Experiment 04 - Advanced Git Commands

Roll No.	70
Name	MAYURI SHRIDATTA YERANDE
Class	D15-B
Subject	DevOps Lab
LO Mapped	LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements LO2: To obtain complete knowledge of the “version control system” to effectively track changes augmented with Git and GitHub

Aim: To implement Feature Branch workflow strategies in real-time scenarios

Introduction:

git branch: The git branch command is actually something of a branch management tool. It can list the branches you have, create a new branch, delete branches and rename branches.

git checkout: The git checkout command is used to switch branches and check content out into your working directory.

git merge: The git merge tool is used to merge one or more branches into the branch you have checked out. It will then advance the current branch to the result of the merge.

git mergetool: The git mergetool command simply launches an external merge helper in case you have issues with a merge in Git.

git log: The git log command is used to show the reachable recorded history of a project from the most recent commit snapshot backwards. By default it will only show the history of the branch you're currently on, but can be given different or even multiple heads or branches from which to traverse. It is also often used to show differences between two or more branches at the commit level.

git stash: The git stash command is used to temporarily store uncommitted work in order to clean out your working directory without having to commit unfinished work on a branch.

git tag: The git tag command is used to give a permanent bookmark to a specific point in the code history. Generally this is used for things like releases.

Workflow:-

A Workflow is defined as a sequence of tasks that processes data through a specific path from initiation to completion. They can be used to structure any kind of business function regardless of industry. Essentially, anytime data is passed between humans and/or systems, a workflow is created.
The purpose of the workflow is to specify paths that describe how something goes from being undone to done, or raw to processed

1. Project workflow

A project workflow is ideal for keeping complex projects on track. So, in a project workflow, a manager might make a list of all project deliverables. Then, a series of activities could be diagrammed out, illustrating which tasks need to occur and in what order, to create each project deliverable. For projects that have a lot of moving parts to them, a project workflow can be invaluable. For one thing, once a project workflow is represented visually, it's very easy for project stakeholders to get a bird's eye view of what needs to happen for the project to run smoothly, as well as identify any potential project bottlenecks.

Example: There's an IT project to create new software. One of the deliverables for such a project would be a document detailing the software's specifications. The process of gathering those specifications probably wouldn't change much from project to project—allowing for that section of the project workflow to be reused.

2. Case workflow

When envisioning a case workflow, it's useful to think in terms of a problem that requires a solution—like an incoming IT help desk ticket. Unlike other types of workflows, a case workflow doesn't occur in a sequential, orderly fashion. Instead, two help desk tickets might go through completely different workflows, depending on the initial problem. So, a case workflow is unique because the actual progression of steps isn't known at the onset of the problem. Those steps can change along the way, or you might even reach one step before realizing you need to return to an earlier one.

Example: Let's look at a hypothetical example of two employees who can't access the Internet. One employee might need their laptop settings reconfigured. By contrast, the other employee might have their laptop configured correctly already, and even so, still can't access the Internet. Now, perhaps the data security department will need to get involved, since the problem appears to be unrelated to the laptop's settings. At any rate, case workflows are used for items where the correct path isn't known from the very beginning and is instead, determined along the way.

3. Process workflow

Of the three types of workflows mentioned in this article, process workflows are what most people tend to be familiar with. Process workflows are used to depict repetitive, predictable tasks. So, for instance, a process workflow might illustrate how vendor invoices get paid, how website content gets created, or how vacation time gets approved. Within a process workflow diagram, the following things are clearly delineated:

- Which tasks need to be performed and when
- Which departments are responsible for handling those tasks

