Radha Tumbre - T23 - 121

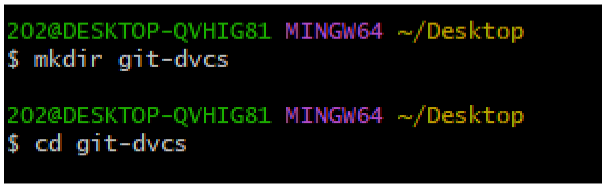
EXPERIMENT 3

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.

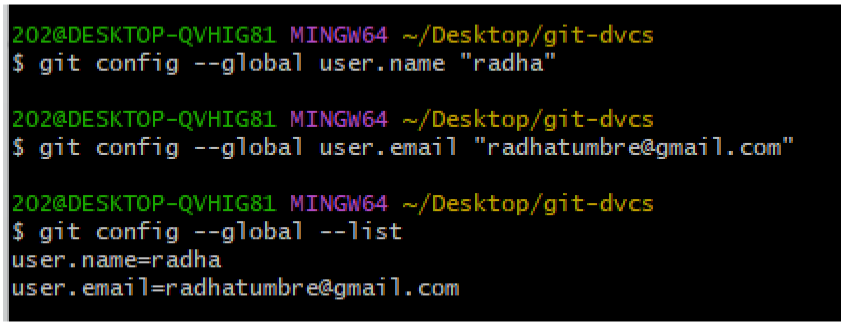
The command `cd git` is used to change the current working directory to the directory named "git." After executing this command, any subsequent commands or file operations will occur within the "git" directory. "cd" stands for "change directory."



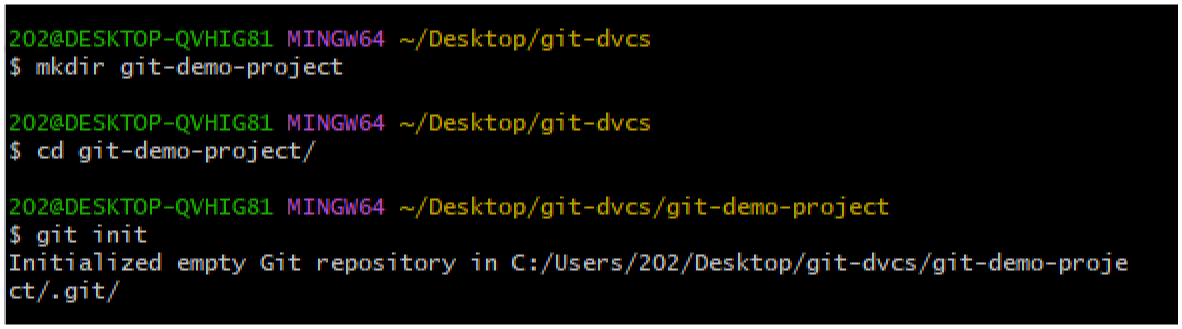
```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop
$ mkdir git-dvcs

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop
$ cd git-dvcs
```

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.

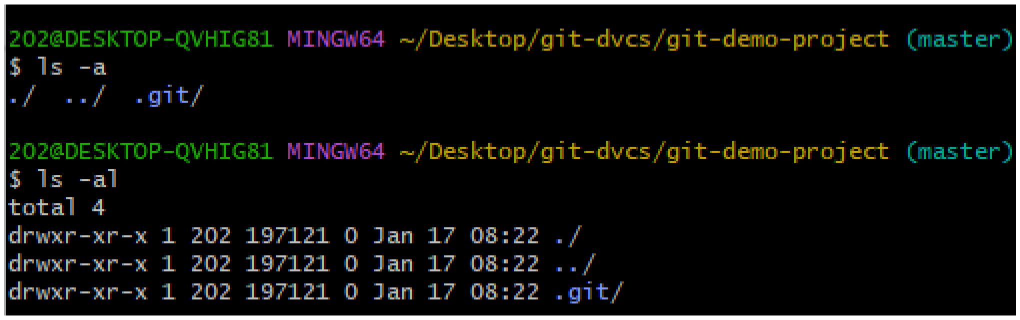
If you want to check your configuration settings, you can use the `git config --list` command to list all the settings Git can find at that point



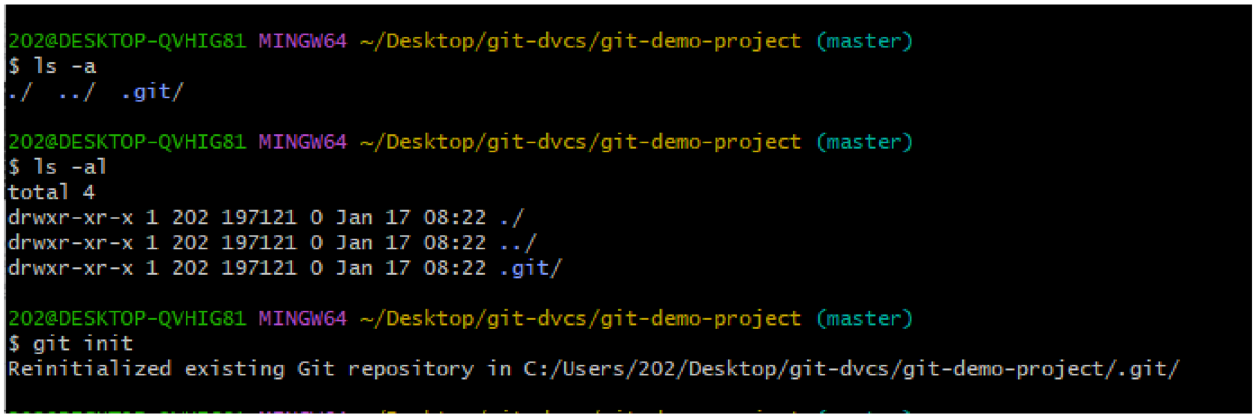
```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs
$ git config --global user.name "radha"

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs
$ git config --global user.email "radhatumbre@gmail.com"

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs
$ git config --global --list
user.name=radha
user.email=radhatumbre@gmail.com
```



```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs
$ mkdir git-demo-project

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs
$ cd git-demo-project/

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project
$ git init
Initialized empty Git repository in C:/Users/202/Desktop/git-dvcs/git-demo-proje
ct/.git/
```



