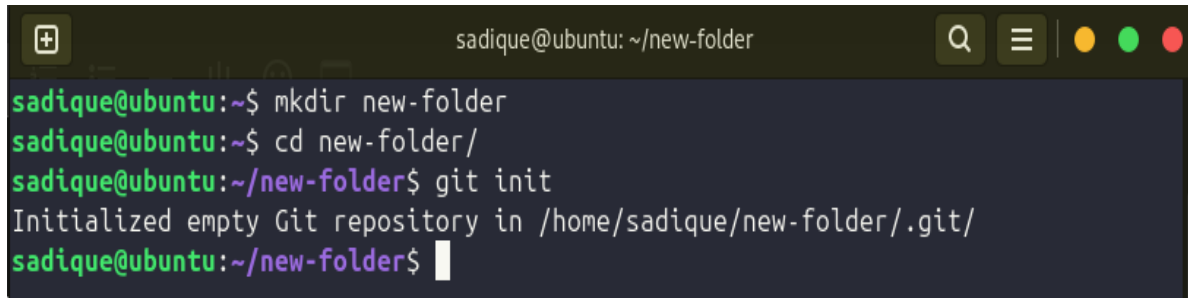


Git – Assignment

1. Tasks to be performed

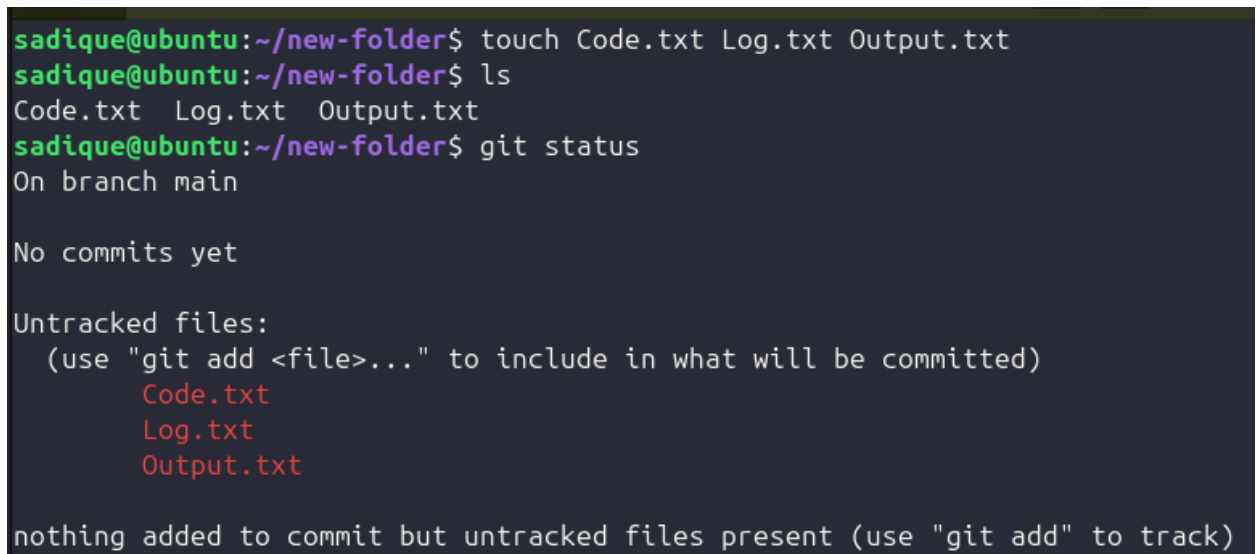
- a. Create a new folder



```
sadique@ubuntu: ~/new-folder
sadique@ubuntu:~$ mkdir new-folder
sadique@ubuntu:~$ cd new-folder/
sadique@ubuntu:~/new-folder$ git init
Initialized empty Git repository in /home/sadique/new-folder/.git/
sadique@ubuntu:~/new-folder$
```

- b. Put the following files in the folder - Code.txt, Log.txt and Output.txt

To create files in a terminal there are many ways but we can use **touch** command which will create empty files within repository. We can use **ls** command to see if the file created successfully or not and also we can use **git status** command to check whether git monitored the changes within the repository (It will not track the files until files are moved to staging area).



```
sadique@ubuntu:~/new-folder$ touch Code.txt Log.txt Output.txt
sadique@ubuntu:~/new-folder$ ls
Code.txt  Log.txt  Output.txt
sadique@ubuntu:~/new-folder$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Code.txt
    Log.txt
    Output.txt

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

- c. Stage the Code.txt and Output.txt files

To move the above mentioned two files in the staging area we will use **git add {file-name}** command. To verify we will again use **git status** command which will tell the following files need to be committed which also means that the files have been moved to staging area and are ready to be part of next version.

