admin@DESKTOP-VHIRI0L MINGW64 ~

$ cd d:

**(CHANGE DIRECTORY)**

admin@DESKTOP-VHIRI0L MINGW64 /d

$ ls

'FRONTEND TECHNOLOGY'/ 'System Volume Information'/ Batches/

**(LIST FILES AND DIRECTORY)**

admin@DESKTOP-VHIRI0L MINGW64 /d

$ git

usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]

[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]

[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]

[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]

[--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink

restore Restore working tree files

rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)

bisect Use binary search to find the commit that introduced a bug

diff Show changes between commits, commit and working tree, etc

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

grow, mark and tweak your common history

branch List, create, or delete branches

commit Record changes to the repository

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

reset Reset current HEAD to the specified state

switch Switch branches

tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some

concept guides. See 'git help <command>' or 'git help <concept>'

to read about a specific subcommand or concept.

See 'git help git' for an overview of the system.

admin@DESKTOP-VHIRI0L MINGW64 /d

$ git -v

git version 2.40.1.windows.1

admin@DESKTOP-VHIRI0L MINGW64 /d

$ git -h

usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]

[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]

[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]

[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]

[--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)

clone Clone a repository into a new directory

init Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)

add Add file contents to the index

mv Move or rename a file, a directory, or a symlink

restore Restore working tree files

rm Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)

bisect Use binary search to find the commit that introduced a bug

diff Show changes between commits, commit and working tree, etc

grep Print lines matching a pattern

log Show commit logs

show Show various types of objects

status Show the working tree status

grow, mark and tweak your common history

branch List, create, or delete branches

commit Record changes to the repository

merge Join two or more development histories together

rebase Reapply commits on top of another base tip

reset Reset current HEAD to the specified state

switch Switch branches

tag Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)

fetch Download objects and refs from another repository

pull Fetch from and integrate with another repository or a local branch

push Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some

concept guides. See 'git help <command>' or 'git help <concept>'

to read about a specific subcommand or concept.

See 'git help git' for an overview of the system.

admin@DESKTOP-VHIRI0L MINGW64 /d

$ git -c

-c expects a configuration string

usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]

[--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]

[-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]

[--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]

[--config-env=<name>=<envvar>] <command> [<args>]

admin@DESKTOP-VHIRI0L MINGW64 /d

$ git --exec-path

C:/Program Files/Git/mingw64/libexec/git-core

admin@DESKTOP-VHIRI0L MINGW64 /d

$ mkdir GITPROJECT

**(MAKE DIRECTORY)**

admin@DESKTOP-VHIRI0L MINGW64 /d

$ cd GITPROJECT

**(CHANGE DIRECTORY)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT

$ ls

'FRONTEND TECHNOLOGY'/ 'System Volume Information'/ Batches/ GITPROJECT/

**(LIST FILES AND DIRECTORY)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT

$ clear

**(CLEAR SCREEN)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT

$ git status

fatal: not a git repository (or any of the parent directories): .git

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git config –global

usage: git config [<options>]

Config file location

--global use global config file

--system use system config file

--local use repository config file

--worktree use per-worktree config file

-f, --file <file> use given config file

--blob <blob-id> read config from given blob object

Action

--get get value: name [value-pattern]

--get-all get all values: key [value-pattern]

--get-regexp get values for regexp: name-regex [value-pattern]

--get-urlmatch get value specific for the URL: section[.var] URL

--replace-all replace all matching variables: name value [value-pattern]

--add add a new variable: name value

--unset remove a variable: name [value-pattern]

--unset-all remove all matches: name [value-pattern]

--rename-section rename section: old-name new-name

--remove-section remove a section: name

-l, --list list all

--fixed-value use string equality when comparing values to 'value-pattern'

-e, --edit open an editor

--get-color find the color configured: slot [default]

--get-colorbool find the color setting: slot [stdout-is-tty]

Type

-t, --type <type> value is given this type

--bool value is "true" or "false"

--int value is decimal number

--bool-or-int value is --bool or --int

--bool-or-str value is --bool or string

--path value is a path (file or directory name)

