

**NAME : KAMBHAM HEMANTH REDDY**

**SRM UNIVERSITY**

**REG NO : RA2111030010087**

**EMAIL : kk8325@srmist.edu.in**

**COGNIZANT WEEK1 TASK**

**CSE Cybersecurity**

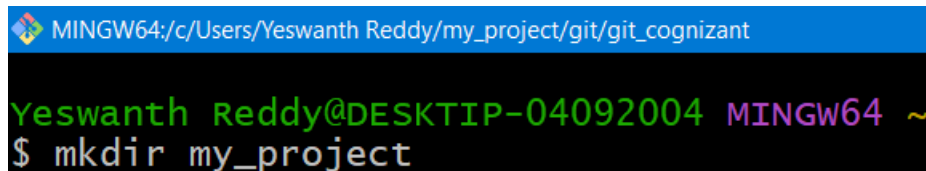
## **Exercise 1: Introduction to Version Control**

### **Objective:**

Initialize a new Git repository and commit your first file.

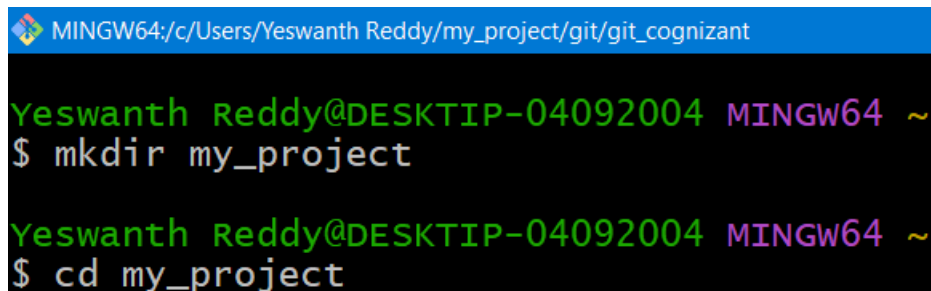
### **Instructions:**

1. Create a new directory for your project.



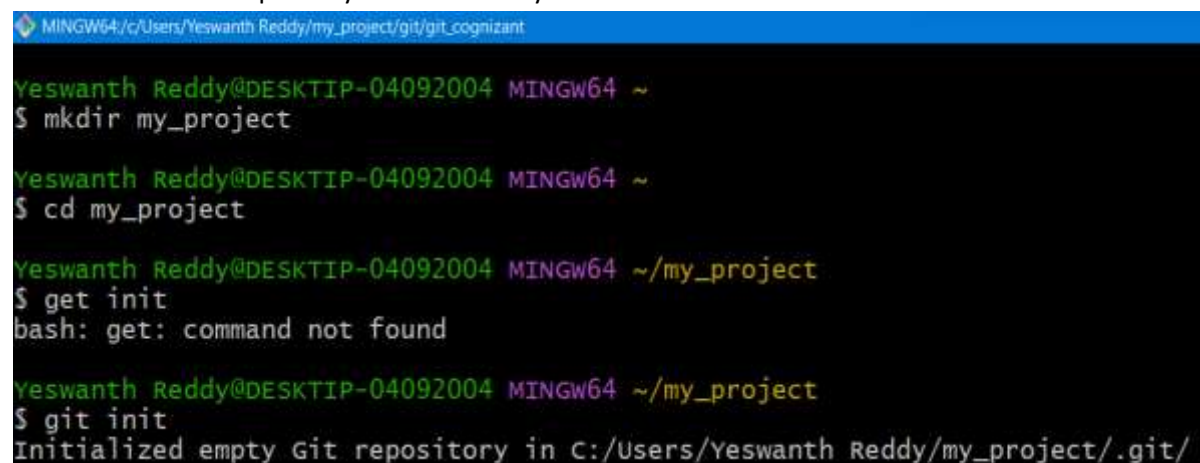
A terminal window with a blue title bar containing the path 'MINGW64:/c/Users/Yeswanth Reddy/my\_project/git/git\_cognizant'. The prompt is 'Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~'. The command '\$ mkdir my\_project' has been entered.

2. Navigate into the directory.



A terminal window with a blue title bar containing the path 'MINGW64:/c/Users/Yeswanth Reddy/my\_project/git/git\_cognizant'. The prompt is 'Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~'. The command '\$ mkdir my\_project' has been entered. The prompt is now 'Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~'. The command '\$ cd my\_project' has been entered.

