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

THEORY:

Introduc on to Git

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

Configuring Git

Before using Git for the first me, it is necessary to configure the user's iden ty. The following commands set up the user's name and email, which will be associated with all commits: bash CopyEdit git config --global user.name "Your Name" git config --global user.email "your.email@example.com"

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

Ini alizing a Git Repository

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

bash CopyEdit git init

Execu ng this command creates a hidden .git directory within the project folder, which stores all version control informa on.

Checking the Status of a Repository

To check the current state of the repository, including untracked and modified files, the following command is used: bash CopyEdit git status

This

command
provides an
overview of
changes that
need to be
staged,
commi ed,
or pushed.

Adding Files to the Staging Area

Before commi ng changes, files must be added to the staging area. This can be done using the following commands: bash CopyEdit 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 commi ng changes.

Commi ng Changes

A commit captures the current state of the repository and saves it locally. Each commit requires a message that describes the changes made: bash CopyEdit git commit -m "Descrip ve commit message"

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

Connec ng to a Remote Repository

To link the local repository with a remote repository (e.g., GitHub), the following command is used:

bash CopyEdit git remote add origin <repository_URL>

For example: bash CopyEdit git remote add origin h ps://github.com/username/repository.git

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

bash CopyEdit git remote -v

Pushing Changes to a Remote Repository

To upload commits to a remote repository, the git push command is used:

bash CopyEdit git push origin main

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

For the first push, use:

bash CopyEdit 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, the git pull command is used:

bash CopyEdit git pull origin main

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

Cloning an Exis ng Repository

To create a local copy of an exis ng remote repository, the git clone command is used: bash

CopyEdit git clone

<repository_URL>

For example: bash CopyEdit git clone h

ps://github.com/username/repository.git This

command downloads the repository and sets up a

connec on to the remote repository.

Branching and Merging

Git allows working with mul ple branches to develop new features without affec ng the main codebase.

Crea ng a new branch: bash

CopyEdit git branch new-

branch Switching to the new

branch: bash CopyEdit git

checkout new-branch

Merging a branch into the main branch:

bash CopyEdit git merge new-branch

Dele ng a branch: bash CopyEdit git

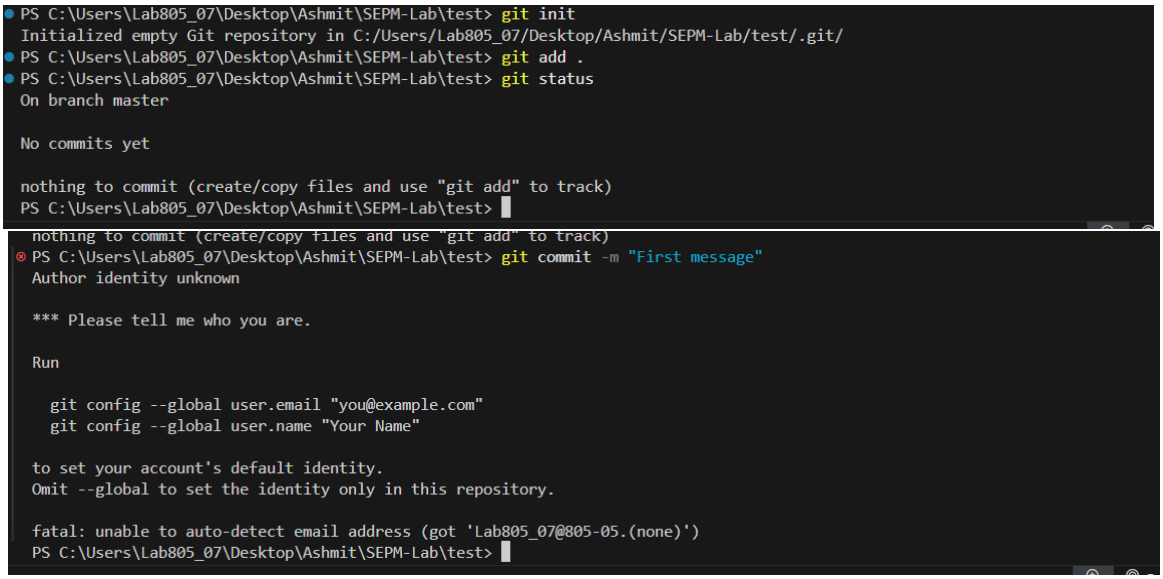
branch -d new-branch

Branches help in parallel development and version control management.

Output:

```
PS C:\Users\Lab805_07\Desktop\Ashmit\SEPM-Lab> 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-regex [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-pattern'
    -e, --[no-]edit       open an editor
    --[no-]get-color      find the color configured: slot [default]
    --[no-]get-colorbool  find the color setting: slot [stdout-is-tty]

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
    -z, --[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, command line)
    --[no-]show-scope     show scope of config (worktree, local, global, system, command)
    --[no-]default <value>
                          with --get, use default value when missing entry
```

```
PS C:\Users\Lab805_07\Desktop\Ashmit\SEPM-Lab> mkdir test


    Directory: C:\Users\Lab805_07\Desktop\Ashmit\SEPM-Lab


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/28/2025   1:47 PM                test


PS C:\Users\Lab805_07\Desktop\Ashmit\SEPM-Lab> cd test
PS C:\Users\Lab805_07\Desktop\Ashmit\SEPM-Lab\test>
```



Conclusion: Successfully implemented various Git opera ons on local and remote repositories using Git cheat sheet.