EXPERIMENT 3

Aim: Cmd version control commands

Theory:

The command `mkdir git` creates a new directory (folder) named "git" in the current working directory. This command is used to make a new directory in a Unix-like operating system.

The command `cd git` is used to change the current working directory to the directory named "git." After executing this command, any subsequent commands or file operations will occur within the "git" directory. "cd" stands for "change directory."

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~
$ cd desktop

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop (master)
$ mkdir git-dvcs
mkdir: cannot create directory ‘git-dvcs’: File exists

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop (master)
$ cd git-dvcs

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs (master)
$ cd git-dvcs/
bash: cd: git-dvcs/: No such file or directory
```

The `git config --global user.name` and `git config --global user.email` commands are used to set your global Git username and email address, respectively. They are part of the configuration settings in Git and are associated with the commits you make.

If you want to check your configuration settings, you can use the `git config --list` command to list all the settings Git can find at that point

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs (master)
$ git config --global user.name "Khyati"

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs (master)
$ git config --global user.email "khyati428@gmail.com"

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs (master)
$ git config --global --list
gui.recentrepo=C:/Users/AI&DS 202/Desktop
user.name=Khyati
user.email=khyati428@gmail.com

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs (master)
$ cat ~/.gitconfig
[gui]
        recentrepo = C:/Users/AI&DS 202/Desktop
[user]
        name = Khyati
        email = khyati428@gmail.com

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs (master)
$ mkdir git-demo-project

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs (master)
$ cd git-demo-project/

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git init
Initialized empty Git repository in C:/Users/AI&DS 202/Desktop/git-dvcs/git-demo-project/.git/
```

git commit -am "commit message"` stages and commits all changes in tracked files with a commit message in a single command.
The command `nano index.html` opens the Nano text editor for the file named "index.html." Nano is a simple command-line text editor that allows you to view and edit files directly in the terminal.

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git init
Initialized empty Git repository in C:/Users/AI&DS 202/Desktop/git-dvcs/git-demo-project/.git/

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ ls -a
./  ../  .git/

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git add .

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git commit -am "express Commit"
[master (root-commit) f6ada53] express Commit
 1 file changed, 47 insertions(+)
 create mode 100644 index.html.txt

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ nano index.html
```

The command `touch teststatus` creates an empty file named "teststatus" in the current directory. The `touch` command is commonly used to update the timestamps of a file or create an empty file if it doesn't exist.
git checkout -- teststatus: Discards changes to the file "teststatus" in the working directory.This reverts the file to the state it has in the last commit.

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    index.html.txt

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

no changes added to commit (use "git add" and/or "git commit -a")

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ touch teststatus

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git checkout -- teststatus
error: pathspec 'teststatus' did not match any file(s) known to git
```

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git checkout -- index.html
error: pathspec 'index.html' did not match any file(s) known to git
```

The `git add` command is used to stage changes in the working directory for the next commit in Git. It prepares modifications, additions, or deletions to be included in the upcoming commit.

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git add index.html

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git checkout -- index.html

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.html

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    index.html.txt

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

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git add teststatus

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index.html
        new file:   teststatus

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    index.html.txt
```

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git commit -am "express Commit"
[master 337386c] express Commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 rename index.html.txt => index.html (100%)
 create mode 100644 teststatus

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master
nothing to commit, working tree clean
```

The `git log` command is used to display the commit history of a Git repository. It shows a chronological list of commits, including commit hashes, author information, timestamps, and commit messages.

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git log
commit 337386c30e699833155a6398244589 39e8062781 (HEAD -> master)
Author: Khyati <khyati428@gmail.com>
Date:   Wed Jan 17 09:16:52 2024 +0530

    express Commit

commit f6ada535f1606dd72e3e752362c4c58847c95f13
Author: Khyati <khyati428@gmail.com>
Date:   Wed Jan 17 09:00:56 2024 +0530

    express Commit
```

The command `git log --oneline` displays a simplified and concise one-line representation of the commit history in a Git repository, showing only the commit SHA-1 hash and the commit message.

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git log --oneline
337386c (HEAD -> master) express Commit
f6ada53 express Commit
```

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git log --oneline f6ada53..337386c
337386c (HEAD -> master) express Commit
```

The `git clone` command is used to create a copy of a Git repository. When you run this command, it duplicates the entire repository, including its files, commit history, and branches, and downloads it to your local machine. This is often the initial step when you want to work with a project hosted on a remote Git repository.

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git log --oneline -n 2
337386c (HEAD -> master) express Commit
f6ada53 express Commit

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ git clone https://github.com/khyaaati/siesworkshop.git
Cloning into 'siesworkshop'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 12 (delta 0), reused 9 (delta 0), pack-reused 0
Receiving objects: 100% (12/12), done.

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project (master)
$ cd siesworkshop/
```

The `git pull` command is used to fetch and integrate changes from a remote repository into the current branch of your local repository. It combines two actions: it fetches the changes from the remote repository, and then it automatically merges those changes into your local branch. This is a convenient way to update your local repository with the latest changes from the remote repository.

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git pull
Already up to date.

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git add .

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git commit -m "commit"
[master ad1b06a] commit
 1 file changed, 1 insertion(+), 1 deletion(-)
```

The `git push` command is used to upload or push the local changes in your Git repository to a remote repository. It updates the remote repository with the latest changes made in your local branch, making them accessible to others who share the same remote repository.
The `git fetch` command is used to retrieve changes from a remote repository. It fetches any new branches or changes made in the remote repository since your last interaction. However, it does not automatically merge these changes into your local branches. After using `git fetch`, you can inspect the changes and decide whether to integrate them using `git merge` or `git rebase`.

```
AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git push
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 296 bytes | 296.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/khyaaati/siesworkshop.git
   0a63357..ad1b06a  master -> master

AI&DS 202@DESKTOP-EEAH3F9 MINGW64 ~/desktop/git-dvcs/git-demo-project/siesworkshop (master)
$ git fetch
```