DevOps 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.

-> git stash temporarily shelves (or stashes) changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on. Stashing is handy if you need to quickly switch context and work on something else, but you're mid-way through a code change and aren't quite ready to commit.

Example: Disha and Vedant are working on a website . Disha is working on the home page and vedant is working on some other page .Disha has some uncommitted changes and she wants to check the progress of vedant's work but she doesn't want to commit yet as the code is incomplete.So in this case she can stash current changes and switch to vedant's branch to check the progress .After that she can come back and take the changes from the stash stack and resume her work.

1. Git stash push -m "message" : used to create a stash.

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment
$ git init
Initialized empty Git repository in D:/git/git_class_assignment/.git/

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ ls
file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git add .

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git commit -m "created file1"
[master (root-commit) 67367e4] created file1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git branch second_branch

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ vi file2

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ ls
file1  file2

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file2

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

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git add .
warning: in the working copy of 'file2', LF will be replaced by CRLF the next time Git touches it

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ ^C

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ vi file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ ls
file1  file2
```

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ ls
file1  file2

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2

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


krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash push -m "First stash"
warning: in the working copy of 'file1', LF will be replaced by CRLF the next time Git touches it
Saved working directory and index state On master: First stash

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stautus
git: 'stautus' is not a git command. See 'git --help'.

The most similar command is
        status

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git status
On branch master
nothing to commit, working tree clean

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git branch
* master
  second_branch

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git checkout second_branch
Switched to branch 'second_branch'

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (second_branch)
$ git branch
  master
* second_branch

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (second_branch)
$ git checkout master
Switched to branch 'master'

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git branch
* master
  second_branch
```

2. Git stash list : Will list the all of the stashes that've been created.

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash list
stash@{0}: On master: First stash
```

3. Git stash show [—index]: Shows the changes lines in that particular stash with respective to the files.

If you don't give the index it'll take the 0.

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash show 0
 file1 | 1 +
 file2 | 1 +
 2 files changed, 2 insertions(+)
```

4. Git stash apply [—index]: is used to get and apply the changes the of a particular stash .

**This will keep the stash in stack.

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git status
On branch master
nothing to commit, working tree clean

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash apply 0
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2

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
```

5. Git stash pop [—index]: is used to get and apply the changes the of a particular stash .

It'll will remove that particular stash from the stack.**This will remove the stash in stack.

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ vi file2

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ ls
file1  file2

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2

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
        modified:   file2


krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash push -m "second stash"
Saved working directory and index state On master: second stash

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash list
stash@{0}: On master: second stash
stash@{1}: On master: First stash

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash pop 1
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2

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

Dropped refs/stash@{1} (9f381e2ac056b153c214fe8ebcc1c27b70e790c6)

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash list
stash@{0}: On master: second stash

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$
```

6.Git stash drop: Used to drop/pop a particular stash from the stack

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ vi file2

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash push -m "third stash"
Saved working directory and index state On master: third stash

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash list
stash@{0}: On master: third stash
stash@{1}: On master: second stash

