

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.

Stash command: stashing your changes is a great way to keep up with your current work without committing them to working branch. This allows you to work between several branches without pushing any changes.

Here the stash will use stack data structure. It stores item like stack.

You can stash your changes by running a simple command.

I have a created 4 files and added and committed them. Now I have modified the content in the file and checked the status using git status.

```
$ vi test.py

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ 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:   test.py

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

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
```

Now add the file which we modified using the command “git add <filename>”.

Now click git status again.

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git add test.py
warning: in the working copy of 'test.py', LF will be replaced by CRLF the next
time Git touches it

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py
```

Now let's add this to the stash using the command “git stash push -m <your name>”.

Now to view the details of the stash you made execute the following command “git stash list” this command is useful when we have multiple stashes.

It shows them like a list.

Now modify the same file and add the file and stash repeat the same for three stashes.

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git stash list
stash@{0}: On master: stash 3
stash@{1}: On master: stash 2
stash@{2}: On master: stash 1
```

To retrieve the stashed changes to keep a copy of your changes in the stash and also bring them over to your working branch use the command git apply.

Now apply the any of the stash. We can apply any stash using the command “git stash show --index <index number>”.

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git stash apply --index 2
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   test.py

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
```

It showed what we have done in the stash 2.

We can delete a particular stash using pop command. After applying a particular stash delete it using pop function.

Use the following command “git stash pop”.

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git stash list
stash@{0}: On master: stash 2
stash@{1}: On master: stash 1
```

Git stash drop is a command that deletes a stash from the queue. By default git stash drop deletes the most recent stash but we can also specify the index to delete a particular stash. Use the following command “git stash drop”.

```
$ git stash list
stash@{0}: On master: stash 2
stash@{1}: On master: stash 1

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git stash drop
Dropped refs/stash@{0} (fbf2159d61700e0a8f01db08c62e56030734072e)

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git stash list
stash@{0}: On master: stash 1

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ |
```

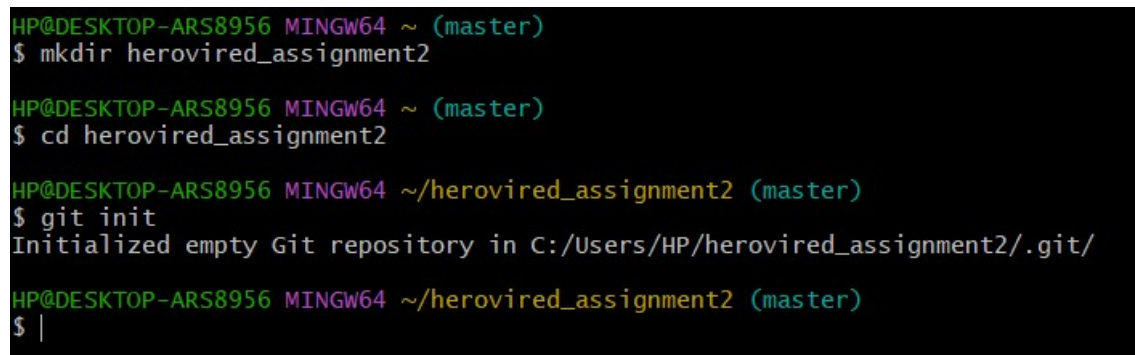
Q2. By using a sample example of your choice, use the git fetch command and also use the git merge command and describe the whole process through a screenshot with all the commands and their output in git bash.

git fetch command retrieves commits, files, and tags from a remote repository. The general syntax for command is: “git fetch <options> <remote name> <branch name>”. Git fetch command downloads commits, files, and refs from a remote repository into your local repo.

Open the terminal and create a directory for the project using the command “mkdir <directory name>”.

Enter the directory using the cd command “cd <directory name>”.

Initialize the local repository with the command “git init”.



```
HP@DESKTOP-ARS8956 MINGW64 ~ (master)
$ mkdir herovired_assignment2

HP@DESKTOP-ARS8956 MINGW64 ~ (master)
$ cd herovired_assignment2

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment2 (master)
$ git init
Initialized empty Git repository in C:/Users/HP/herovired_assignment2/.git/

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment2 (master)
$ |
```

Use the following command to add a remote URL to the local repo “git remote add <short remote name> <remote URL>”

For example: git remote add origin [git@github.com:jjdhfhjd-KB/text.git](https://github.com/jjdhfhjd-KB/text.git)

To confirm the remote added successfully “git remote -v”.



```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment2 (master)
$ git remote add origin https://github.com/KAVYA-SRI-TIPPANI/KAVYA-SRI-TIPPANI.github.io.git

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment2 (master)
$ git remote -v
origin https://github.com/KAVYA-SRI-TIPPANI/KAVYA-SRI-TIPPANI.github.io.git (fetch)
origin https://github.com/KAVYA-SRI-TIPPANI/KAVYA-SRI-TIPPANI.github.io.git (push)

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment2 (master)
$ |
```

To fetch a remote repository git fetch is the command “git fetch <remote name>”.

