Deep Chavan
T11-15

LAB ASSIGNMENT 3

AIM: To perform various Git Operations.

LAB OUTCOME:

LO1, LO2 Mapped.

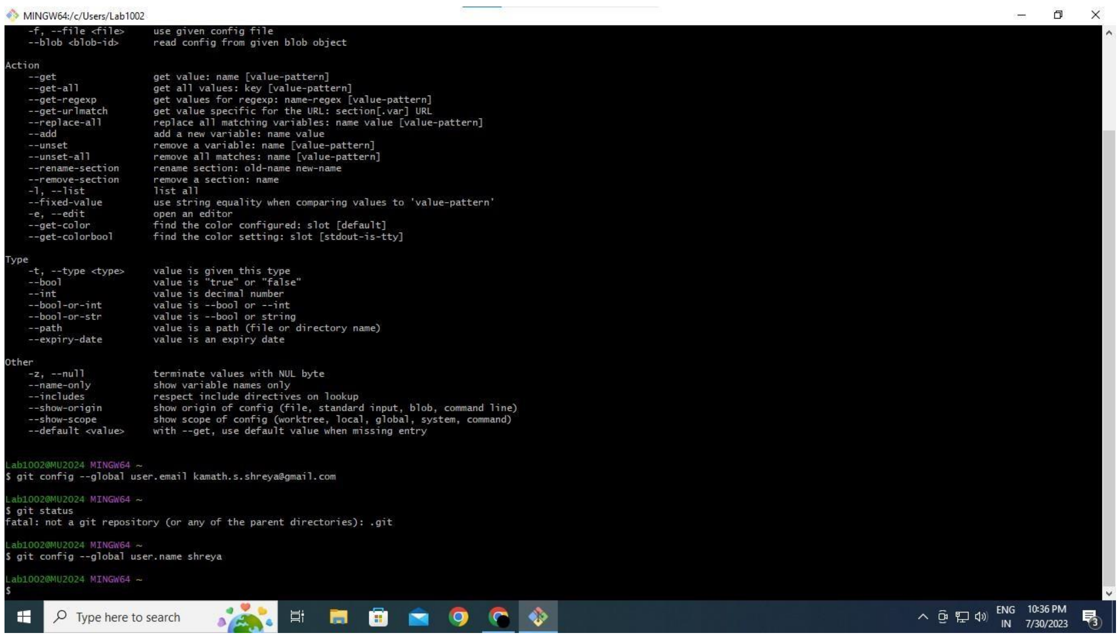
THEORY:

Git is a distributed version control system that enables collaboration among developers by tracking changes in code. It allows multiple people to work on the same project simultaneously, maintaining different versions of the code through commits and branches. This system facilitates efficient code merging, conflict resolution, and the ability to work on new features independently. Git also provides a remote repository for sharing and backup, making it an essential tool for managing and tracking code changes in software development projects.

Git commands are instructions that you give to the Git version control system to perform various tasks and operations on your source code repository. These commands allow you to interact with your code, track changes, collaborate with others, and manage your project's history effectively. Each Git command corresponds to a specific action, such as initialising a repository, staging changes, committing code, creating branches, merging changes, fetching remote updates, and more. Some of these are given below:

1. git init: Initialises a new Git repository. Example: git init
2. git clone: Copies a remote repository to your local machine. Example: git clone https://github.com/username/repo.git
3. git add: Stages changes for commit. Example: git add file.txt
4. git commit: Records staged changes with a message. Example: git commit -m "Added newfeature"
5. git status: Shows the status of your working directory. Example: git status
6. git log: Displays commit history. Example: git log
7. git branch: Lists, creates, or deletes branches. Example: git branch new-feature 8. git checkout: Switches to a different branch. Example: git checkout new-feature
9. git merge: Combines changes from different branches.
10. Example: git merge feature-branch
11. git pull: Fetches remote changes and merges them into the current branch. Example: git pull origin main
12. git push: Uploads local changes to a remote repository. Example: git push origin main
13. git remote: Manages remote repositories.
14. Example: git remote add origin https://github.com/username/repo.git

15. git fetch: Retrieves remote changes but doesn't merge them. Example: git fetch origin
16. git diff: Shows differences between working directory and last commit. Example: git diff
17. git reset: Unstaged files or discards changes. Example: git reset file.txt
18. COMMANDS & OUTPUT:



```
    -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


Lab1002@MU2024 MINGW64 ~
$ git config --global user.email kamath.s.shreya@gmail.com

Lab1002@MU2024 MINGW64 ~
$ git status
fatal: not a git repository (or any of the parent directories): .git

Lab1002@MU2024 MINGW64 ~
$ git config --global user.name shreya

Lab1002@MU2024 MINGW64 ~
$
```

Lab1002@MU2024 MINGW64 ~

$ git config 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
Lab1002@MU2024 MINGW64 ~
$ git config --global user.email kamath.s.shreya@gmail.com

Lab1002@MU2024 MINGW64 ~
$ git status fatal: not a git repository (or any of the parent
directories):
.git
Lab1002@MU2024 MINGW64 ~
$ git config --global user.name shreya

Lab1002@MU2024 MINGW64 ~
$ git init
Initialized empty Git repository in C:/Users/Lab1002/.git/

Lab1002@MU2024 MINGW64 ~ (master)
$ git status

On branch master No commits yet
Lab1002@MU2024 MINGW64 ~ (master)
$ git --version git version
2.41.0.windows.3

Lab1002@MU2024 MINGW64 ~ (master)
$ git remote add origin "https://github.com/ShreyaKamath09/DevOps.git"

Lab1002@MU2024 MINGW64 ~ (master)
$ git pull origin main remote: Enumerating objects: 9, done. remote: Counting
objects: 100% (9/9), done. remote: Compressing objects: 100% (4/4), done.
```

remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0 Unpacking objects: 100% (9/9), 1.83 KiB | 5.00 KiB/s, done. From https://github.com/ShreyaKamath09/DevOps

* branch main -> FETCH_HEAD * [new

branch] main -> origin/main

CONCLUSION:

In conclusion, I've effectively used different Git commands, demonstrating their crucial role in version control and teamwork. This hands-on experience highlights Git's essential role in making coding processes more organised and collaborative.