3. Initialize a new Git repository in the directory.



A terminal window with a blue title bar containing the path 'MINGW64:/c/Users/Yeswanth Reddy/my\_project/git/git\_cognizant'. The prompt is 'Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~'. The command '\$ mkdir my\_project' has been entered. The prompt is now 'Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~'. The command '\$ cd my\_project' has been entered. The prompt is now 'Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my\_project'. The command '\$ get init' has been entered, resulting in the error 'bash: get: command not found'. The prompt is now 'Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my\_project'. The command '\$ git init' has been entered, resulting in the output 'Initialized empty Git repository in C:/Users/Yeswanth Reddy/my\_project/.git/'.

4. Create a new file named file1.txt and add some content to it.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project
$ git init
Initialized empty Git repository in C:/Users/Yeswanth Reddy/my_project/.git/

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ echo "This is my first file" > file1.txt
```

4. Add the file to the staging area.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ git add file1.txt
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it
```

5. Commit the file with a commit message.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ git commit -m "Initial commit: Add file1.txt"
[master (root-commit) 78adffc] Initial commit: Add file1.txt
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
```

## Exercise:1 picture

```
MINGW64/C:/Users/Yeswanth Reddy/my_project/git/git_cognizant

Yeswanth Reddy@DESKTOP-04092004 MINGW64 -
$ mkdir my_project

Yeswanth Reddy@DESKTOP-04092004 MINGW64 -
$ cd my_project

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project
$ get init
bash: get: command not found

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project
$ git init
Initialized empty Git repository in C:/Users/Yeswanth Reddy/my_project/.git/

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ echo "This is my first file" > file1.txt

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ git add file1.txt
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ git commit -m "Initial commit: Add file1.txt"
[master (root-commit) 78adffc] Initial commit: Add file1.txt
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
```

## Exercise 2: Understanding Git

### Objective:

Clone an existing repository and explore its history.

### Instructions:

1. Clone a public repository from a platform like GitHub.

```
MINGW64:/c/Users/Yeswanth Reddy/my_project/git/git_cognizant

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ git clone https://github.com/git/git.git
Cloning into 'git'...
remote: Enumerating objects: 371962, done.
remote: Counting objects: 100% (432/432), done.
remote: Compressing objects: 100% (208/208), done.
remote: Total 371962 (delta 252), reused 387 (delta 224), pack-reused 371530
Receiving objects: 100% (371962/371962), 243.04 MiB | 997.00 KiB/s, done.
Resolving deltas: 100% (280066/280066), done.
Updating files: 100% (4509/4509), done.
```

2. Navigate into the cloned repository.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ cd git
```

3. Check the commit history.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project (master)
$ cd git

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git log
commit 39bf06adf96da25b87c9aa7d35a32ef3683eb4a4 (HEAD -> master, tag: v2.46.0)
Author: Junio C Hamano <gitster@pobox.com>
Date: Mon Jul 29 07:14:09 2024 -0700

    Git 2.46
```

4. Show changes introduced by a specific commit.

```
MINGW64:/c/Users/Yeswanth Reddy/my_project/git/git_cognizant

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git show 114bff72ac030b9e9c931a9efd2bd0af8137692b
commit 114bff72ac030b9e9c931a9efd2bd0af8137692b
Author: Derrick Stolee <stolee@gmail.com>
Date:   Fri Jun 28 12:43:25 2024 +0000

    sparse-index: improve lstat caching of sparse paths

    The clear_skip_worktree_from_present_files() method was first introduced
    in af6a51875a (repo_read_index: clear SKIP_WORKTREE bit from files
    present in worktree, 2022-01-14) to allow better interaction with the
    working directory in the presence of paths outside of the
    sparse-checkout. The initial implementation would lstat() every single
    SKIP_WORKTREE path to see if it existed; if it ran across a sparse
    directory that existed (when a sparse index was in use), then it would
    expand the index and then check every SKIP_WORKTREE path.

    Since these lstat() calls were very expensive, this was improved in
    d79d299352 (Accelerate clear_skip_worktree_from_present_files() by
    caching, 2022-01-14) by caching directories that do not exist so it
    could avoid lstat()ing any files under such directories. However, there
    are some inefficiencies in that caching mechanism.