For example “git fetch origin”.

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment2 (master)
$ git fetch origin
remote: Enumerating objects: 17, done.
remote: Total 17 (delta 0), reused 0 (delta 0), pack-reused 17
Unpacking objects: 100% (17/17), 12.03 KiB | 68.00 KiB/s, done.
From https://github.com/KAVYA-SRI-TIPPANI/KAVYA-SRI-TIPPANI.github.io
* [new branch]      main      -> origin/main

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment2 (master)
$ |
```

The output shows the command fetched all the contents from the remote repository, including branches and tags.

List all the fetched remote branches with “git branch -r”.

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment2 (master)
$ git branch -r
origin/main
```

Git merge command:

The concept of git merging is basically to merge multiple sequences of commits, stored in multiple branches in a unified history or to be simple you can say in a single branch.

The command for git merge is “git merge <query>”.

Git allows merging the whole branch in another branch. Suppose you have made many changes on a branch and want to merge all of that at a time. Git allows you to do so.

Now create a branch with any name with the command “git branch <branchname>”.

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git branch assignment

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git branch --list
assignment
* master
```

Now to switch to newly created branch and create a file and commit it .

```

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git checkout assignment
Switched to branch 'assignment'
D      test.py

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (assignment)
$ vi hello.py

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (assignment)
$ git commit -m "committing in the first branch"
[assignment dc1e549] committing in the first branch
1 file changed, 1 deletion(-)
delete mode 100644 test.py

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (assignment)
$ |

```

Now switch back to master branch and give the merge command “git merge <branchname>”

```

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (assignment)
$ git switch master
Switched to branch 'master'

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ git merge assignment
Updating fbbc1bb..dc1e549
Fast-forward
 test.py | 1 -
1 file changed, 1 deletion(-)
delete mode 100644 test.py