--expiry-date value is an expiry date

Other

-z, --null terminate values with NUL byte

--name-only show variable names only

--includes respect include directives on lookup

--show-origin show origin of config (file, standard input, blob, command line)

--show-scope show scope of config (worktree, local, global, system, command)

--default <value> with --get, use default value when missing entry

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git config --global user.name "javedalihosamani"

**(USER NAME CONFIG)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git config --global user.email "javed.hosamani007@gmail.com"

**(USER EMAIL CONFIG)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git config --global –list

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

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

filter.lfs.required=true

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

user.name=javedalihosamani

user.email=javed.hosamani007@gmail.com

**(CONFIG STATUS OF USER NAME AND USER EMAIL)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT

$ git init

Initialized empty Git repository in D:/Batches/12-00 C.V Raman/HTML/.git/

**(INITIALIZED EMPTY GIT -> Helps to track a records of project)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ ls -a

./ ../ .git/

**(DISPLAYING HIDDEN FOLDERS)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ ls .git

HEAD config description hooks/ info/ objects/ refs/

**(DISPLAYING HIDDEN FOLDER CONTENT)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ touch names.txt

**(NAME.TXT FILE IS BEING CREATED)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

No commits yet

Untracked files:

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

names.txt

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

**(DISPLAYS THE STATE OF THE WORKING DIRECTORY AND THE STAGING AREA)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ rm -rf names.txt

**(REMOVE A FILE)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ ls

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

No commits yet

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

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ touch names.txt

**(CREATING A FILE)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ ls

names.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

No commits yet

Untracked files: **(NO ONE KNOWS ABOUT THIS FILE BECAUSE ITS IN THE FOLDER)**

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

names.txt **(COLOUR IS IN RED BECAUSE UNKNOWN FILE)**

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

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git add names.txt

**(ADDING FILE INTO THE GIT)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

No commits yet

Changes to be committed:

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

new file: names.txt **(COLOUR IS IN GREEN BECAUSE GIT KNOWS THE FILE)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git commit -m "names.txt file added"

[master (root-commit) 9b9df9f] names.txt file added **(FILE NOT MODIFIED)**

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

create mode 100644 names.txt

**(PERMANENT STORE => PROVIDING GIT HISTORY. -m IS USED FOR MESSAGE)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

nothing to commit, working tree clean

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ vi names.txt

**(OPEN OR EDIT A FILE USING CMD)**

**NOTE :**

1. If you want to insert a text into the names.txt file, press i => Switch to insert mode.
2. After editing press Esc button => Switch to command mode.
3. After type :wq or ZZ => Save and quit / exit vi.
4. :q => Quit vi and do not save changes.
5. yy => Yank (Copy) a line of text.
6. p => Past a line of yanked text below the current line.

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ cat names.txt

WELCOME TO GITHUB PROJECT CLASS.

THIS IS JAVED ALI FROM BEPRACTICAL TECH SOLUTIONS.

THIS IS UNIX VI EDITOR.

THIS IS UNIX VI EDITOR.

TEST CASE IS WORKING.

**(DISPLAY THE THINGS ARE AVAILABLE IN NAMES.TXT FILE)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (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: names.txt **(FILE MODIFIED STATUS)**

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

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git add names.txt

warning: in the working copy of 'names.txt', LF will be replaced by CRLF the next time Git touches it

**(ADDING GITMODIFIED FILE)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

Changes to be committed:

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

modified: names.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git restore --staged names.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (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: names.txt (OUTSDE THE STAGE)

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

NOTE : There are three commands with similar names **GIT RESET, GIT RESTORE AND GIT REVERT.**

1. **GIT REVERT** => is about making a new commit that reverts the changes made by other commits.
2. **GIT RESTORE** => is about restoring files in the working tree from either the index or another commit. This command doesn’t update your branch. The command can also be used to retore files in the index from another commit.
3. **GIT RESET** => is about updating your branch, moving the tip in order to add or remove commits from the branch. This operation changes the commit history. GIT RESET can also be used to restore the index, overlapping with GIT RESTORE.