    The caching mechanism stored only the parent directory as not existing,
    even if a higher parent directory also does not exist. This means that
    wasted lstat() calls would occur when the paths passed to path_found()
    change immediate parent directories but within the same parent directory
    that does not exist.

    To create an example repository that demonstrates this problem, it helps
    to have a directory outside of the sparse-checkout that contains many
    deep paths. In particular, the first paths (in lexicographic order)
    underneath the sparse directory should have deep directory structures,
    maximizing the difference between the old caching algorithm that looks
    to a single parent and the new caching algorithm that looks to the
    top-most missing directory.

    The performance test script p2000-sparse-operations.sh takes the sample
    repository and copies its HEAD to several copies nested in directories
    of the form f<i>/f<j>/f<k> where i, j, and k are numbers from 1 to 4.
    The sparse-checkout cone is then selected as "f2/f4/". Creating "f1/f1/"
    will trigger the behavior and also lead to some interesting cases for
    the caching algorithm since "f1/f1/" exists but "f1/f2/" and "f3/" do
    not.

    This is difficult to notice when running performance tests using the Git
    repository (or a blow-up of the Git repository, as in
    p2000-sparse-operations.sh) because Git has a very shallow directory
    structure.
```

## Exercise 3: Setting Up Git

### Objective:

Set up Git configuration and verify it.

### Instructions:

1. Set your username for Git.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git config --global user.name "KAMBHAM HEMANTH REDDY"
```

2. Set your email for Git.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git config --global user.email "kk8325@srmist.edu.in"
```

3. Verify your configuration settings.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=kk8325@srmist.edu.in
user.name=KAMBHAM HEMANTH REDDY
user.nmae=KAMBHAM_HEMANTH_REDDY
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
remote.origin.url=https://github.com/git/git.git
remote.origin.fetch=+refs/heads/*:refs/remotes/origin/*
branch.master.remote=origin
branch.master.merge=refs/heads/master
user.email=kk8325@srmist.edu.in
user.name=KAMBHAM HEMANTH REDDY
user.nmae=KAMBHAM_HEMANTH_REDDY
```

## Exercise 4: Basic Git Commands

### Objective:

Practice basic Git commands by modifying files and tracking changes.

### Instructions:

1. Create a new file named file2.txt and add some content to it.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ echo "This is the content for file2.txt" > file2.txt
```

2. Add the file to the staging area.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git add file2.txt
warning: in the working copy of 'file2.txt', LF will be replaced by CRLF the next time Git touches it
```

3. Commit the new file with a commit message.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git commit -m "Add file2.txt with initial content"
[master ced5b8e54a] Add file2.txt with initial content
1 file changed, 1 insertion(+)
create mode 100644 file2.txt
```

4. Modify the existing file1.txt and add more content to it.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ echo "Adding more content to the file1.txt" >> file1.txt
```

5. Add the modified file to the staging area.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git add file1.txt
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it
```

6. Commit the changes with a commit message.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git commit -m "Update file1.txt with additional content"
[master 0d489c8235] Update file1.txt with additional content
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
```



7. View the current status of your repository.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

8. View the differences between your working directory and the repository.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git diff
```

## Exercise 4 : Picture

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ echo "This is the content for file2.txt" > file2.txt

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git add file2.txt
warning: in the working copy of 'file2.txt', LF will be replaced by CRLF the next time Git touches it

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git commit -m "Add file2.txt with initial content"
[master ced5b8e54a] Add file2.txt with initial content
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ echo "Adding more content to the file1.txt" >> file1.txt

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git add file1.txt
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git commit -m "Update file1.txt with additional content"
[master 0d489c8235] Update file1.txt with additional content
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git diff

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git diff --staged
```

## Exercise 5: Branching and Merging

### Objective:

Create a new branch, make changes, and merge it back to the main branch.

