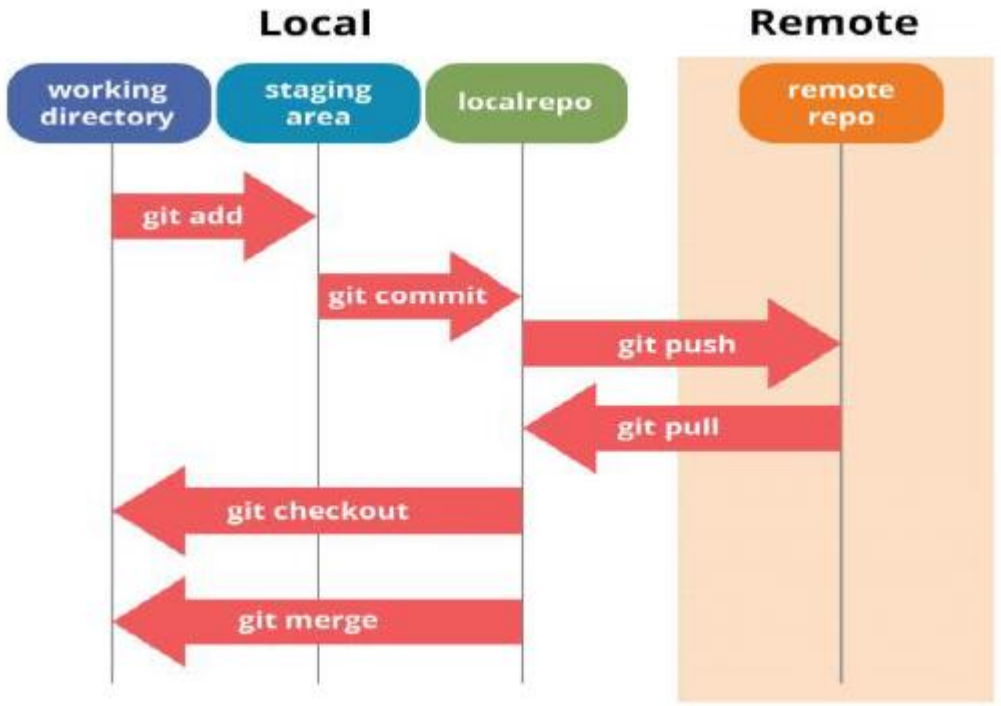
Experiment no. 3: To perform various git operations on local and remote repositories using GIT cheat-sheet

Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easy to learn and has a tiny footprint with lightning-fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.
Some of the basic operations in Git are:

1. Initialize
2. Add
3. Commit
4. Pull
5. Push

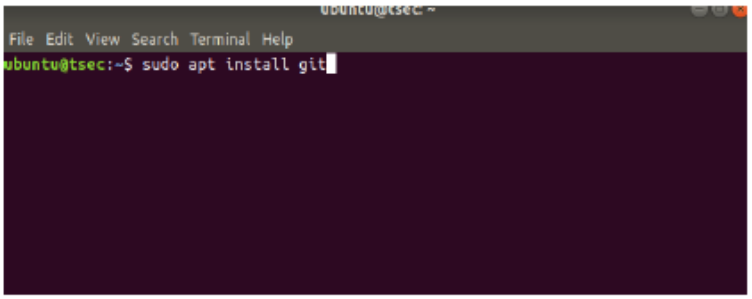
Some advanced Git operations are:
1. Branching
2. Merging
3. Rebasing

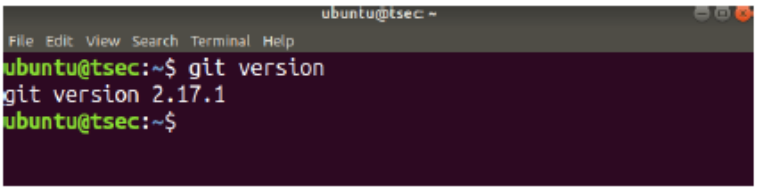
The following diagram depicts the all supported operations in GIT



