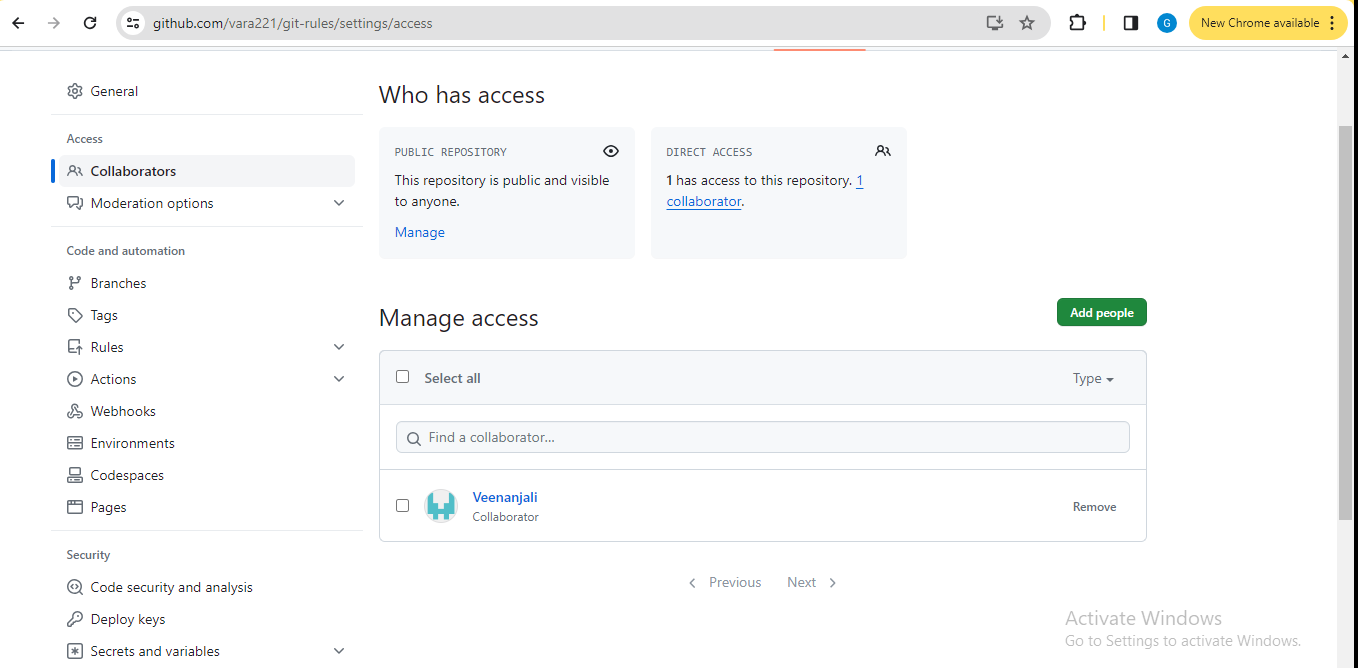
Then add people





Click on the add veenanjali to this repository

One email will be sent to the user Accept that invitation



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/vara221/git-rules.git

Cloning into 'git-rules'...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), done.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-rules/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ git config user.email "varalaxmiganta066@gmail.com"

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ git commit -m "commiting the code in main"

[main 6aa59e9] commiting the code in main

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ git push

Enumerating objects: 4, done.

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

Delta compression using up to 4 threads

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

Writing objects: 100% (3/3), 287 bytes | 287.00 KiB/s, done.

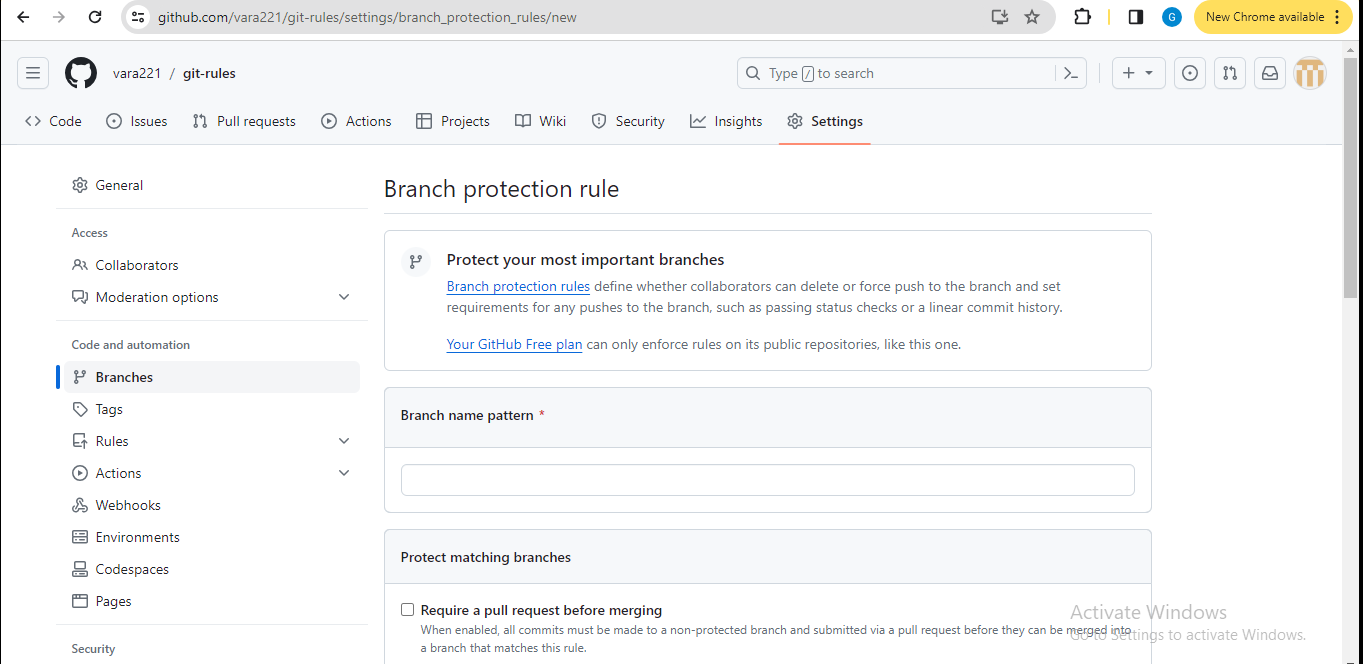
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-rules.git