```
sadique@ubuntu: ~/new-folder
sadique@ubuntu:~/new-folder$ git add Code.txt Output.txt
sadique@ubuntu:~/new-folder$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Code.txt
        new file:   Output.txt

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

sadique@ubuntu:~/new-folder$
```

d. Commit them

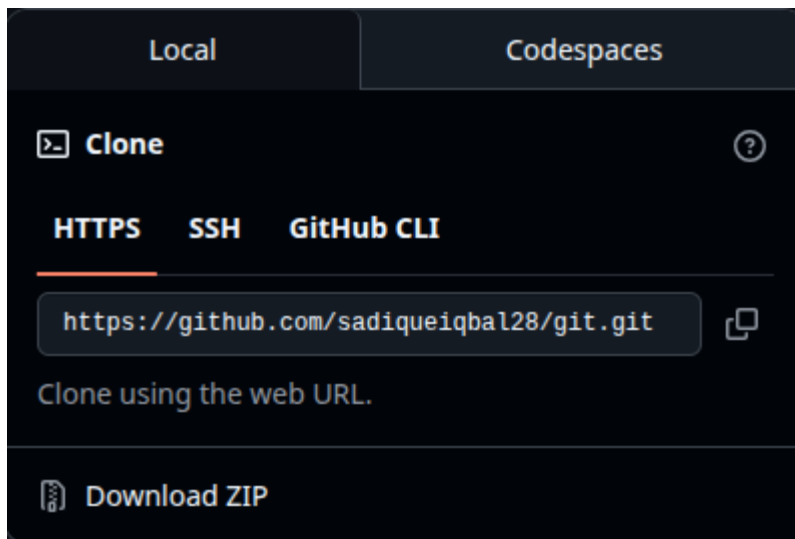
To commit the files from staging area to git repository we will use **git commit -m "some-message"** command. Here **-m** flag is used to pass a commit message with the commit. We can use **git log** command to verify that whether commit was successful or not.

```
sadique@ubuntu: ~/new-folder
sadique@ubuntu:~/new-folder$ git commit -m "initial commit"
[main (root-commit) 1cd152b] initial commit
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Code.txt
create mode 100644 Output.txt
sadique@ubuntu:~/new-folder$ git log
commit 1cd152bc5a235be113d171249bdc7c7dac766df2 (HEAD -> main)
Author: Sadique Iqbal <sadiqueiqbal28@gmail.com>
Date:   Sun Jun 15 13:21:55 2025 +0530

    initial commit
sadique@ubuntu:~/new-folder$
```

e. And finally push them to GitHub

To push the code on GitHub first we need to initialise empty git repository on GitHub then we need to add remote url of GitHub repository in our local repository using the command **git remote add origin https://....**



Here, origin is the name of GitHub repo we are trying to add as there can be multiple remote repository so we can give any name of our like. After adding the remote repo in our local repo we can confirm it by the command **git remote -v**

```
sadique@ubuntu: ~/new-folder
sadique@ubuntu:~/new-folder$ git remote add origin https://github.com/sadiqueiqbal28/git.git
sadique@ubuntu:~/new-folder$ git remote -v
origin https://github.com/sadiqueiqbal28/git.git (fetch)
origin https://github.com/sadiqueiqbal28/git.git (push)
```

Then we need retrieve **Personal Access Token** from GitHub which can be found in GitHub > Settings > Developer Settings > Personal Access Tokens > Fine Grain/Classic Token and set the settings with our need and which can be used in password prompt while pushing the code as Git forbid password based authentication from GitHub

To push the code we can use the **git push origin main** command where origin is name of GitHub repo and main is the default branch or the branch we are currently working on which will be pushed to GitHub. It will prompt for the username and password which can entered respectively and in password field we can Personal Access Token we retrieved.