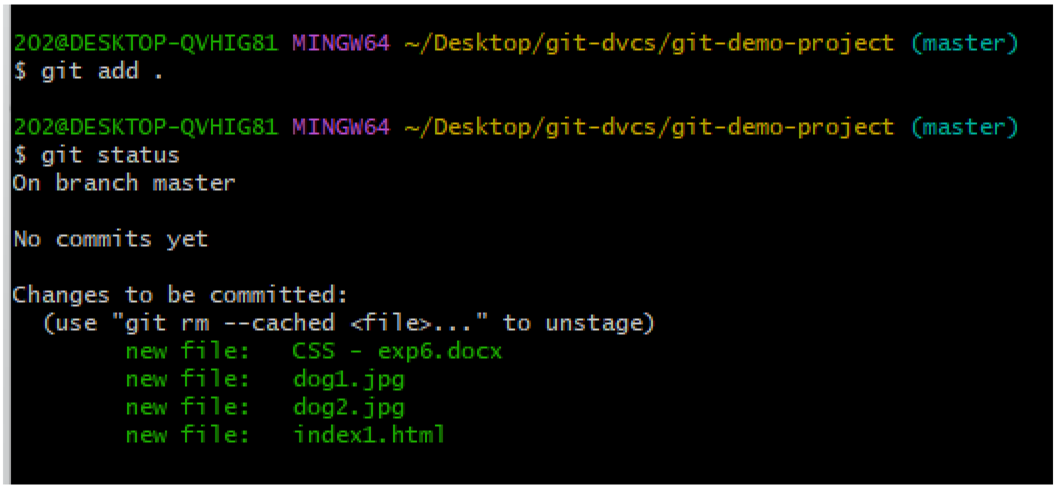
```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ ls -a
./  ../  .git/

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ ls -al
total 4
drwxr-xr-x 1 202 197121 0 Jan 17 08:22 ./
drwxr-xr-x 1 202 197121 0 Jan 17 08:22 ../
drwxr-xr-x 1 202 197121 0 Jan 17 08:22 .git/
```



```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ ls -a
./  ../  .git/

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ ls -al
total 4
drwxr-xr-x 1 202 197121 0 Jan 17 08:22 ./
drwxr-xr-x 1 202 197121 0 Jan 17 08:22 ../
drwxr-xr-x 1 202 197121 0 Jan 17 08:22 .git/

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git init
Reinitialized existing Git repository in C:/Users/202/Desktop/git-dvcs/git-demo-project/.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.

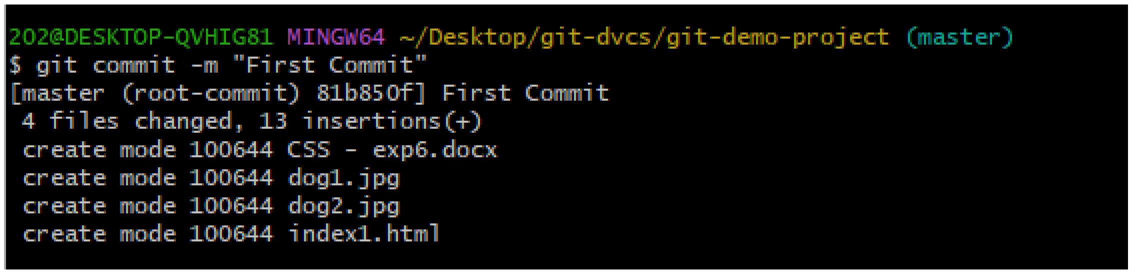
The ‘git status’ command is used in Git to display the current state of your working directory and staging area. When you run ‘git status’, Git provides information about which files have been modified, which files are staged for commit, and which files are untracked. Here's how you can use the git status command:



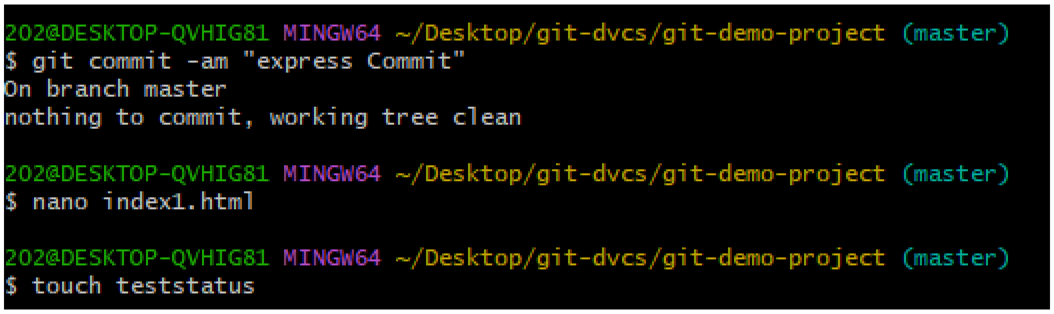
```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git add .

