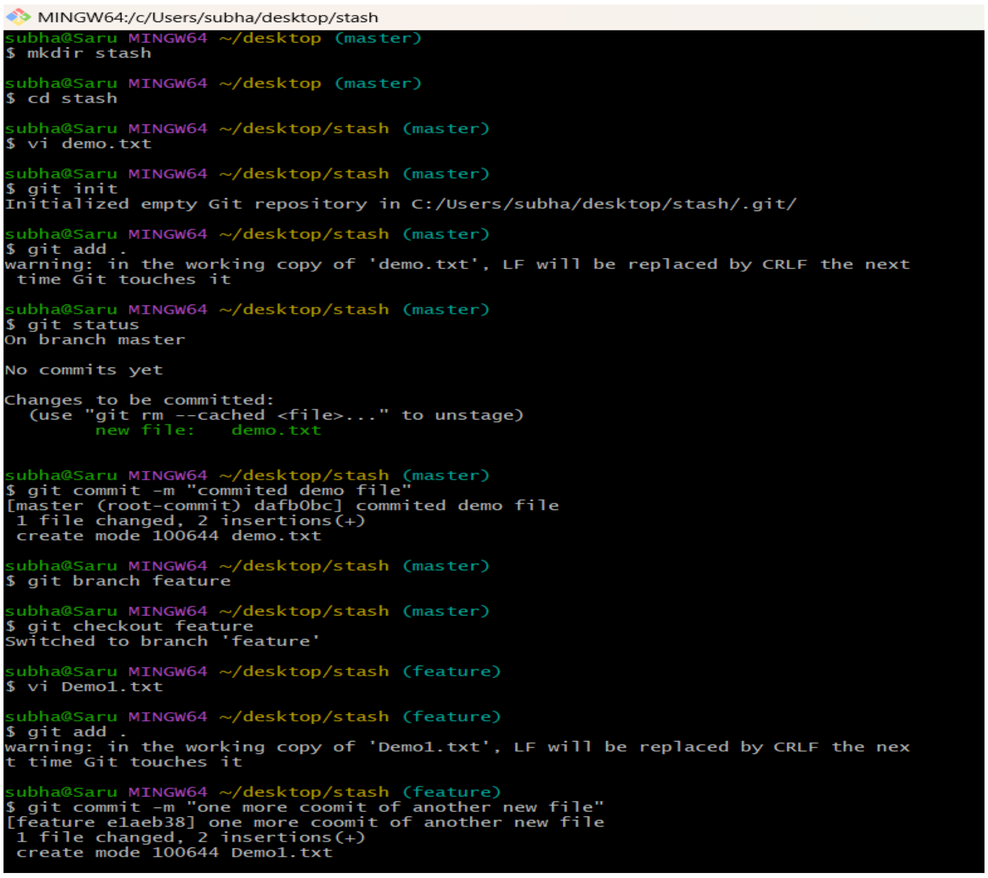
ASSIGNMENT

Q1. Describe the usage of the git stash command by using an example and also state the process by giving the screenshot of all the commands written in git bash.

Answer:

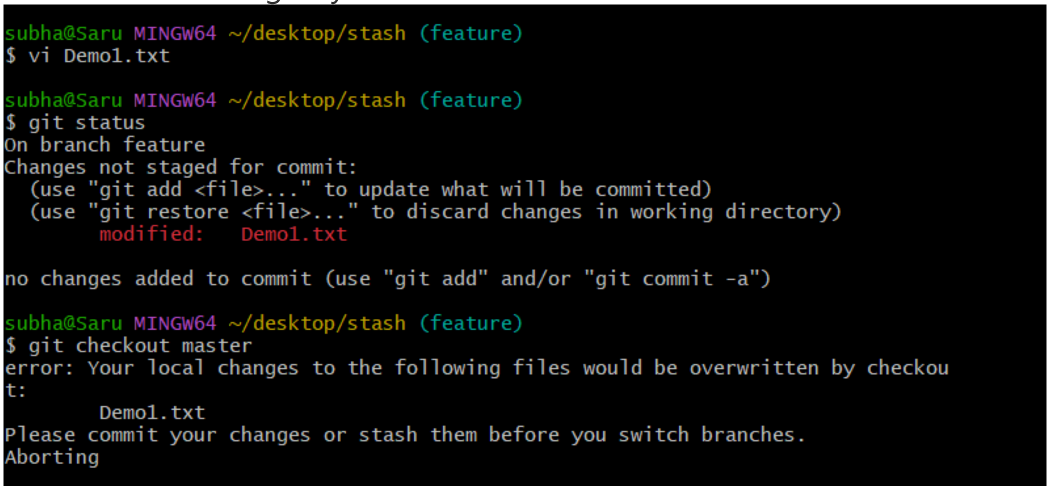
Usage of the git statsh:-

- The git stash commad is used to temporarily save changes in a Git repository that are not yet ready to be committed.
- The **git stash command** enables you to switch branches without committing the current branch.
- Generally, the stash's meaning is "**store something safely in a hidden place**."
- The sense in Git is also the same for stash; Git temporarily saves your data safely without committing.
- Git stash uses **STACK** data structure.
- Here are the steps to use git stash:-
- **Step-1:** Create a directory within a directory created a file and initialize the file,add the file into stagging area and commited the file.



```
MINGW64:/c/Users/subha/desktop/stash
subha@Saru MINGW64 ~/desktop (master)
$ mkdir stash

subha@Saru MINGW64 ~/desktop (master)
$ cd stash

subha@Saru MINGW64 ~/desktop/stash (master)
$ vi demo.txt

subha@Saru MINGW64 ~/desktop/stash (master)
$ git init
Initialized empty Git repository in C:/Users/subha/desktop/stash/.git/

subha@Saru MINGW64 ~/desktop/stash (master)
$ git add .
warning: in the working copy of 'demo.txt', LF will be replaced by CRLF the next
 time Git touches it

subha@Saru MINGW64 ~/desktop/stash (master)
$ git status
On branch master

No commits yet

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


subha@Saru MINGW64 ~/desktop/stash (master)
$ git commit -m "commited demo file"
[master (root-commit) dafb0bc] commited demo file
 1 file changed, 2 insertions(+)
 create mode 100644 demo.txt

subha@Saru MINGW64 ~/desktop/stash (master)
$ git branch feature

subha@Saru MINGW64 ~/desktop/stash (master)
$ git checkout feature
Switched to branch 'feature'

subha@Saru MINGW64 ~/desktop/stash (feature)
$ vi Demo1.txt

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git add .
warning: in the working copy of 'Demo1.txt', LF will be replaced by CRLF the nex
t time Git touches it

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git commit -m "one more coomit of another new file"
[feature e1aeb38] one more coomit of another new file
 1 file changed, 2 insertions(+)
 create mode 100644 Demo1.txt
```