a597a21..6aa59e9 main -> main

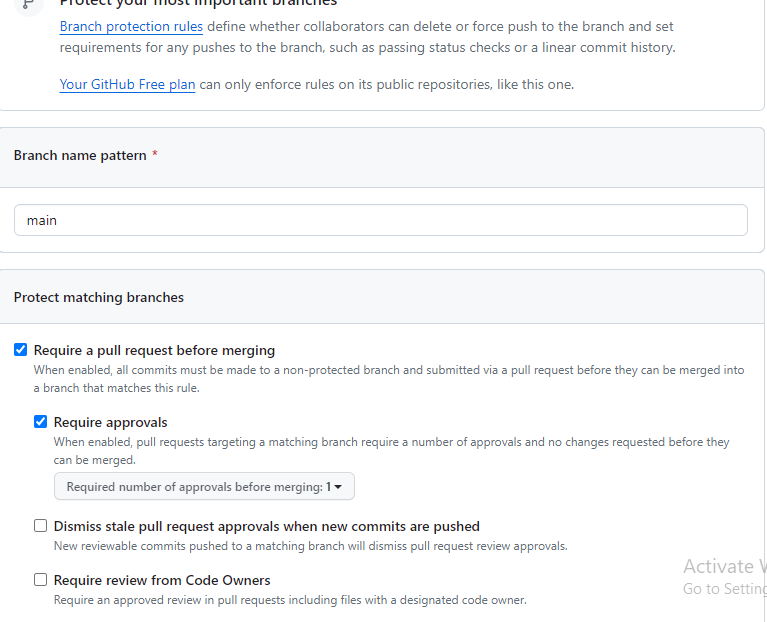
**Add restriction :**

Settings --> Branches --> add Branch protection rules

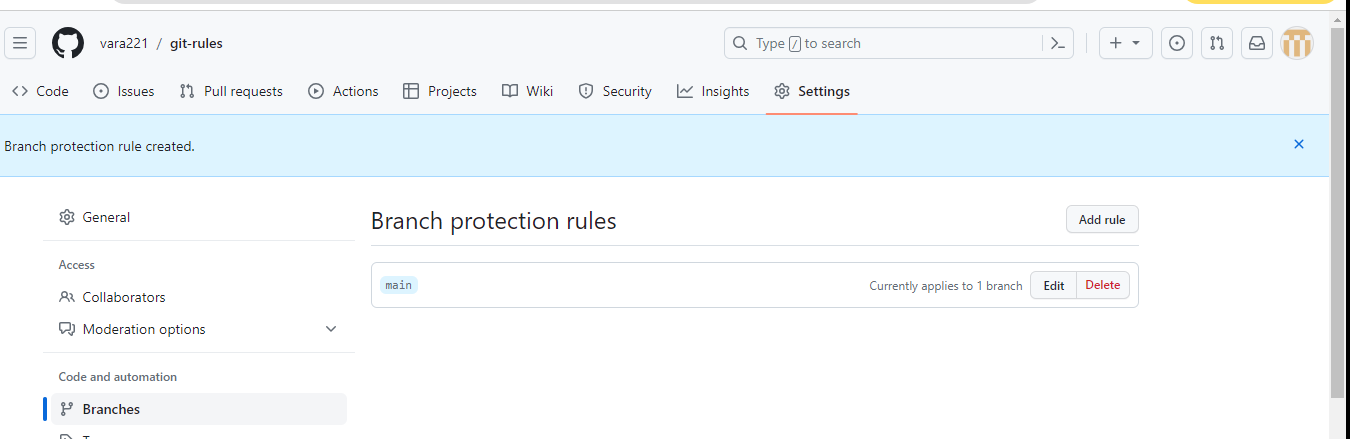


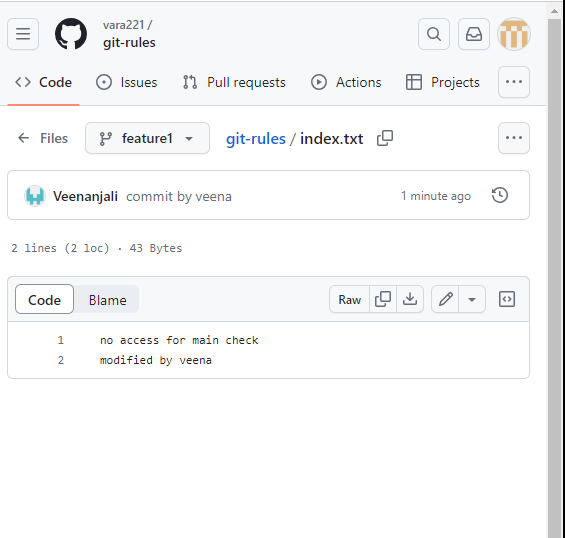
Branch name pattern : main

It means here collaborators can’t be modify the main branch file without merge because owner set the branch protection rules



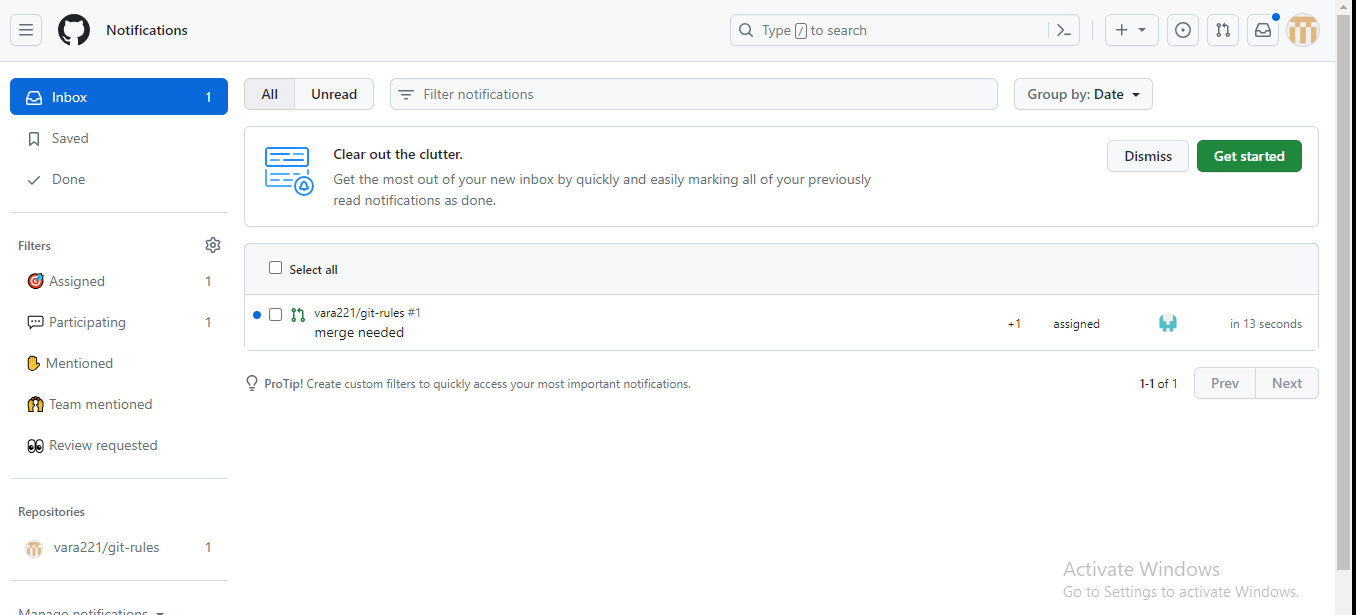
Create it.