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   CSS - exp6.docx
        new file:   dog1.jpg
        new file:   dog2.jpg
        new file:   index1.html
```

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.



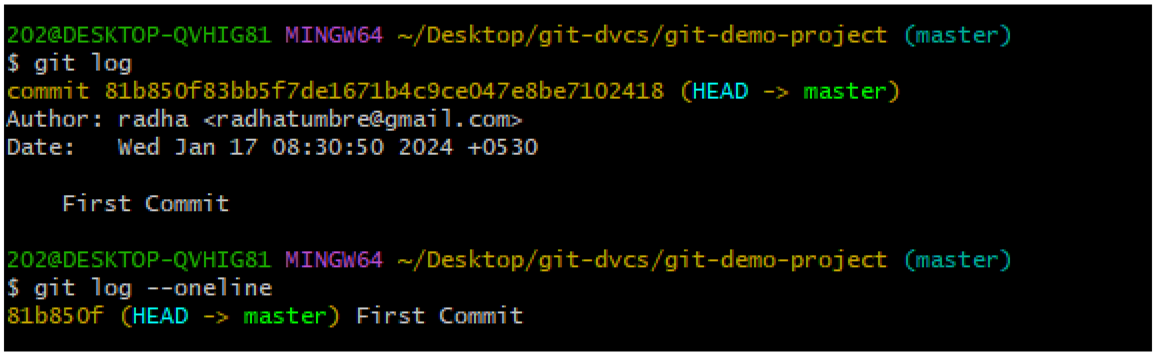
```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git commit -m "First Commit"
[master (root-commit) 81b850f] First Commit
 4 files changed, 13 insertions(+)
 create mode 100644 CSS - exp6.docx
 create mode 100644 dog1.jpg
 create mode 100644 dog2.jpg
 create mode 100644 index1.html
```



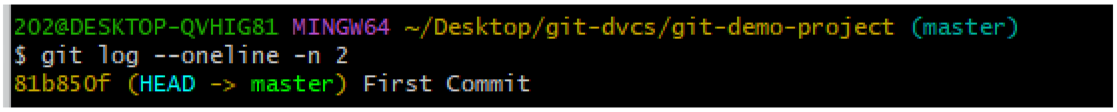
```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git commit -am "express Commit"
On branch master
nothing to commit, working tree clean

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ nano index1.html

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ touch teststatus
```

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.

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.



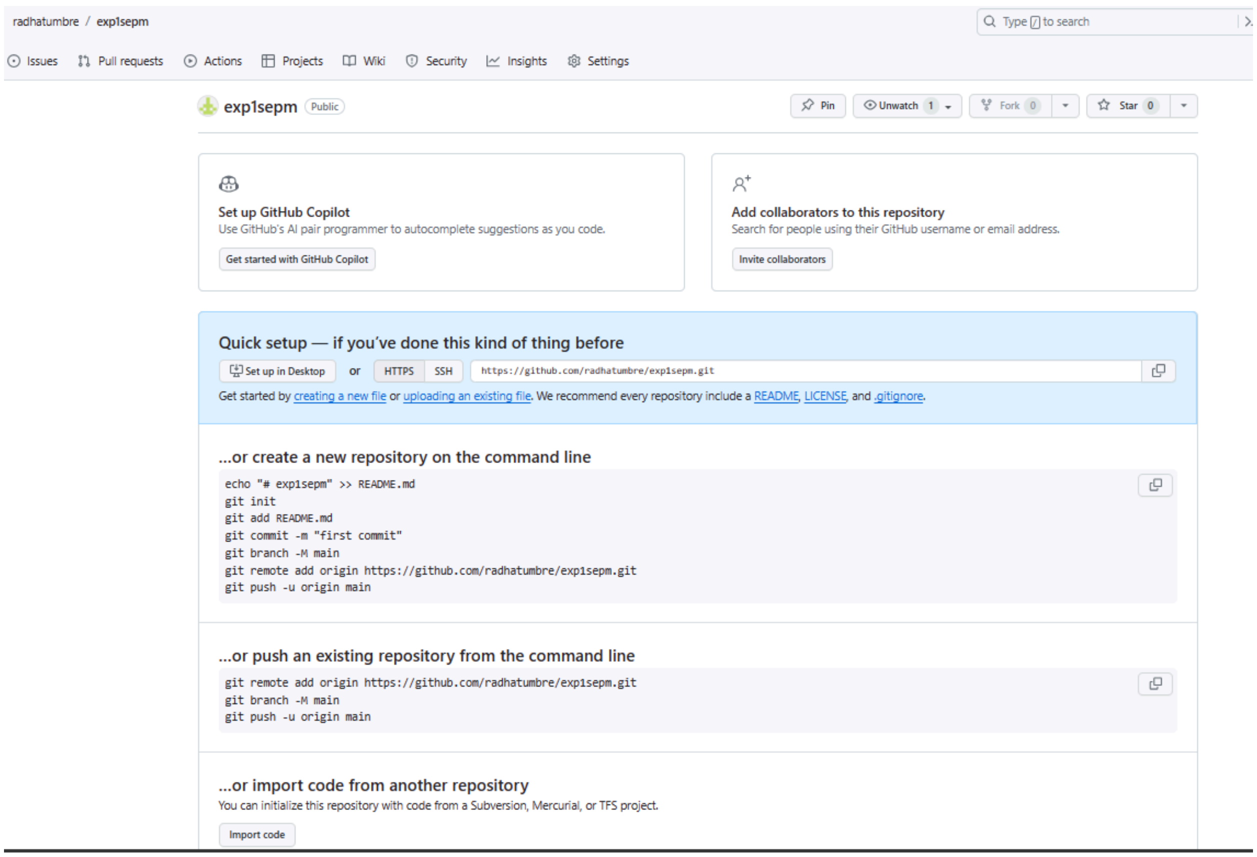
```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log
commit 81b850f83bb5f7de1671b4c9ce047e8be7102418 (HEAD -> master)
Author: radha <radhatumbre@gmail.com>
Date:   Wed Jan 17 08:30:50 2024 +0530

