

**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.**

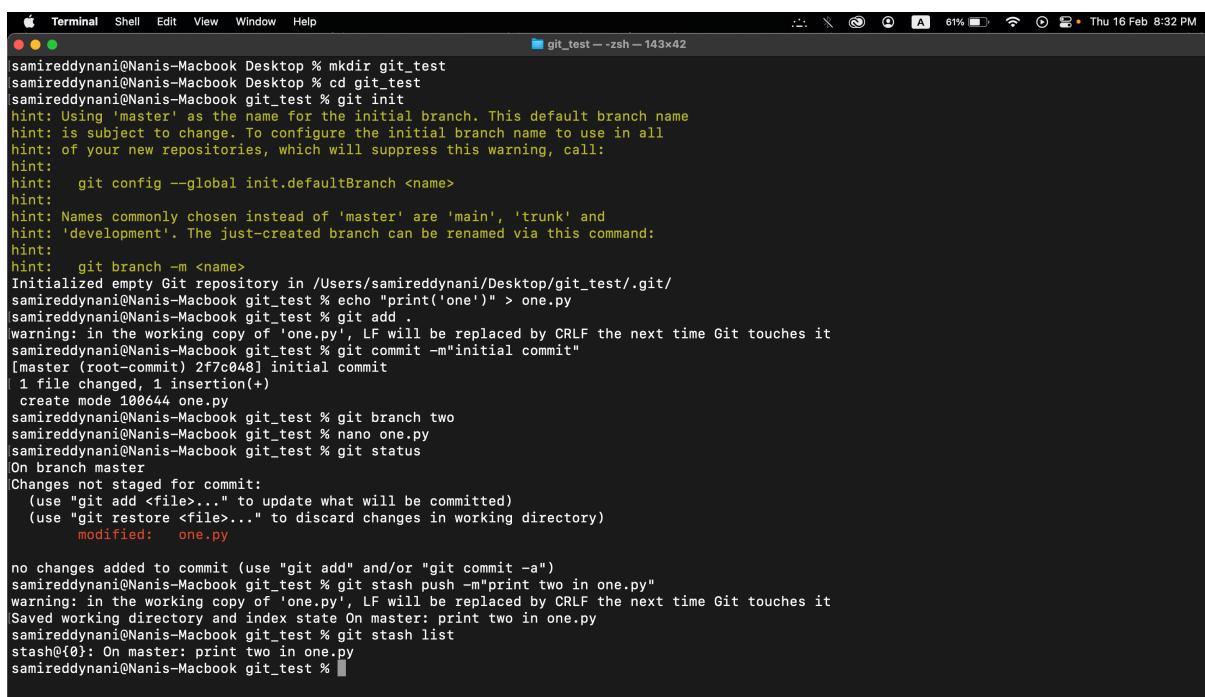
---

**Git stash:** Git stash is used to temporarily store the changes which are not committed and under tracked . 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.

**For example:** If we are working on a group project and the project is divided into modules and each module has a branch. And If we want to checkout the progress of the others without committing the current changes then we can stash our current changes and checkout to the other branch. After completing that we can come back and resume our work by taking the changes from the stash to current working directory.

-> Following are the commands used in Git stash:

- To create a stash we use the command
  - **git stash push -m “message”**

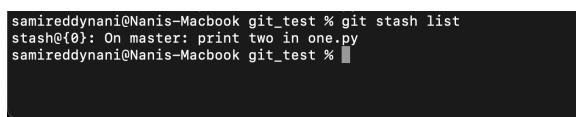


The screenshot shows a Mac OS X terminal window titled "git\_test -- zsh". The session starts with creating a new directory "git\_test" and navigating into it. Then, "git init" is run, which outputs several hints about the default branch name "master". It suggests renaming it to something like "main" or "trunk" or "development". A warning is shown about LF being replaced by CRLF. The user then runs "git commit -m 'initial commit'" and adds a file "one.py". Next, "git branch two" is run to create a new branch. Finally, "git stash push -m 'print two in one.py'" is executed, creating a stash entry named "stash@{0}" containing the changes from the "two" branch.

```
Terminal Shell Edit View Window Help
git_test -- zsh - 143x42
samireddynani@Nanis-Macbook Desktop % mkdir git_test
samireddynani@Nanis-Macbook Desktop % cd git_test
samireddynani@Nanis-Macbook git_test % git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/samireddynani/Desktop/git_test/.git/
samireddynani@Nanis-Macbook git_test % echo "print('one')" > one.py
samireddynani@Nanis-Macbook git_test % git add .
warning: in the working copy of 'one.py', LF will be replaced by CRLF the next time Git touches it
samireddynani@Nanis-Macbook git_test % git commit -m"initial commit"
[master (root-commit) 2f7c048] initial commit
1 file changed, 1 insertion(+)
create mode 100644 one.py
samireddynani@Nanis-Macbook git_test % git branch two
samireddynani@Nanis-Macbook git_test % nano one.py
samireddynani@Nanis-Macbook git_test % 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:   one.py

no changes added to commit (use "git add" and/or "git commit -a")
samireddynani@Nanis-Macbook git_test % git stash push -m"print two in one.py"
warning: in the working copy of 'one.py', LF will be replaced by CRLF the next time Git touches it
Saved working directory and index state On master: print two in one.py
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: print two in one.py
samireddynani@Nanis-Macbook git_test %
```

- To list all the stashes that are present in the stash stack we use the command
  - **git stash list**



The screenshot shows a Mac OS X terminal window titled "git\_test -- zsh". The user runs "git stash list", which displays a single stash entry at index 0, labeled "stash@{0}: On master: print two in one.py".

```
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: print two in one.py
samireddynani@Nanis-Macbook git_test %
```

- To see the changes in a particular stash we use the command. We use index option to specify the stash index in the stash stack
  - **git stash show [index]**

```
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: print two in one.py
samireddynani@Nanis-Macbook git_test % git stash show 0
one.py | 1 +
1 file changed, 1 insertion(+)
samireddynani@Nanis-Macbook git_test %
```

- To apply back the changes from the stash to the current head/working directory we use the following command. NOTE: if we use apply it won't remove the stash from the stack
  - **git stash apply [index]**

```
samireddynani@Nanis-Macbook git_test % git status
On branch master
nothing to commit, working tree clean
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: print two in one.py
samireddynani@Nanis-Macbook git_test % git stash apply 0
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:   one.py

no changes added to commit (use "git add" and/or "git commit -a")
samireddynani@Nanis-Macbook git_test %
```

- To apply back the changes from the stash to the current head/working directory and also remove the stash from the stack we use the following command
  - **git stash pop [index]**

```
no changes added to commit (use "git add" and/or "git commit -a")
samireddynani@Nanis-Macbook git_test % git stash push -m" stash 2"
Saved working directory and index state On master: stash 2
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: stash 2
stash@{1}: On master: print two in one.py
samireddynani@Nanis-Macbook git_test % git stash pop 0
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:   one.py

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (a72652657d448a532e419f6880a485598f79413f)
samireddynani@Nanis-Macbook git_test %
```

- To remove a particular stash from the stack we use the command
  - **git stash drop [index]**

```
samireddynani@Nanis-Macbook git_test % git stash push -m"updated one.py added line 3"
Saved working directory and index state On master: updated one.py added line 3
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: updated one.py added line 3
stash@{1}: On master: print two in one.py
samireddynani@Nanis-Macbook git_test % git stash drop 1
Dropped refs/stash@{1} (ffb10c9c62fa6da4b0f68716b5b0d85179ceb92c)
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: updated one.py added line 3
samireddynani@Nanis-Macbook git_test %
```

- To remove all stashes from the stack we use the command
  - **git stash clear [index]**

```
samireddynani@Nanis-Macbook git_test % git stash drop 1
Dropped refs/stash@{1} (ffb10c9c62fa6da4b0f68716b5b0d85179ceb92c)
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: updated one.py added line 3
samireddynani@Nanis-Macbook git_test % git stash list
stash@{0}: On master: updated one.py added line 3
samireddynani@Nanis-Macbook git_test % git stash clear
samireddynani@Nanis-Macbook git_test % git stash list
samireddynani@Nanis-Macbook git_test %
```