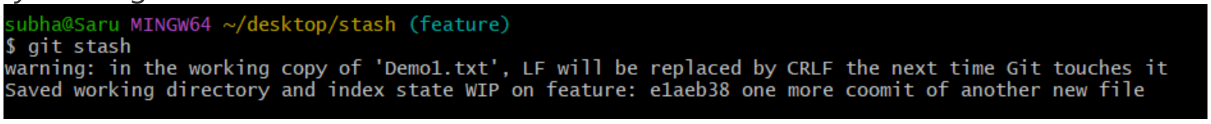
- **Step-2 :** Next, Created a branch named as feature and again created a file ,add the file into stagging area and commited the file.
- Later,done some modifications in one file and the status of a file .When the file in modified stage try to switch the branch then it shows an error.



```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ vi Demo1.txt

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git status
On branch feature
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:   Demo1.txt

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

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git checkout master
error: Your local changes to the following files would be overwritten by checkou
t:
        Demo1.txt
Please commit your changes or stash them before you switch branches.
Aborting
```

- **Git Stash:-** Stashes the modified files and creates new stash

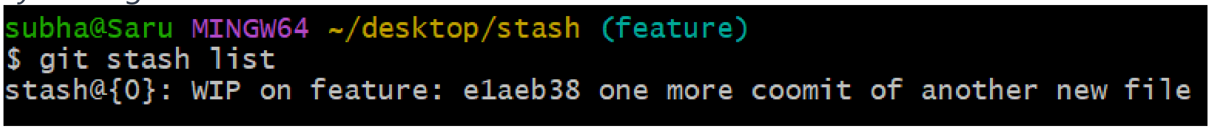
  Syntax :- git stash



```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash
warning: in the working copy of 'Demo1.txt', LF will be replaced by CRLF the next time Git touches it
Saved working directory and index state WIP on feature: e1aeb38 one more coomit of another new file
```

- **Git Stash List:-** To see all the stashes list we use

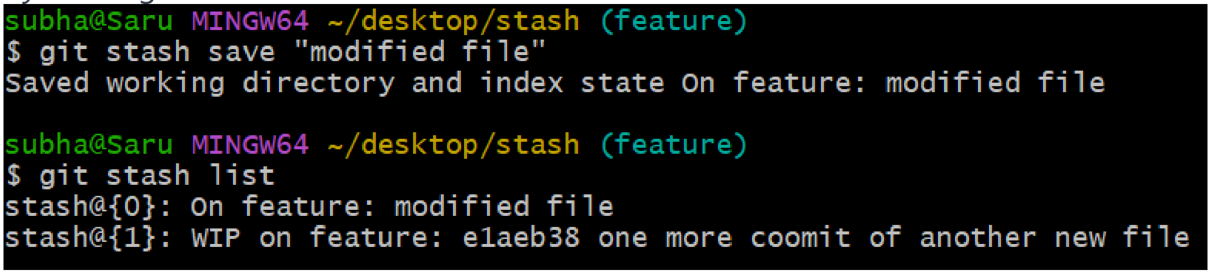
  Syntax:- git stash list



```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash list
stash@{0}: WIP on feature: e1aeb38 one more coomit of another new file
```

- **Git Stash Save:-** If you want to give the name for that particular stash

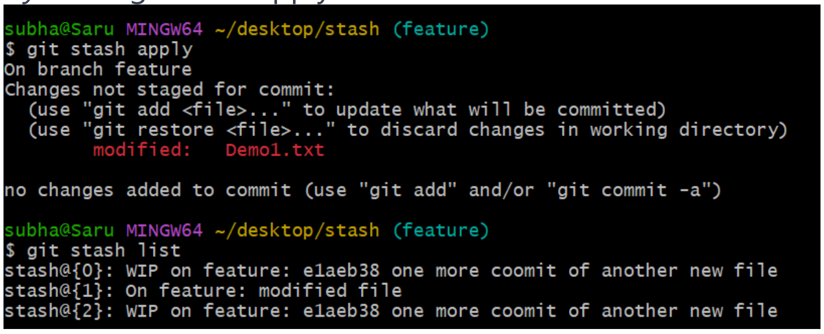
  Syntax:- git stash save<name>



```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash save "modified file"
Saved working directory and index state On feature: modified file

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash list
stash@{0}: On feature: modified file
stash@{1}: WIP on feature: e1aeb38 one more coomit of another new file
```

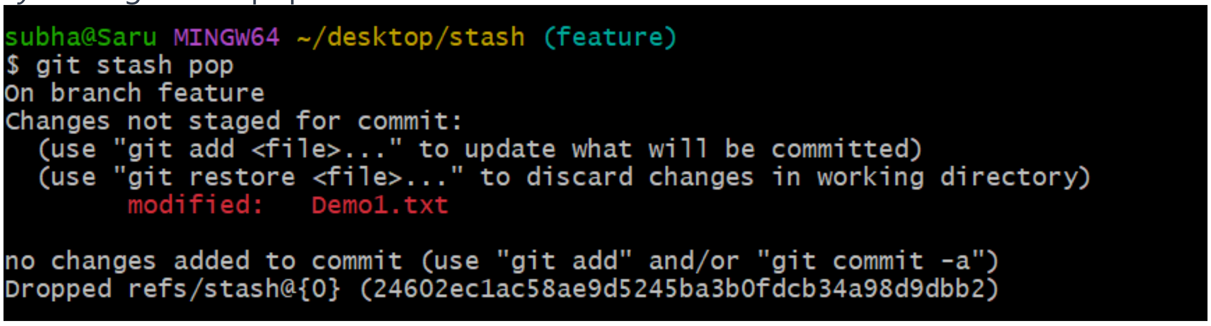
- **Git stash apply:-** To apply the particular stash