    First Commit

202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline
81b850f (HEAD -> master) First Commit
```



```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline -n 2
81b850f (HEAD -> master) First Commit
```

In GitHub, forking is a way to create a personal copy of someone else's project or repository. Forking a repository allows you to make changes to your own copy without affecting the original project. This is particularly useful for contributing to open-source projects or experimenting with changes to existing codebases.

Radha Tumbre - T23 - 121



Set up GitHub Copilot

Use GitHub's AI pair programmer to autocomplete suggestions as you code.

Get started with GitHub Copilot

Add collaborators to this repository

Search for people using their GitHub username or email address.

Invite collaborators

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH `https://github.com/radhatumbre/exp1sepm.git`

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# exp1sepm" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/radhatumbre/exp1sepm.git
git push -u origin main
```

...or push an existing repository from the command line

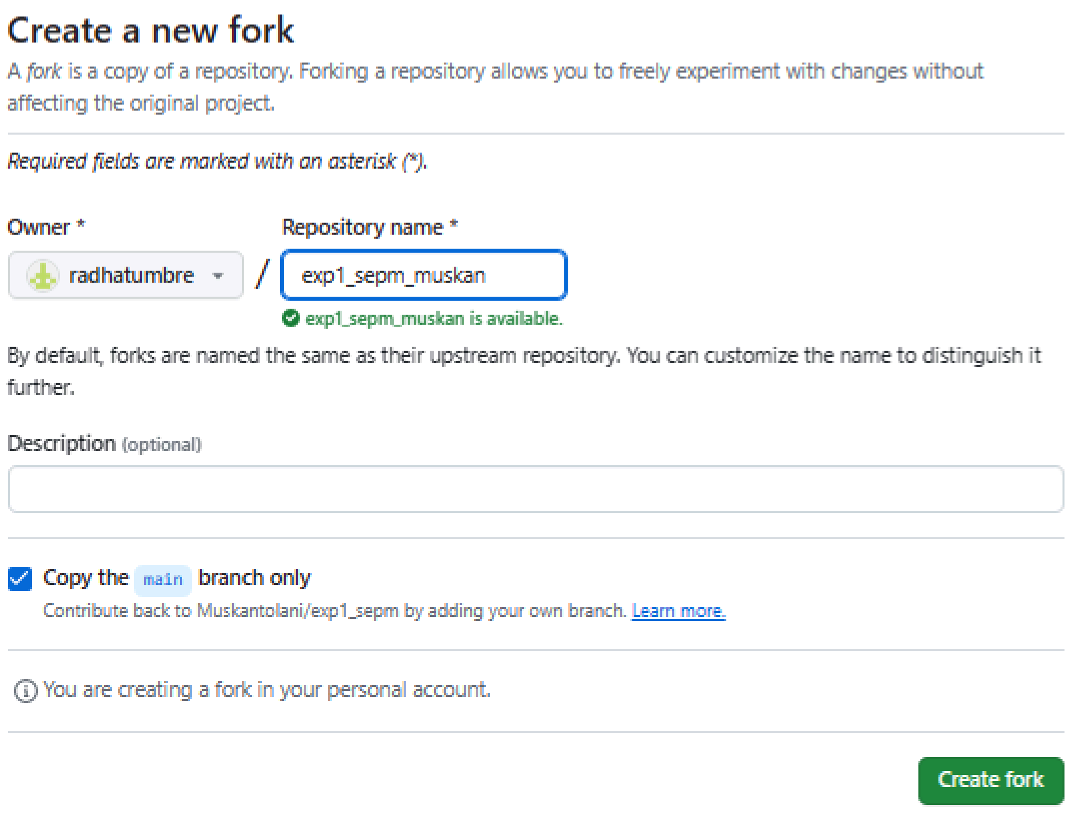
```
git remote add origin https://github.com/radhatumbre/exp1sepm.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Radha Tumbre - T23 - 121



Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (*).

Owner * radhatumbre / Repository name * exp1_sepm_muskan

exp1_sepm_muskan is available.

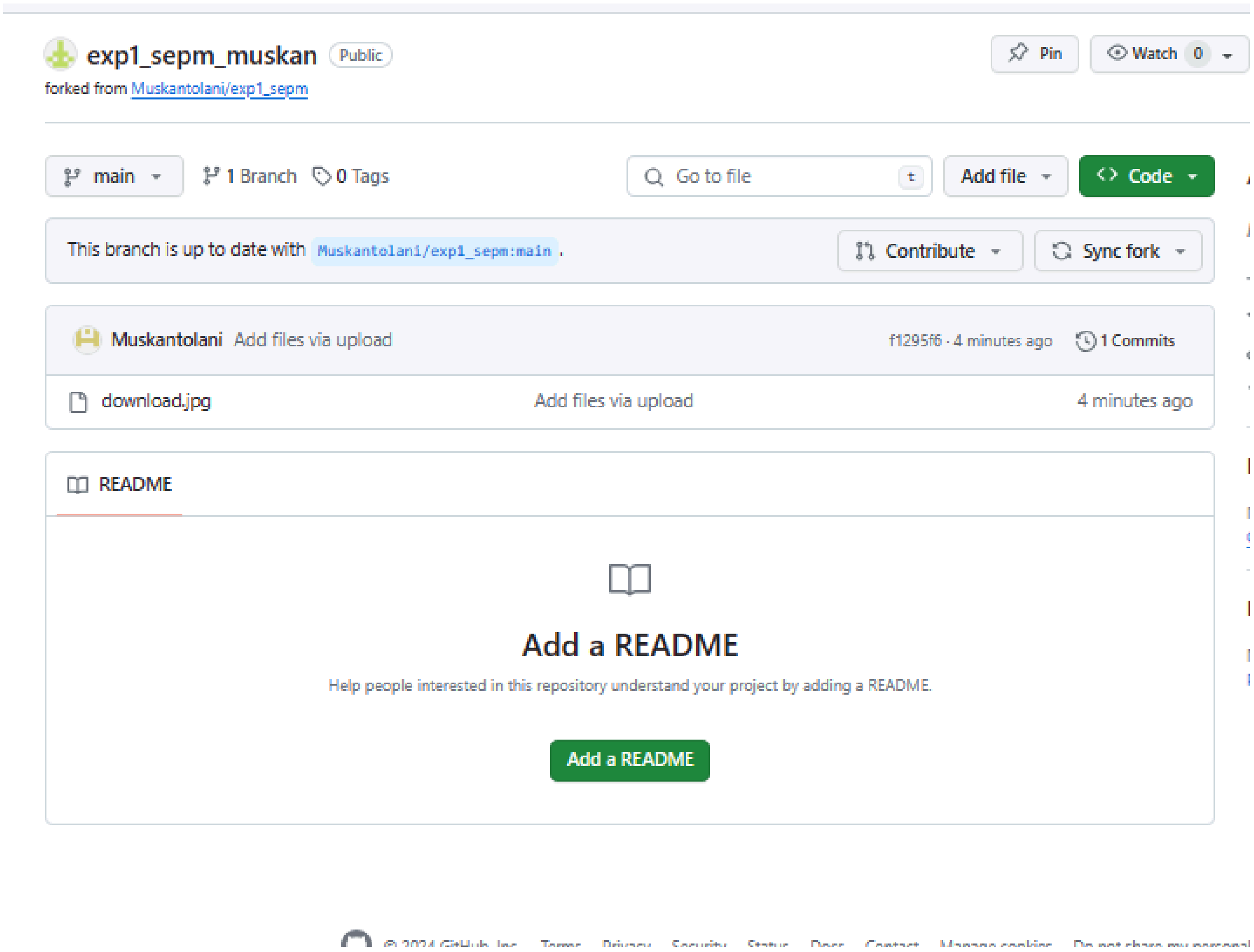
By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

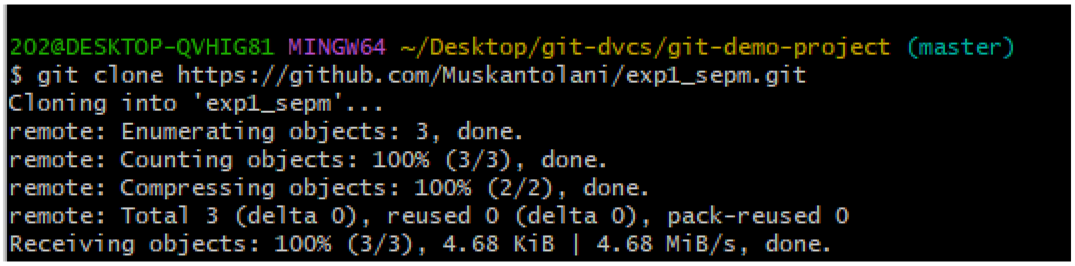
[✓] Copy the `main` branch only

Contribute back to Muskantolani/exp1_sepm by adding your own branch. Learn more.

You are creating a fork in your personal account.


exp1_sepm_muskan
Public
forked from Muskantolani/exp1_sepm
Pin
Watch 0
main
1 Branch
0 Tags
Go to file
Add file
Code
This branch is up to date with Muskantolani/exp1_sepm:main .
Contribute
Sync fork
Muskantolani Add files via upload
f1295f6 · 4 minutes ago
1 Commits
download.jpg
Add files via upload
4 minutes ago
README
Add a README
Help people interested in this repository understand your project by adding a README.
Add a README

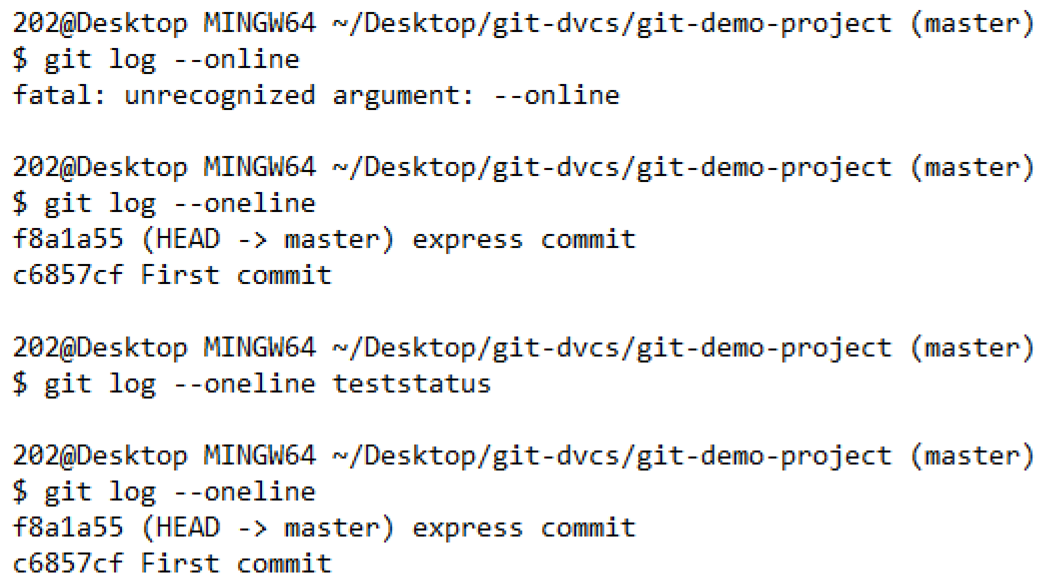
The `git clone` command is used to create a copy of a Git repository. When you run this command, it duplicates the entire repository, including its files, commit history, and branches, and downloads it to your local machine. This is often the initial step when you want to work with a project hosted on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



