

# 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:

```
bash
```

```
CopyEdit
```

```
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:

```
bash
```

```
CopyEdit
```

```
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, the following command is used:

```
bash
```

```
CopyEdit
```

```
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 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 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:

bash

CopyEdit

```
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), the following command is used:

bash

CopyEdit

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

For example:

bash

CopyEdit

```
git remote add origin https://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 Existing Repository**

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

bash

CopyEdit

```
git clone <repository_URL>
```

For example:

bash

CopyEdit

```
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:**

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

#### **Deleting a branch:**

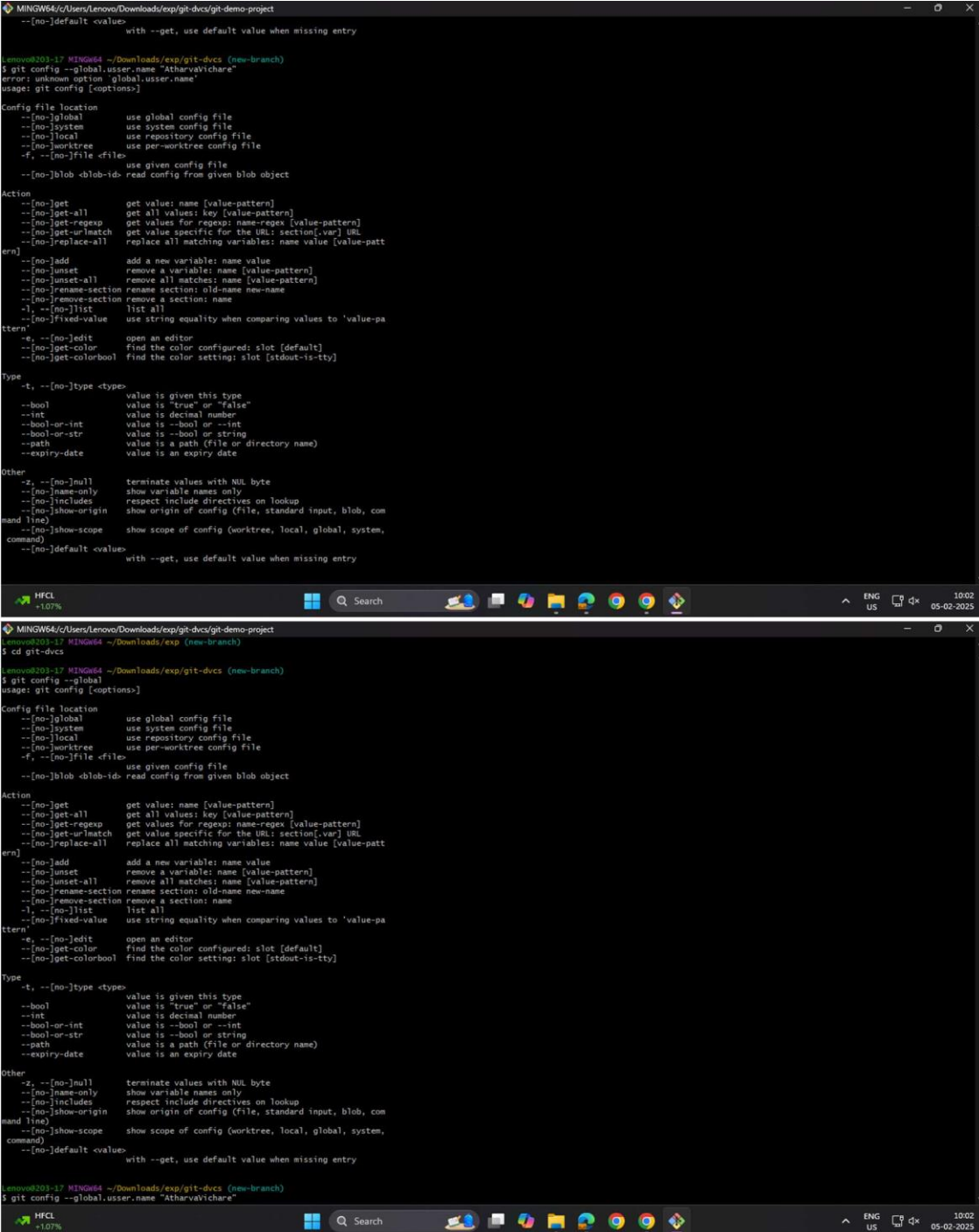
bash

CopyEdit

git branch -d new-branch

Branches help in parallel development and version control management.