  Syntax:- git stash apply<stash-id>



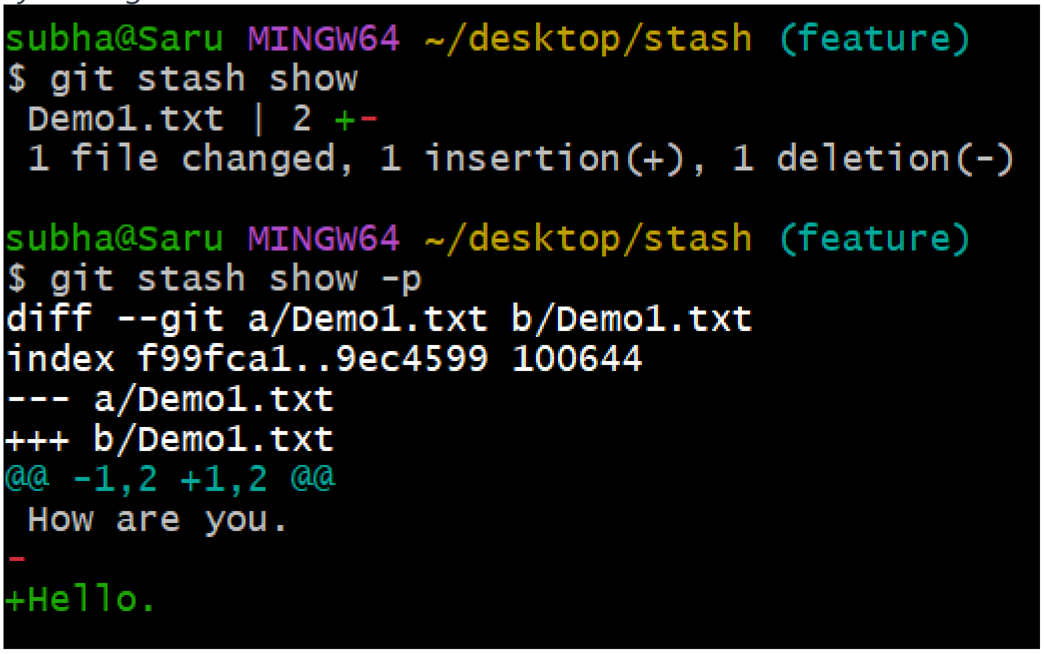
```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash apply
On branch feature
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Demo1.txt

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

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash list
stash@{0}: WIP on feature: e1aeb38 one more coomit of another new file
stash@{1}: On feature: modified file
stash@{2}: WIP on feature: e1aeb38 one more coomit of another new file
```

- **Git Stash Pop:-** To retrive the recent stashed data into the branch . After getting the stash data it removes automatically until and unless if there are no conflicts
  Syntax:- git stash pop



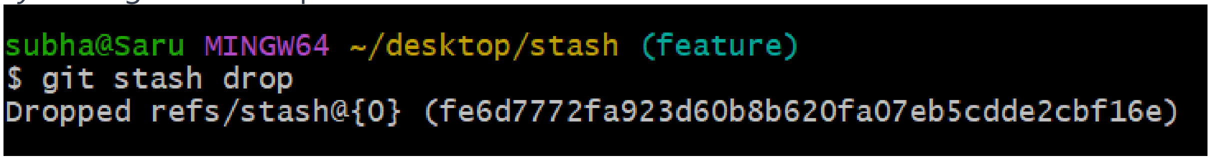
```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash pop
On branch feature
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Demo1.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (24602ec1ac58ae9d5245ba3b0fdcb34a98d9dbb2)
```

- **Git Stash Show:-** To check the changes of the stashed data before pulling
  Syntax:- git stash show



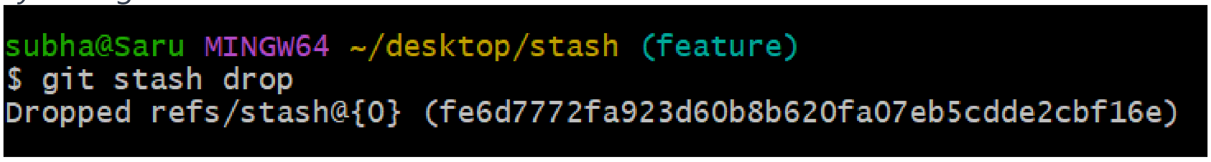
```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash show
 Demo1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash show -p
diff --git a/Demo1.txt b/Demo1.txt
index f99fca1..9ec4599 100644
--- a/Demo1.txt
+++ b/Demo1.txt
@@ -1,2 +1,2 @@
 How are you.
-
+Hello.
```

- **Git Stash Drop:-** To delete particular stash details
  Syntax:- git stash drop<stash-id>



```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash drop
Dropped refs/stash@{0} (fe6d7772fa923d60b8b620fa07eb5cdde2cbf16e)
```

- **Git Clear:-** Clear the complete stash details
  *Syntax:-*g



```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git stash drop
Dropped refs/stash@{0} (fe6d7772fa923d60b8b620fa07eb5cdde2cbf16e)
```

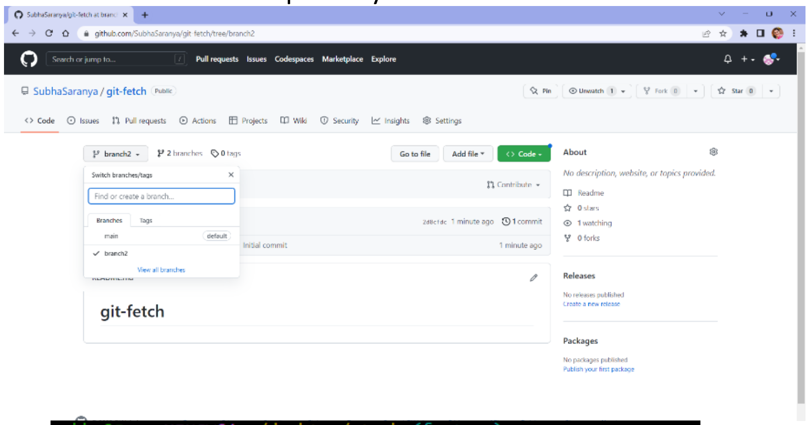
Q2. By using a sample example of your choice, use the git fetch command and also use the git merge command and describe the whole process through a screenshot with all the commands and their output in git bash.
Answer:

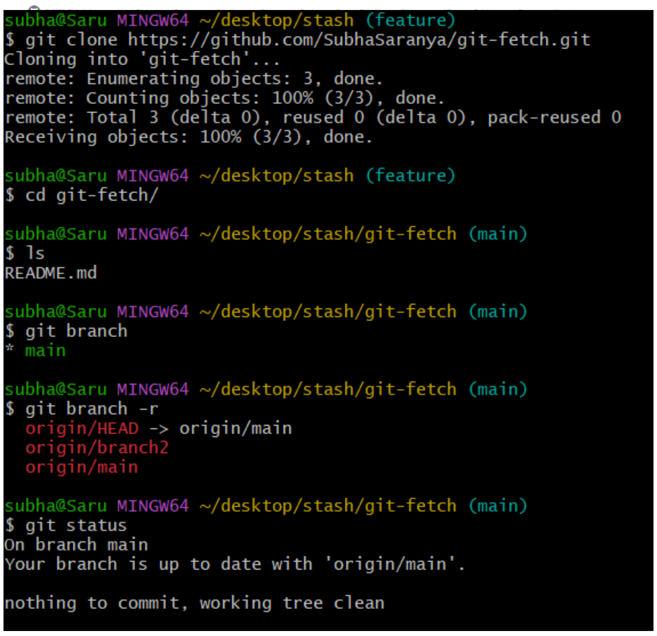
Git Fetch:

- The git fetch command downloads commits, files, and refs from a remote repository into your local repository.Fetching allows us to download changes from remote repository.But those changes will not be automatically integrated to our working files.
Syntax:- git fetch<remote>

