DAY 2 TASK

SUTHARSAN M

73152113110

Exercise 1:

Main Task

1. Create a new directory and change into it.

2. Use the init command to create a Git repository in that directory.

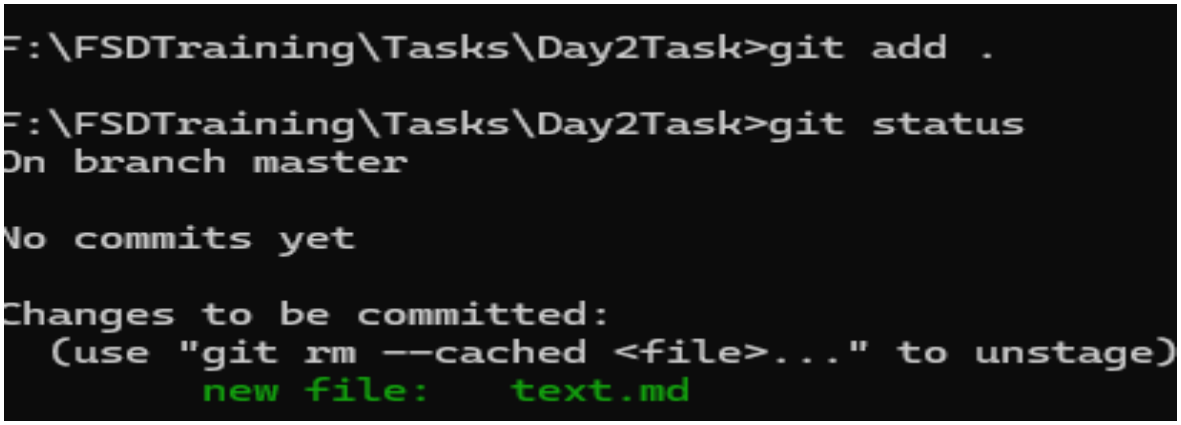
3. Observe that there is now a .git directory.

4. Create a README file.

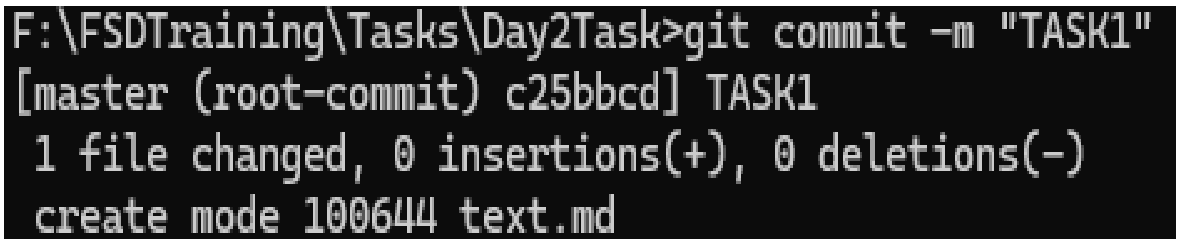
5. Look at the output of the status command; the README you created should appear as an untracked file.

```
F:\FSDTraining\Tasks\Day2Task>git init
Initialized empty Git repository in F:/FSDTraining/Tasks/Day2Task/.git/

F:\FSDTraining\Tasks\Day2Task>git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to tr
ack)
```

6. Use the add command to add the new file to the staging area. Again, look at the output of the status command.



7. Now use the commit command to commit the contents of the staging area.



8. Create a src directory and add a couple of files to it.

9. Use the add command, but name the directory, not the individual files. Use the status command. See how both files have been staged. Commit them.

```
F:\FSDTraining\Tasks\Day2Task>mkdir src

F:\FSDTraining\Tasks\Day2Task>git status
On branch master
nothing to commit, working tree clean

F:\FSDTraining\Tasks\Day2Task>git add .

F:\FSDTraining\Tasks\Day2Task>git status
On branch master
nothing to commit, working tree clean

F:\FSDTraining\Tasks\Day2Task>git add .

F:\FSDTraining\Tasks\Day2Task>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/file1.txt
        new file:   src/file2.txt


F:\FSDTraining\Tasks\Day2Task>git commit -m "TASK2"
[master b2a1176] TASK2
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 src/file1.txt
 create mode 100644 src/file2.txt
```