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git add names.txt

warning: in the working copy of 'names.txt', LF will be replaced by CRLF the next time Git touches it

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

Changes to be committed:

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

modified: names.txt (INSIDE THE STAGE)

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git commit -m "names.txt files modified"

[master d2d08ac] names.txt files modified

1 file changed, 10 insertions(+)

admin@DESKTOP-VHIRI0L MINGW64 /d/FRONTEND TECHNOLOGY/GITPROJECT (master)

$ git commit -am "File updated"

WITHOUT GIT ADD NAMES.TXT, WE CAN UPDATE THE FILE BY USING ABOVE FILE

warning: in the working copy of 'naveen.txt', LF will be replaced by CRLF the next time Git touches it

[master 30b29e0] File updated

1 file changed, 3 insertions(+)

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git log

commit d2d08ac1999a84dfc8951cf88ebd84202888b41f (HEAD -> master)

Author: javedalihosamani <javed.hosamani007@gmail.com>

Date: Wed Jun 7 11:26:54 2023 +0530

names.txt files modified

commit 9b9df9fff122626186f89ac0565fed09ef17003f

Author: javedalihosamani <javed.hosamani007@gmail.com>

Date: Wed Jun 7 10:28:02 2023 +0530

names.txt file added

**(HELPS TO DISPLAY COMMENT HISTORY)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ rm -rf names.txt **(DELETING NAMES.TXT FILE)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git add . **(ADDING FILES INTO GIT)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git commit -m "names.txt file deleted" **(SAVE FILES INTO GIT)**

[master 3180ceb] names.txt file deleted

1 file changed, 10 deletions(-)

delete mode 100644 names.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git log **(DISPLAYING COMMENT HISTORY)**

commit 3180ceb726cb6cd847d127cf917fcf7ac871184d (HEAD -> master)

Author: javedalihosamani <javed.hosamani007@gmail.com>

Date: Thu Jun 8 10:14:08 2023 +0530

names.txt file deleted

commit d2d08ac1999a84dfc8951cf88ebd84202888b41f

Author: javedalihosamani <javed.hosamani007@gmail.com>

Date: Wed Jun 7 11:26:54 2023 +0530

names.txt files modified

commit 9b9df9fff122626186f89ac0565fed09ef17003f

Author: javedalihosamani <javed.hosamani007@gmail.com>

Date: Wed Jun 7 10:28:02 2023 +0530

names.txt file added

**(THERE ARE THREE COMMENTS WITHOUT BRANCH)**

**NOTE** :

1. We can’t remove or delete one particular comment from middle (light grey colour comment) because each comment has hash id and each comment build on top of each other.
2. If you want to remove first two comment from comment history, copy the HASH Id just below it. Example : I want to delete first two comment so I copied third comment HASH id because it helps to remove the above comments.

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git reset 9b9df9fff122626186f89ac0565fed09ef17003f

Unstaged changes after reset:

D names.txt

**(DELETED FIRST TWO COMMENTS)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git log

commit 9b9df9fff122626186f89ac0565fed09ef17003f (HEAD -> master)

Author: javedalihosamani <javed.hosamani007@gmail.com>

Date: Wed Jun 7 10:28:02 2023 +0530

names.txt file added

**(DISPLAYING COMMENT HISTORY)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ touch surname.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ vi surname.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ touch houses.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git add .

warning: in the working copy of 'surname.txt', LF will be replaced by CRLF the next time Git touches it

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

Changes to be committed:

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

renamed: names.txt -> houses.txt

new file: surname.txt (Observe this changes)

**(NOTE :STILL NOT YET COMMITTED)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git stash (GO TO BACK STAGE. If you don’t want the above command changes)

Saved working directory and index state WIP on master: 9b9df9f names.txt file added

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

nothing to commit, working tree clean

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ ls

names.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git log

commit 9b9df9fff122626186f89ac0565fed09ef17003f (HEAD -> master)

Author: javedalihosamani <javed.hosamani007@gmail.com>

Date: Wed Jun 7 10:28:02 2023 +0530

names.txt file added

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ cat names.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git stash pop (COME BACK STAGE)

On branch master

Changes to be committed:

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

new file: houses.txt

new file: surname.txt

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: names.txt

Dropped refs/stash@{0} (0a03e5a9af36cb3c2e56362daaddcbe76f5770cb)