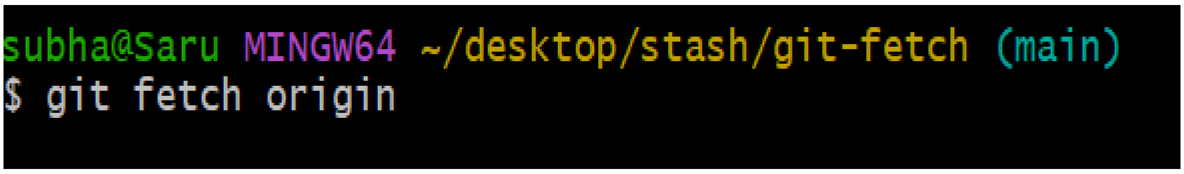
- **Process of git fetch command:-**

1.Check the status of your local repository with the command git status.
this will show you the current state of your local repository.
2.Create a new repository named as git-fetch in github
3.In that repository create two branches named as newBranch, branch2
4.Clone the repository and add some files in the git branches.
5.Fetches the remote repository

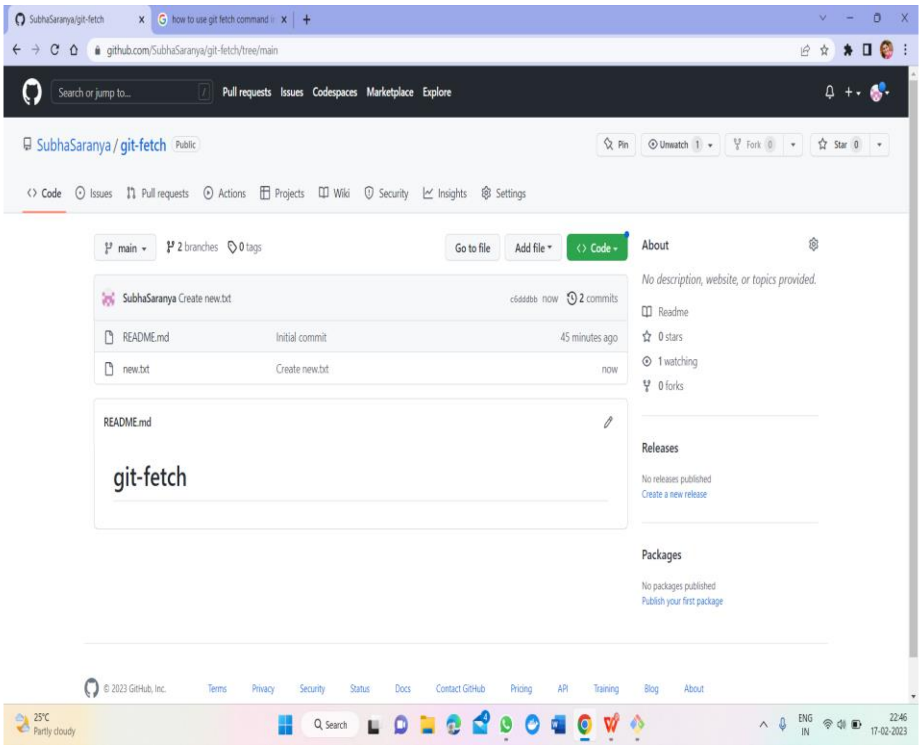




```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git clone https://github.com/SubhaSaranya/git-fetch.git
Cloning into 'git-fetch'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

subha@Saru MINGW64 ~/desktop/stash (feature)
$ cd git-fetch/

subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ ls
README.md

subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ git branch
* main

subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ git branch -r
  origin/HEAD -> origin/main
  origin/branch2
  origin/main

subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```



```
subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ git fetch origin
```



```
SubhaSaranya Create new.txt          c6dddbb now   2 commits
README.md    Initial commit          45 minutes ago
new.txt      Create new.txt          now
```

README.md

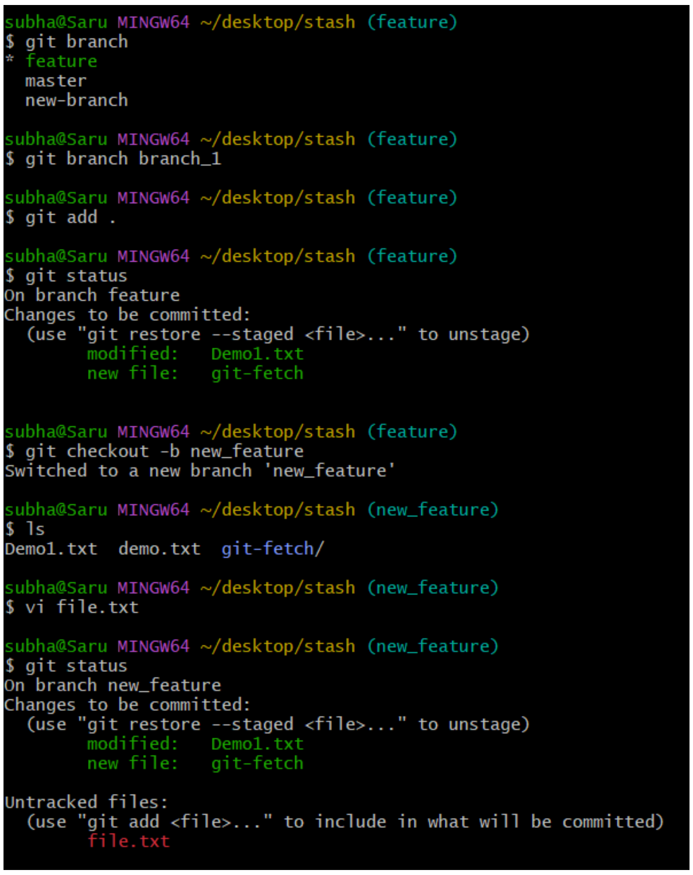
git-fetch

No description, website, or topics provided.

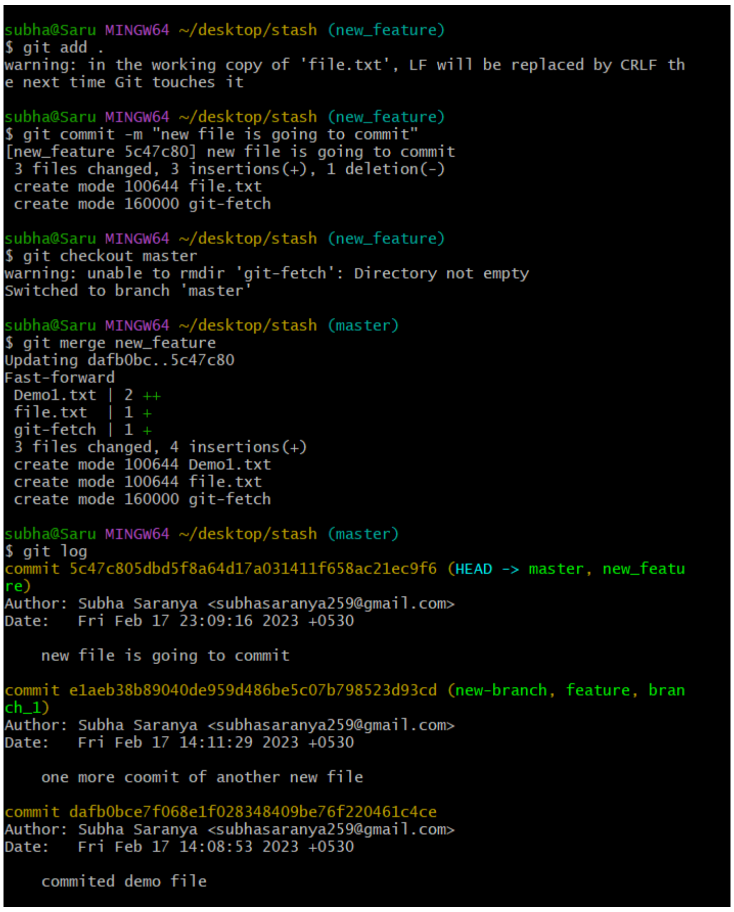
Git Merge:-

- Git merging is basically to merge multiple sequences of commits, stored in multiple branches.
- When you merge one branch into the another,Git takes the changes that were made on the source and applies them to the destination branch

  Syntax:-**git merge <filename>**