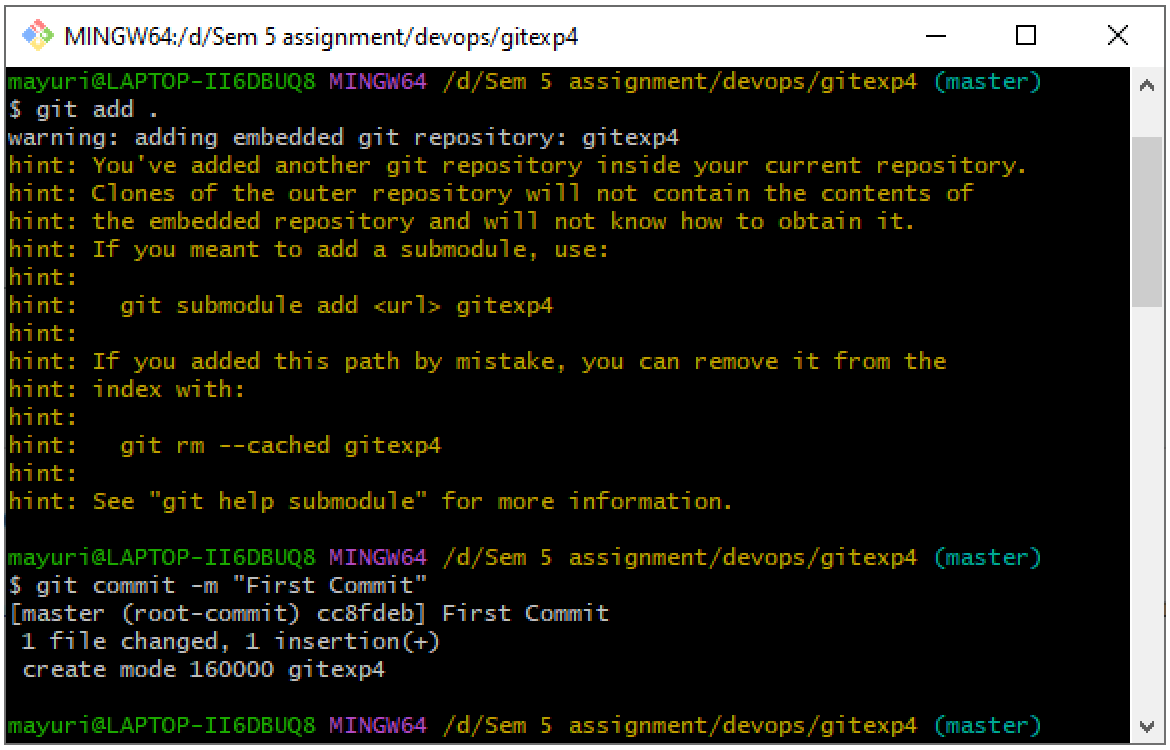
A process workflow also spells out what should occur if there's a problem within the process.
Example: that you have a process workflow illustrating how vendors get paid. The workflow diagram should include exceptions within the process, such as what occurs if a vendor's invoice doesn't get approved.

4. Branch Workflow

Git Feature Branch Workflow is branching model focused, meaning that it is a guiding framework for managing and creating branches. Other workflows are more repo focused. The Git Feature Branch Workflow can be incorporated into other workflows

Feature Branch Workflow

1. Create an empty repository and commit the changes

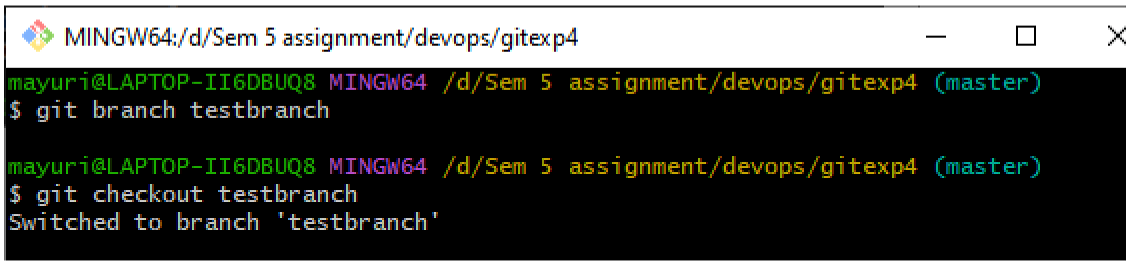


MINGW64:/d/Sem 5 assignment/devops/gitexp4

```
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git add .
warning: adding embedded git repository: gitexp4
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> gitexp4
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached gitexp4
hint:
hint: See "git help submodule" for more information.

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git commit -m "First Commit"
[master (root-commit) cc8fdeb] First Commit
 1 file changed, 1 insertion(+)
 create mode 160000 gitexp4

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
```

2. Create a branch using the command 'git branch testbranch'. You can check all existing branches by the command 'git branch'. The green dot indicates the branch you currently are working on.