```
202@DESKTOP-QVHIG81 MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git clone https://github.com/Muskantolani/exp1_sepm.git
Cloning into 'exp1_sepm'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), 4.68 KiB | 4.68 MiB/s, done.
```

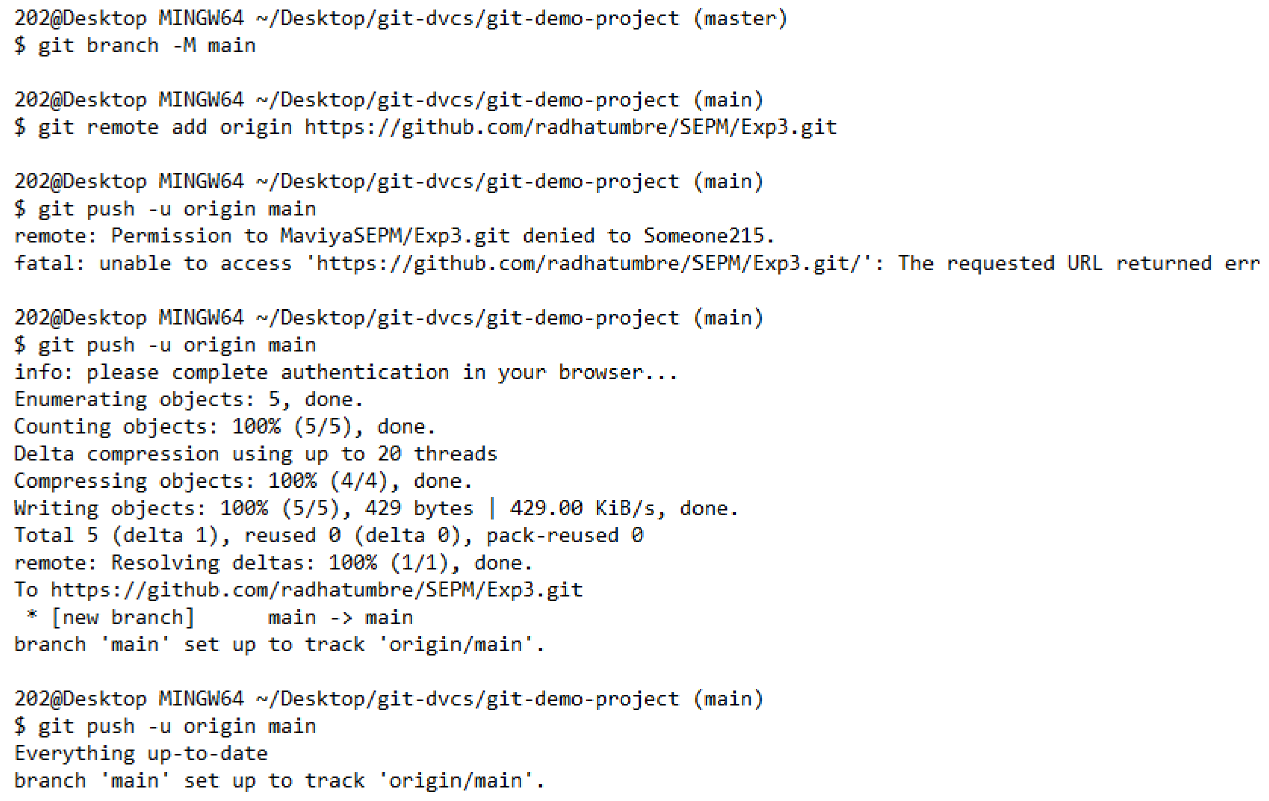
The command `touch teststatus` creates an empty file named "teststatus" in