```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git branch
* feature
  master
  new-branch

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git branch branch_1

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git add .

subha@Saru MINGW64 ~/desktop/stash (feature)
$ git status
On branch feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Demo1.txt
        new file:   git-fetch


subha@Saru MINGW64 ~/desktop/stash (feature)
$ git checkout -b new_feature
Switched to a new branch 'new_feature'

subha@Saru MINGW64 ~/desktop/stash (new_feature)
$ ls
Demo1.txt  demo.txt  git-fetch/

subha@Saru MINGW64 ~/desktop/stash (new_feature)
$ vi file.txt

subha@Saru MINGW64 ~/desktop/stash (new_feature)
$ git status
On branch new_feature
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Demo1.txt
        new file:   git-fetch

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

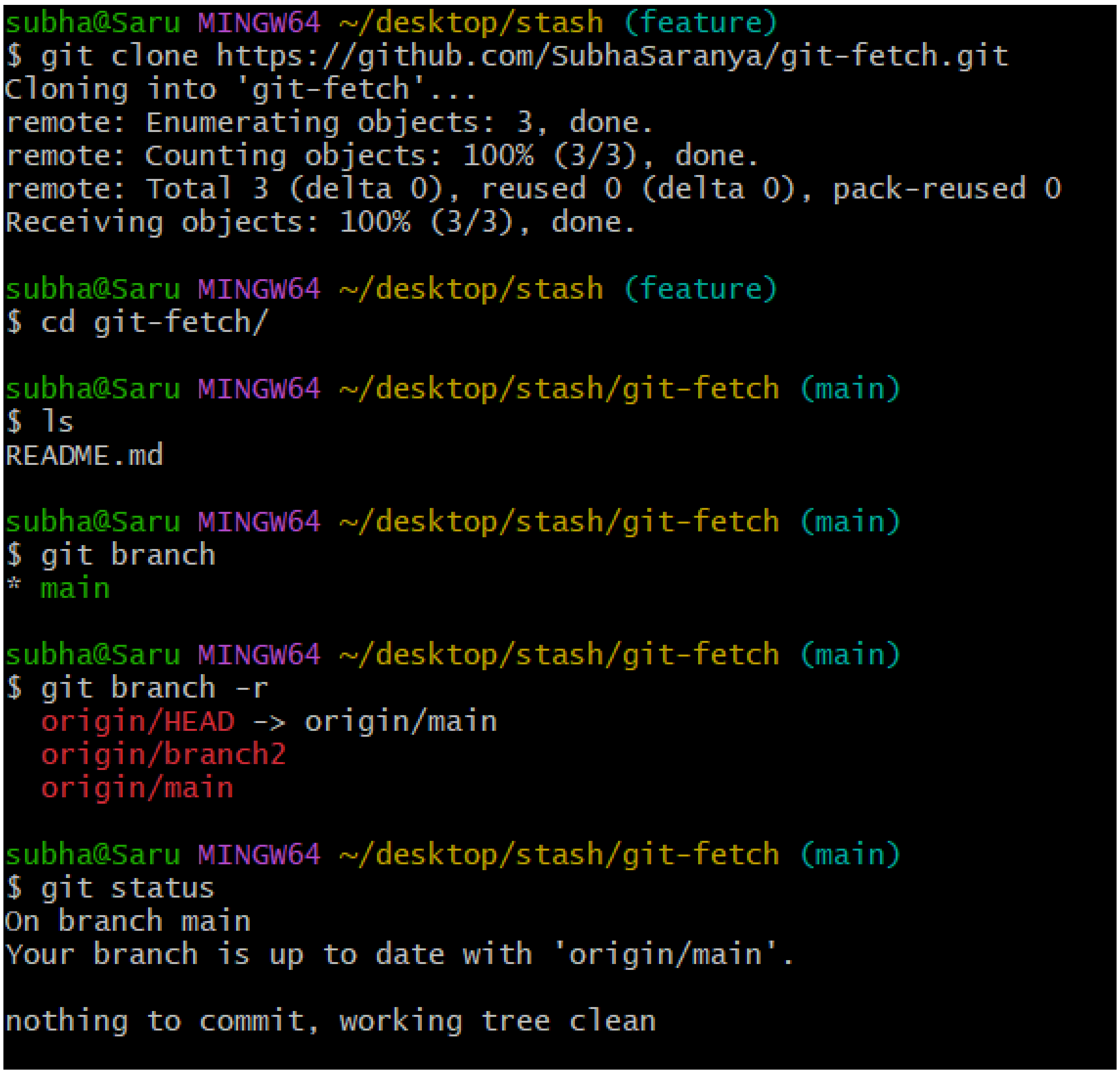


```
subha@Saru MINGW64 ~/desktop/stash (new_feature)
$ git add .
warning: in the working copy of 'file.txt', LF will be replaced by CRLF th
e next time Git touches it

subha@Saru MINGW64 ~/desktop/stash (new_feature)
$ git commit -m "new file is going to commit"
[new_feature 5c47c80] new file is going to commit
 3 files changed, 3 insertions(+), 1 deletion(-)
 create mode 100644 file.txt
 create mode 160000 git-fetch

subha@Saru MINGW64 ~/desktop/stash (new_feature)
$ git checkout master
warning: unable to rmdir 'git-fetch': Directory not empty
Switched to branch 'master'

subha@Saru MINGW64 ~/desktop/stash (master)
$ git merge new_feature
Updating dafb0bc..5c47c80
Fast-forward
 Demo1.txt | 2 ++
 file.txt  | 1 +
 git-fetch | 1 +
 3 files changed, 4 insertions(+)
 create mode 100644 Demo1.txt
 create mode 100644 file.txt
 create mode 160000 git-fetch

subha@Saru MINGW64 ~/desktop/stash (master)
$ git log
commit 5c47c805dbd5f8a64d17a031411f658ac21ec9f6 (HEAD -> master, new_featu
re)
Author: Subha Saranya <subhasaranya259@gmail.com>
Date:   Fri Feb 17 23:09:16 2023 +0530

    new file is going to commit

commit e1aeb38b89040de959d486be5c07b798523d93cd (new-branch, feature, bran
ch_1)
Author: Subha Saranya <subhasaranya259@gmail.com>
Date:   Fri Feb 17 14:11:29 2023 +0530

    one more coomit of another new file

commit dafb0bce7f068e1f028348409be76f220461c4ce
Author: Subha Saranya <subhasaranya259@gmail.com>
Date:   Fri Feb 17 14:08:53 2023 +0530

    commited demo file
```