HP@DESKTOP-ARS8956 MINGW64 ~/herovired_assignment (master)
$ |

```


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 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.

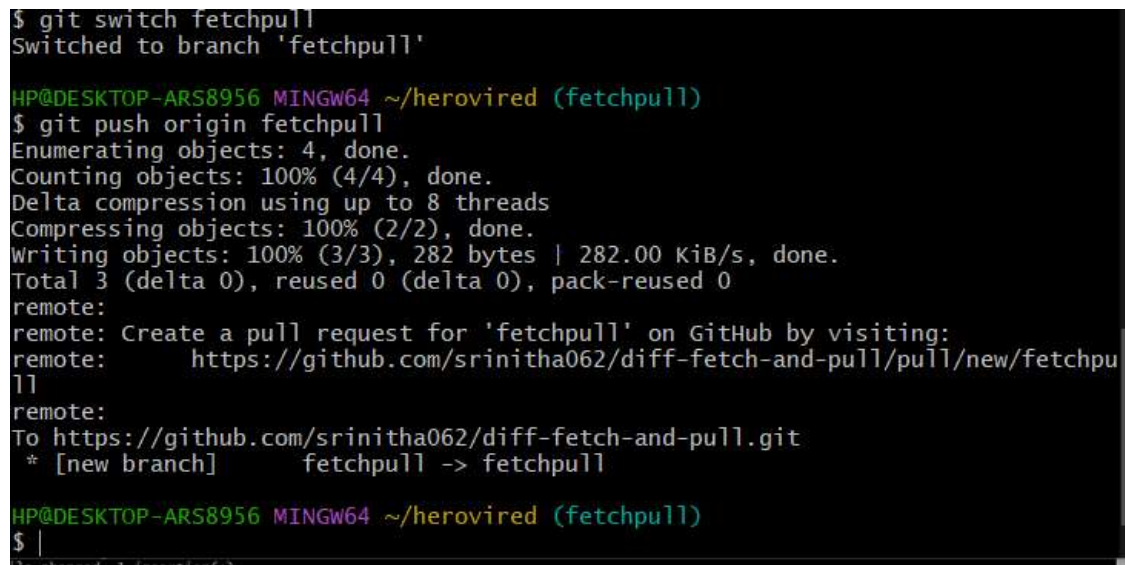
Git push:

Make a new directory and run the command git init and create a file and commit it in local repository.

Run the following commands

Git remote add origin <link to your repository>

Git push origin.

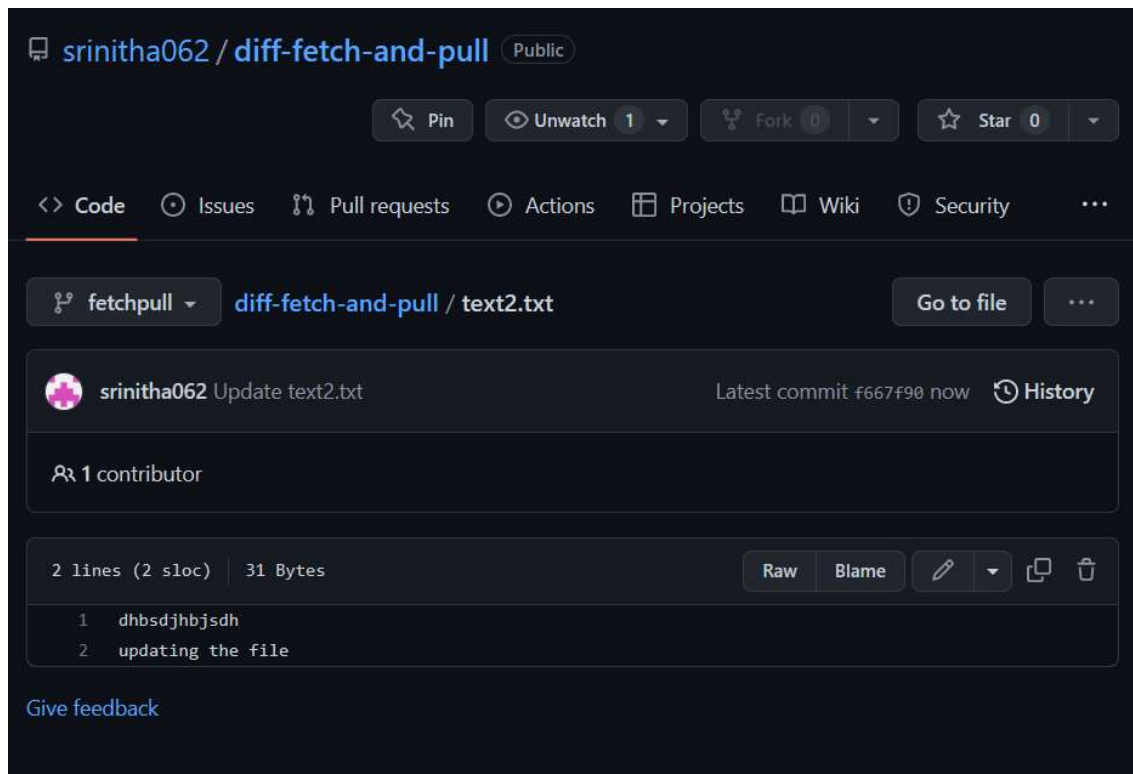


```
$ git switch fetchpull
Switched to branch 'fetchpull'

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (fetchpull)
$ git push origin fetchpull
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 282 bytes | 282.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'fetchpull' on GitHub by visiting:
remote:   https://github.com/srinitha062/diff-fetch-and-pull/pull/new/fetchpu
ll
remote:
To https://github.com/srinitha062/diff-fetch-and-pull.git
 * [new branch]      fetchpull -> fetchpull

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (fetchpull)
$ |
```

Now change the content in text1.txt in remote repository. And commit it.



Now using git fetch

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ git merge origin/fetchpull
Updating 9b61a97..f667f90
Fast-forward
 text2.txt | 1 +
 1 file changed, 1 insertion(+)

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ git log
commit f667f90732e36e3a9456e99ef868922cdea42f33 (HEAD -> master, origin/fetchpull)
Author: srinitha062 <84458794+srinitha062@users.noreply.github.com>
Date:   Fri Feb 17 20:55:39 2023 +0530

    Update text2.txt

commit 9b61a977c9a8d07e5bef0bc4240ebe7fe33d8048 (fetchpull)
Author: srinitha <velechetsrinitha@gmail.com>
Date:   Fri Feb 17 20:48:13 2023 +0530

    helloworld

commit f1372f70e95aeaa1e436bacd79134b881d48d8c2
Author: srinitha <velechetsrinitha@gmail.com>
Date:   Fri Feb 17 20:38:42 2023 +0530

    first commit
```

Now lets use the git pull command:

Now again make changes to the file In remote repository and commit it there.

srinitha062 / diff-fetch-and-pull Public

Pin Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security

fetchpull diff-fetch-and-pull / text2.txt Go to file ...