3. To switch to other branch use the command 'git checkout'

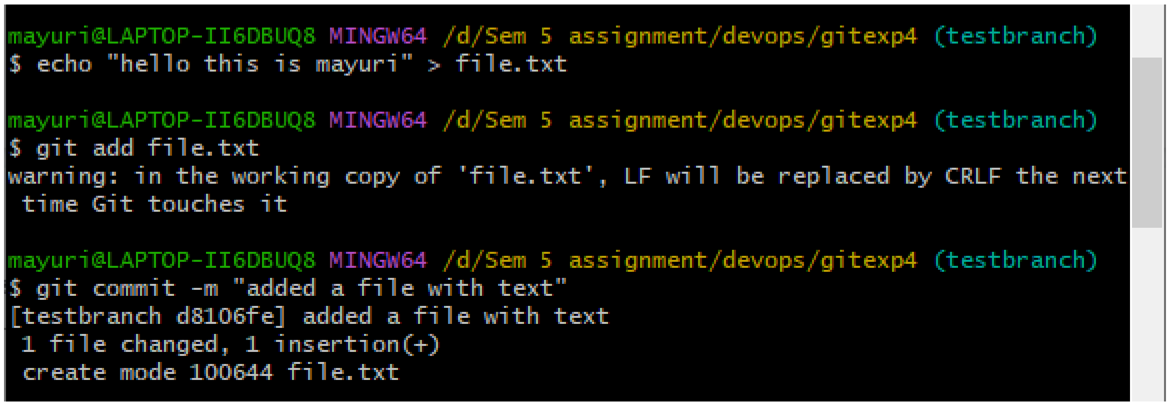


MINGW64:/d/Sem 5 assignment/devops/gitexp4

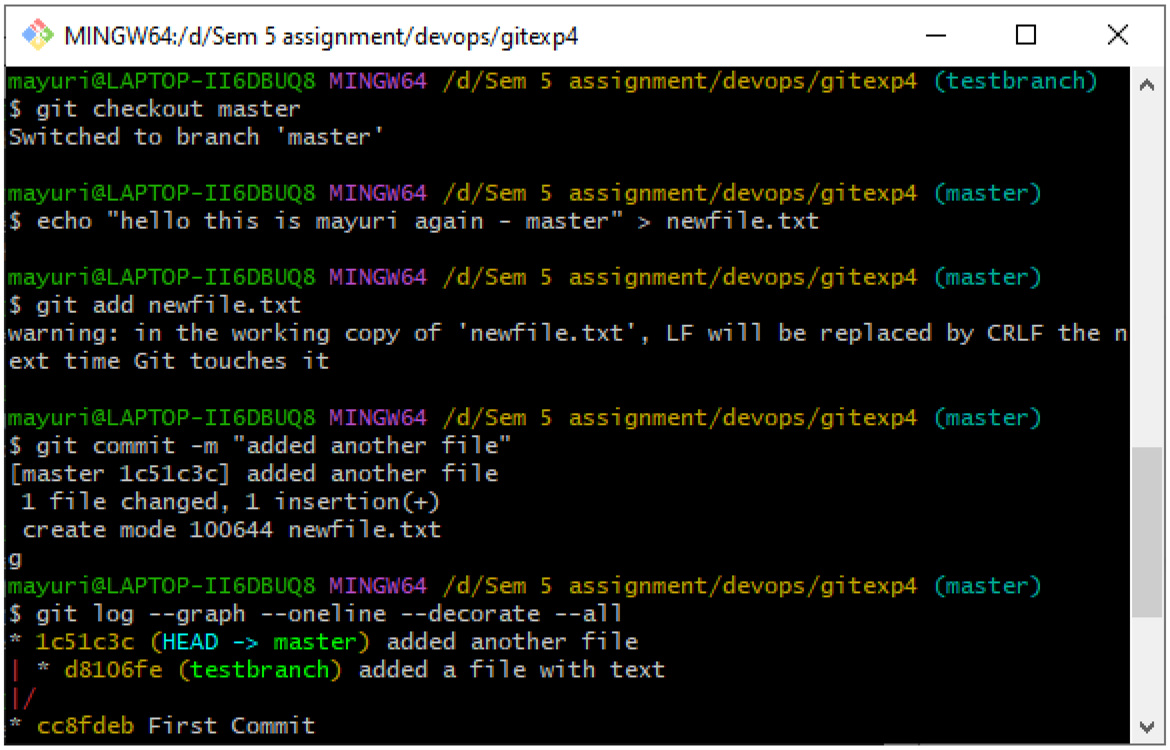
```
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git branch testbranch

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git checkout testbranch
Switched to branch 'testbranch'
```

4. Make some changes to any file while in b1 and commit those changes.

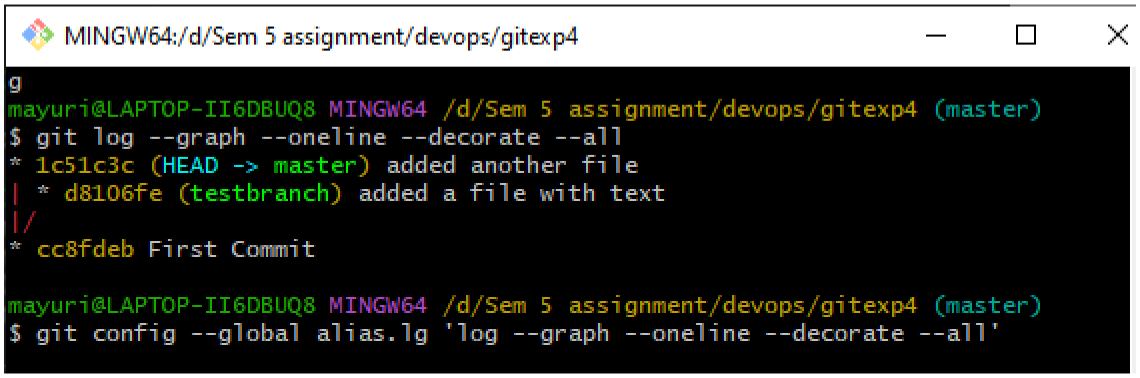
(We added a file in "test branch" and added another file in "master")



```
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (testbranch)
$ echo "hello this is mayuri" > file.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (testbranch)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next
 time Git touches it

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (testbranch)
$ git commit -m "added a file with text"
[testbranch d8106fe] added a file with text
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt
```