**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.**

---

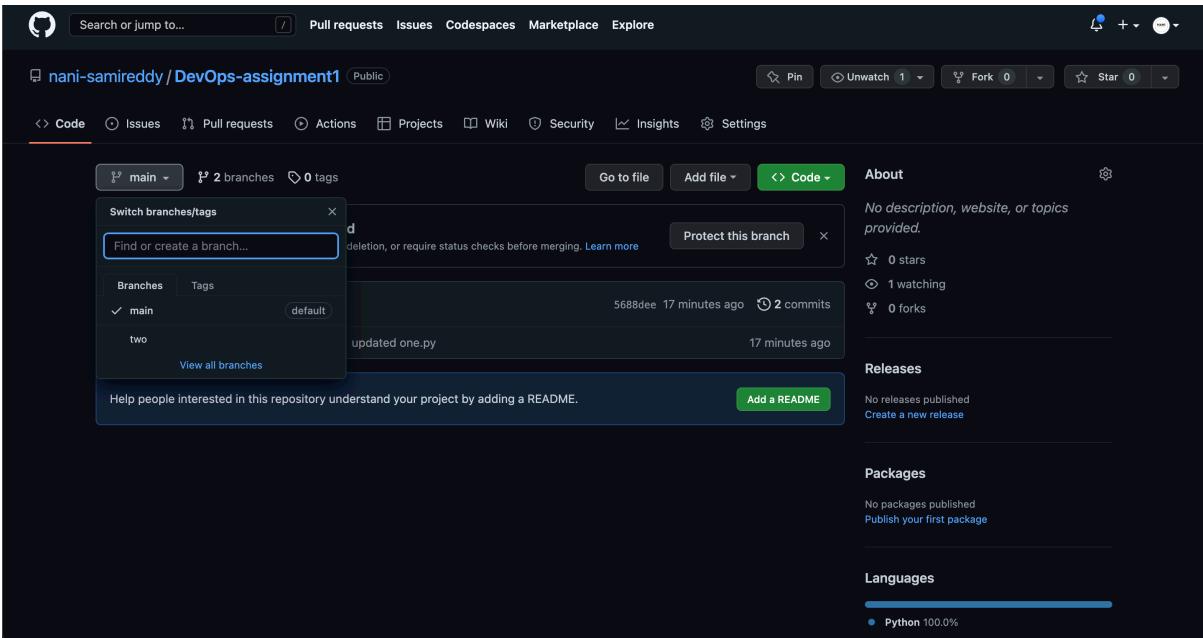
Before actually using fetch and merge first we have to know about the **git fetch** and **git merge** commands.

**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. This makes fetching a safe way to review commits before integrating them with your local repository.

**Git merge:** Incorporates changes from the named commits (since the time their histories diverged from the current branch) into the current branch.

To demonstrate the usage of the git fetch and git merge we are taking the following example we have a repository that has two branches and we will update the second branch in the remote(github) by making a commit there. So we'll fetch those updated commits using **git fetch** and then merges with the main branch in the local repo.

As we know that fetch only downloads but doesn't merge those changes to the local repo. We have to do it manually.