### Instructions:

1. Create a new branch named new-feature.

```
MINGW64:/c/Users/Yeswanth Reddy/my_project/git/git_cognizant

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git checkout -b new-feature
Switched to a new branch 'new-feature'
```

2. Switch to the new branch (if not already switched).

```
MINGW64:/c/Users/Yeswanth Reddy/my_project/git/git_cognizant

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git checkout -b new-feature
Switched to a new branch 'new-feature'

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
```

3. Create a new file named feature.txt and add some content to it.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ echo "this s the content for feature.txt" > feature.txt
```

4. Add the file to the staging area.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ git add feature.txt
warning: in the working copy of 'feature.txt', LF will be replaced by CRLF the next time Git touches it
```

5. Commit the new file with a commit message.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ git commit -m "Add feature.txt with initial content"
[new-feature 3cf1e67c2f] Add feature.txt with initial content
1 file changed, 1 insertion(+)
create mode 100644 feature.txt
```

6. Switch back to the main branch.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)
```

7. Merge the new-feature branch into the main branch.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git merge new-feature
Updating 0d489c8235..3cf1e67c2f
Fast-forward
 feature.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 feature.txt
```



8. Resolve any conflicts if they arise and commit the merge.  
No conflicts raised during the commit the merge.

## Exercise 6 : Picture

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git checkout -b new-feature
Switched to a new branch 'new-feature'

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ echo "This s the content for feature.txt" > feature.txt

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ git add feature.txt
warning: in the working copy of 'feature.txt', LF will be replaced by CRLF the next time Git touches it

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ git commit -m "Add feature.txt with initial content"
[new-feature 3cf1e67c2f] Add feature.txt with initial content
1 file changed, 1 insertion(+)
create mode 100644 feature.txt

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ git checkout main
error: pathspec 'main' did not match any file(s) known to git

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (new-feature)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git merge new-feature
Updating 0d489c8235..3cf1e67c2f
Fast-forward
 feature.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 feature.txt
```

## Exercise 6: Remote Repositories

### Objective:

Add a remote repository and push your local changes.

### Instructions:

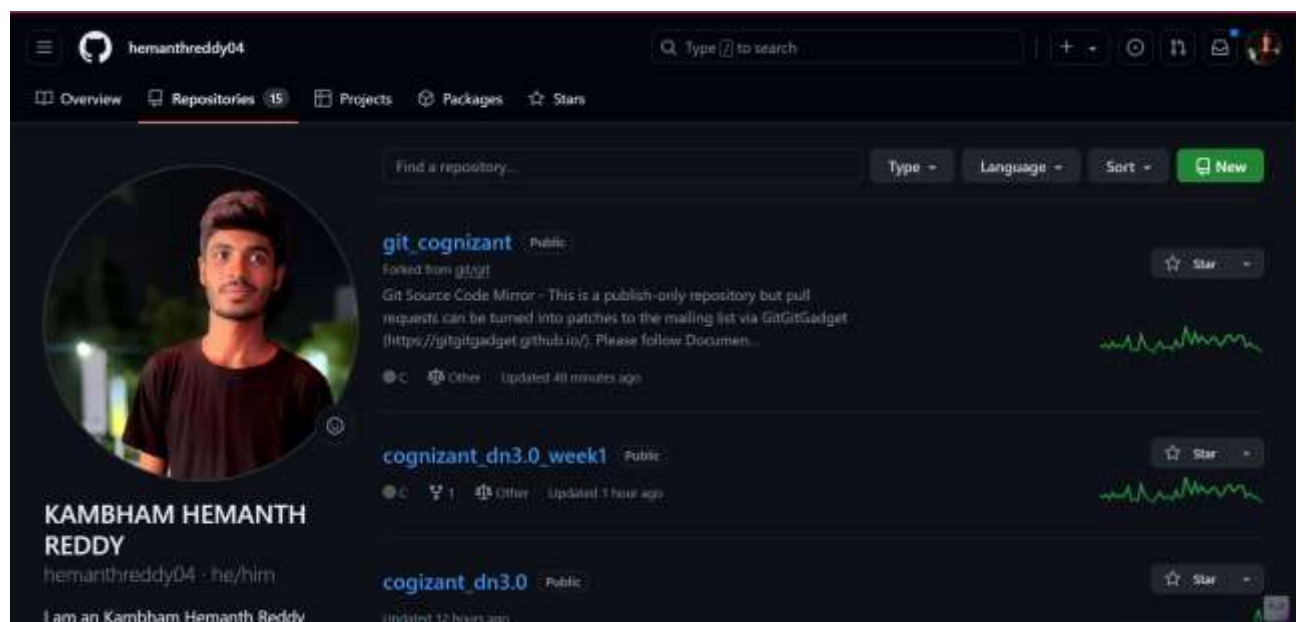
1. Add a remote repository URL to your local Git repository.