## Output:



```

MINGW64/C:/Users/Lenovo/Downloads/exp/git-dvcs/git-demo-project
--[no-]default <value>          with --get, use default value when missing entry

Lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ git config --global.user.name "AtharvaVichare"
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 values: 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]

ern
--[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'

Item
-e, --[no-]edit        open an editor
--[no-]get-color       find the color configured: slot [default]
--[no-]get-colorbool   find the color settings: slot [stdout-is-tyt]

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

```

```

MINGW64/C:/Users/Lenovo/Downloads/exp/git-dvcs/git-demo-project
Lenovo@2023-17 MINGW64 ~/Downloads/exp (new-branch)
$ cd git-dvcs

Lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (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 values: 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]

ern
--[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'

Item
-e, --[no-]edit        open an editor
--[no-]get-color       find the color configured: slot [default]
--[no-]get-colorbool   find the color settings: slot [stdout-is-tyt]

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

Lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ git config --global.user.name "AtharvaVichare"

```

```

MINGW64~/Users/Lenovo/Downloads/exp/git-dvcs/git-demo-project
lenovo@2023-12 MINGW64 ~/Downloads/exp (new-branch)
$ mkdir git-dvcs
lenovo@2023-12 MINGW64 ~/Downloads/exp (new-branch)
$ cd git-dvcs
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs (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(.ver) URL
  --[no-]replace-all    replace all matching variables: name value [value-pattern]

ern
  --[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'

ttern
  -e, --[no-]edit        open an editor
  --[no-]get-color       find the color configured: slot [default]
  --[no-]get-colorbool   find the color settings: slot [stdout-is-ty]

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

```

```

MINGW64~/Users/Lenovo/Downloads/exp/git-dvcs/git-demo-project
$ cd git-demo-project
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/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.
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/git-demo-project (master)
$ git push origin main
error: src refspec main does not match any
error: Failed to push some refs to 'origin'
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/git-demo-project (master)
$ git status
On branch master
nothing to commit, working tree clean
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/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.
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/git-demo-project (master)
$ git remote add origin https://github.com/AtharvaVichare/SEPM-Lab.git
git branch -M main
git push -u origin main

remote: Permission to AtharvaVichare/SEPM-Lab.git denied to yatish20.
fatal: unable to access 'https://github.com/AtharvaVichare/SEPM-Lab.git/': The requested URL returned error: 403
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/git-demo-project (main)
$ git remote add origin https://github.com/AtharvaVichare/SEPM-Lab.git
error: remote origin already exists.
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/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/AtharvaVichare/SEPM-Lab.git'
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/git-demo-project (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Writing objects: 100% (4/4), 267 bytes | 267.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/AtharvaVichare/SEPM-Lab.git
 * [new branch] main -> main
branch 'main' set up to track 'origin/main'.
lenovo@2023-12 MINGW64 ~/Downloads/exp/git-dvcs/git-demo-project (main)
$

```

```
MINGW64~/Users/Lenovo/Downloads/exp/git-dvcs/git-demo-project
lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ cd git-dvcs/git-demo-project
bash: cd: git-dvcs/git-demo-project: No such file or directory

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ git init
Initialized empty Git repository in C:/Users/Lenovo/Downloads/exp/git-dvcs/.git/

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (master)
$ git add .
$ git status
On branch master

No commits yet

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

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (master)
$ git commit -m "First Commit"
On branch master

Initial commit

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

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (master)
$ git add .
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   git-demo-project/1.txt.txt

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (master)
$ git commit -m "First Commit"
[master (root-commit) fc3b092] First Commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 git-demo-project/1.txt.txt

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (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.

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (master)
$ cd git-demo-project

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs/git-demo-project

MINGW64~/Users/Lenovo/Downloads/exp/git-dvcs/git-demo-project
-1, --[no-]list          list all
--[no-]fixed-value      use string equality when comparing values to 'value-pa
tern'
-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-tyt]

Type
-t, --[no-]type <type> value is given this type
                        value is "true" or "false"
--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, com
mand line)
--[no-]show-scope       show scope of config (worktree, local, global, system,
command)
--[no-]default <value> with --get, use default value when missing entry

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ git config --global user.name "AtharvaVichare"

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ git config --global --list
user.name=AtharvaVichare
user.email=tammy123@gmail.com
color.ui=true
user.name=AtharvaVichare

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ git config --global user.name "AtharvaVichare"

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ git config --global user.email "atharvavichare37@gmail.com"

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ git config --global --list
user.name=AtharvaVichare
user.email=atharvavichare37@gmail.com
color.ui=true
user.name=AtharvaVichare

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ mkdir git-demo-project

lenovo@2023-17 MINGW64 ~/Downloads/exp/git-dvcs (new-branch)
$ cd git-demo-prjct
bash: cd: git-demo-prjct: No such file or directory
```

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