```
sadique@ubuntu: ~/new-folder
sadique@ubuntu:~/new-folder$ git push origin main
Username for 'https://github.com': sadiqueiqbal28
Password for 'https://sadiqueiqbal28@github.com':
Everything up-to-date
```

2. Tasks to be performed

- 1) Create a Git working directory with feature1.txt and feature2.txt in the master branch

```
sadique@ubuntu: ~/new-project
sadique@ubuntu:~$ mkdir new-project
sadique@ubuntu:~$ cd new-project/
sadique@ubuntu:~/new-project$ git init
Initialized empty Git repository in /home/sadique/new-project/.git/
sadique@ubuntu:~/new-project$ touch feature1.txt feauture2.txt
sadique@ubuntu:~/new-project$ git add .
sadique@ubuntu:~/new-project$ git commit -m "committing feature files"
[main (root-commit) 9a5f35b] committing feature files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 feature1.txt
 create mode 100644 feauture2.txt
sadique@ubuntu:~/new-project$
```

- 2) Create 3 branches develop, feature1 and feature2

```
sadique@ubuntu: ~/new-project
sadique@ubuntu:~/new-project$ git branch develop
sadique@ubuntu:~/new-project$ git branch feature1
sadique@ubuntu:~/new-project$ git branch feature2
sadique@ubuntu:~/new-project$ git branch
  develop
  feature1
  feature2
* main
sadique@ubuntu:~/new-project$
```

- 3) In develop branch create develop.txt, do not stage or commit it

```
sadique@ubuntu:~/new-project$ git checkout develop
Switched to branch 'develop'
sadique@ubuntu:~/new-project$ touch develop.txt
sadique@ubuntu:~/new-project$
```

- 4) Stash this file and checkout to feature1 branch

```
sadique@ubuntu:~/new-project$ git stash --include-untracked
Saved working directory and index state WIP on develop: 9a5f35b committing feature file
sadique@ubuntu:~/new-project$ git checkout feature1
Switched to branch 'feature1'
sadique@ubuntu:~/new-project$
```

- 5) Create new.txt file in feature1 branch, stage and commit this file

```
Switched to branch 'feature1'
sadique@ubuntu:~/new-project$ touch new.txt
sadique@ubuntu:~/new-project$ git add new.txt
sadique@ubuntu:~/new-project$ git commit -m "commit new.txt file"
[feature1 4d98995] commit new.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 new.txt
sadique@ubuntu:~/new-project$ git status
On branch feature1
nothing to commit, working tree clean
sadique@ubuntu:~/new-project$ git log
commit 4d98995d5dcdb3de0ea9b8eb02b062b99ca750af (HEAD -> feature1)
Author: Sadique Iqbal <sadiqueiqbal28@gmail.com>
Date:   Sun Jun 15 15:15:19 2025 +0530

    commit new.txt file
```

- 6) Checkout to develop, unstash this file and commit

```
sadique@ubuntu:~/new-project$ git stash apply
Already up to date.
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        develop.txt

nothing added to commit but untracked files present (use "git add" to track)
sadique@ubuntu:~/new-project$ git add .
sadique@ubuntu:~/new-project$ git commit "committing develop.txt"
> ^C
sadique@ubuntu:~/new-project$ git commit -m "committing develop.txt"
>
> /
> ^C
sadique@ubuntu:~/new-project$ git commit -m "committing develop.txt"
> ^C
sadique@ubuntu:~/new-project$ git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   develop.txt

sadique@ubuntu:~/new-project$ git commit -m "develop commit"
[develop 30418a2] develop commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 develop.txt
sadique@ubuntu:~/new-project$ git log
commit 30418a24ade8bc98a6f8ea3905b15578216916e7 (HEAD -> develop)
Author: Sadique Iqbal <sadiqueiqbal28@gmail.com>
Date:   Sun Jun 15 15:20:50 2025 +0530

    develop commit
```

7) Please submit all the Git commands used to do the above step

- **mkdir new-project && cd new-project** - This will create a directory named new-project and change it to new-project
- **git init** - This will initialise git repository in our project folder for the tracking, versioning etc.
- **touch feauture1.txt feature2.txt** - This will create 2 empty files named feature1.txt feature2.txt in our default main branch.
- **git branch develop** - This will create a new branch with the name develop
- **git branch feature1** - This will create the new branch with the name feature1.
- **git branch feature2** - This will create the new branch with the name feature2.
- **git checkout develop** - This will help us to switch to develop branch from our default branch which is main.
- **touch develop.txt** - This will create an empty file develop.txt within develop branch.
- **git status** - This will help us to see on which branch we are and whether git sensed new file created.
- **git stash --include-untracked** - This will help us to stash our file and we are using **--include-untracked** flag as by default git stash only the files which are already in staging area with this flag it will stash untracked file as well.
- **git checkout feature1** - This will help us to switch our branch to feauture1.
- **touch new.txt** - This will create an empty file touch.txt in our current branch which is feature1
- **git add new.txt** - This will add the file in staging area to be tracked.
- **git commit -m "..."** - This will help us to commit our file.
- **git checkout develop** - This will switch to develop branch
- **git stash apply** - This command will bring back our file from stashed area back to working directory as it was not staged. In our case it was just one.
- **git add develop.txt** - This will move the file from working dir to staging area.
- **git commit -m "...."** - This will move the file from staging area to git repository (local).

3. Tasks to be performed

1. Create a git working directory with the following branches - Develop, f1, f2