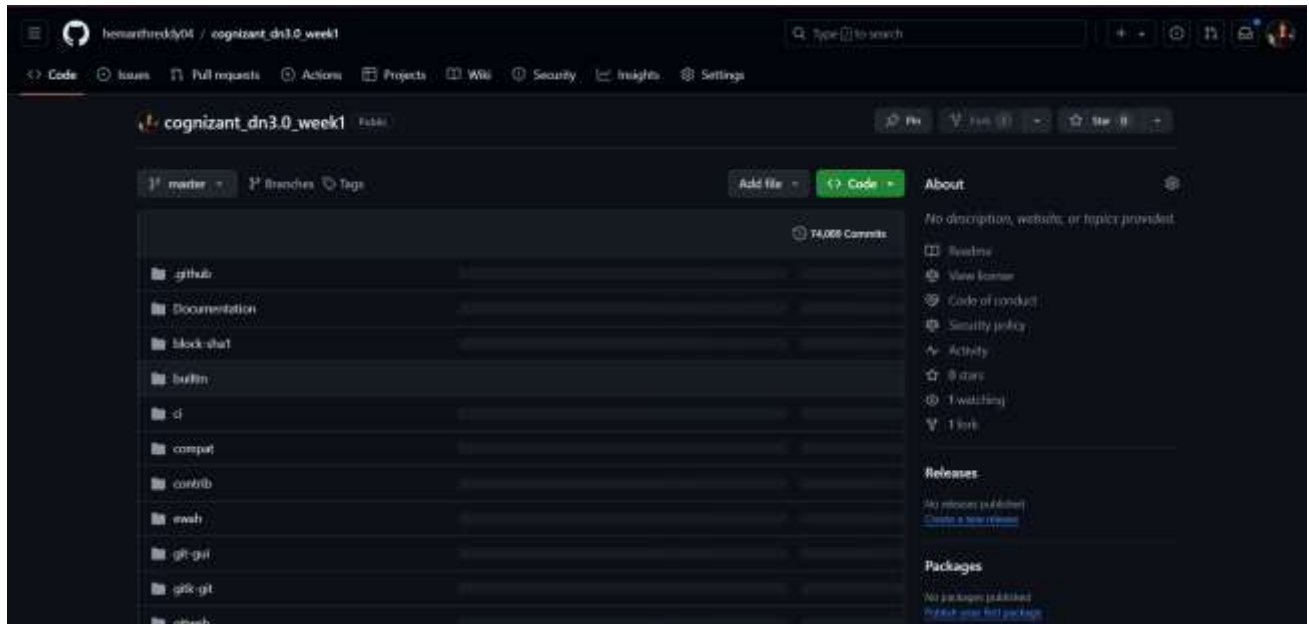
```
MINGW64:/c/Users/Yeswanth Reddy/my_project/git/git_cognizant  
  
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)  
$ git remote add cognizantweek1 https://github.com/hemanthreddy04/cognizant_dn3.0_week1.git
```

2. Push your local changes to the remote repository.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)  
$ git push -u cognizantweek1 master  
Enumerating objects: 360812, done.  
Counting objects: 100% (360812/360812), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (84742/84742), done.  
Writing objects: 100% (360812/360812), 235.13 MiB | 1.91 MiB/s, done.  
Total 360812 (delta 273713), reused 360794 (delta 273702), pack-reused 0 (from 0)
```

## Exercise 6 : Pictures





```
To https://github.com/hemanthreddy04/cognizant_dn3.0_week1.git
* [new branch]      master -> master
branch 'master' set up to track 'cognizantweek1/master'.
```

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git (master)
$ git remote -v
cognizantweek1 https://github.com/hemanthreddy04/cognizant_dn3.0_week1.git (fetch)
cognizantweek1 https://github.com/hemanthreddy04/cognizant_dn3.0_week1.git (push)
origin https://github.com/git/git.git (fetch)
origin https://github.com/git/git.git (push)
```