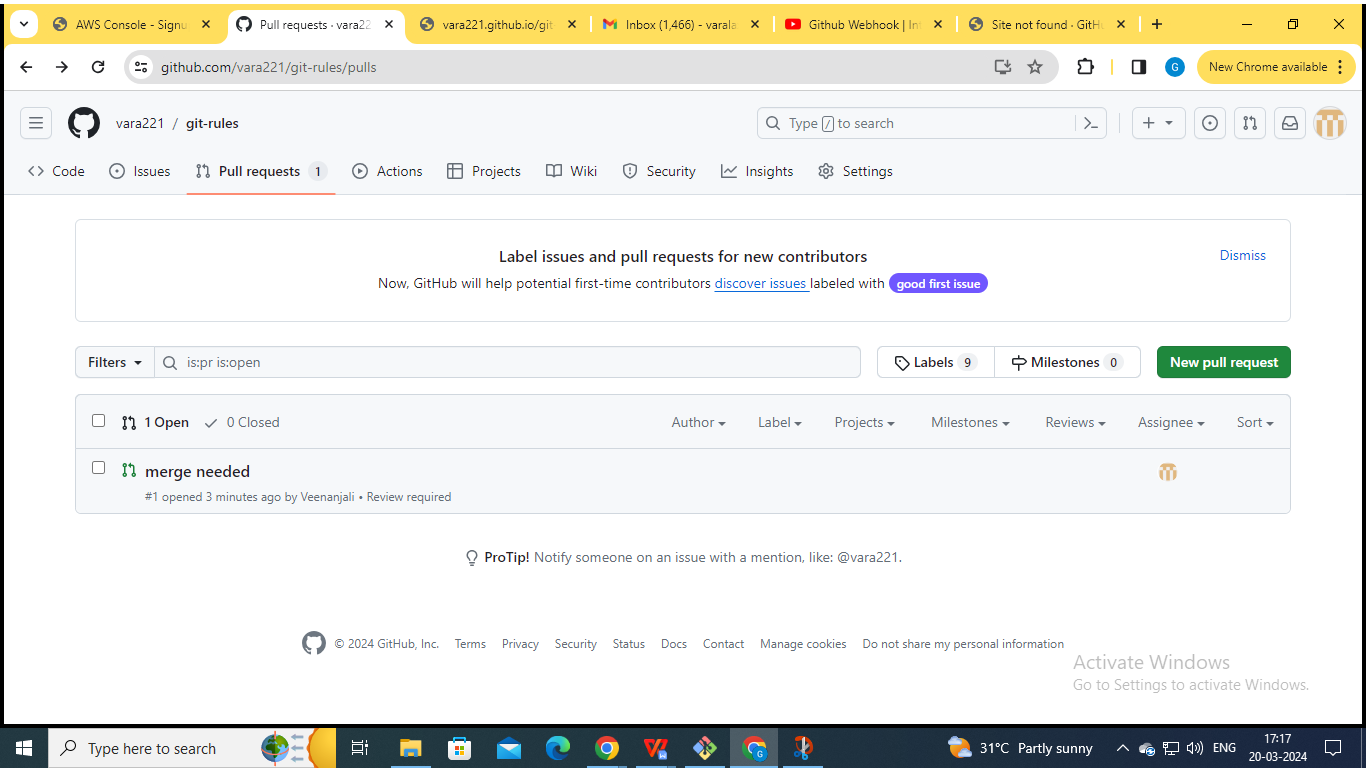


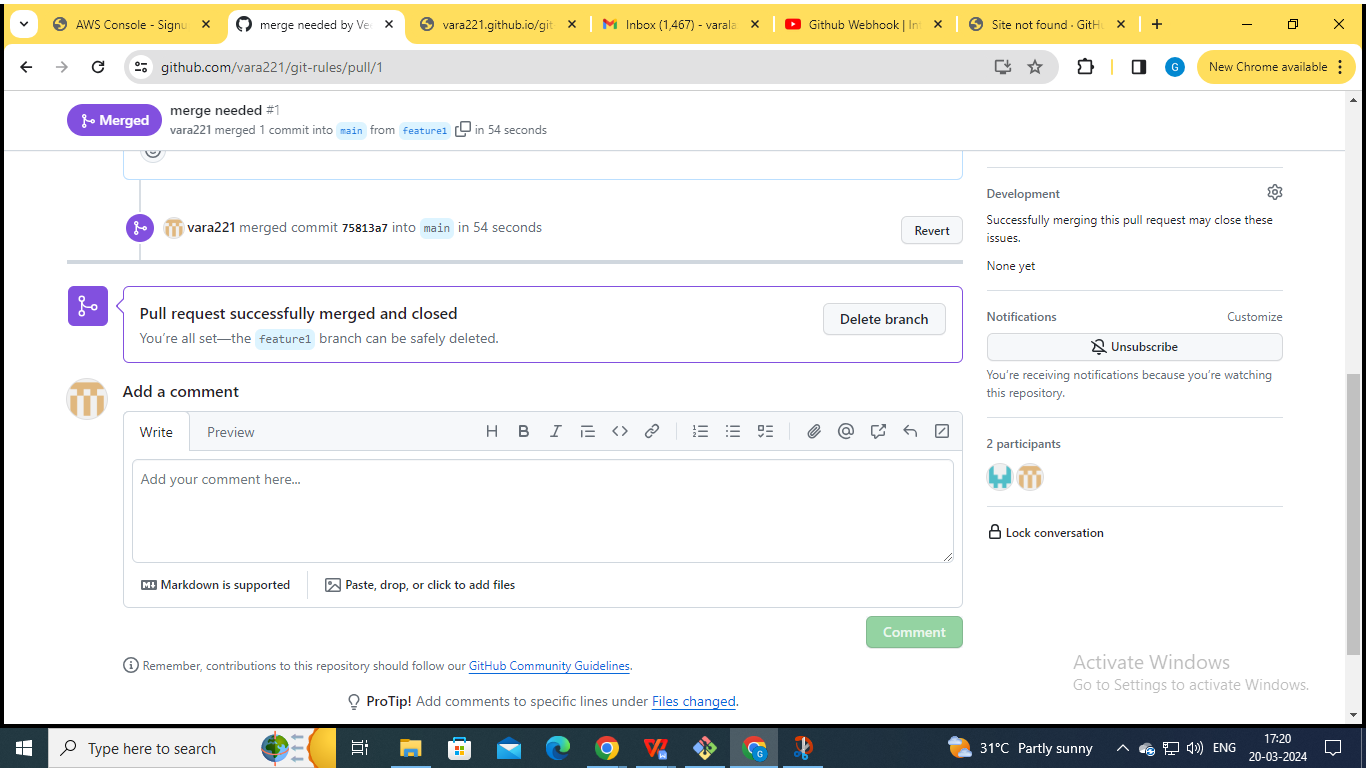


Here create new branch and git add and commit after

If user can send the request to merge then click on merge needed message







vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ git pull

remote: Enumerating objects: 6, done.

remote: Counting objects: 100% (6/6), done.

remote: Compressing objects: 100% (3/3), done.

remote: Total 4 (delta 0), reused 3 (delta 0), pack-reused 0

Unpacking objects: 100% (4/4), 1.15 KiB | 5.00 KiB/s, done.

From https://github.com/vara221/git-rules

1e98ddd..75813a7 main -> origin/main

\* [new branch] feature1 -> origin/feature1

Updating 1e98ddd..75813a7

Fast-forward

index.txt | 1 +

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ ls

README.md index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ cat index.txt

no access for main check

modified by veena

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-rules (main)

$ git log --oneline

75813a7 (HEAD -> main, origin/main, origin/HEAD) Merge pull reque

st #1 from vara221/feature1

3792872 (origin/feature1) commit by veena

1e98ddd check access for main branch

6aa59e9 commiting the code in main

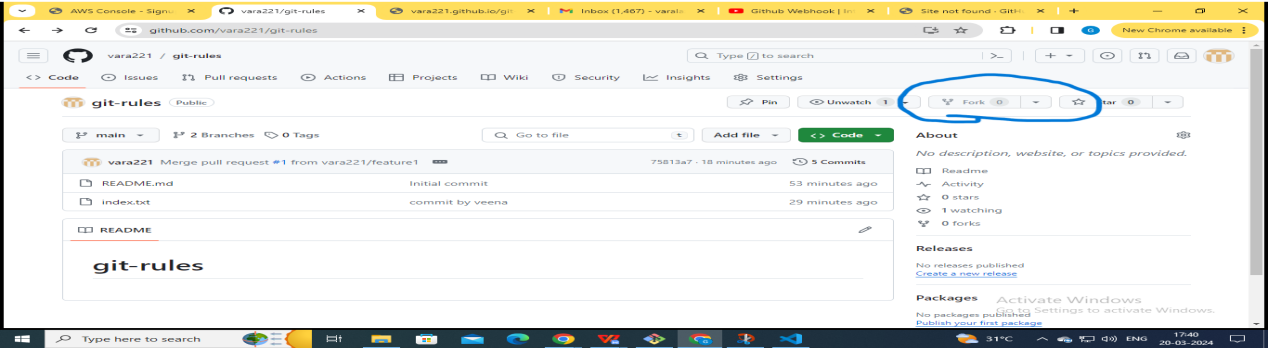
a597a21 Initial commit

### Git Forking

When you are having large open source projects with lots of contributors

They employ this forking strategy or work flow where there might be a handful of actual maintainers.

They cannot add thousands of people as direct contributors or collaborators.



This forking workflow enables anybody to try and make a contribution for the repository

There is no permissions needed.you can make your own copy .you try making changes and then you make a pull request

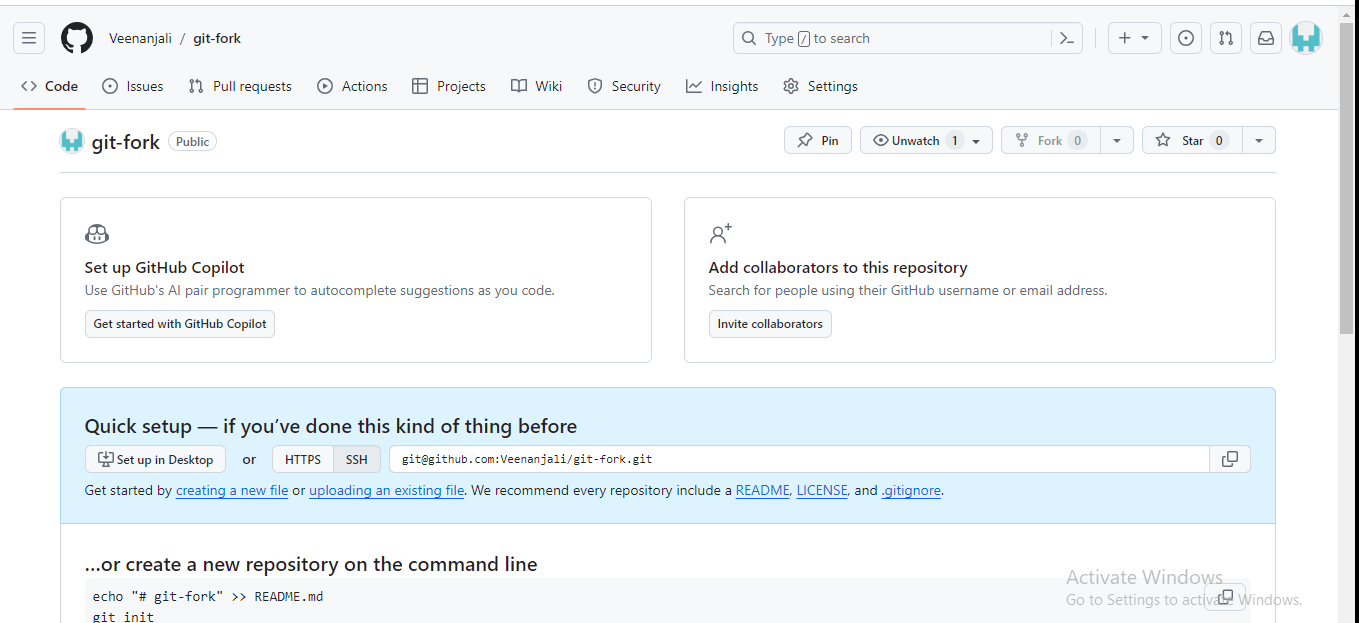
Anyone can make a pull request

Github and other similar tools allow us to create personal copies of other peoples repositories.