the current directory. The `touch` command is commonly used to update the timestamps of a file or create an empty file if it doesn't exist.

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.



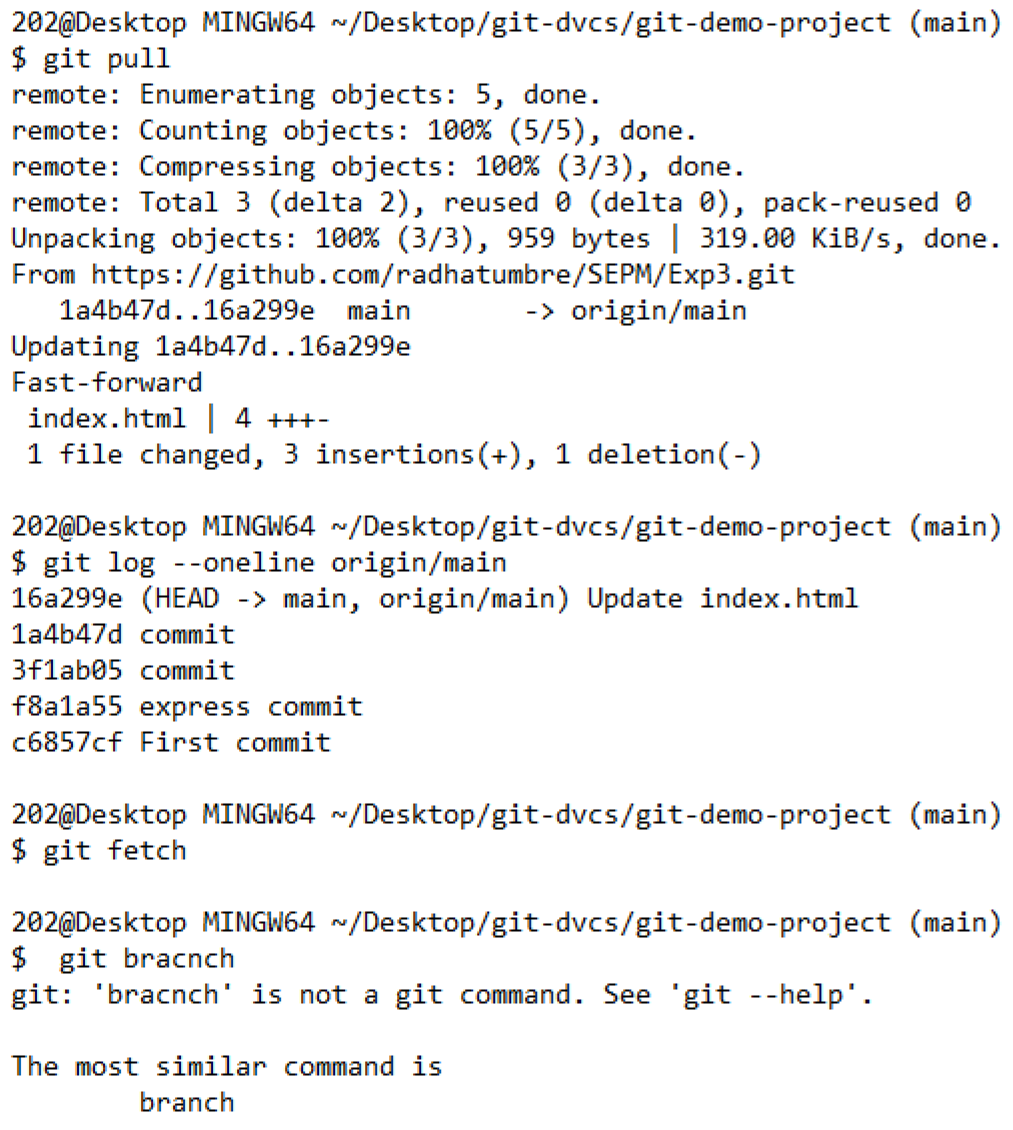
```
202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --online
fatal: unrecognized argument: --online