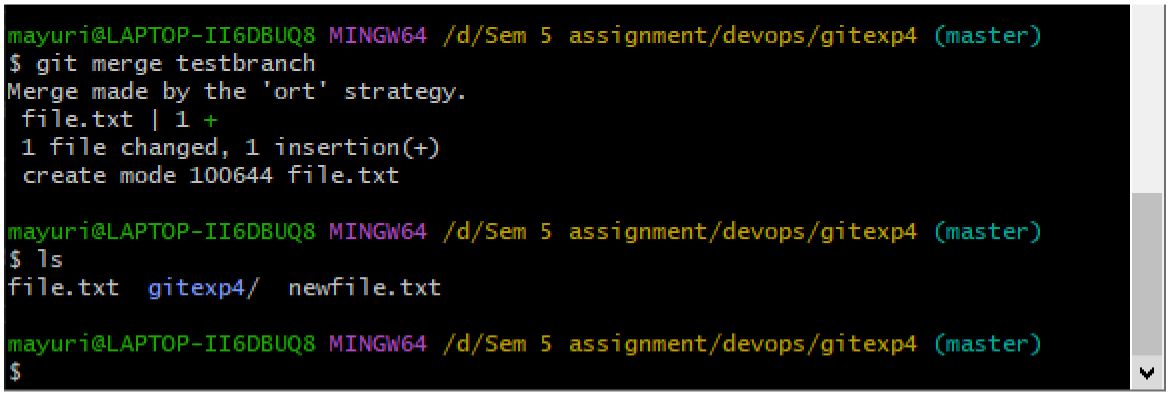
```
MINGW64:/d/Sem 5 assignment/devops/gitexp4
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (testbranch)
$ git checkout master
Switched to branch 'master'

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ echo "hello this is mayuri again - master" > newfile.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git add newfile.txt
warning: in the working copy of 'newfile.txt', LF will be replaced by CRLF the n
ext time Git touches it

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git commit -m "added another file"
[master 1c51c3c] added another file
 1 file changed, 1 insertion(+)
 create mode 100644 newfile.txt
g
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git log --graph --oneline --decorate --all
* 1c51c3c (HEAD -> master) added another file
| * d8106fe (testbranch) added a file with text
|/
* cc8fdeb First Commit
```



MINGW64:/d/Sem 5 assignment/devops/gitexp4

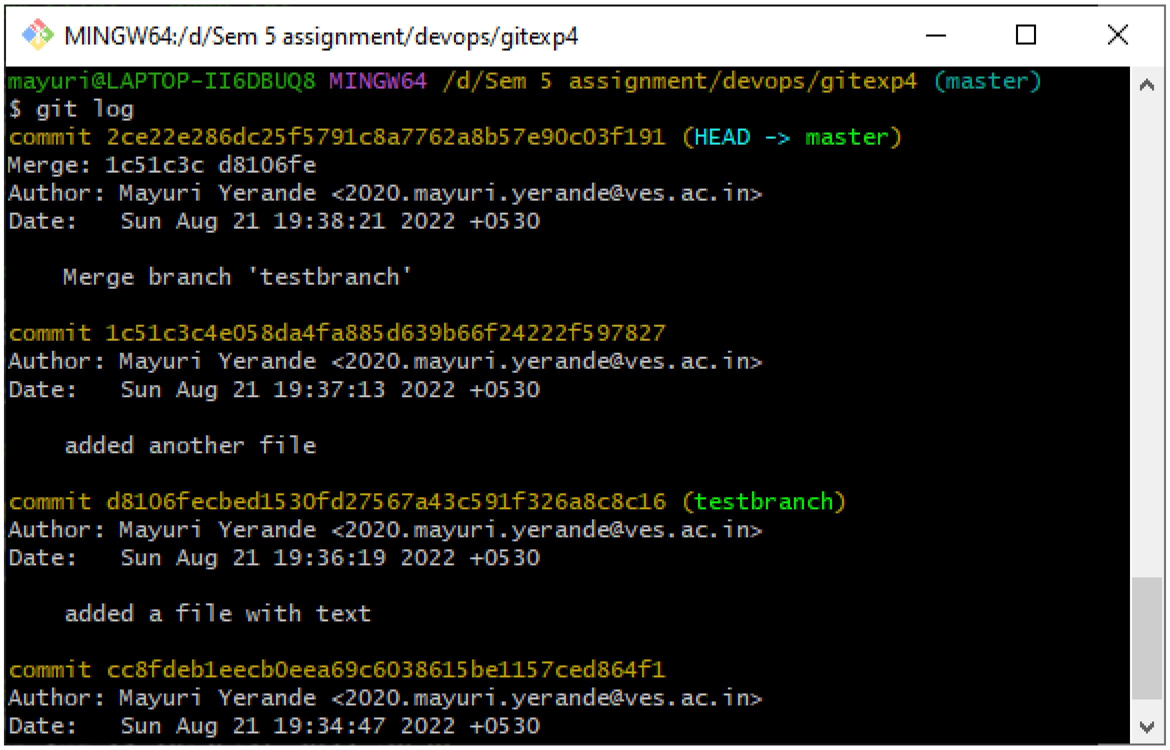
```
g
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git log --graph --oneline --decorate --all
* 1c51c3c (HEAD -> master) added another file
| * d8106fe (testbranch) added a file with text
|/
* cc8fdeb First Commit

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git config --global alias.lg 'log --graph --oneline --decorate --all'
```