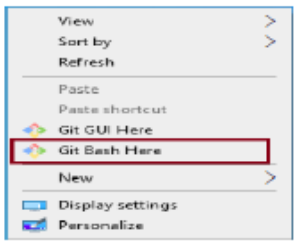
Installation of GIT

1) In windows, download GIT from https://git-scm.com/ and perform the straightforward installation.
2) In Ubuntu, install GIT using $sudo apt install git, Confirm the version after installation using command $git –version





Once installation is done, open the terminal in Ubuntu and perform the following steps or in windows Right click and select Git bash here.



Git Cheat Sheet: Local & Remote Repository Operations

Basic Git Commands

1. Git Configuration

- git config --global user.name "Your Name" → Set your name
- git config --global user.email "youremail@example.com" → Set your email
- git config --list → View Git configurations

Local Repository Operations

2. Initialize and Clone

- git init → Initialize a new Git repository
- git clone <repo_url> → Clone a remote repository

3. Basic Workflow

- git status → Check the status of changes

- git add <file> → Stage a specific file
- git add . → Stage all files
- git commit -m "Commit message" → Commit changes
- git log → View commit history
- git diff → View unstaged changes

4. Branching and Merging

- git branch → List branches
- git branch <branch_name> → Create a new branch
- git checkout <branch_name> → Switch to a branch
- git merge <branch_name> → Merge a branch into the current branch
- git branch -d <branch_name> → Delete a branch

Remote Repository Operations

5. Connect to Remote Repository

- git remote add origin <repo_url> → Connect local repo to remote
- git remote -v → View remote connections

6. Push and Pull Changes

- git push origin <branch_name> → Push local changes to remote
- git pull origin <branch_name> → Fetch and merge changes from remote
- git fetch → Fetch changes from remote without merging
- git add <file> & git commit -m "Resolved conflicts" → Finalize merge

8. Undo Changes

- git reset --hard <commit_id> → Reset to a specific commit
- git revert <commit_id> → Create a new commit that undoes previous changes
- git checkout -- <file> → Discard changes in a file

Useful Commands

- git stash → Save uncommitted changes temporarily
- git stash pop → Restore stashed changes
- git tag -a v1.0 -m "Version 1.0" → Create a tag

- git show <commit_id> → View commit details

By using these commands,we can efficiently manage both local and remote repositories with Git.