We call those copies as a fork of the original

When we fork a repository , we basically asking github “make me my own copy of this repository please”

**I am using another account to track the fork usage.**



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/Veenanjali/git-fork.git

Cloning into 'git-fork'...

warning: You appear to have cloned an empty repository.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-fork

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git commit -m "Initial commit"

[main (root-commit) 9fe610d] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 index.txt

**Here, permission is denied because we are in Vara221 git.**

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git push

remote: Permission to Veenanjali/git-fork.git denied to vara221.

fatal: unable to access 'https://github.com/Veenanjali/git-fork.gi

t/': The requested URL returned error: 403

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

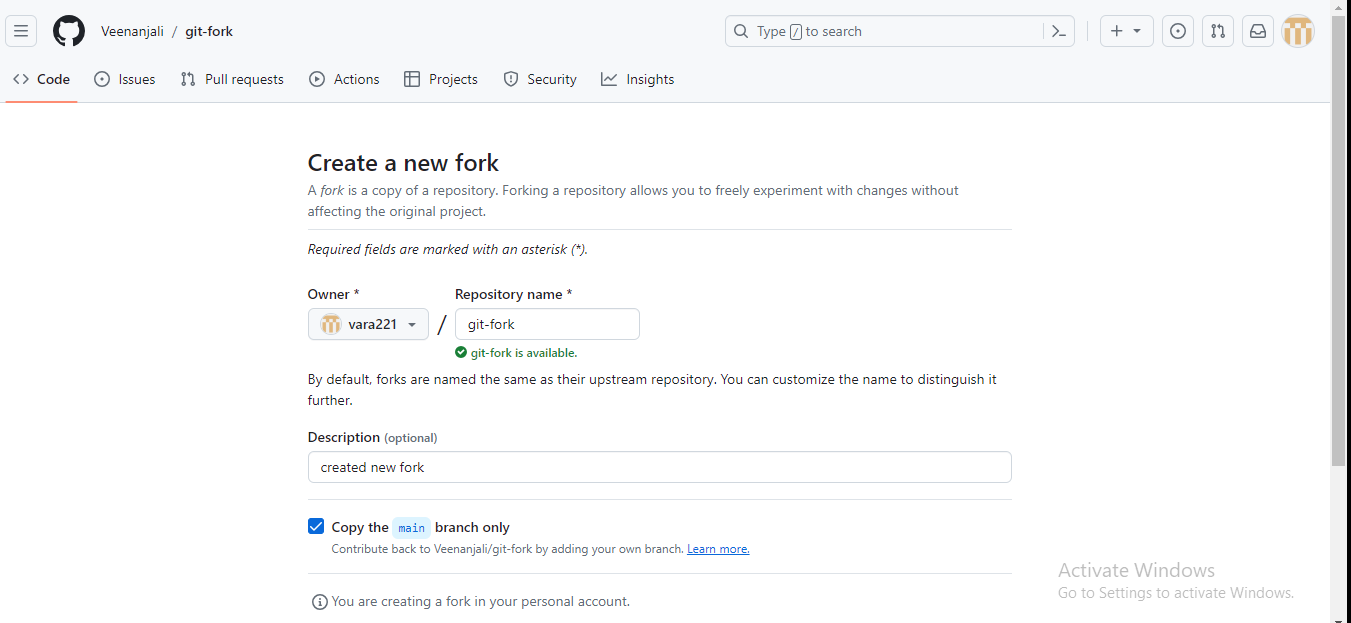
$ cd ../

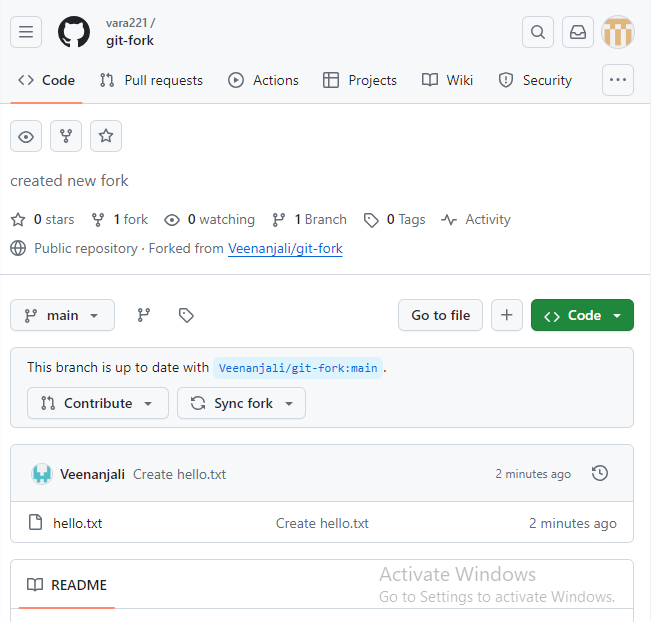
vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ rm -rf git-fork

**Go to the Vara221**

**git hub, then click on the fork option.**





vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/vara221/git-fork.git

Cloning into 'git-fork'...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), done.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-fork

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git commit -m "created index.txt file"

[main 7d95aad] created index.txt file

1 file changed, 1 insertion(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git push

Enumerating objects: 4, done.

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

Delta compression using up to 4 threads

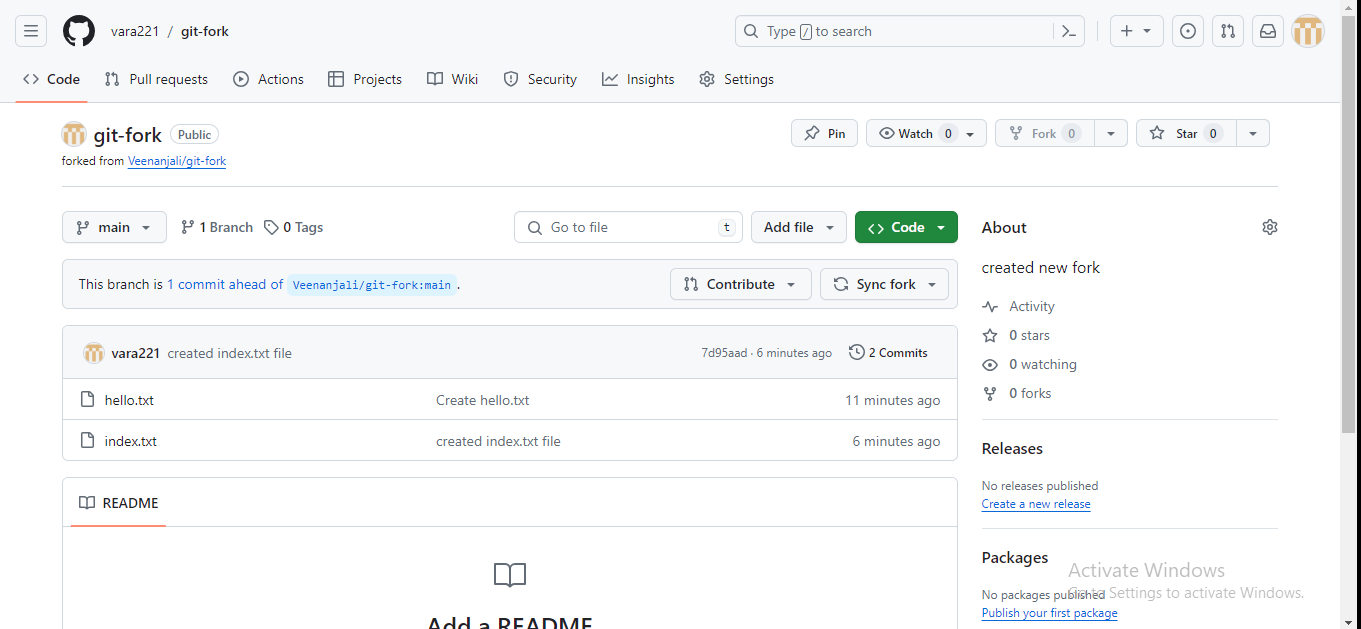
Compressing objects: 100% (2/2), done.