```
sadique@ubuntu:~$ mkdir task-3
sadique@ubuntu:~$ cd task-3/
sadique@ubuntu:~/task-3$ git init
Initialized empty Git repository in /home/sadique/task-3/.git/
sadique@ubuntu:~/task-3$ git remote add origin https://github.com/sadiqueiqbal28/task-3.git
sadique@ubuntu:~/task-3$ git checkout -b develop
Switched to a new branch 'develop'
sadique@ubuntu:~/task-3$ git checkout -b f1
Switched to a new branch 'f1'
sadique@ubuntu:~/task-3$ git checkout -b f2
Switched to a new branch 'f2'
sadique@ubuntu:~/task-3$
```

2. In the master branch, commit main.txt file

```
sadique@ubuntu:~/task-3$ git log
commit 5c2dea6f4c2844f28b59272262e9a0dc8059e03d (HEAD -> main)
Author: Sadique Iqbal <sadiqueiqbal28@gmail.com>
Date: Sun Jun 15 16:35:54 2025 +0530

    committed main.txt
```

3. Put develop.txt in develop branch, f1.txt and f2.txt in f1 and f2 respectively

```
sadique@ubuntu:~/task-3$ git status
On branch develop
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    develop.txt

nothing added to commit but untracked files present (use "git add" to track)
sadique@ubuntu:~/task-3$ git add .
sadique@ubuntu:~/task-3$ git commit -m "Committed develop.txt"
[develop f80ca45] Committed develop.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 develop.txt
sadique@ubuntu:~/task-3$ git log
commit f80ca45b567b51eeb7ffcf6cb5e649b888455ac7 (HEAD -> develop)
Author: Sadique Iqbal <sadiqueiqbal28@gmail.com>
Date: Sun Jun 15 16:39:37 2025 +0530

    Committed develop.txt
```





4. Push all these branches to GitHub

```
sadique@ubuntu:~/task-3$ git push --all
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 443 bytes | 443.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/sadiqueiqbal28/task-3.git
 * [new branch]      develop -> develop
 * [new branch]      main -> main
sadique@ubuntu:~/task-3$
```

5. On local delete f2 branch

```
sadique@ubuntu:~/task-3$ git status
On branch main
nothing to commit, working tree clean
sadique@ubuntu:~/task-3$ git branch -D f2
Deleted branch f2 (was f80ca45).
sadique@ubuntu:~/task-3$
```

6. Delete the same branch on GitHub as well

| Your branches | | | | |
|---------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|--------------|--------|-------|
| Branch | Updated | Check status | Behind | Ahead |
| f2  |  4 minutes ago | | 0 | 1 |
| develop  |  7 minutes ago | | 0 | 1 |

4. Tasks to be performed

1. Put master.txt on master branch, stage and commit

```
sadique@ubuntu:~$ mkdir task-4
sadique@ubuntu:~$ cd task-4/
sadique@ubuntu:~/task-4$ git init
Initialized empty Git repository in /home/sadique/task-4/.git/
sadique@ubuntu:~/task-4$ touch master.txt
sadique@ubuntu:~/task-4$ git add master.txt
sadique@ubuntu:~/task-4$ git commit -m "committedmaster.txt"
[main (root-commit) 9472725] committedmaster.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 master.txt
sadique@ubuntu:~/task-4$ git log
commit 9472725c36a921b4f051a36100250224be09c314 (HEAD -> main)
Author: Sadique Iqbal <sadiqueiqbal28@gmail.com>
Date:   Sun Jun 15 16:58:54 2025 +0530

    committedmaster.txt
sadique@ubuntu:~/task-4$
```

2. Create 3 branches: public 1, public 2 and private