Output of the executed commands

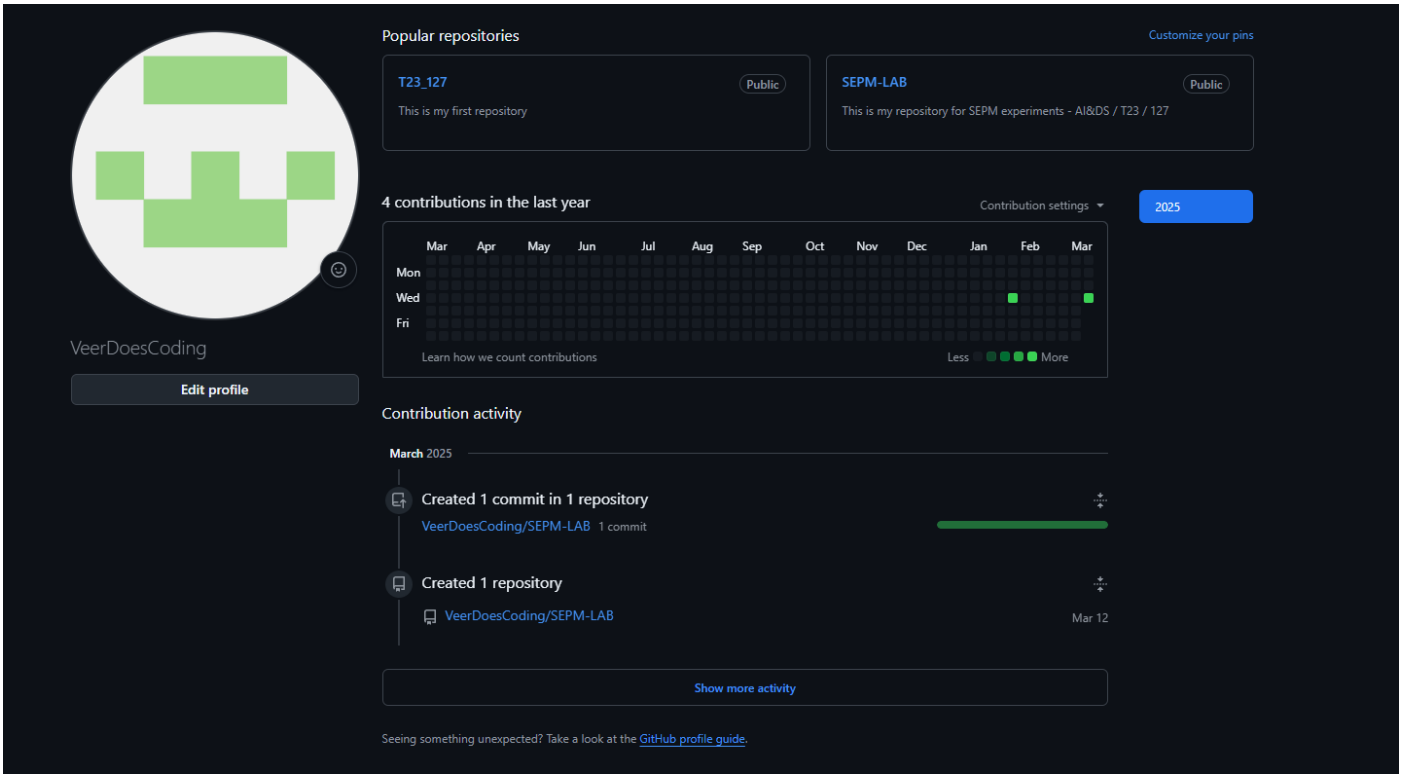
```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git add index.html

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        teststatus
```

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html

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


ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git add teststatus

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   index.html
        new file:   teststatus
```

```
ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git commit -am "Express commit"
[master d3a6a76] Express commit
 2 files changed, 2 insertions(+), 2 deletions(-)
 create mode 100644 teststatus

ADMIN@DESKTOP-J582V2L MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
nothing to commit, working tree clean
```

```
MINGW64:/c/Users/Lenovo/git-dvcs

Lenovo@203-016 MINGW64 ~
$ cd git-dvcs

Lenovo@203-016 MINGW64 ~/git-dvcs
$ 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-patt
ern]
    --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-pa
ttern'
    -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, com
mand line)
    --show-scope          show scope of config (worktree, local, global, system,
 command)
    --default <value>     with --get, use default value when missing entry


Lenovo@203-016 MINGW64 ~/git-dvcs
$ git config --global usser.name "VeerDoesCoding"
```

```
Lenovo@203-016 MINGW64 ~/git-dvcs
$ git config --global user.email "veer.bhatt2005@gmail.com"

Lenovo@203-016 MINGW64 ~/git-dvcs
$ git config --global --list
user.name=mathesh
user.email=veer.bhatt2005@gmail.com
color.ui=true
usser.name=VeerDoesCoding

Lenovo@203-016 MINGW64 ~/git-dvcs
$ git config --global user.name "Veer"

Lenovo@203-016 MINGW64 ~/git-dvcs
$ git config --global --list
user.name=Veer
user.email=veer.bhatt2005@gmail.com
color.ui=true
usser.name=VeerDoesCoding
```



Conclusion: Thus, we have successfully executed git operations on local and remote repositories