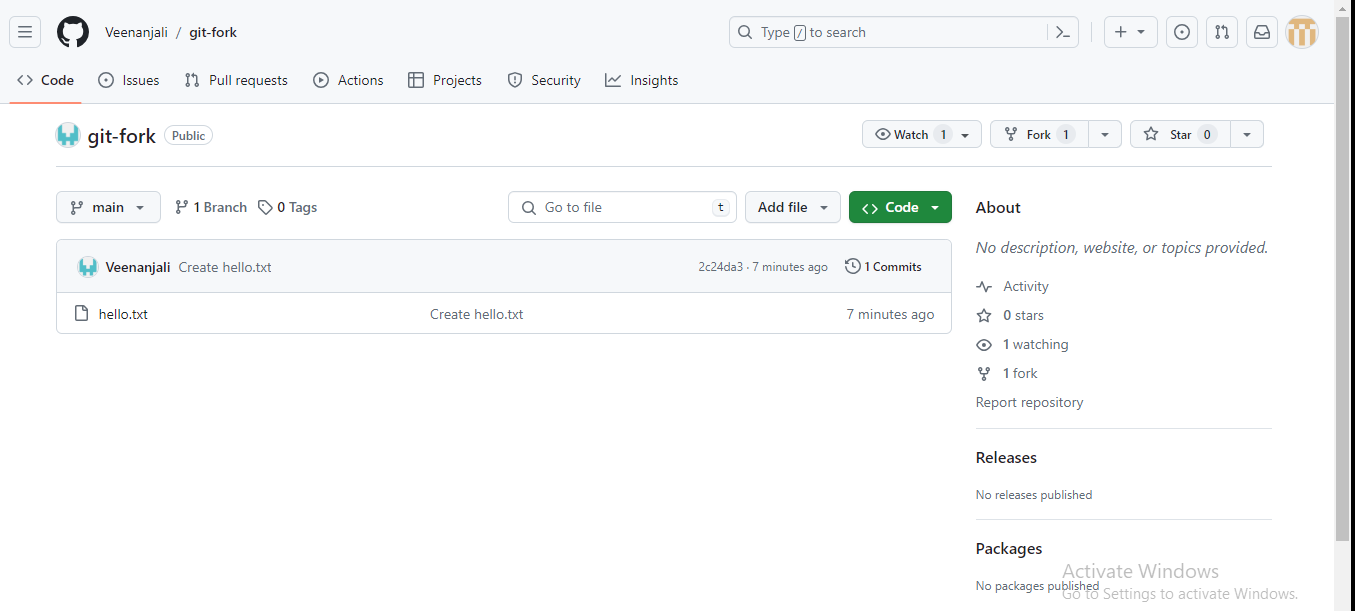
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-fork.git

2c24da3..7d95aad main -> main

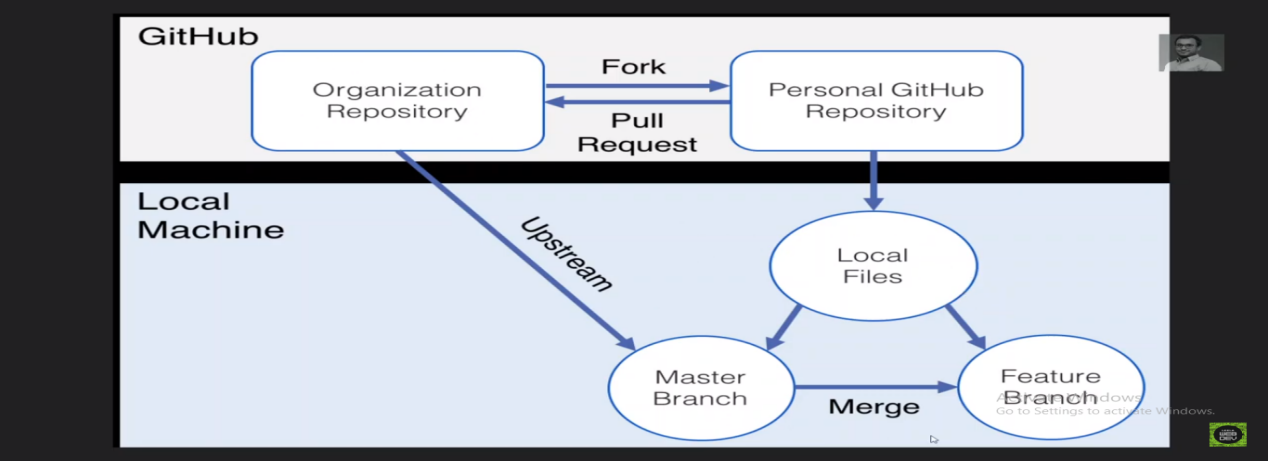




I can clone my fork and make changes,add features and break things without fear of disturbing the original repository

If I want to share my work, I can make a pull request from my fork to the original repository.

This means that whole bunch of people can fork and can work on the project without actually having permissions to them.



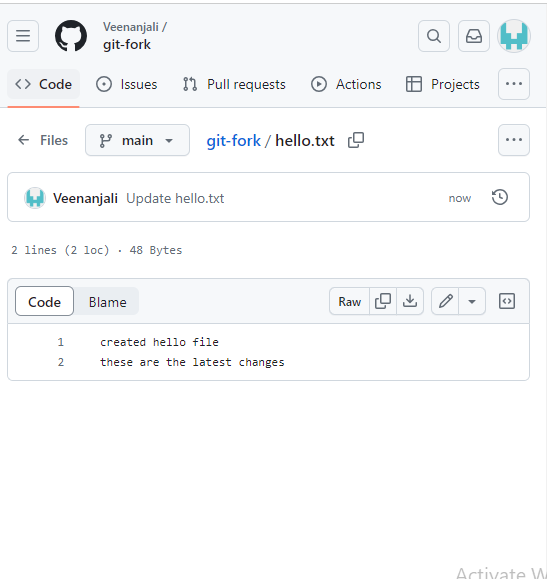
Organization repository : veenanjali

Personal GitHub repository : vara221

The “fork and clone” Work flow might seem complicated, but it’s extremely common for good reason.

It allows a project manager to accept contributions from developers all around the world without having adding them as actual owners of the main project repository or worry about giving them all permissions to push to the Repository.

Go to the veenanjali website



Modify the hello.txt

Clone the url <https://github.com/Veenanjali/git-fork.git>

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git remote -v

origin https://github.com/vara221/git-fork.git (fetch)

origin https://github.com/vara221/git-fork.git (push)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git remote add upstream https://github.com/Veenanjali/git-fork.git

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git remote -v

origin https://github.com/vara221/git-fork.git (fetch)

origin https://github.com/vara221/git-fork.git (push)

upstream https://github.com/Veenanjali/git-fork.git (fetch)

upstream https://github.com/Veenanjali/git-fork.git (push)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git pull upstream main

remote: Enumerating objects: 5, done.

remote: Counting objects: 100% (5/5), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Unpacking objects: 100% (3/3), 934 bytes | 6.00 KiB/s, done.

From https://github.com/Veenanjali/git-fork

\* branch main -> FETCH\_HEAD

\* [new branch] main -> upstream/main

Merge made by the 'ort' strategy.

hello.txt | 1 +

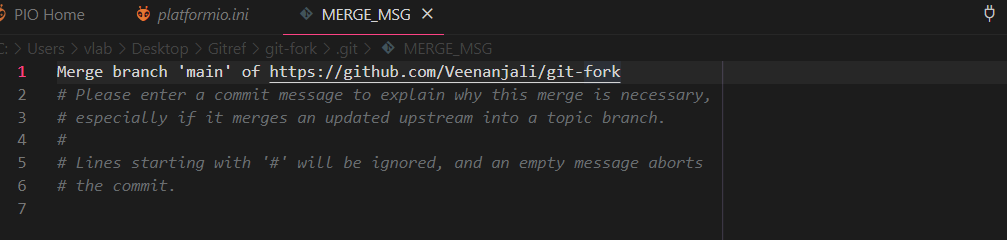
1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ cat hello.txt

created hello file

these are the latest changes



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git log --oneline

7d533c8 (HEAD -> main) Merge branch 'main' of https://github.com/V

eenanjali/git-fork

acc5b80 (upstream/main) Update hello.txt

7d95aad (origin/main, origin/HEAD) created index.txt file

2c24da3 Create hello.txt

Change the hello.txt file

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ vi hello.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git commit -m "changes add in hello.txt"

[main 886f900] changes add in hello.txt

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-fork (main)

$ git push origin main

Enumerating objects: 9, done.

Counting objects: 100% (9/9), done.

Delta compression using up to 4 threads

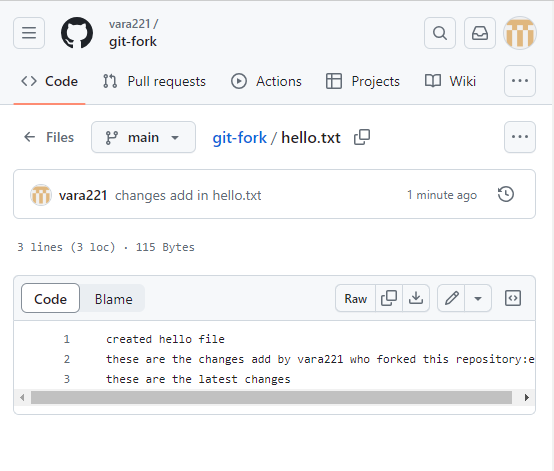
Compressing objects: 100% (5/5), done.