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline
f8a1a55 (HEAD -> master) express commit
c6857cf First commit

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline teststatus

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git log --oneline
f8a1a55 (HEAD -> master) express commit
c6857cf First commit
```

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.



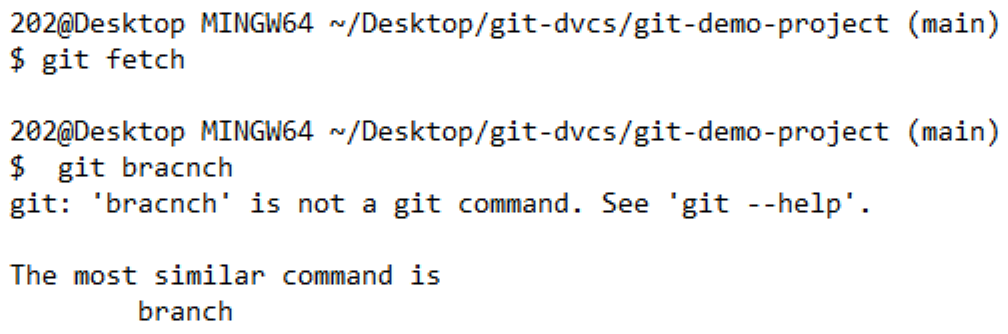
```
202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (master)
$ git branch -M main

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git remote add origin https://github.com/radhatumbre/SEPM/Exp3.git

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git push -u origin main
remote: Permission to MaviyaSEPM/Exp3.git denied to Someone215.
fatal: unable to access 'https://github.com/radhatumbre/SEPM/Exp3.git/': The requested URL returned err

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git push -u origin main
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 20 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 429 bytes | 429.00 KiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/radhatumbre/SEPM/Exp3.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git push -u origin main
Everything up-to-date
branch 'main' set up to track 'origin/main'.
```

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.



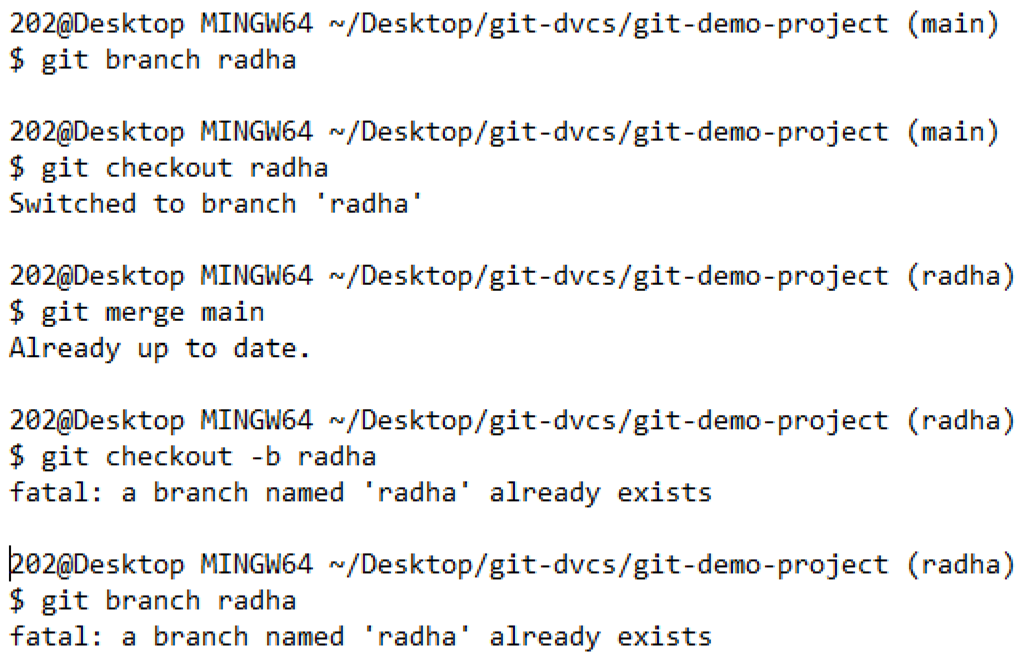
```
202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 959 bytes | 319.00 KiB/s, done.
From https://github.com/radhatumbre/SEPM/Exp3.git
   1a4b47d..16a299e  main       -> origin/main
Updating 1a4b47d..16a299e
Fast-forward
 index.html | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git log --oneline origin/main
16a299e (HEAD -> main, origin/main) Update index.html
1a4b47d commit
3f1ab05 commit
f8a1a55 express commit
c6857cf First commit

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git fetch

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$  git bracnch
git: 'bracnch' is not a git command. See 'git --help'.

The most similar command is
        branch
```

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`.



```
202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git fetch

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$  git bracnch
git: 'bracnch' is not a git command. See 'git --help'.

The most similar command is
        branch
```

The git branch command in Git is used to list, create, or delete branches in a Git repository.
To create a new branch, you use the same ‘git branch’ command followed by the name of the new branch:
However, creating a branch alone doesn't switch to it. You need to use ‘git checkout’ or ‘git switch’ to switch to the new branch:



```
202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git branch radha

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (main)
$ git checkout radha
Switched to branch 'radha'

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (radha)
$ git merge main
Already up to date.

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (radha)
$ git checkout -b radha
fatal: a branch named 'radha' already exists

202@Desktop MINGW64 ~/Desktop/git-dvcs/git-demo-project (radha)
$ git branch radha
fatal: a branch named 'radha' already exists
```