10. Make a change to one of the files. Use the diff command to view the details of the change.

```
F:\FSDTraining\Tasks\Day2Task>git diff b2a11767bab9590443c1de5b2d9040cb7
bd5aca5 c25bbcd5a1d5eb892b0e010c8fec425162746d24
diff --git a/src/file1.txt b/src/file1.txt
deleted file mode 100644
index e69de29..0000000
diff --git a/src/file2.txt b/src/file2.txt
deleted file mode 100644
index e69de29..0000000

F:\FSDTraining\Tasks\Day2Task>cd src

F:\FSDTraining\Tasks\Day2Task\src>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt

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

11. Next, add the changed file, and notice how it moves to the staging area in the status output. Also observe that the diff command you did before using add now gives no output. Why not? What do you have to do to see a diff of the things in the staging area? (Hint: review the slides if you can't remember.)

```
F:\FSDTraining\Tasks\Day2Task\src>git add .

F:\FSDTraining\Tasks\Day2Task\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file1.txt
```

12. Now – without committing – make another change to the same file you changed in step 10. Look at the status output, and the diff output. Notice how you can have both staged and unstaged changes, even when you're talking about a single file. Observe the difference when you use the add command to stage the latest round of changes. Finally, commit them. You should now have started to get a feel for the staging area.

```
F:\FSDTraining\Tasks\Day2Task\src>cd ..

F:\FSDTraining\Tasks\Day2Task>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   src/file1.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/file1.txt


F:\FSDTraining\Tasks\Day2Task>cd src

F:\FSDTraining\Tasks\Day2Task\src>git add .

F:\FSDTraining\Tasks\Day2Task\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file1.txt


F:\FSDTraining\Tasks\Day2Task\src>git commit -m "TASK3"
[master 053379e] TASK3
 1 file changed, 1 insertion(+)
```

13. Use the log command in order to see all of the commits you made so far.

```
F:\FSDTraining\Tasks\Day2Task\src>git log
commit 053379e8da364db854a83049a3156e0c3712913d (HEAD -> master)
Author: Sutharsan <sutharsanmurugan87@gmail.com>
Date:   Fri Jul 12 14:49:34 2024 +0530

    TASK3

commit b2a11767bab9590443c1de5b2d9040cb7bd5aca5
Author: Sutharsan <sutharsanmurugan87@gmail.com>
Date:   Fri Jul 12 14:46:34 2024 +0530

    TASK2

commit c25bbcd5a1d5eb892b0e010c8fec425162746d24
Author: Sutharsan <sutharsanmurugan87@gmail.com>
Date:   Fri Jul 12 14:43:02 2024 +0530

    TASK1
```

14. Use the show command to look at an individual commit. How many characters of the commit identifier can you get away with typing at a minimum?

```
F:\FSDTraining\Tasks\Day2Task\src>git show
commit 053379e8da364db854a83049a3156e0c3712913d (HEAD -> master)
Author: Sutharsan <sutharsanmurugan87@gmail.com>
Date:   Fri Jul 12 14:49:34 2024 +0530

    TASK3

diff --git a/src/file1.txt b/src/file1.txt
index e69de29..6d15dfe 100644
--- a/src/file1.txt
+++ b/src/file1.txt
@@ -0,0 +1 @@
+abcdefghijklmnopqustuvwxyz--12345678910
```

15. Make a couple more commits, at least one of which should add an extra file.

```
F:\FSDTraining\Tasks\Day2Task\src>git add .

F:\FSDTraining\Tasks\Day2Task\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file3.txt


F:\FSDTraining\Tasks\Day2Task\src>git commit -m "TASK4"
[master 97f6869] TASK4
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 src/file3.txt
```

Stretch Task

1.Use the Git rm command to remove a file. Look at the status afterwards. Now commit the deletion.