Writing objects: 100% (5/5), 673 bytes | 336.00 KiB/s, done.

Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

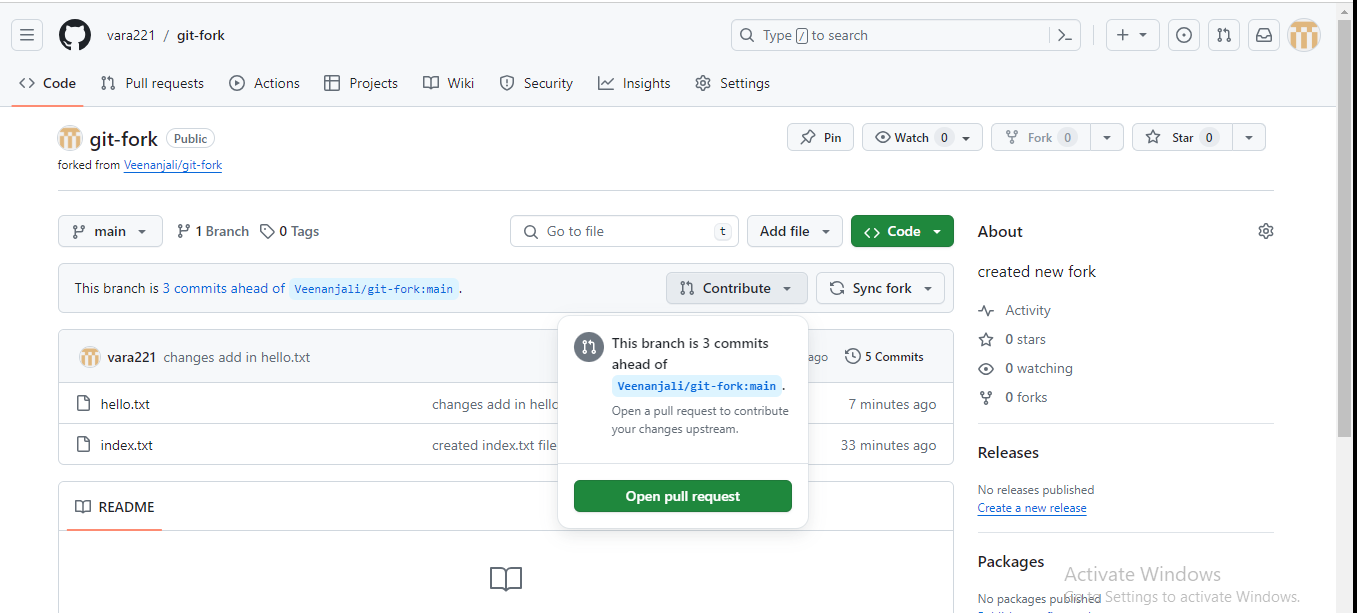
To https://github.com/vara221/git-fork.git

7d95aad..886f900 main -> main

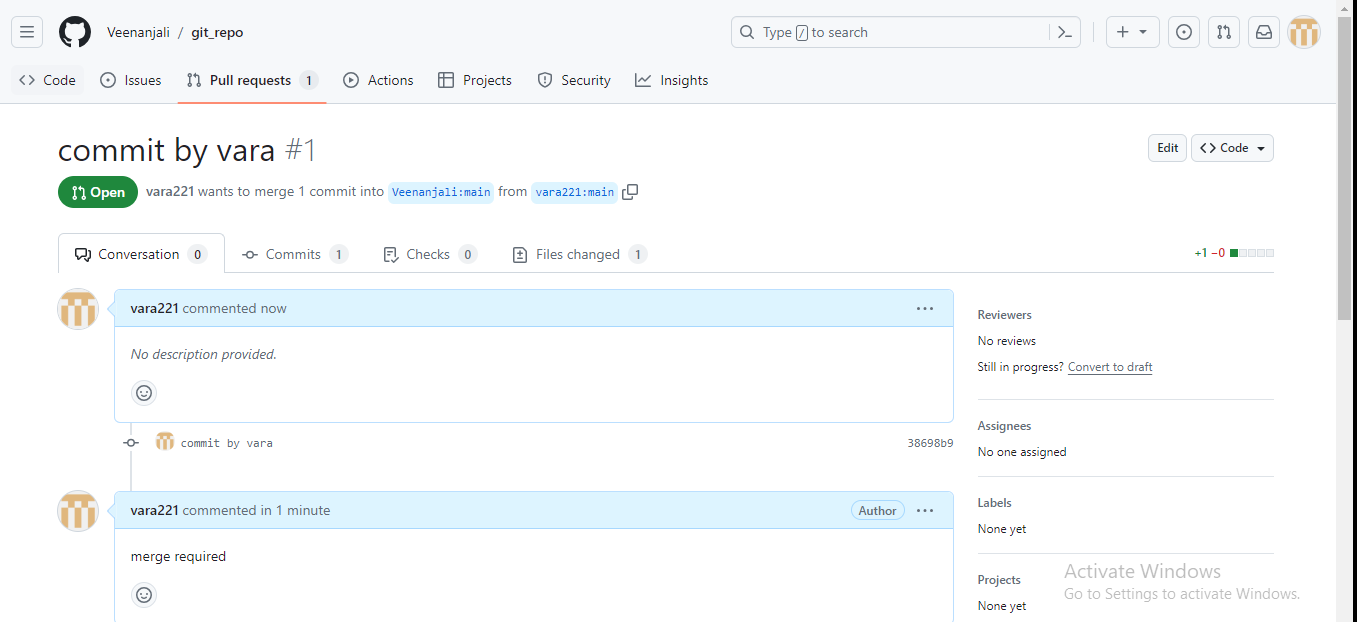


**So that here I want to raise a pull request to Veenanjali from Vara221**

Go to the contribute



Click on the open pull request and create a pull request



After merging again, clone the Veenanjali repository, then verify the changes.

### Git Tags

Git tags main idea is that we can tag particular commit so we can label commits by creating a tag, a reference to a moment in time.

Tags are pointers that refer to particular points in Git history.we can mark a particular moment in time with a tag.

Tags are most often used to mark version releases in projects (v4.1.0,v4.1.1 etc)

Think of tags as branch references that do not change . Once a tag is created , it always refers to the same commit .

It’s just label for a commit.

There are two types of tags

1. Lightweight Tags : lightweight tags are light weight.They are just a name / label that points to a particular commit. It is much like a branch that doesn’t change -its just a pointer to a commit.
2. Annotated Tags :Annotated tags store extra meta data including the author name and email , the date and a tagging message ( like a commit message) . Annotated tags, however are stored as full objects in the git database. It’s generally recommended that you create annotated tags so you have full information.

#### Viewing Tags

* Git tag : git tag will print a list of all tags in the current repository .
* We can search for tags that match a particular by using git tag -l and then passing wild-card. For example git tag -l “\*beta\*” will print a list of tags that include beta in their name
* Git checkout <tag> : to view the state of a repository at a particular tag,we can use git checkout <tag>. This puts us in detached Head.

For example open this link [git@github.com:facebook/react.git](mailto:git@github.com:facebook/react.git)

Create one fork and clone that code

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ git clone https://github.com/vara221/react.git

Cloning into 'react'...

remote: Enumerating objects: 173169, done.

remote: Counting objects: 100% (204/204), done.

remote: Compressing objects: 100% (142/142), done.

remote: Total 173169 (delta 99), reused 119 (delta 61), pack-reuse

d 172965

Receiving objects: 100% (173169/173169), 154.90 MiB | 6.07 MiB/s,

done.

Resolving deltas: 100% (130961/130961), done.

Updating files: 100% (2595/2595), done.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd react/

View all the tags

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/react (main)

$ git tag

#### Creating lightweight tags

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ mkdir git-tag

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref (master)

$ cd git-tag

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git init

Initialized empty Git repository in C:/Users/vlab/Desktop/Gitref/g

it-tag/.git/

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ ls

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git add .

g

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git commit -m "Initial commit"

[master (root-commit) 648cc11] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git log

commit 648cc1185f86a1566015c87ede44b8071b953125 (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:15:15 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag v1.0.0

For patch release v1.0.1

For major release v2.0.0

For minor release v1.1.0

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag

v1.0.0

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git add .

gi

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git commit -m "this is the second commit"

[master da71058] this is the second commit

1 file changed, 1 insertion(+)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git log

commit da710580eebdfc0dece0ab2d897e10bfa4842c03 (HEAD -> master)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:16:44 2024 +0530

this is the second commit

commit 648cc1185f86a1566015c87ede44b8071b953125 (tag: v1.0.0)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:15:15 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git checkout v1.0.0

Note: switching to 'v1.0.0'.