5. We added the “file.txt” into testbranch and “newfile.txt” in master branch. After merging testbranch using the “git merge testbranch” command. Now use “ls” commad to see the files,we can see the files of testbranch in master branch. Thus the branches are merged.



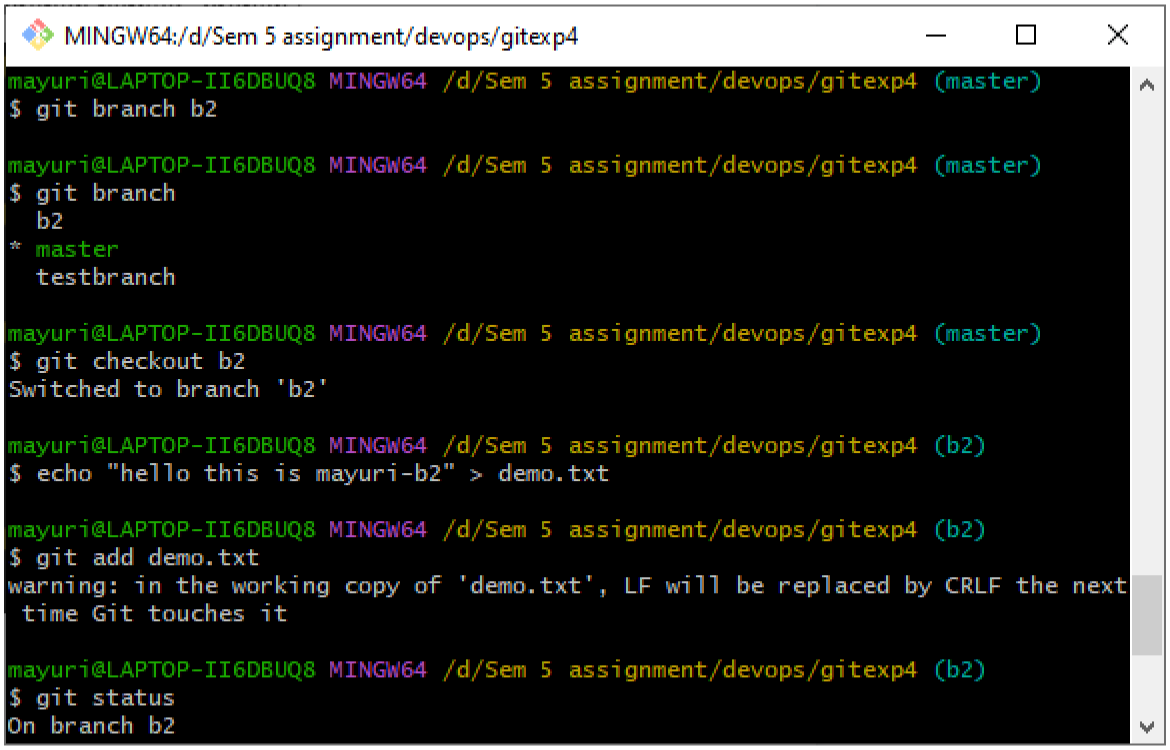
```
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git merge testbranch
Merge made by the 'ort' strategy.
 file.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 file.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ ls
file.txt  gitexp4/  newfile.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$
```