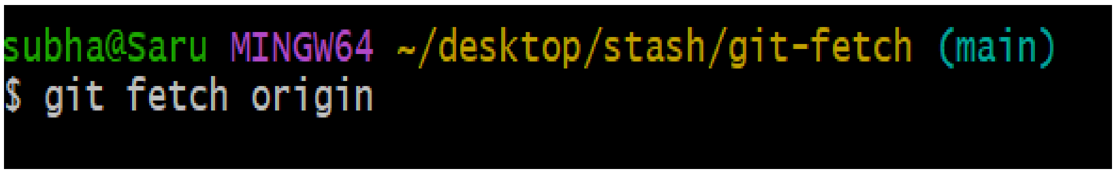
Q3)State the difference between git fetch and git pull by doing a practical example in your git bash and attach a screenshot of all the processes.
Answer:

Git Fetch:-

- Git fetch downloads the changes from a remote repository to your local repository, but it does not apply those changes to your current working branch.
- Instead, it updates your remote tracking branch to reflect any changes that have occurred on the remote repository.
- Git fetch is useful when you want to check for changes in a remote repository without merging them into your current working branch.



```
subha@Saru MINGW64 ~/desktop/stash (feature)
$ git clone https://github.com/SubhaSaranya/git-fetch.git
Cloning into 'git-fetch'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

subha@Saru MINGW64 ~/desktop/stash (feature)
$ cd git-fetch/

subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ ls
README.md

subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ git branch
* main

subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ git branch -r
  origin/HEAD -> origin/main
  origin/branch2
  origin/main

subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```



```
subha@Saru MINGW64 ~/desktop/stash/git-fetch (main)
$ git fetch origin
```

Git Pull:-

- **git pull**, on the other hand, does both a git fetch and a git merge in one step.
- It downloads the changes from a remote repository to your local repository and immediately applies those changes to your current working branch. git pull is useful when you want to update your local branch to the latest changes in a remote branch and immediately see those changes in your working copy.

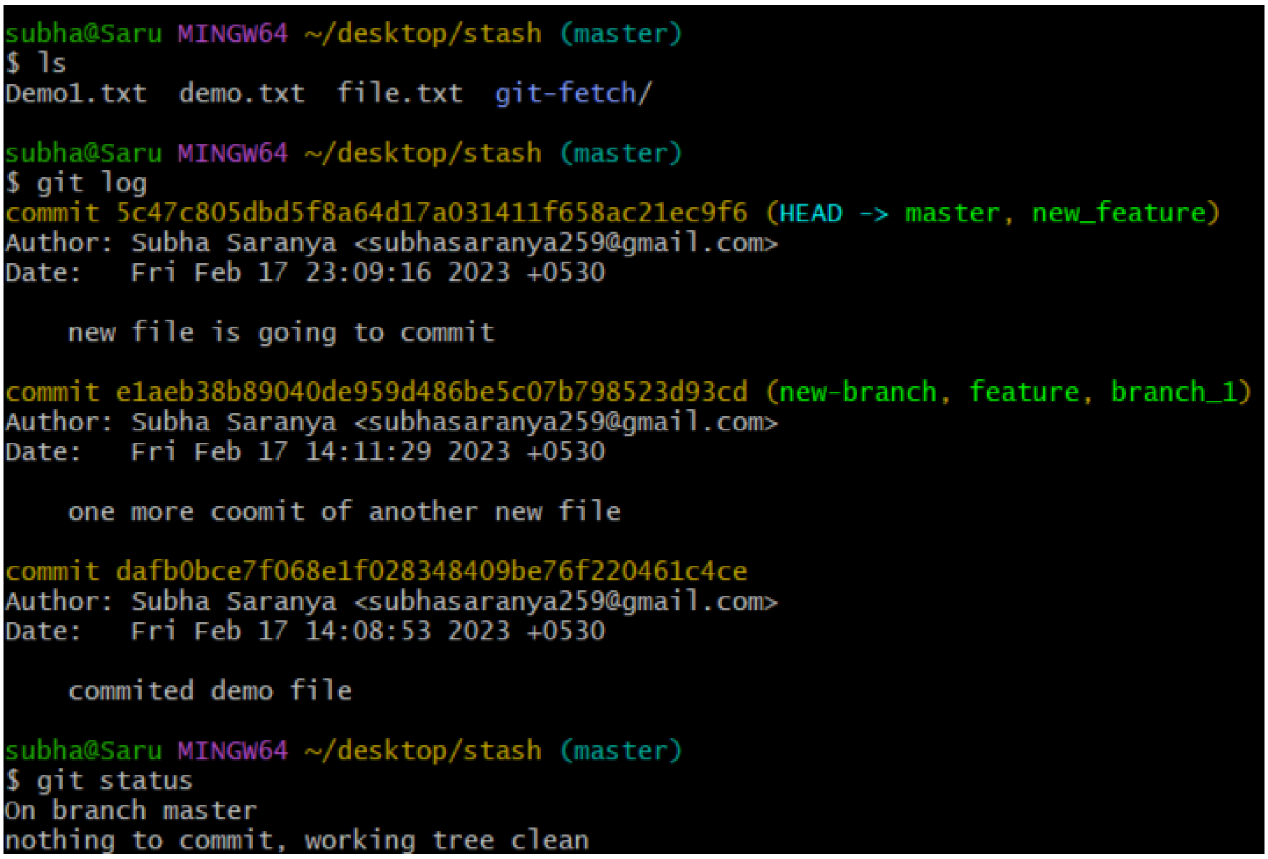
Steps for git pull:-

Step1:- At first, check the status of the git repository and check what files are present in.

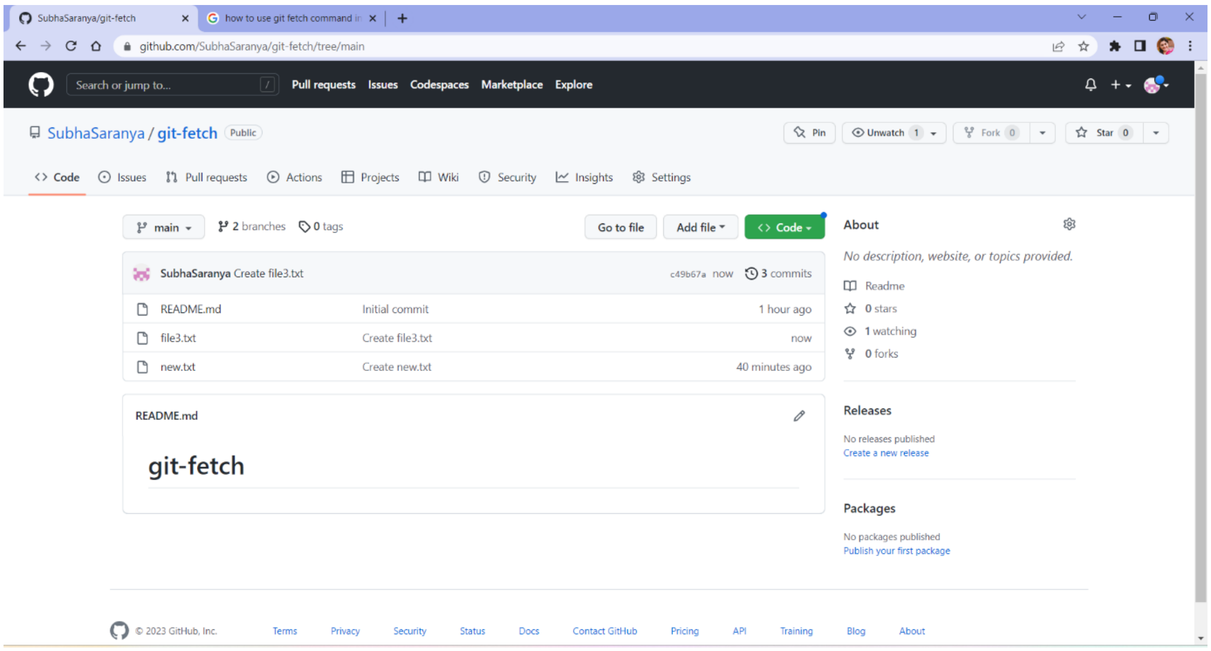
Step2:- Add a file in git repository and commit the file

Step3:- From the steps the changes we done in a git repository in not visible in local r repository

Step4:- The changes we are done in git repository are visible in local repository when you excute the git pull command