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash drop 0
Dropped refs/stash@{0} (a9a49a9564cbee1d3114ecce339217aa547fa914)
```

7. Git stash clear: This will remove all the stashes from the stack.

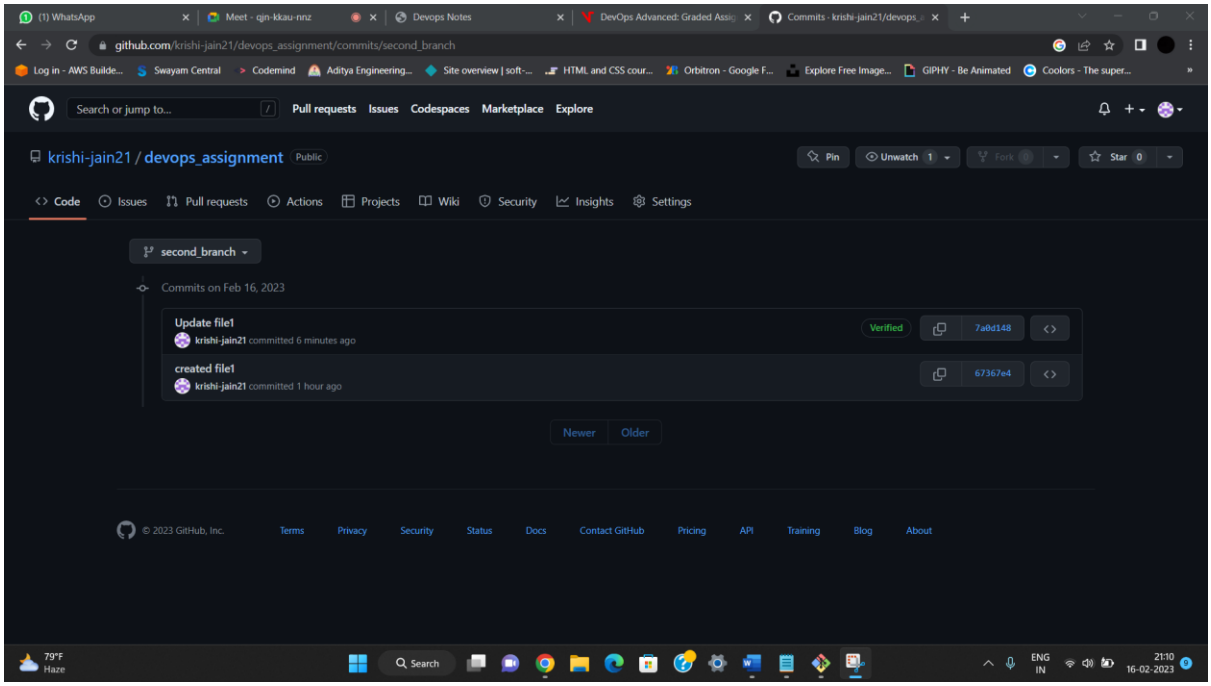
```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash clear

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git stash list
```

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.

->Git fetch : The git fetch command downloads commits, files, and refs from a remote repository into your local repo. Fetching is what you do when you want to see what everybody else has been working on. Git fetch downloads but it won't merge .

->Git merge: Merging is Git's way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.

For Example: We are creating a repository in the git hub and cloning it to the local then we will make a new branch in the remote and create a commit in that and fetch that update by git fetch as we know that it only downloads that update/commit but does not merge, so we will use git merge to merge that updated branch with the local repository.




```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git remote add origin git@github.com:krishi-jain21/devops_assignment.git

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (master)
$ git branch -M main

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 210 bytes | 210.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:krishi-jain21/devops_assignment.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git push --all
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'second_branch' on GitHub by visiting:
remote:      https://github.com/krishi-jain21/devops_assignment/pull/new/second_branch
remote:
To github.com:krishi-jain21/devops_assignment.git
 * [new branch]      second_branch -> second_branch

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git fetch --all
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 627 bytes | 52.00 KiB/s, done.
From github.com:krishi-jain21/devops_assignment
   67367e4..7a0d148  second_branch -> origin/second_branch

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git log --all
commit 7a0d1484f1bfe4a08dd1e12d746a7d40f2ec6486 (origin/second_branch)
Author: Krishi Jain <84382254+krishi-jain21@users.noreply.github.com>
Date:   Thu Feb 16 21:03:22 2023 +0530

    Update file1

commit 67367e45f8dd40acc8536bbc3fcd3bdc65499cc1 (HEAD -> main, origin/main, second_branch)
Author: krishi-jain21 <krishijain212121@gmail.com>
Date:   Thu Feb 16 20:12:11 2023 +0530

    created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git log --oneline --all
7a0d148 (origin/second_branch) Update file1
67367e4 (HEAD -> main, origin/main, second_branch) created file1
```

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git merge origin/second_branch
Updating 67367e4..7a0d148
Fast-forward
 file1 | 1 +
 1 file changed, 1 insertion(+)

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git log --oneline --all
7a0d148 (HEAD -> main, origin/second_branch) Update file1
67367e4 (origin/main, second_branch) created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git log --oneline
7a0d148 (HEAD -> main, origin/second_branch) Update file1
67367e4 (origin/main, second_branch) created file1
```

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.