You are in 'detached HEAD' state. You can look around, make experi

mental

changes and commit them, and you can discard any commits you make

in this

state without impacting any branches by switching back to a branch

.

If you want to create a new branch to retain commits you create, y

ou may

do so (now or later) by using -c with the switch command. Example:

git switch -c <new-branch-name>

Or undo this operation with:

git switch -

Turn off this advice by setting config variable advice.detachedHea

d to false

HEAD is now at 648cc11 Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag ((v1.0.0))

$ vi index.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag ((v1.0.0))

$ cat index.txt

this is the first commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag ((v1.0.0))

$ git checkout -

Previous HEAD position was 648cc11 Initial commit

Switched to branch 'master'

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ cat index.txt

this is the first commit

this is the second commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag

v1.0.0

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git log --oneline

da71058 (HEAD -> master) this is the second commit

648cc11 (tag: v1.0.0) Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag v1.0.1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag

v1.0.0

v1.0.1

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git diff v1.0.0 v1.0.1

diff --git a/index.txt b/index.txt

index ccee652..fce55d4 100644

--- a/index.txt

+++ b/index.txt

@@ -1 +1,2 @@

this is the first commit

+this is the second commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git log --oneline

da71058 (HEAD -> master, tag: v1.0.1) this is the second commit

648cc11 (tag: v1.0.0) Initial commit

#### Create Annotated Tags

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag

v1.0.0

v1.0.1

These lightweight tags

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ vi newfeature.txt

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git add .

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git commit -m "new feature done"

[master 54d59e6] new feature done

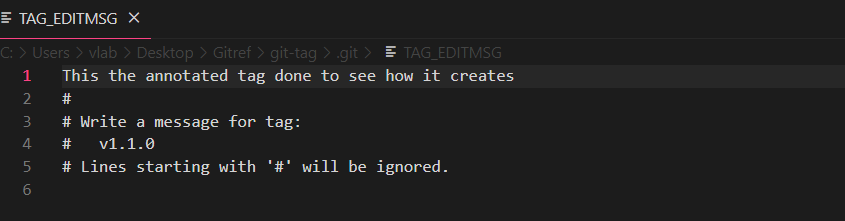
1 file changed, 1 insertion(+)

create mode 100644 newfeature.txt

By using option -a to create a annotated tag

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag -a v1.1.0



vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git show v1.1.0

tag v1.1.0

Tagger: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:26:28 2024 +0530

This the annotated tag done to see how it creates

commit 54d59e6ff6a12bc4c99e20923f3899257b3a5a06 (HEAD -> master, t

ag: v1.1.0)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:26:07 2024 +0530

new feature done

diff --git a/newfeature.txt b/newfeature.txt

new file mode 100644

index 0000000..c099020

--- /dev/null

+++ b/newfeature.txt

@@ -0,0 +1 @@

+this is first line

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git show v1.0.0

commit 648cc1185f86a1566015c87ede44b8071b953125 (tag: v1.0.0)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:15:15 2024 +0530

Initial commit

diff --git a/index.txt b/index.txt

new file mode 100644

index 0000000..ccee652

--- /dev/null

+++ b/index.txt

@@ -0,0 +1 @@

+this is the first commit

#### Pushing Tags to the GIthub Remote Repository

Create one repository with the name git-tag in the Github

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git remote add origin https://github.com/vara221/git-tags.git

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git remote -v

origin https://github.com/vara221/git-tags.git (fetch)

origin https://github.com/vara221/git-tags.git (push)

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git log

commit 54d59e6ff6a12bc4c99e20923f3899257b3a5a06 (HEAD -> master, t

ag: v1.1.0)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:26:07 2024 +0530

new feature done

commit da710580eebdfc0dece0ab2d897e10bfa4842c03 (tag: v1.0.1)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:16:44 2024 +0530

this is the second commit

commit 648cc1185f86a1566015c87ede44b8071b953125 (tag: v1.0.0)

Author: varalaxmi ganta <varalaxmiganta066@gmail.com>

Date: Thu Mar 21 14:15:15 2024 +0530

Initial commit

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git push -u origin master

Enumerating objects: 9, done.

Counting objects: 100% (9/9), done.

Delta compression using up to 4 threads

Compressing objects: 100% (5/5), done.

Writing objects: 100% (9/9), 763 bytes | 127.00 KiB/s, done.

Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-tags.git

\* [new branch] master -> master

branch 'master' set up to track 'origin/master'.

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git tag

v1.0.0

v1.0.1

v1.1.0

When tags are not pushed by default, they are pushed separately.

#### Moving the tags into two ways

1. Moving all tags at a time
2. Moving a particular tag

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git push origin v1.0.0

Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

To https://github.com/vara221/git-tags.git

\* [new tag] v1.0.0 -> v1.0.0

vlab@VEDA2F118 MINGW32 ~/Desktop/Gitref/git-tag (master)

$ git push origin --tags

Enumerating objects: 1, done.

Counting objects: 100% (1/1), done.

Writing objects: 100% (1/1), 193 bytes | 193.00 KiB/s, done.

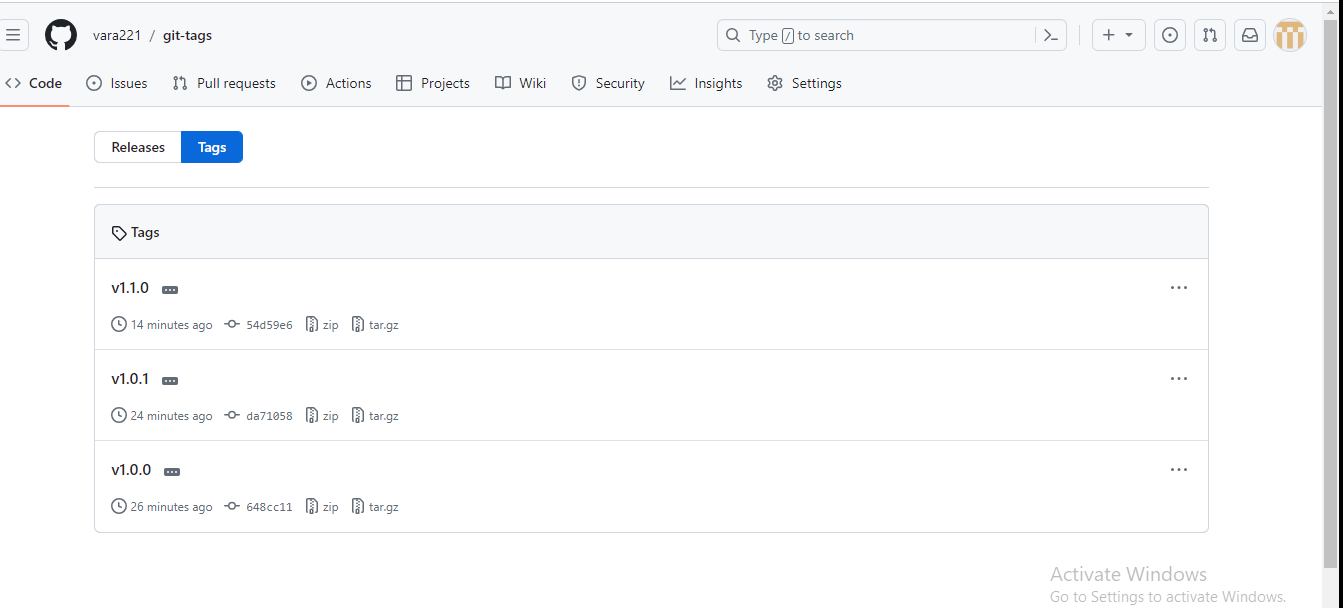
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)