```
sadique@ubuntu:~/task-4$ git switch -c public1
Switched to a new branch 'public1'
sadique@ubuntu:~/task-4$ git switch -c public2
Switched to a new branch 'public2'
sadique@ubuntu:~/task-4$ git switch -c private
Switched to a new branch 'private'
sadique@ubuntu:~/task-4$ git branch --list
  main
* private
  public1
  public2
sadique@ubuntu:~/task-4$
```

3. Put public1.txt on public 1 branch, stage and commit

```
sadique@ubuntu:~/task-4$ git checkout public1
Switched to branch 'public1'
sadique@ubuntu:~/task-4$ touch public1.txt
sadique@ubuntu:~/task-4$ git add public1.txt
sadique@ubuntu:~/task-4$ git commit -m "committed public.txt"
[public1 e773b7c] committed public.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 public1.txt
sadique@ubuntu:~/task-4$
```

4. Merge public 1 on master branch

```
sadique@ubuntu:~/task-4$ git switch main
Switched to branch 'main'
sadique@ubuntu:~/task-4$ git merge public1
Updating 9472725..e773b7c
Fast-forward
 public1.txt | 0
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 public1.txt
```

5. Merge public 2 on master branch

```
sadique@ubuntu:~/task-4$ git merge public2
Already up to date.
sadique@ubuntu:~/task-4$
```

6. Edit master.txt on private branch, stage and commit

```
sadique@ubuntu:~/task-4$ git checkout private
Switched to branch 'private'
sadique@ubuntu:~/task-4$ ls
master.txt
sadique@ubuntu:~/task-4$ echo "Hello World" > master.txt
sadique@ubuntu:~/task-4$ git status
On branch private
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:   master.txt

no changes added to commit (use "git add" and/or "git commit -a")
sadique@ubuntu:~/task-4$ git add .
sadique@ubuntu:~/task-4$ git commit -m "master.txt on priv branch"
[private 307c3e9] master.txt on priv branch
 1 file changed, 1 insertion(+)
sadique@ubuntu:~/task-4$
```

7. Now update branch public 1 and public 2 with new master code in private

```
sadique@ubuntu:~/task-4$ git status
On branch public1
nothing to commit, working tree clean
sadique@ubuntu:~/task-4$ echo "Hello 1" > master.txt
sadique@ubuntu:~/task-4$ git add .
sadique@ubuntu:~/task-4$ git commit -m "modified on public1 master1.txt"
[public1 7919aab] modified on public1 master1.txt
 1 file changed, 1 insertion(+)
sadique@ubuntu:~/task-4$ git switch public2
Switched to branch 'public2'
sadique@ubuntu:~/task-4$ echo "Hello 2" > master.txt
sadique@ubuntu:~/task-4$ git add master.txt
sadique@ubuntu:~/task-4$ git commit -m "modified on public2 master.txt"
[public2 a450db5] modified on public2 master.txt
 1 file changed, 1 insertion(+)
sadique@ubuntu:~/task-4$
```

8. Also update new master code on master

```
sadique@ubuntu:~/task-4$ git switch main
Switched to branch 'main'
sadique@ubuntu:~/task-4$ git status
On branch main
nothing to commit, working tree clean
sadique@ubuntu:~/task-4$ echo "Hello-master" > master.txt
sadique@ubuntu:~/task-4$ git add .
sadique@ubuntu:~/task-4$ git commit -m "new master code on master"
[main 142dc02] new master code on master
1 file changed, 1 insertion(+)
sadique@ubuntu:~/task-4$
```

9. Finally update all the code on private branch

```
sadique@ubuntu:~/task-4$ git switch private
Switched to branch 'private'
sadique@ubuntu:~/task-4$ git status
On branch private
nothing to commit, working tree clean
sadique@ubuntu:~/task-4$ ls
master.txt
sadique@ubuntu:~/task-4$ echo "private-master" > master.txt
sadique@ubuntu:~/task-4$ git add master.txt
sadique@ubuntu:~/task-4$ git commit -m "committed master on private"
[private 5cc368b] committed master on private
1 file changed, 1 insertion(+), 1 deletion(-)
sadique@ubuntu:~/task-4$
```

5. Tasks to be performed

1. Create a Git Flow workflow architecture on Git

In Git Flow workflow there are many branches: master : This branch contains code which are going to be part of next version. develop : It is a different branch where features branch will be integrated. features : It is the branch where new features or new updates will be added. release : This branch will contain the part which will be deployed to production. hotfix : This branch is for directly fixing bugs and code on production server.