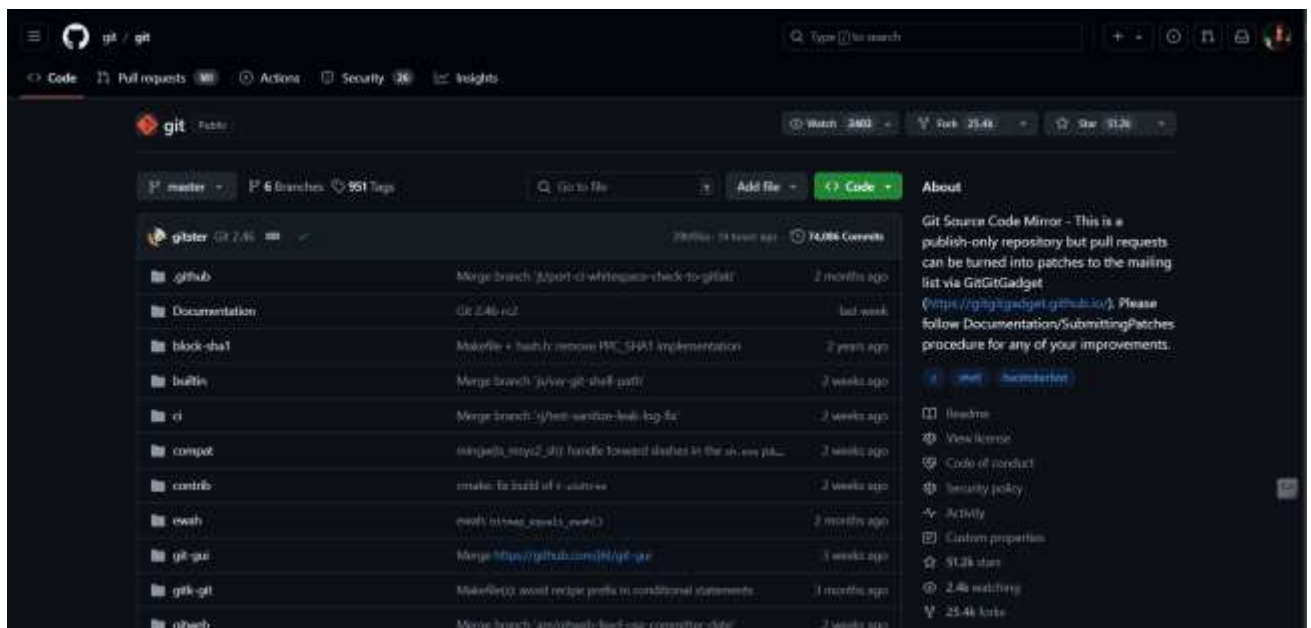
## Exercise 7: Collaborating with Git

### Objective:

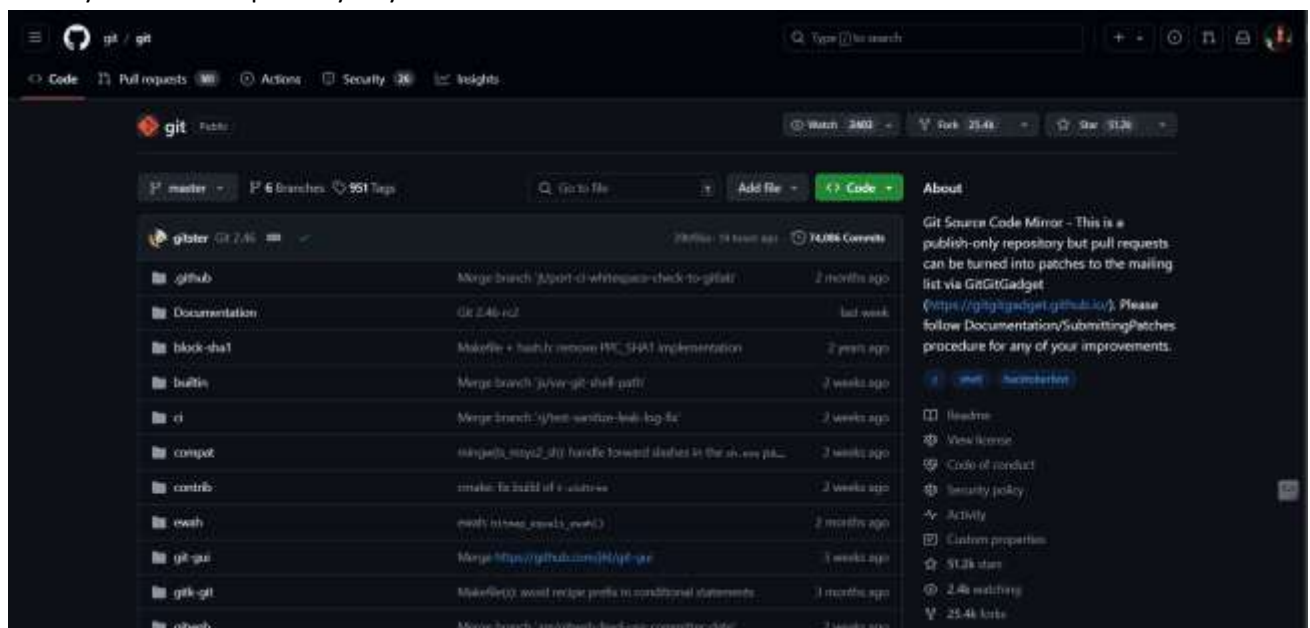
Collaborate on a repository by creating a pull request.