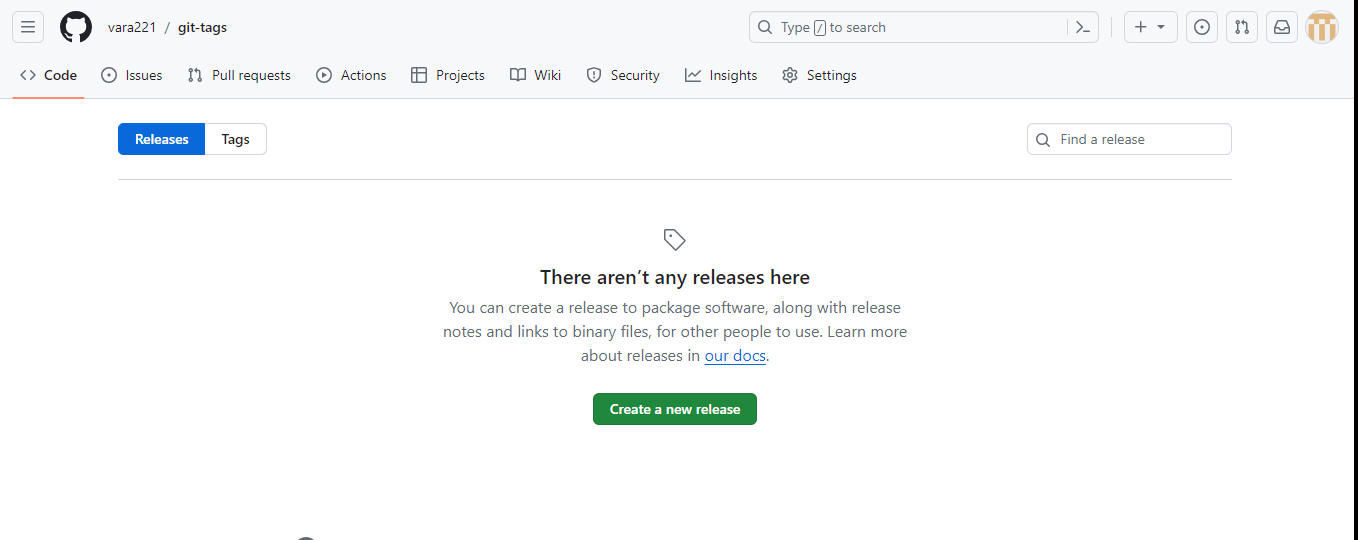
To https://github.com/vara221/git-tags.git

\* [new tag] v1.0.1 -> v1.0.1

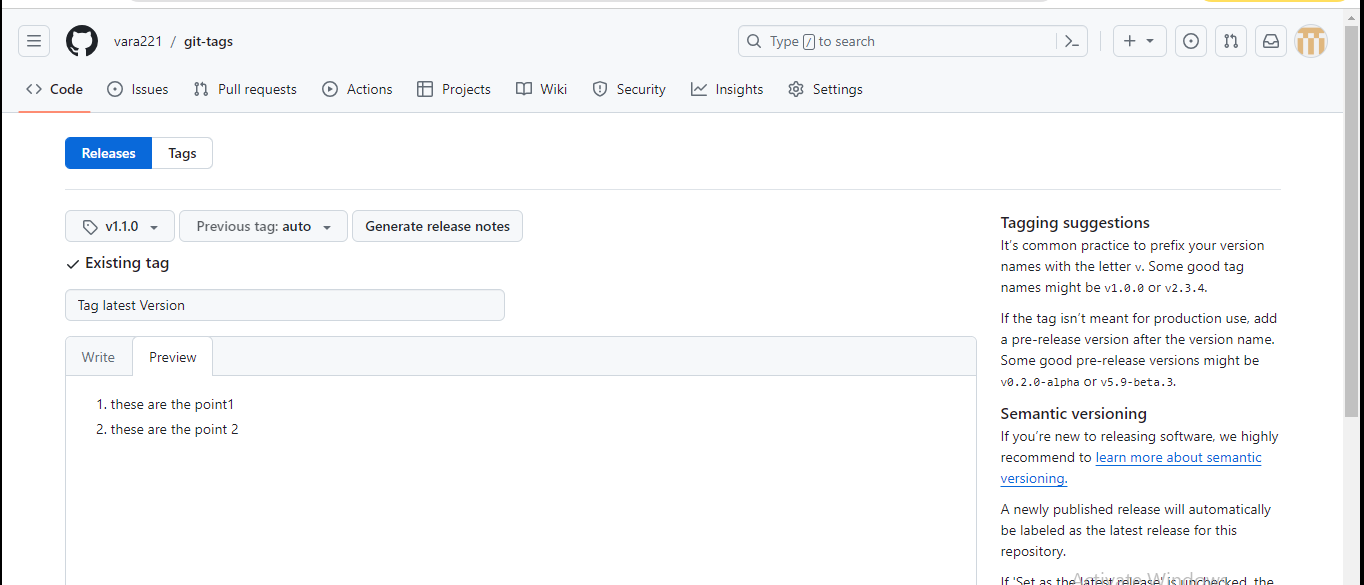
\* [new tag] v1.1.0 -> v1.1.0

**See all the tags in the github. Go to the tags and open it.**

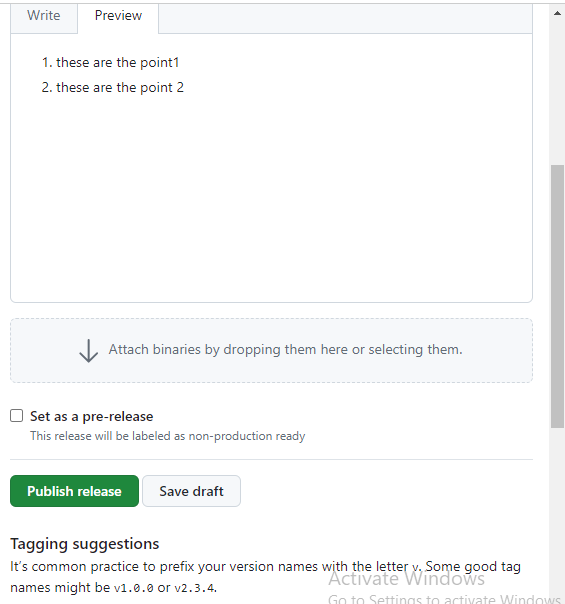


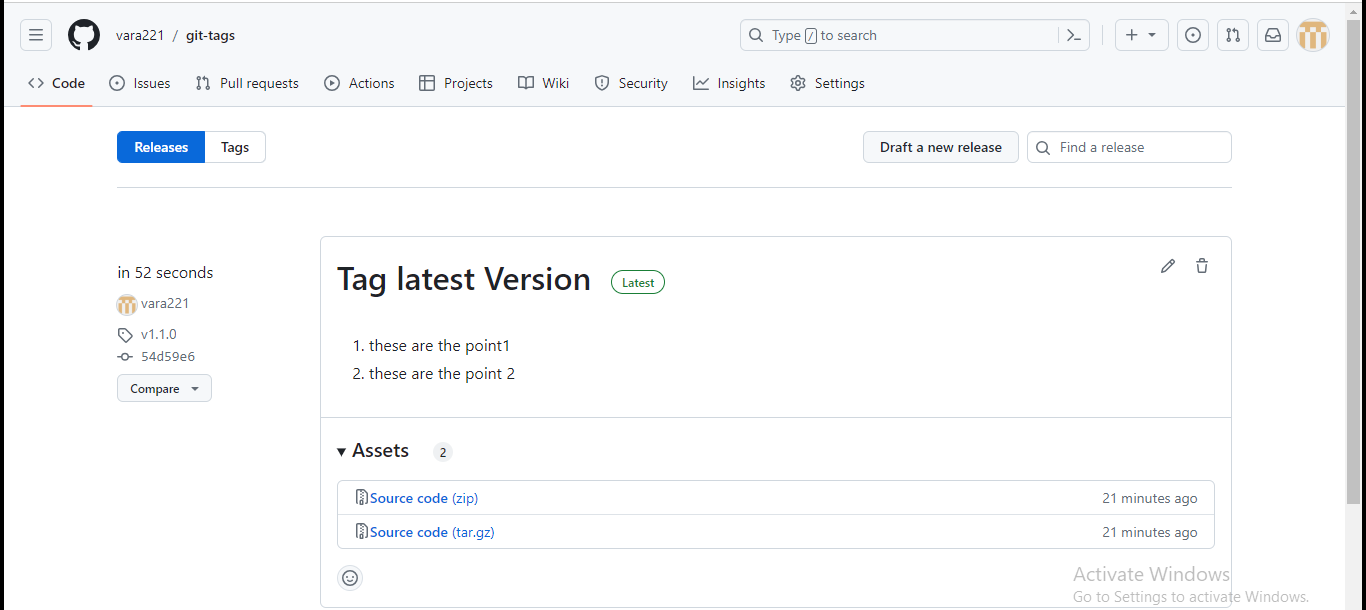


To see the releases then draft that release so click on create a new release



After click on the publish release





Finally go to tags it’s show the latest version

### Git Reflog

* The term Reflogs is a Short form for Reference Logs.
* These are just logs that git keeps us for as a record
* Git keeps a record of when the tips of branches and other references were updated in the repository.
* We can view and update these reflogs using the git reflog command
* Git only keeps reflogs on your local activity .They are not shared with collaborators
* Reflogs also expire.Git cleans out old entries after around 90 days, though this can be configured
* The git reflog command accepts sub commands like show,expire,delete,and exists.
* Show is the only command used variant, and it is the default sub command
* Git reflog show command will show the log of a specific reference (it defaults to HEAD)
* For example , to view the logs for the tip of the main branch we could run

Git reflog show main