

Experiment 3: To Perform various Git operations on local and remote repositories using Git cheat sheet.

THEORY:

Introduction to Git

Git is a distributed version control system used for tracking changes in source code. It allows multiple developers to work on a project simultaneously while keeping track of changes and enabling collaboration through remote repositories like GitHub, GitLab, and Bitbucket.

Configuring Git

Before using Git for the first time, it is necessary to configure the user's identity. The following commands set up the user's name and email, which will be associated with all commits:

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your.email@example.com"
```

The `--global` flag ensures that the configuration applies to all repositories on the system.

Initializing a Git Repository

A Git repository must be initialized before tracking changes. This is done using the `git init` command:

```
git init
```

Executing this command creates a hidden `.git` directory within the project folder, which stores all version control information.

Checking the Status of a Repository

To check the current state of the repository, including untracked and modified files, use:

```
git status
```

This command provides an overview of changes that need to be staged, committed, or pushed.

Adding Files to the Staging Area

Before committing changes, files must be added to the staging area. This can be done using:

```
git add <file_name> # Adds a specific file
```

```
git add . # Adds all modified and new files
```

The staging area acts as an intermediate step before committing changes.

Committing Changes

A commit captures the current state of the repository and saves it locally. Each commit requires a message that describes the changes made:

```
git commit -m "Descriptive commit message"
```

Commits are local and do not affect the remote repository until they are pushed.

Connecting to a Remote Repository

To link the local repository with a remote repository (e.g., GitHub), use:

```
git remote add origin <repository_URL>
```

For example:

```
git remote add origin https://github.com/username/repository.git
```

To verify that the remote repository has been added, use:

```
git remote -v
```

Pushing Changes to a Remote Repository

To upload commits to a remote repository:

```
git push origin main
```

- **origin** refers to the remote repository.
- **main** refers to the branch being pushed.

For the first push, use:

```
git push -u origin main
```

The **-u** flag sets **origin main** as the default upstream branch, allowing future pushes to be done with **git push** alone.

Pulling Changes from a Remote Repository

To retrieve and merge updates from the remote repository:

```
git pull origin main
```

This command ensures the local repository is up-to-date with the remote repository.

Cloning an Existing Repository

To create a local copy of an existing remote repository:

```
git clone <repository_URL>
```

For example:

```
git clone https://github.com/username/repository.git
```

This command downloads the repository and sets up a connection to the remote repository.

Branching and Merging

Git allows working with multiple branches to develop new features without affecting the main codebase.

Creating a new branch:

```
git branch new-branch
```

Switching to the new branch:

```
git checkout new-branch
```

Merging a branch into the main branch:

```
git merge new-branch
```

Deleting a branch:

```
git branch -d new-branch
```

Branches help in parallel development and version control management.

Implementation:

```
MINGW64 ~/Users/lenovo/Downloads/exp/git-docs/git-demo project
--no-[default] <value>
    with --get, use default value when missing entry

MINGW64 17 MINGW64 --Downloads/exp/git-docs (new-branch)
$ git config --global user.name "ShrirangZend"
error: unknown option 'global.user.name'
usage: git config [<options>]

Config file location
--no-[global]          use global config file
--no-[system]         use system config file
--no-[local]          use repository config file
--no-[worktree]       use per-worktree config file
-f, --no-[file] <file> use given config file
--no-[blob] <blob-id> read config from given blob object

Action
--no-[get]             get value: name [value-pattern]
--no-[get-all]        get all values: key [value-pattern]
--no-[get-regexp]      get values for regexp: name-regexp [value-pattern]
--no-[get-urlmatch]    get value specific for the URL: section[.var] URL
--no-[replace-all]    replace all matching variables: name value [value-pattern]

--no-[add]            add a new variable: name value
--no-[unset]          remove a variable: name [value-pattern]
--no-[unset-all]      remove all matches: name [value-pattern]
--no-[rename-section]  rename section: old-name new-name
--no-[remove-section] remove a section: name
-l, --no-[list]        list all
--no-[fixed-value]     use string equality when comparing values to 'value-as'

Items
-h, --no-[edit]       open an editor
--no-[get-color]      find the color configured: slot [default]
--no-[get-colorbool]  find the color settings: slot [stand-is-ity]

Type
-t, --no-[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
-h, --no-[null]       terminate values with NUL byte
--no-[name-only]      show variable names only
--no-[includes]       respect include directives on lookup
--no-[show-origin]    show origin of config (file, standard input, blob, cmd)
--no-[show-scope]     show scope of config (worktree, local, global, system,
command)
--no-[default] <value>
    with --get, use default value when missing entry
```