The screenshot shows a GitHub repository page for 'nani-samireddy / DevOps-assignment1'. The 'Code' tab is selected. A modal window titled 'Switch branches/tags' is open, showing the 'main' branch is the default. A commit for 'updated one.py' is visible in the main branch's history. The right sidebar displays repository statistics: 0 stars, 1 watching, 0 forks, and no releases or packages published. The 'Languages' section shows Python at 100%.

```
git checkout -b two
git fetch origin
git merge two
```

```
samireddy@Nanis-Macbook git-test % git branch
* main
  two
samireddy@Nanis-Macbook git-test % 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), 667 bytes | 333.00 KiB/s, done.
From github.com:nani-samireddy/DevOps-assignment1
  5688dee..31df64d main      -> origin/main
samireddy@Nanis-Macbook git-test % git log --oneline
5688dee (HEAD -> main) updated one.py
2f7c048 (origin/two, two) initial commit
samireddy@Nanis-Macbook git-test % git log --oneline --all
31df64d (origin/main) update one.py and added line three in one.py
5688dee (HEAD -> main) updated one.py
2f7c048 (origin/two, two) initial commit
samireddy@Nanis-Macbook git-test %
```

```
Terminal Shell Edit View Window Help git_test --zsh - 130x42
samireddynani@Nanis-Macbook git_test % git stash list
samireddynani@Nanis-Macbook git_test % git remote add origin git@github.com:nani-samireddy/DevOps-assignment1.git
samireddynani@Nanis-Macbook git_test % git branch -M main
samireddynani@Nanis-Macbook git_test % git push -u origin --all
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (6/6), 456 bytes | 456.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:nani-samireddy/DevOps-assignment1.git
 * [new branch]      main -> main
 * [new branch]      two -> two
branch 'main' set up to track 'origin/main'.
branch 'two' set up to track 'origin/two'.
samireddynani@Nanis-Macbook git_test % git branch
* main
  two
samireddynani@Nanis-Macbook git_test % 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), 667 bytes | 333.00 KiB/s, done.
From github.com:nani-samireddy/DevOps-assignment1
  5688dee..31df64d main      -> origin/main
samireddynani@Nanis-Macbook git_test % git log --oneline
5688dee (HEAD -> main) updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git log --oneline --all
31df64d (origin/main) update one.py and added line three in one.py
5688dee (HEAD -> main) updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git merge origin/main
Updating 5688dee..31df64d
Fast-forward
 one.py | 1 +
 1 file changed, 1 insertion(+)
samireddynani@Nanis-Macbook git_test % git log --oneline --all
31df64d (HEAD -> main, origin/main) update one.py and added line three in one.py
5688dee updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test %
```

### **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. This makes fetching a safe way to review commits before integrating them with your local repository.

**Git pull:** Incorporates changes from a remote repository into the current branch. If the current branch is behind the remote, then by default it will fast-forward the current branch to match the remote.

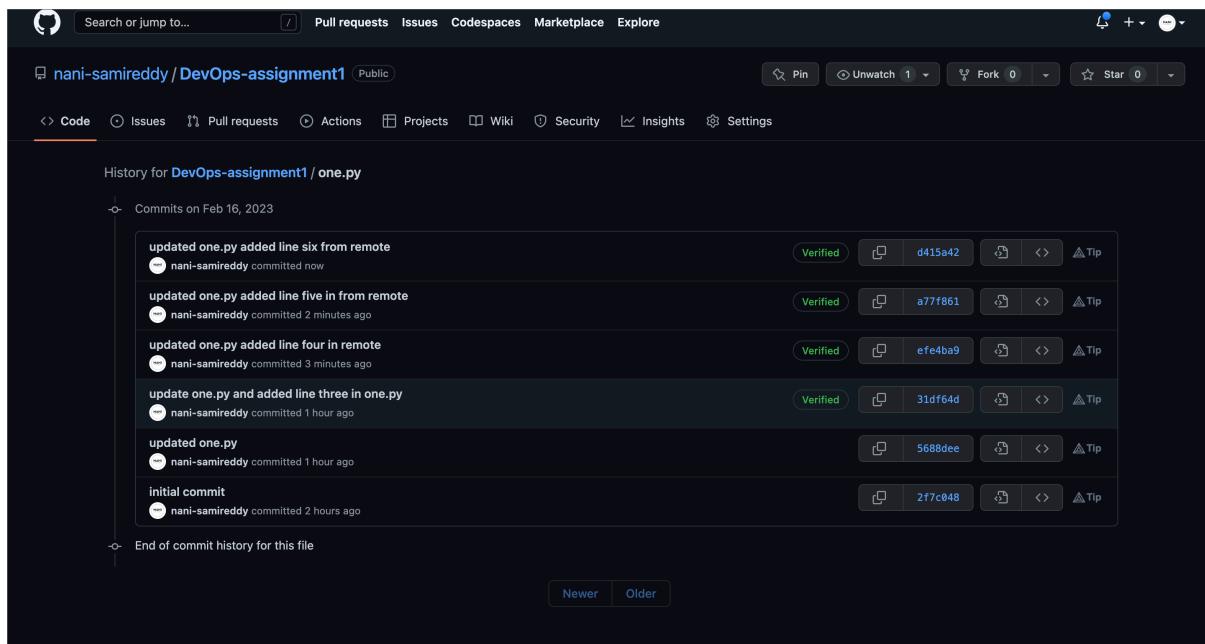
Git pull command pulls new changes or commits from a particular branch from your central repository and updates your target branch in your local repository.