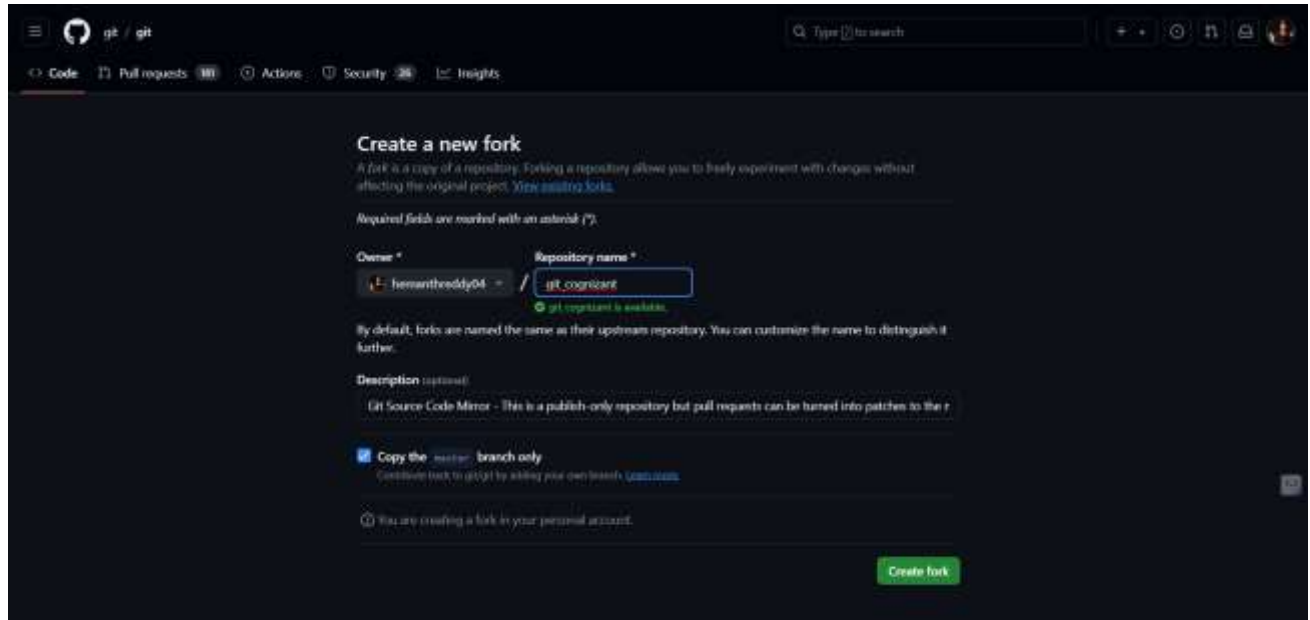
### Instructions:

1. Fork a repository on GitHub.



2. Clone your forked repository to your local machine.





```
MINGW64/c/Users/Yeswanth Reddy/my_project/git/git_cognizant

Yeswanth Reddy@DESKTIP-04092004 MINGW64 ~/my_project/git (master)
$ git clone https://github.com/hemanthreddy04/git_cognizant
Cloning into 'git_cognizant'...
remote: Enumerating objects: 360803, done.
remote: Counting objects: 100% (38/38), done.
remote: Compressing objects: 100% (18/18), done.
Receiving objects: 100% (360803/360803), 235.53 MiB | 4.25 MiB/s, done.
remote: Total 360803 (delta 21), reused 35 (delta 20), pack-reused 360765
Resolving deltas: 100% (273730/273730), done.
Updating files: 100% (4509/4509), done.
```