srinitha062 Update text2.txt Latest commit 86bca65 now History

1 contributor

3 lines (3 sloc) 69 Bytes Raw Blame

```

1 dhbsdjhbjsdh
2 updating the file
3 updating the file for the second time

```

[Give feedback](#)

```

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ git pull origin fetchpull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 700 bytes | 21.00 KiB/s, done.
From https://github.com/srinitha062/diff-fetch-and-pull
 * branch          fetchpull    -> FETCH_HEAD
  f667f90..86bca65  fetchpull    -> origin/fetchpull
Updating f667f90..86bca65
Fast-forward
 text2.txt | 1 +
 1 file changed, 1 insertion(+)

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ git log
commit 86bca6515e8086f323401b9ca127fa3522ec4c15 (HEAD -> master, origin/fetchpull)
Author: srinitha062 <84458794+srinitha062@users.noreply.github.com>
Date:   Fri Feb 17 21:04:13 2023 +0530

    Update text2.txt

commit f667f90732e36e3a9456e99ef868922cdea42f33
Author: srinitha062 <84458794+srinitha062@users.noreply.github.com>
Date:   Fri Feb 17 20:55:39 2023 +0530

```



```
commit f667f90732e36e3a9456e99ef868922cdea42f33
Author: srinitha062 <84458794+srinitha062@users.noreply.github.com>
Date:   Fri Feb 17 20:55:39 2023 +0530

    Update text2.txt

commit 9b61a977c9a8d07e5bef0bc4240ebe7fe33d8048 (fetchpull)
Author: srinitha <velechetsrinitha@gmail.com>
Date:   Fri Feb 17 20:48:13 2023 +0530

    helloworld

commit f1372f70e95aeaale436bacd79134b881d48d8c2
Author: srinitha <velechetsrinitha@gmail.com>
Date:   Fri Feb 17 20:38:42 2023 +0530

    first commit

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$
```

Here pull command is the combination of both git fetch and git merge

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.

The awk is a powerful scripting language used for text scripting. It searches and replaces the texts and sorts, validates, and indexes the database.

It is one of the most widely used tools for the programmer, as they write the scaled-down effective program in the form of a statement to define the next patterns and designs.

The syntax is awk options 'selection _criteria {action }' input-file > output-file.

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ cat friends.txt
srinitha v
kavya t
sushma b
pushpa y
rosy t
ramya p
iswarya t
sahasra k
blessy ch

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ |
```

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ awk '/t/{print}' friends.txt
srinitha v
kavya t
rosy t
iswarya t

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ |
```

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ awk '{print}' friends.txt
srinitha v
kavya t
sushma b
pushpa y
rosy t
ramya p
iswarya t
sahasra k
blessy ch

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ |
```

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ awk '{print $1,$5}' friends.txt
srinitha
kavya
sushma
pushpa
rosy
ramya
iswarya
sahasra
blessy

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ !
```

Prime number in shell scripting:

```
#!/bin/bash

echo "Prime numbers in the range of 1 to 20 are:"

for num in {1..20}; do
    prime=true
    for (( i=2; i<$num; i++ )); do
        if (( $num % $i == 0 )); then
            prime=false
            break
        fi
    done
    if [ $prime == true ]; then
        echo $num
    fi
done
```

Output:

```
HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
$ sh prime.sh
Prime numbers in the range of 1 to 20 are:
1
2
3
5
7
11
13
17
19

HP@DESKTOP-ARS8956 MINGW64 ~/herovired (master)
```

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.

Docker: docker Is an open-source platform designed to create, deploy, and run applications. Docker uses container on the host's operating system to run applications. It allows applications to use the same linux kernel as a system on the host computer, rather than creating a whole virtual operating system. Containers ensure that our applications works in any environment.

Ubuntu: Ubuntu is a popular free and open-source Linux-based operating system you can use on a computer virtual private server. It has user friendliness, strong security, more software options, enhanced privacy, lightweight performance.

```
C:\Users\HP>docker pull ubuntu
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\HP>docker run -it ubuntu
```

```
C:\Users\HP>docker run -it ubuntu
```

Now it runs the Ubuntu server.

Now it opens the terminal. And in the terminal run the following command “apt update” it will install all the required packages.

X

```
root@16247235c9bd:/# apt update
Get:1 http://archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [752 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [5557 B]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [807 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [860 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [266 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1136 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [10.9 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1091 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [808 kB]
Get:17 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [22.4 kB]
Get:18 http://archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [49.0 kB]
Fetched 25.8 MB in 16s (1654 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

Now if you open docker desktop it shows exited .

Containers

[Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

☐ Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	Started	Actions
<input type="checkbox"/>	competent_banach 16247235c9bd	ubuntu	Exited (130)			<div><div></div><div></div><div></div></div>