Git fetch is also used for the same purpose but it works in a slightly different way. When you perform a git fetch, it pulls all new commits from the desired branch and stores it in a new branch in your local repository.

If you want to reflect these changes in your target branch, git fetch must be followed with a git merge.

our target branch will only be updated after merging the target branch and fetched branch.

**Git pull = git fetch + git merge**



The screenshot shows a GitHub repository page for 'nani-samireddy / DevOps-assignment1'. The 'Code' tab is selected. Below it, the commit history for the file 'one.py' is displayed. The commits are listed in reverse chronological order, starting with the most recent at the top. Each commit includes the author ('nani-samireddy'), the date ('committed now' or '2 hours ago'), the commit message, and several action buttons (Verified, copy, download, etc.).

Commit Details	Action Buttons
updated one.py added line six from remote nani-samireddy committed now	Verified, copy, d415a42, download, <>, Tip
updated one.py added line five in from remote nani-samireddy committed 2 minutes ago	Verified, copy, a77f861, download, <>, Tip
updated one.py added line four in remote nani-samireddy committed 3 minutes ago	Verified, copy, efe4ba9, download, <>, Tip
update one.py and added line three in one.py nani-samireddy committed 1 hour ago	Verified, copy, 31df64d, download, <>, Tip
updated one.py nani-samireddy committed 1 hour ago	copy, 5688dee, download, <>, Tip
initial commit nani-samireddy committed 2 hours ago	copy, 2f7c048, download, <>, Tip

```
Terminal Shell Edit View Window Help git_test --zsh - 130x42
From github.com:nani-samireddy/DevOps-assignment1
 5688dee..31df64d main      -> origin/main
samireddynani@Nanis-Macbook git_test % git log --oneline
5688dee (HEAD -> main) updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git log --oneline --all
31df64d (origin/main) update one.py and added line three in one.py
5688dee (HEAD -> main) updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git merge origin/main
Updating 5688dee..31df64d
Fast-forward
 one.py | 1 +
 1 file changed, 1 insertion(+)
samireddynani@Nanis-Macbook git_test % git log --oneline --all
31df64d (HEAD -> main, origin/main) update one.py and added line three in one.py
5688dee updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
samireddynani@Nanis-Macbook git_test % git log --oneline --all
31df64d (HEAD -> main, origin/main) update one.py and added line three in one.py
5688dee updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git fetch
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 1.28 KiB | 262.00 KiB/s, done.
From github.com:nani-samireddy/DevOps-assignment1
 31df64d..a77f861 main      -> origin/main
samireddynani@Nanis-Macbook git_test % git log --oneline --all
a77f861 (origin/main) updated one.py added line five in from remote
efe4ba9 updated one.py added line four in remote
31df64d (HEAD -> main) update one.py and added line three in one.py
5688dee updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test %
```

```
Terminal Shell Edit View Window Help git_test --zsh - 130x42
Thu 16 Feb 10:54 PM
5688dee updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git fetch
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 1.28 KiB | 262.00 KiB/s, done.
From github.com:nani-samireddy/DevOps-assignment1
 31df64d..a77f861 main      -> origin/main
samireddynani@Nanis-Macbook git_test % git log --oneline --all
a77f861 (origin/main) updated one.py added line five in from remote
efe4ba9 updated one.py added line four in remote
31df64d (HEAD -> main) update one.py and added line three in one.py
5688dee updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git log --oneline --all
a77f861 (origin/main) updated one.py added line five in from remote
efe4ba9 updated one.py added line four in remote
31df64d (HEAD -> main) update one.py and added line three in one.py
5688dee updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test % git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 680 bytes | 226.00 KiB/s, done.
From github.com:nani-samireddy/DevOps-assignment1
 a77f861..d415a42 main      -> origin/main
Updating 31df64d..d415a42
Fast-forward
 one.py | 3 +++
 1 file changed, 3 insertions(+)
samireddynani@Nanis-Macbook git_test % git log --oneline --all
d415a42 (HEAD -> main, origin/main) updated one.py added line six from remote
a77f861 updated one.py added line five in from remote
efe4ba9 updated one.py added line four in remote
31df64d update one.py and added line three in one.py
5688dee updated one.py
2f7c048 (origin/two, two) initial commit
samireddynani@Nanis-Macbook git_test %
```