6. Use git log command to check the changes made



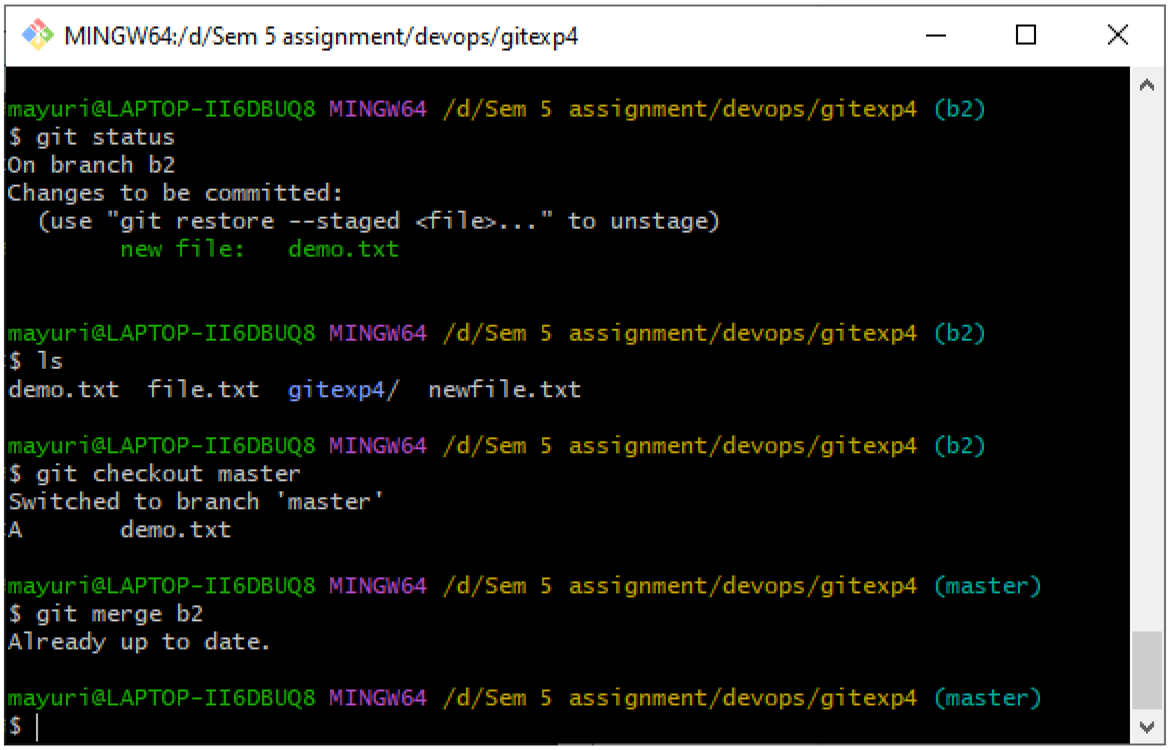
```
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git log
commit 2ce22e286dc25f5791c8a7762a8b57e90c03f191 (HEAD -> master)
Merge: 1c51c3c d8106fe
Author: Mayuri Yerande <2020.mayuri.yerande@ves.ac.in>
Date:   Sun Aug 21 19:38:21 2022 +0530

    Merge branch 'testbranch'

commit 1c51c3c4e058da4fa885d639b66f24222f597827
Author: Mayuri Yerande <2020.mayuri.yerande@ves.ac.in>
Date:   Sun Aug 21 19:37:13 2022 +0530

    added another file

commit d8106fecbed1530fd27567a43c591f326a8c8c16 (testbranch)
Author: Mayuri Yerande <2020.mayuri.yerande@ves.ac.in>
Date:   Sun Aug 21 19:36:19 2022 +0530

    added a file with text

commit cc8fdeb1eecb0eea69c6038615be1157ced864f1
Author: Mayuri Yerande <2020.mayuri.yerande@ves.ac.in>
Date:   Sun Aug 21 19:34:47 2022 +0530
```

7. Checkout to master branch and create another branch named ‘b2’.



```
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git branch b2

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git branch
  b2
* master
  testbranch

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git checkout b2
Switched to branch 'b2'

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (b2)
$ echo "hello this is mayuri-b2" > demo.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (b2)
$ git add demo.txt
warning: in the working copy of 'demo.txt', LF will be replaced by CRLF the next
 time Git touches it

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (b2)
$ git status
On branch b2
```