->**Git fetch** :The git fetch command downloads commits, files, and refs from a remote repository into your local repo. Fetching is what you do when you want to see what everybody else has been working on. It's similar to svn update in that it lets you see how the central history has progressed, but it doesn't force you to actually merge the changes into your repository. Git isolates fetched content from existing local content; it has absolutely no effect on your local development work. Fetched content has to be explicitly checked out using the git checkout command. This makes fetching a safe way to review commits before integrating them with your local repository.

->**Git pull**:The git pull command first runs git fetch which downloads content from the specified remote repository. Then a git merge is executed to merge the remote content refs and heads into a new local merge commit.

**Git Fetch is the command that tells the local repository that there are changes available in the remote repository without bringing the changes into the local repository. Git Pull on the other hand brings the copy of the remote directory changes into the local repository.

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git log --oneline
7a0d148 (HEAD -> main, origin/second_branch) Update file1
67367e4 (origin/main, second_branch) created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 630 bytes | 126.00 KiB/s, done.
From github.com:krishi-jain21/devops_assignment
   67367e4..f47ebbb  main       -> origin/main

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git log --oneline
7a0d148 (HEAD -> main, origin/second_branch) Update file1
67367e4 (second_branch) created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git log --all
commit f47ebbbb8091730d932b5c441e91e2f4c1376c86 (origin/main)
Author: Krishi Jain <84382254+krishi-jain21@users.noreply.github.com>
Date:   Thu Feb 16 22:21:59 2023 +0530

    Update file1

commit 7a0d1484f1bfe4a08dd1e12d746a7d40f2ec6486 (HEAD -> main, origin/second_branch)
Author: Krishi Jain <84382254+krishi-jain21@users.noreply.github.com>
Date:   Thu Feb 16 21:03:22 2023 +0530

    Update file1

commit 67367e45f8dd40acc8536bbc3fcd3bdc65499cc1 (second_branch)
Author: krishi-jain21 <krishijain212121@gmail.com>
Date:   Thu Feb 16 20:12:11 2023 +0530

    created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git log --all --oneline
f47ebbb (origin/main) Update file1
7a0d148 (HEAD -> main, origin/second_branch) Update file1
67367e4 (second_branch) created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$
```

```
$ git log --oneline --all
f47ebbb (origin/main) Update file1
7a0d148 (HEAD -> main, origin/second_branch) Update file1
67367e4 (second_branch) created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 641 bytes | 160.00 KiB/s, done.
From github.com:krishi-jain21/devops_assignment
   f47ebbb..deb1ce9  main       -> origin/main
Auto-merging file1
CONFLICT (content): Merge conflict in file1
Automatic merge failed; fix conflicts and then commit the result.

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ git log --oneline
7a0d148 (HEAD -> main, origin/second_branch) Update file1
67367e4 (second_branch) created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ git log --all
commit deb1ce95d68daa8d5723f0ab793372a12cf11e7f (origin/main)
Author: Krishi Jain <84382254+krishi-jain21@users.noreply.github.com>
Date:   Thu Feb 16 22:31:37 2023 +0530

    Update file1

commit f47ebbbb8091730d932b5c441e91e2f4c1376c86
Author: Krishi Jain <84382254+krishi-jain21@users.noreply.github.com>
Date:   Thu Feb 16 22:21:59 2023 +0530

    Update file1

commit 7a0d1484f1bfe4a08dd1e12d746a7d40f2ec6486 (HEAD -> main, origin/second_bra
Author: Krishi Jain <84382254+krishi-jain21@users.noreply.github.com>
Date:   Thu Feb 16 21:03:22 2023 +0530

    Update file1

commit 67367e45f8dd40acc8536bbc3fcd3bdc65499cc1 (second_branch)
Author: krishi-jain21 <krishijain212121@gmail.com>
Date:   Thu Feb 16 20:12:11 2023 +0530

    created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ git log --oneline --all
deb1ce9 (origin/main) Update file1
f47ebbb Update file1
7a0d148 (HEAD -> main, origin/second_branch) Update file1
67367e4 (second_branch) created file1