```
F:\FSDTraining\Tasks\Day2Task\src>git rm file3.txt
rm 'src/file3.txt'

F:\FSDTraining\Tasks\Day2Task\src>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    file3.txt

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


F:\FSDTraining\Tasks\Day2Task\src>git commit -m "TASK5"
[master 3a21276] TASK5
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 src/file3.txt
```

2. Delete another file, but this time do not use Git to do it; e.g. if you are on Linux, just use the normal (non-Git) rm command; on Windows use del.

3. Look at the status. Compare it to the status output you had after using the Git built-in rm command. Is anything different? After this, commit the deletion.

```
F:\FSDTraining\Tasks\Day2Task\src>del file2.txt

F:\FSDTraining\Tasks\Day2Task\src>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:    file2.txt

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

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

F:\FSDTraining\Tasks\Day2Task\src>git commit -m "TASK6"
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:    file2.txt

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

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

4. Use the Git mv command to move or rename a file; for example, rename README to README.txt. Look at the status. Commit the change.

```
F:\FSDTraining\Tasks\Day2Task>git mv text.md text.txt

F:\FSDTraining\Tasks\Day2Task>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:    text.md -> text.txt

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:    src/file2.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        src/New Text Document.txt


F:\FSDTraining\Tasks\Day2Task>git commit -m "TASK7"
[master e5c8d08] TASK7
 1 file changed, 0 insertions(+), 0 deletions(-)
 rename text.md => text.txt (100%)
```

5. Now do another rename, but this time using the operating system's command to do so. How does the status look? Will you get the right outcome if you were to commit at this point? (Answer: almost certainly not, so don't.) Work out how to get the status to show that it will not lose the file, and then commit. Did Git at any point work out that you had done a rename?

```
F:\FSDTraining\Tasks\Day2Task\src>ren file1.txt file.txt

F:\FSDTraining\Tasks\Day2Task\src>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:    file1.txt
        deleted:    file2.txt

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

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

F:\FSDTraining\Tasks\Day2Task\src>git commit -m "TASK8"
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:    file1.txt
        deleted:    file2.txt

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

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

6. Use git help log to find out how to get Git to display just the most recent 3 commits. Try it.

```
F:\FSDTraining\Tasks\Day2Task\src>git help log

F:\FSDTraining\Tasks\Day2Task\src>git log -n 3
commit e5c8d08ce3456adc59f02f10ba266514e6bb73b7 (HEAD -> master)
Author: Sutharsan <sutharsanmurugan87@gmail.com>
Date:   Fri Jul 12 15:39:32 2024 +0530

    TASK7

commit 3a21276ad5c5d728eeec0b5ec5535c6a51318001
Author: Sutharsan <sutharsanmurugan87@gmail.com>
Date:   Fri Jul 12 15:34:42 2024 +0530

    TASK5

commit 97f68694bfd8f1c9bdf5786df70b0970dd85eb1a
Author: Sutharsan <sutharsanmurugan87@gmail.com>
Date:   Fri Jul 12 14:51:29 2024 +0530

    TASK4
```

7. If you don’t remember, look back in the slides to see what the --stat option did on the diff command. Find out if this also works with the show command. How about the log command?

```
F:\FSDTraining\Tasks\Day2Task\src>git diff --stat
 src/file1.txt | 1 -
 src/file2.txt | 0
 2 files changed, 1 deletion(-)
```

8. Imagine you want to see a diff that summarizes all that happened between two commit identifiers. Use the diff command, specifying two commit identifiers joined by two dots (that is, something like abc123..def456). Check the output is what you expect.

```
D:\FULLSTACK70DAYS\GIT\day2\src>git diff 9e9aea0af16d950733d773262ac99217a69
30907..82b57b833b2c2b55bf6e507dc7c6ffea5050b893
diff --git a/READme.txt b/README.md
similarity index 100%
rename from READme.txt
rename to README.md
```