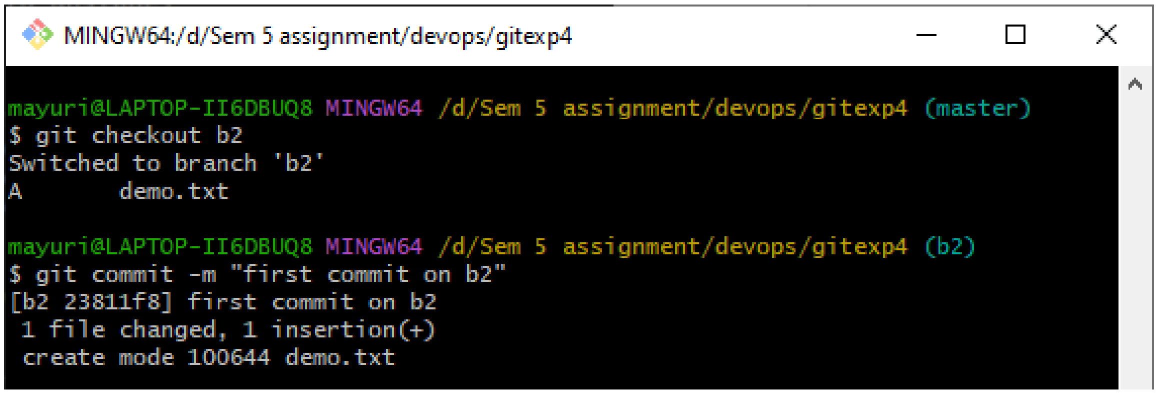
8. If you do a ‘ls’ while in master branch, you won't see a file ‘demo.txt’ as it was created in b2. When you merge the b2 to master using command ‘git merge b2’ and then do a ‘ls’ you’ll see that all the files of branch1 have been added/merged with master branch.



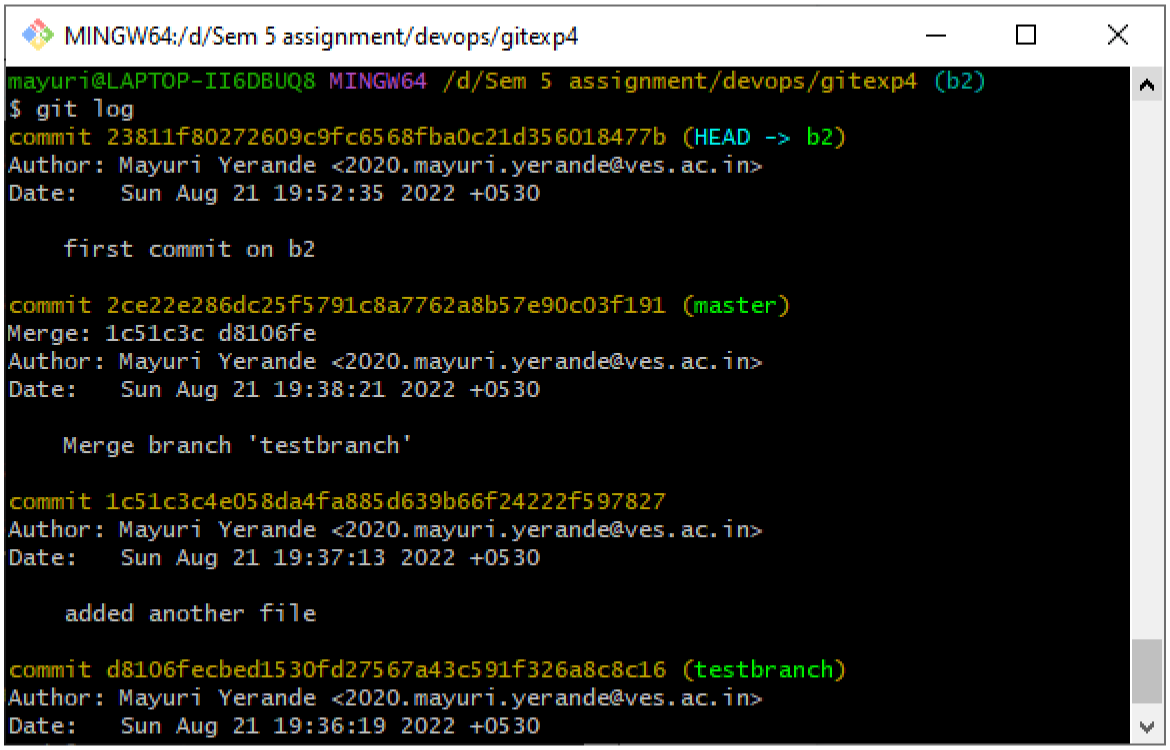
MINGW64:/d/Sem 5 assignment/devops/gitexp4

```
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (b2)
$ git status
On branch b2
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   demo.txt


mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (b2)
$ ls
demo.txt  file.txt  gitexp4/  newfile.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (b2)
$ git checkout master
Switched to branch 'master'
A       demo.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git merge b2
Already up to date.

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ |
```