```
subha@Saru MINGW64 ~/desktop/stash (master)
$ ls
Demo1.txt  demo.txt  file.txt  git-fetch/

subha@Saru MINGW64 ~/desktop/stash (master)
$ git log
commit 5c47c805dbd5f8a64d17a031411f658ac21ec9f6 (HEAD -> master, new_feature)
Author: Subha Saranya <subhasaranya259@gmail.com>
Date:   Fri Feb 17 23:09:16 2023 +0530

    new file is going to commit

commit e1aeb38b89040de959d486be5c07b798523d93cd (new-branch, feature, branch_1)
Author: Subha Saranya <subhasaranya259@gmail.com>
Date:   Fri Feb 17 14:11:29 2023 +0530

    one more coomit of another new file

commit dafb0bce7f068e1f028348409be76f220461c4ce
Author: Subha Saranya <subhasaranya259@gmail.com>
Date:   Fri Feb 17 14:08:53 2023 +0530

    commited demo file

subha@Saru MINGW64 ~/desktop/stash (master)
$ git status
On branch master
nothing to commit, working tree clean
```

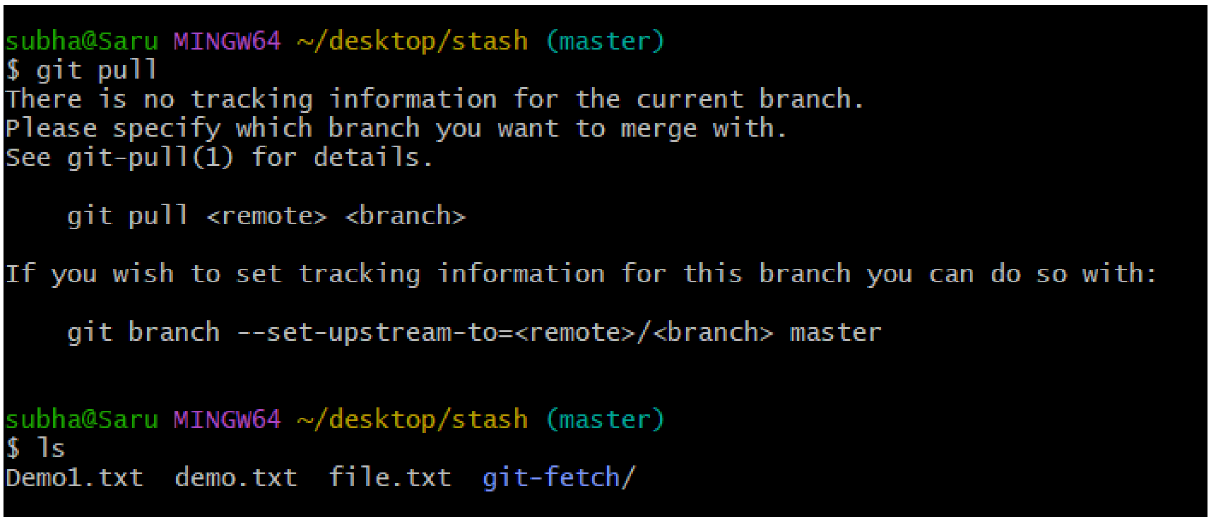


SubhaSaranya Create file3.txt c49b67a now 3 commits

README.md	Initial commit	1 hour ago
file3.txt	Create file3.txt	now
new.txt	Create new.txt	40 minutes ago

README.md

git-fetch

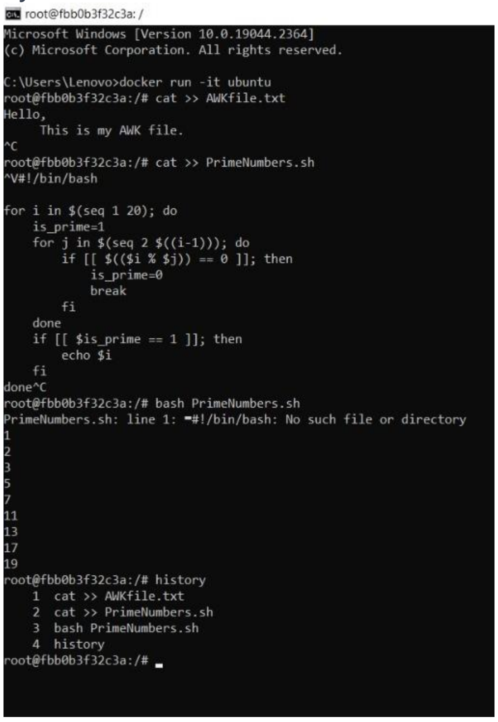