```
sadique@ubuntu:~$ mkdir task-5
sadique@ubuntu:~$ cd task-5/
sadique@ubuntu:~/task-5$ git init
Initialized empty Git repository in /home/sadique/task-5/.git/
sadique@ubuntu:~/task-5$ echo "Task 5 Assignment" > README.md
sadique@ubuntu:~/task-5$ git add README.md
sadique@ubuntu:~/task-5$ git commit -m "Initial commit on main branch"
[main (root-commit) dcf2f78] Initial commit on main branch
1 file changed, 1 insertion(+)
create mode 100644 README.md
sadique@ubuntu:~/task-5$ git checkout -b develop
Switched to a new branch 'develop'
sadique@ubuntu:~/task-5$
```

2. Create all the required branches

```
sadique@ubuntu:~/task-5$ git checkout -b feature/my-feature develop
Switched to a new branch 'feature/my-feature'
sadique@ubuntu:~/task-5$ echo "This is a new feature" > feature.txt
sadique@ubuntu:~/task-5$ git add feature.txt
sadique@ubuntu:~/task-5$ git commit -m "Add feature.txt"
[feature/my-feature b79b146] Add feature.txt
1 file changed, 1 insertion(+)
create mode 100644 feature.txt
sadique@ubuntu:~/task-5$
```

3. starting from the feature branch, push the branch to the master, following the architecture

```
sadique@ubuntu:~/task-5$ git checkout develop
Switched to branch 'develop'
sadique@ubuntu:~/task-5$ git merge feature/my-feature
Updating dcf2f78..b79b146
Fast-forward
 feature.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt
sadique@ubuntu:~/task-5$ git checkout -b release/1.0
Switched to a new branch 'release/1.0'
sadique@ubuntu:~/task-5$ git merge develop
Already up to date.
sadique@ubuntu:~/task-5$ git log --all --decorate --oneline --graph
* b79b146 (HEAD -> release/1.0, feature/my-feature, develop) Add feature
* dcf2f78 (main) Initial commit on main branch
sadique@ubuntu:~/task-5$ git checkout master
error: pathspec 'master' did not match any file(s) known to git
sadique@ubuntu:~/task-5$ git checkout main
Switched to branch 'main'
sadique@ubuntu:~/task-5$ git merge release/1.0
Updating dcf2f78..b79b146
Fast-forward
 feature.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 feature.txt
sadique@ubuntu:~/task-5$ git tag -a v1.0 -m "Release version 1.0"
sadique@ubuntu:~/task-5$ git checkout develop
Switched to branch 'develop'
sadique@ubuntu:~/task-5$ git merge release/1.0
Already up to date.
sadique@ubuntu:~/task-5$
```

4. Push an urgent.txt on master using hotfix

```
sadique@ubuntu:~/task-5$ git checkout hotfix/urgent-fix
error: pathspec 'hotfix/urgent-fix' did not match any file(s) known to git
sadique@ubuntu:~/task-5$ git checkout -b hotfix/urgent-fix
Switched to a new branch 'hotfix/urgent-fix'
sadique@ubuntu:~/task-5$ echo "Urgent fix" > urgent.txt
sadique@ubuntu:~/task-5$ git add urgent.txt
sadique@ubuntu:~/task-5$ git commit -m "Added urgent.txt as a hotfix"
[hotfix/urgent-fix eba839b] Added urgent.txt as a hotfix
1 file changed, 1 insertion(+)
create mode 100644 urgent.txt
sadique@ubuntu:~/task-5$ git remote add origin https://github.com/sadiqueiqbal28/task-5
sadique@ubuntu:~/task-5$ git push origin hotfix/urgent-fix
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 755 bytes | 755.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/sadiqueiqbal28/task-5.git
 * [new branch]      hotfix/urgent-fix -> hotfix/urgent-fix
sadique@ubuntu:~/task-5$ git checkout develop
Switched to branch 'develop'
sadique@ubuntu:~/task-5$ git merge hotfix/urgent-fix
Updating b79b146..eba839b
Fast-forward
 urgent.txt | 1 +
1 file changed, 1 insertion(+)
create mode 100644 urgent.txt
sadique@ubuntu:~/task-5$ git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/sadiqueiqbal28/task-5/pull/new/develop
remote:
To https://github.com/sadiqueiqbal28/task-5.git
 * [new branch]      develop -> develop
sadique@ubuntu:~/task-5$
```