**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 scripting language, and it is helpful when working in the command line. It's also a widely used command for text processing.

When using awk, you are able to select data – one or more pieces of individual text – based on a pattern you provide.

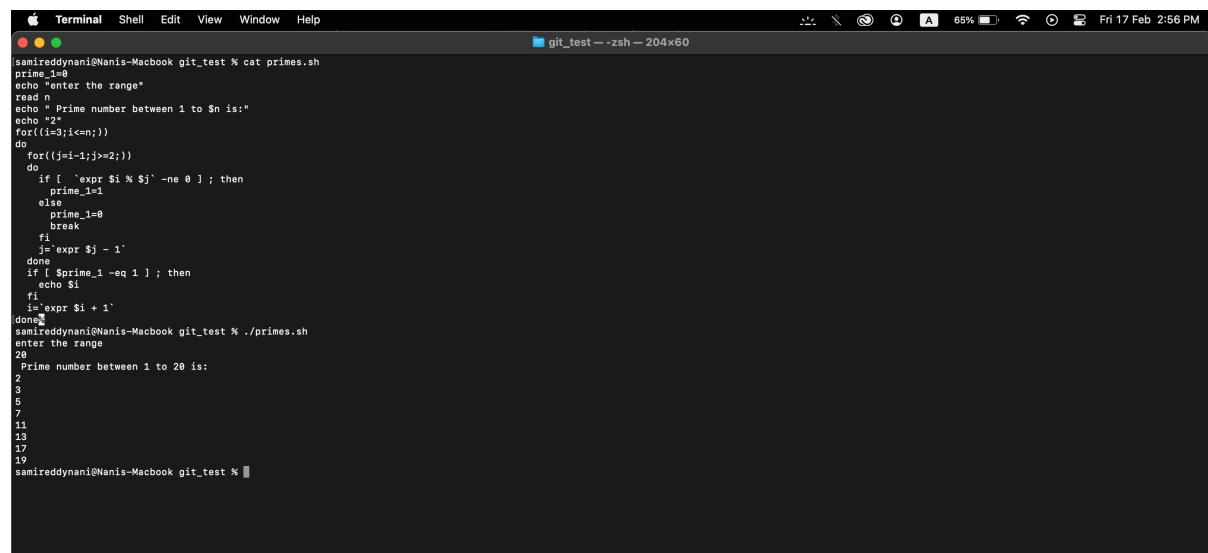
For example, some of the operations you can do with awk are searching for a specific word or pattern in a piece of text given, or even select a certain line or a certain column in a file you provide.