```
MINGW64 ~/Users/lenovo/Downloads/exp/git-docs/git-demo project
MINGW64 17 MINGW64 --Downloads/exp (new-branch)
$ cd git-docs

MINGW64 17 MINGW64 --Downloads/exp/git-docs (new-branch)
$ git config --global
usage: git config [<options>]

Config file location
--no-[global]          use global config file
--no-[system]         use system config file
--no-[local]          use repository config file
--no-[worktree]       use per-worktree config file
-f, --no-[file] <file> use given config file
--no-[blob] <blob-id> read config from given blob object

Action
--no-[get]             get value: name [value-pattern]
--no-[get-all]        get all values: key [value-pattern]
--no-[get-regexp]      get values for regexp: name-regexp [value-pattern]
--no-[get-urlmatch]    get value specific for the URL: section[.var] URL
--no-[replace-all]    replace all matching variables: name value [value-pattern]

--no-[add]            add a new variable: name value
--no-[unset]          remove a variable: name [value-pattern]
--no-[unset-all]      remove all matches: name [value-pattern]
--no-[rename-section]  rename section: old-name new-name
--no-[remove-section] remove a section: name
-l, --no-[list]        list all
--no-[fixed-value]     use string equality when comparing values to 'value-as'

Items
-h, --no-[edit]       open an editor
--no-[get-color]      find the color configured: slot [default]
--no-[get-colorbool]  find the color settings: slot [stand-is-ity]

Type
-t, --no-[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
-h, --no-[null]       terminate values with NUL byte
--no-[name-only]      show variable names only
--no-[includes]       respect include directives on lookup
--no-[show-origin]    show origin of config (file, standard input, blob, cmd)
--no-[show-scope]     show scope of config (worktree, local, global, system,
command)
--no-[default] <value>
    with --get, use default value when missing entry

MINGW64 17 MINGW64 --Downloads/exp/git-docs (new-branch)
$ git config --global user.name "ShrirangZend"
```

```
MINGW64 ~/Downloads/esp/git-demos/git-demo-project
$ mkdir git-demos
$ cd git-demos
$ 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

Section
--get         get value: name [value-pattern]
--get-all    get all values: key [value-pattern]
--get-regexp  get values for regexp: name-regexp [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
--fix=string  use string equality when comparing values to 'value-pattern'

Colors
-c, --color    find the color configured: slot [default]
--color=bool  find the color setting: slot [stdout-is-tyt]

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 loading
--show-origin show origins of config (file, standard input, blob, command line)
--show-scope  show scope of config (worktree, local, global, system, command)
--default=string with --get, use default value when missing entry
```

```
MINGW64 ~/Downloads/esp/git-demos/git-demo-project
$ cd git-demo-project
$ git push origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

MINGW64 ~/Downloads/esp/git-demos/git-demo-project (master)
$ git push origin main
error: src refspec main does not match any
error: failed to push some refs to 'origin'

MINGW64 ~/Downloads/esp/git-demos/git-demo-project (master)
$ git status
On branch master
nothing to commit, working tree clean

MINGW64 ~/Downloads/esp/git-demos/git-demo-project (master)
$ git push origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

MINGW64 ~/Downloads/esp/git-demos/git-demo-project (master)
$ git remote add origin https://github.com/AdharvarVishare/SERP-Lab.git
git branch -M main
git push -u origin main

remote: Permission to AdharvarVishare/SERP-Lab.git denied to yatic123.
fatal: unable to access 'https://github.com/AdharvarVishare/SERP-Lab.git/': The r
equested URL returned error 403

MINGW64 ~/Downloads/esp/git-demos/git-demo-project (main)
$ git remote add origin https://github.com/AdharvarVishare/SERP-Lab.git
error: remote origin already exists.

MINGW64 ~/Downloads/esp/git-demos/git-demo-project (main)
$ git push -u origin master
error: src refspec master does not match any
error: failed to push some refs to 'https://github.com/AdharvarVishare/SERP-Lab.g
it'

MINGW64 ~/Downloads/esp/git-demos/git-demo-project (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Writing objects: 100% (4/4), 287 bytes | 287.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
to https://github.com/AdharvarVishare/SERP-Lab.git
 * [new branch] main -> main
branch 'main' set up to track 'origin/main'.
```

Conclusion:

Successfully implemented various Git operations on local and remote repositories using the Git cheat sheet.