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$
```

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.

->**awk command**:AWK is suitable for pattern search and processing. The script runs to search one or more files to identify matching patterns and if the said patterns perform specific tasks. In this guide, we take a look into AWK Linux command and see what it can do.

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ vi first_file

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ awk '{print($0)}'


krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ awk '{print($0)}' first_file
Sno      Name       Language           Native place
1        Krishi      Hindi/Marwadi     Rajashthan
2        Nani        Telugu            Andhra Pradesh
3        Pratik      Hindi             Jharkhand
4        Bhanu       Telugu            Andhra Pradesh

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ awk '{print($2)}' first_file
Name
Krishi
Nani
Pratik
Bhanu

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ awk '/i/ {print($0)}' first_file
Sno      Name       Language           Native place
1        Krishi      Hindi/Marwadi     Rajashthan
2        Nani        Telugu            Andhra Pradesh
3        Pratik      Hindi             Jharkhand

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$
```

Program to print primes between 1 to 20:

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ vi prime.sh
```

MINGW64:/d/git/git_class_assignment

```
read n
echo " Prime number between 1 to $n is:"
echo "2"
for((i=3;i<=n;))
do
  for((j=i-1;j>=2;))
  do
    if [  `expr $i % $j` -ne 0 ] ; then
      prime_1=1
    else
      prime_1=0
      break
    fi
    j=`expr $j - 1`
  done
  if [ $prime_1 -eq 1 ] ; then
    echo $i
  fi
  i=`expr $i + 1`
done
```

```
prime.sh [unix] (14:54 17/02/2023)
"prime.sh" [unix] 23L, 335B
```

Output:

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ ./prime.sh
enter the range
20
 Prime number between 1 to 20 is:
2
3
5
7
11
13
17
19

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
```

History command: Linux history command is used to display the history of the commands executed by the user.

```
krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$ history
    1  ssh-keygen -t ed25519 -C "krishijain212121@gmail.com"
    2  eval "$(ssh-agent -s)"
    3  touch file1
    4  ls
    5  git add.
    6  git add .
    7  clear
    8  git init
    9  ls
   10  git add .
   11  git commit -m "created file1"
   12  git branch second_branch
   13  vi file2
   14  ls
   15  git status
   16  git add .
   17  vi file1
   18  ls
   19  git status
   20  git stash push -m "First stash"
   21  git stautus
   22  git status
   23  git branch
   24  git checkout second_branch
   25  git branch
   26  git checkout master
   27  git branch
   28  git stash --list
   29  git stash list
   30  git stash show --0
   31  git stash show 0
   32  git status
   33  git stash apply 0
   34  vi file2
   35  ls
   36  git status
   37  git stash push -m "second stash"
   38  git stash list
   39  git stash pop 1
   40  git stash list
   41  vi file2
   42  git stash push -m "third stash"
   43  git stash list
   44  git stash drop 0
   45  git stash clear
   46  git stash list
   47  git remote add origin git@github.com:krishi-jain21/devops_assignment.git
   48  git branch -M main
   49  git push -u origin main
   50  git push --all
   51  git fetch --all
   52  git log --all
   53  git log --oneline --all
   54  git merge origin/second_branch
   55  git log --oneline --all
   56  git log --oneline
   57  git log --oneline
   58  git fetch
   59  git log --oneline
   60  git log --all
   61  git log --all --oneline
   62  git log --oneline --all
   63  git pull
   64  git log --oneline
   65  git log --all
   66  git log --oneline --all
   67  awk
   68  awk
   69  vi first_file
   70  awk '{print($0)}'
   71  awk '{print($0)}' first_file
   72  awk '{print($2)}' first_file
   73  awk '/i/ {print($0)}' first_file
   74  history

krishi@KrishiJain MINGW64 /d/git/git_class_assignment (main|MERGING)
$
```

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.

→Docker pull is used to get an image from docker hub so here we use “docker pull ubuntu:latest “

→This will get the latest image of ubuntu from dockerhub.To run ubuntu container we use “docker run -ti ubuntu /bin/bash“.

→This will open an interactive terminal of ubuntu container that started .