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ ls

houses.txt surname.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git stash clear (HELPS TO CLEAR NOT COMMITTED THINGS)

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git status

On branch master

Changes to be committed:

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

new file: houses.txt

new file: surname.txt

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: names.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ ls

houses.txt surname.txt

**Go to GitHub server and create repository. (**[**https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.git**](https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.git)**)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git remote add origin https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.it

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git remote -v

origin https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.git (fetch)

origin https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.git (push)

**(IT DISPLAY THE URL ATTACHED TO THE URL)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git branch -M main

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (main)

$ git push -u origin main

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), 223 bytes | 37.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.git

\* [new branch] main -> main

branch 'main' set up to track 'origin/main'.

**(AFTER THIS COMMAND REFRESH YOUR GITHUB URL)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (main)

$ ls

class.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (main)

$ git branch **(DISPLAYING CURRENT BRANCH NAME)**

\* main

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (main)

$ git branch -M master **(CREATE A NEW BRANCH NAME)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git push origin master **(UPLOADING FILES INTO NEW BRANCH)**

Enumerating objects: 10, done.

Counting objects: 100% (10/10), done.

Delta compression using up to 4 threads

Compressing objects: 100% (7/7), done.

Writing objects: 100% (10/10), 860 bytes | 286.00 KiB/s, done.

Total 10 (delta 2), reused 0 (delta 0), pack-reused 0

remote: Resolving deltas: 100% (2/2), done.

remote:

remote: Create a pull request for 'master' on GitHub by visiting:

remote: https://github.com/javedalihosamani/COMMUNITYCLASS-GIT/pull/new/master

remote:

To https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.git

\* [new branch] master -> master

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git checkout master **(DISPLAYING CURRENT BRANCH FILES)**

Already on 'master'

D houses.txt

D node.txt

D react.txt

D surname.txt

Your branch is up to date with 'origin/master'.

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git branch feature **(CREATING A VERSION OF CURRENT BRANCH)**

**NOTE :**

1. When developers write code, they’re writing it on a branch. When several developers work on the same code base, it becomes difficult to add, modify, or erase code without their work overlapping or overriding each other’s changes.
2. A feature branch is a copy of the main codebase where an individual or team of software developers can work on a new feature until it is complete.

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (master)

$ git checkout feature

Switched to branch 'feature'

D houses.txt

D node.txt

D react.txt

D surname.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

**(BRANCH CHANGED)**

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ git commit

On branch feature

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: houses.txt

deleted: node.txt

deleted: react.txt

deleted: surname.txt

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

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ vi node.txt (INSERTED SOME TEXT)

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ cat node.txt

THSI IS NOTE TEXT FILE

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ git commit

On branch feature

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: houses.txt

modified: node.txt (modified file)

deleted: react.txt

deleted: surname.txt

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

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ git add .

warning: in the working copy of 'node.txt', LF will be replaced by CRLF the next time Git touches it

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ git commit -m "Files are uploaded"

[feature 125a909] Files are uploaded

4 files changed, 1 insertion(+), 1 deletion(-)

delete mode 100644 houses.txt

delete mode 100644 react.txt

delete mode 100644 surname.txt

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ git push origin feature (GO TO SERVER AND OBSERVE THE PULL REQUEST.)

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 4 threads

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

Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.git

9db37f7..125a909 feature -> feature

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ vi node.txt (UPDATING FILE)

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ cat node.txt (DISPLAYING UPDAED CONTENT)

THSI IS NOTE TEXT FILE

THIS IS ANOTHER NODE TEXT FILE

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ git add .

warning: in the working copy of 'node.txt', LF will be replaced by CRLF the next time Git touches it

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ git commit -m "node.txt file is uploaded"

[feature 6b643e6] node.txt file is uploaded

1 file changed, 2 insertions(+)

admin@DESKTOP-VHIRI0L MINGW64 /d/GITPROJECT (feature)

$ git push origin feature (GO TO GITHUB AND COMPLETE COMPARE AND PULL REQUEST)

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 4 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 328 bytes | 328.00 KiB/s, done.

Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

To https://github.com/javedalihosamani/COMMUNITYCLASS-GIT.git

125a909..6b643e6 feature -> feature