3. Navigate into the cloned repository.

```
Yeswanth Reddy@DESKTIP-04092004 MINGW64 ~/my_project/git (master)
$ cd git_cognizant
```

4. Create a new branch for your changes.

```
Yeswanth Reddy@DESKTIP-04092004 MINGW64 ~/my_project/git/git_cognizant (master)
$ git checkout -b my-changes
Switched to a new branch 'my-changes'
```

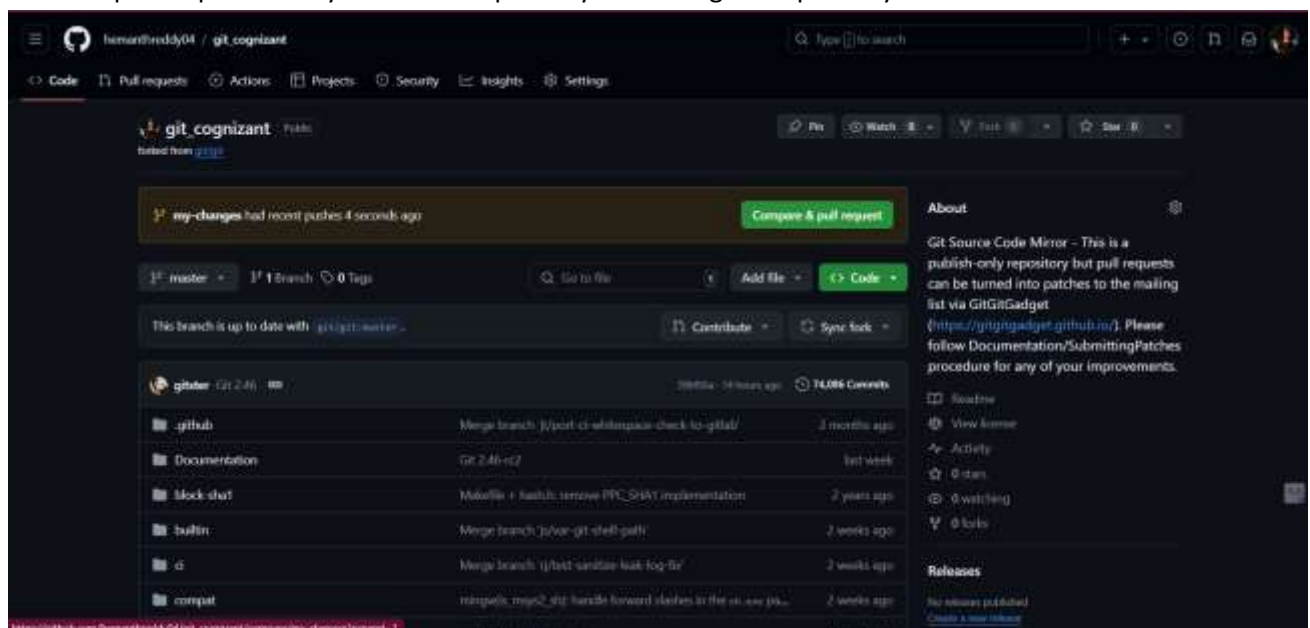
5. Make your changes and commit them.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git/git_cognizant (my-changes)
$ git commit -m "Describe your changes"
[my-changes 8efa602b40] Describe your changes
1 file changed, 1 insertion(+)
create mode 100644 file3.txt
```

6. Push the branch to your forked repository.

```
Yeswanth Reddy@DESKTOP-04092004 MINGW64 ~/my_project/git/git_cognizant (my-changes)
$ git push -u origin my-changes
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 305 bytes | 101.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'my-changes' on GitHub by visiting:
remote:   https://github.com/hemanthreddy04/git_cognizant/pull/new/my-changes
remote:
To https://github.com/hemanthreddy04/git_cognizant
 * [new branch]      my-changes -> my-changes
branch 'my-changes' set up to track 'origin/my-changes'.
```

7. Create a pull request from your forked repository to the original repository.



The screenshot shows the GitHub web interface for the repository 'git\_cognizant' by user 'hemanthreddy04'. The repository is a public fork of 'git\_cognizant' by 'grip'. The 'my-changes' branch is selected, and a 'Compare & pull request' button is visible. The repository is up to date with 'grip:master'. The file list shows files like '.github', 'Documentation', 'lock.sh', 'main', 'ci', and 'comput'. The commit history shows recent merges and commits.