9. Now checkout to b2, add a file, commit the changes2

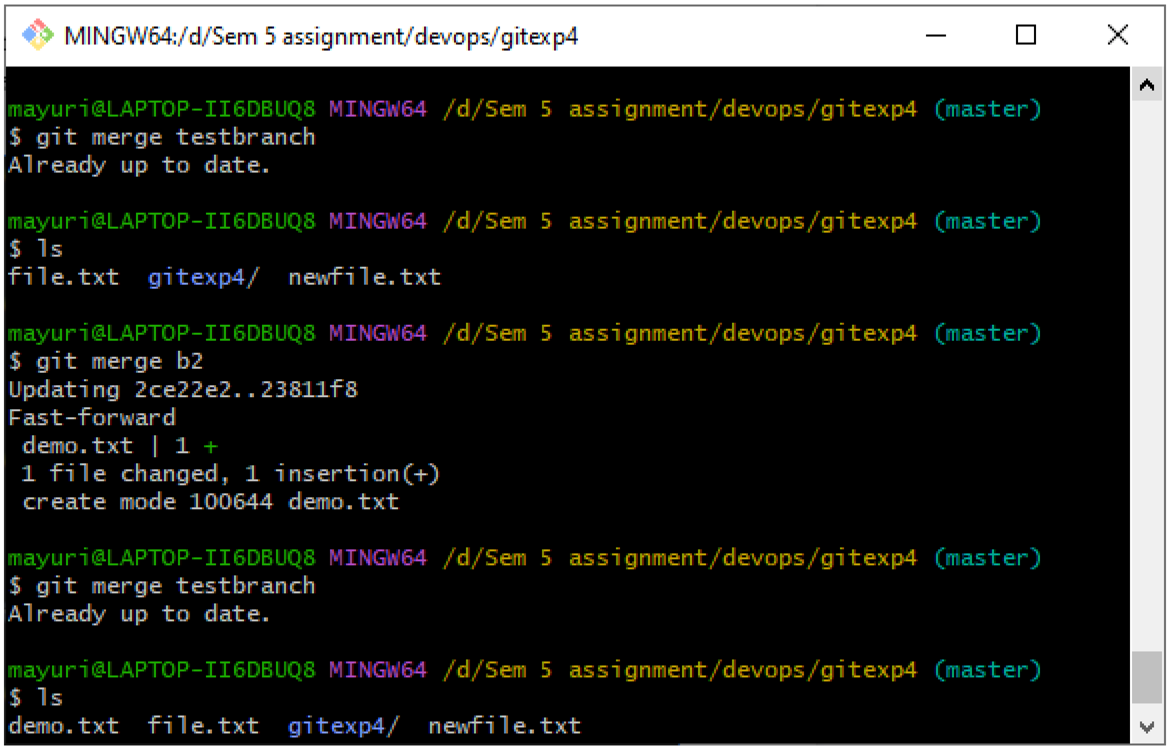


```
MINGW64:/d/Sem 5 assignment/devops/gitexp4

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git checkout b2
Switched to branch 'b2'
A       demo.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (b2)
$ git commit -m "first commit on b2"
[b2 23811f8] first commit on b2
 1 file changed, 1 insertion(+)
 create mode 100644 demo.txt
```



```
MINGW64:/d/Sem 5 assignment/devops/gitexp4
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (b2)
$ git log
commit 23811f80272609c9fc6568fba0c21d356018477b (HEAD -> b2)
Author: Mayuri Yerande <2020.mayuri.yerande@ves.ac.in>
Date:   Sun Aug 21 19:52:35 2022 +0530

    first commit on b2

commit 2ce22e286dc25f5791c8a7762a8b57e90c03f191 (master)
Merge: 1c51c3c d8106fe
Author: Mayuri Yerande <2020.mayuri.yerande@ves.ac.in>
Date:   Sun Aug 21 19:38:21 2022 +0530

    Merge branch 'testbranch'

commit 1c51c3c4e058da4fa885d639b66f24222f597827
Author: Mayuri Yerande <2020.mayuri.yerande@ves.ac.in>
Date:   Sun Aug 21 19:37:13 2022 +0530

    added another file

commit d8106fecbed1530fd27567a43c591f326a8c8c16 (testbranch)
Author: Mayuri Yerande <2020.mayuri.yerande@ves.ac.in>
Date:   Sun Aug 21 19:36:19 2022 +0530
```

10. Perform the same thing as done for branch1. You'll see a new file added to the master branch after the branch2 has been merged with master.



MINGW64:/d/Sem 5 assignment/devops/gitexp4

```
mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git merge testbranch
Already up to date.

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ ls
file.txt  gitexp4/  newfile.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git merge b2
Updating 2ce22e2..23811f8
Fast-forward
 demo.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 demo.txt

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ git merge testbranch
Already up to date.

mayuri@LAPTOP-II6DBUQ8 MINGW64 /d/Sem 5 assignment/devops/gitexp4 (master)
$ ls
demo.txt  file.txt  gitexp4/  newfile.txt
```

<u>Conclusion</u>: We learnt about Features of Branch workflow tactics in real time scenario contexts and how to implement them..