```
subha@Saru MINGW64 ~/desktop/stash (master)
$ git pull
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=<remote>/<branch> master


subha@Saru MINGW64 ~/desktop/stash (master)
$ ls
Demo1.txt  demo.txt  file.txt  git-fetch/
```

Q4). Try to find out about the awk command and use it while reading a file created by yourself. Also, make a bash script file and try to find out the prime number from the range 1 to 20.The whole process should be carried out and by using the history command, give the screenshot of all the processes being carried out.

AWK:-

- **awk** is a powerful command-line tool used for processing and manipulating text files especially when dealing with large amounts of data.
- In the below image, created a file and named as Data.txt and print the same data
- Commands on awk:-
- Syntax:- **awk '{print}' filename**
  This commad prints the data present in the file
- Command:- **awk '{print$column_number}' filename**
  Eg:-awk '{print$2}' Data.txt
  This command prints the second column data in a data.txt file
- Command:- **awk '{print$1,$4}' Data.txt**
  This command prints the first and fourth column data in a Data.txt file
- **BASH SCRIPTING:-**
  Steps to follow in bash Scripting
- Step1:-Create a file with extension .sh
- Step2:- Give the permissions of read, write and excute
- Step3:- open the shell and write the script
- Step4:- Save the code and run the code
  The command to run a code is
  Syntax:- ./filename

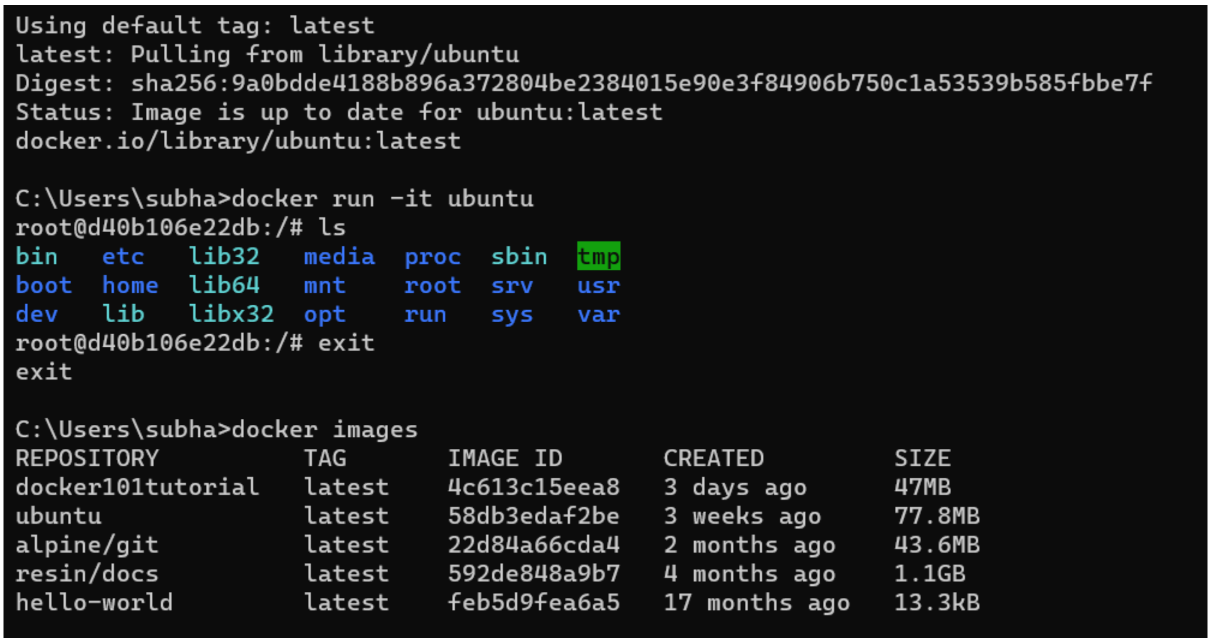


```
root@fbb0b3f32c3a: /
Microsoft Windows [Version 10.0.19044.2364]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>docker run -it ubuntu
root@fbb0b3f32c3a:/# cat >> AWKfile.txt
Hello,
    This is my AWK file.
^C
root@fbb0b3f32c3a:/# cat >> PrimeNumbers.sh
^V#!/bin/bash

for i in $(seq 1 20); do
    is_prime=1
    for j in $(seq 2 $((i-1))); do
        if [[ $(($i % $j)) == 0 ]]; then
            is_prime=0
            break
        fi
    done
    if [[ $is_prime == 1 ]]; then
        echo $i
    fi
done^C
root@fbb0b3f32c3a:/# bash PrimeNumbers.sh
PrimeNumbers.sh: line 1: ▬#!/bin/bash: No such file or directory
1
2
3
5
7
11
13
17
19
root@fbb0b3f32c3a:/# history
    1  cat >> AWKfile.txt
    2  cat >> PrimeNumbers.sh
    3  bash PrimeNumbers.sh
    4  history
root@fbb0b3f32c3a:/# _
```

Q5)Set up a container and run a Ubuntu operating system. For this purpose, you can make use of the docker hub and run the container in interactive mode.All the processes pertaining to this should be provided in a screenshot for grading.
Answer:

- Steps to set up a container and run a ubuntu operating system.
- **Step1**:- Install docker image from a google and set the docker image according to your machine
  Step2:-To check the weather the docker installed correctly in your machine excute the below command in the command prompt.
  Command:- **docker version**
- If you get the description and version about the docker then it installed correctly. Otherwise again install the docker in your machine
- **Step3:**-Pull the ubuntu image from the docker image by running the below
  Command:- **docker pull ubuntu**
- **Step4:**-After the image was downloaded,run the container using the following
  Command:-**docker run -it ubuntu**

- "-it" option runs the container in the interactive mode and opens up a shell within the ubuntu operating system
- To see the downloaded docker images list we use the following command.
  Command:- docker image



```
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbbe7f
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest

C:\Users\subha>docker run -it ubuntu
root@d40b106e22db:/# ls
bin   etc   lib32   media  proc  sbin  tmp
boot  home  lib64   mnt    root  srv   usr
dev   lib   libx32  opt    run   sys   var
root@d40b106e22db:/# exit
exit

C:\Users\subha>docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
docker101tutorial   latest    4c613c15eea8   3 days ago      47MB
ubuntu              latest    58db3edaf2be   3 weeks ago     77.8MB
alpine/git          latest    22d84a66cda4   2 months ago    43.6MB
resin/docs          latest    592de848a9b7   4 months ago    1.1GB
hello-world         latest    feb5d9fea6a5   17 months ago   13.3kB
```