**Syntax: awk options 'selection \_criteria {action }' input-file > output-file**

Let's do an example by Create a file called data.txt as follows

```
samireddymani@Nanis-Macbook git_test % nano data.txt
samireddymani@Nanis-Macbook git_test % cat data.txt
ID      NAME    AGE
1       Nani    20
2       Nano    21
3       Nancy   22
4       Chinna  18
samireddymani@Nanis-Macbook git_test % awk '{print($0)}'
^C
samireddymani@Nanis-Macbook git_test % awk '{print($0)}' data.txt
ID      NAME    AGE
1       Nani    20
2       Nano    21
3       Nancy   22
4       Chinna  18
samireddymani@Nanis-Macbook git_test % awk '{print($2)}' data.txt
NAME
Nani
Nano
Nancy
Chinna
samireddymani@Nanis-Macbook git_test % awk '/c/ {print($2)}' data.txt
Nancy
samireddymani@Nanis-Macbook git_test % awk '/c/ {print($1,$2)}' data.txt
3 Nancy
samireddymani@Nanis-Macbook git_test %
```

### Shell program to find the primes numbers between two numbers(1,20)



The screenshot shows a Mac OS X terminal window titled "git\_test -- zsh -- 204x60". The window displays the following session:

```
Apple Terminal Shell Edit View Window Help
Fri 17 Feb 2:56 PM
samireddymani@Nanis-Macbook git_test %
samireddymani@Nanis-Macbook git_test % cat primes.sh
prime_1=0
echo "enter the range"
read n
echo "Prime number between 1 to $n is:"
echo "#"
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
samireddymani@Nanis-Macbook git_test %
enter the range
28
Prime number between 1 to 20 is:
2
3
5
7
11
13
17
19
samireddymani@Nanis-Macbook git_test %
```

**History command:** history command keeps a list of all the other commands that have been run from that terminal session, then allows you to replay or reuse those commands instead of retyping them.

-> To display all the history of commands we use

```
samireddynani@Nanis-Macbook ~ % history
999 docker rmi f423db7cc57
999 docker rmi 46331d942d63
1000 docker images
1001 bash "/Users/samireddynani/Desktop/git_test/primes.sh"
1002 bash "/Users/samireddynani/Desktop/git_test/primes.sh"
1003 bash "/Users/samireddynani/Desktop/git_test/primes.sh"
1004 bash "/Users/samireddynani/Desktop/git_test/primes.sh"
1005 bash "/Users/samireddynani/Desktop/git_test/primes.sh"
1006 bash "/Users/samireddynani/Desktop/git_test/primes.sh"
1007 ng serve
1008 git add .
1009 git push
1010 history
1011 history | grep ssh
1012 history | grep docker
1013 clear
```

-> To display most n recent commands of the history of commands we use

```
samireddynani@Nanis-Macbook ~ % history 5
5 git commit -a -m "created new repository with combined folders"
6 git commit -a -m "created new repository with combined folders"
7 git push
8 cd front-end
9 npm start
10 cd Desktop
```

-> To remove the complete history of executed command, run the below command:

```
history: bad option: -c
samireddynani@Nanis-Macbook ~ % history c
1013 clear
1014 history
1015 history -n 5
1016 history -n 5
1017 history -d 100
1018 history -d 1
1019 history -d 1
1020 history -c
```

## 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 containers are the light weight environments that contains everything that our applications needs to run.

This time we want to setup a container of Ubuntu OS. So now we have to pull the Ubuntu Container Base Image from the Docker hub using the command:

**docker pull ubuntu:latest**

```
samireddynani@Nanis-Macbook ~ % docker pull ubuntu:latest
latest: Pulling from library/ubuntu
8b150fd943bc: Pull complete
Digest: sha256:9a0bbde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbbe7f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
samireddynani@Nanis-Macbook ~ %
```

Now we got our Container image and we have to run to initiate the container for that we use the command

**docker run -ti ubuntu /bin/bash**

```
 samireddynani@Nanis-Macbook ~ % docker run -ti ubuntu /bin/bash
root@1ccdb2bc9d0a:/# ls
bin  boot  dev  etc  home  lib  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@1ccdb2bc9d0a:/# pwd
/
root@1ccdb2bc9d0a:/# cd home/
root@1ccdb2bc9d0a:/home# ls
root@1ccdb2bc9d0a:/home#
```

As we can see in the above screenshot we are running the in the above command the options -ti Implies to the interactive terminal mode that allows us to interact with Ubuntu.