

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.

1.Git Stash Command:Git stash is a built-in command with the distributed Version control tool in Git that locally stores all the most recent changes in a workspace and resets the state of the workspace to the prior commit state. A user can retrieve all files put into the stash with the git stash pop and git stash apply commands.

Git stash command saves the previously written code and then goes back to the last commit for a fresh start.Now you can add the new feature without disturbing the old one as it is saved locally.After committing the new feature you can go on working with the old one which was incomplete and not committed.

```
MINGW64:/c/Users/JYOTHI
JYOTHI@DESKTOP-84J7GEV MINGW64 ~
$ git config --global user.name "jyothikatakam396"
JYOTHI@DESKTOP-84J7GEV MINGW64 ~
$ git config --global user.email "20a91a05f5@aec.edu.in"
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~
$ git clone "https://github.com/jyothikatakam396/hero-vired.git"
Cloning into 'hero-vired'...
warning: You appear to have cloned an empty repository.
JYOTHI@DESKTOP-84J7GEV MINGW64 ~
$ cd hero-vired
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ vi file1.txt
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git status
On branch main
No commits yet
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    file1.txt
nothing added to commit but untracked files present (use "git add" to track)
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git commit -m "committed"
[main (root-commit) f4004f6] committed
1 file changed, 3 insertions(+)
create mode 100644 file1.txt
```


Now changes are removed.

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it
Saved working directory and index state WIP on main: f4004f6 committed
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
```

```
MINGW64:/c/Users/JYOTHI/hero-vired
Git stash
it stores changes in the stash stack
~
~
~
~
~
```

The file is now stashed and it is under untracked state.

By default, running git stash will stash the changes that have been added to your index(staged changes) and unstages changes. To stash your untracked files, use git stash -u.

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ vi file1.txt

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git status
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)

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.txt

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

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash
Saved working directory and index state WIP on main: f4004f6 committed

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ cat file1.txt
Git stash
it stores changes in the stash stack
```

2.Listing stashesYou can create multiple slashes and view them using git stash list command.

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash list
stash@{0}: WIP on main: f4004f6 committed
stash@{1}: WIP on main: f4004f6 committed

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$
```

3.Providing additional message:

To provide more context to the stash we create the stash using the following command.

```
git stash save "message"
```

4.Getting back stashed changes:

You can reapply the previously stashed changes with the 'git stash pop' or 'git stash apply' command.

- 1.'git stash pop' removes the changes from stash and reapplies the changes in working copy,
- 2.'git stash apply' do not remove changes .but reapplies the changes in working copy.

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash pop
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)

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.txt

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (7bcf6090a78cd7afe29c918edad4d5d096db7781)
```

Now check whether stash is removed or not.

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash list
stash@{0}: WIP on main: f4004f6 committed
```

By using "git stash apply" We got the previous uncommitted changes.

```

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash apply
error: Your local changes to the following files would be overwritten by merge:
       file1.txt
Please commit your changes or stash them before you merge.
Aborting
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
    (use "git branch --unset-upstream" to fixup)

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.txt

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

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ cat file1.txt
Git stash
git stores changes in the stash stack
It stores changes securely

```

5.To view the stash summary:

Git stash show is used to view the summary

```

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash show
file1.txt | 5 +++--
1 file changed, 3 insertions(+), 2 deletions(-)

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$

```

6.Deleting stashes:

To delete a particular stash:

```
git stash drop stash@{1}
```

To delete all stashes at once,use the below command

```
git stash clear
```

```

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash clear

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git stash list

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$

```

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.

Fetch just downloads the objects and refs from a remote repository and normally updates the remote tracking branches. Pull, however, will not only download the changes, but also merges them - it is the combination of fetch and merge (cf. the section called "Merging"). The configured remote tracking branch is selected automatically.

Git pull=git fetch+git merge

git fetch is the command that tells your local git to retrieve the latest meta-data info from the original (yet doesn't do any file transferring. It's more like just checking to see if there are any changes available).

The "git merge" command. The git merge command is used to merge the branches. The syntax for the git merge command is as: \$ git merge <query>

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (master)
$ git config --global user.name "jyothikatakam396"
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (master)
$ git config --global user.email "20a91a05f5@aec.edu.in"
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (master)
$ git clone https://github.com/jyothikatakam396/hero-vired.git
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (master)
$ git clone https://github.com/jyothikatakam396/hero-vired
.git
Cloning into 'hero-vired'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 12 (delta 0), reused 0 (delta 0), pack-reuse
d 0
Receiving objects: 100% (12/12), done.
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (m
aster)
$ cd hero-vired
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/he
ro-vired (main)
$ vim first
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/he
ro-vired (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
```

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

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

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/he
ro-vired (main)
$ git add .
warning: in the working copy of 'first', LF will be replac
ed by CRLF the next time Git touches it
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/he
ro-vired (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.
```

```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   first
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/he
ro-vired (main)
$ git commit -m "test"
[main 22db8ce] test
1 file changed, 1 insertion(+)
```


Git Fetch

git fetch is the command that tells your local git to retrieve the latest meta-data info from the original (yet doesn't do any file transferring. It's more like just checking to see if there are any changes available).

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~  
$ cd hero-vired  
  
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)  
$ git fetch  
remote: Enumerating objects: 4, done.  
remote: Counting objects: 100% (4/4), done.  
remote: Compressing objects: 100% (2/2), done.  
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0  
Unpacking objects: 100% (3/3), 293 bytes | 22.00 KiB/s, done.  
From https://github.com/jyothikatakam396/hero-vired  
   c06822a..22db8ce  main       -> origin/main  
  
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)  
$ git log  
commit 315738deff9a0735d792861644a691e7d48489a9 (HEAD -> main)  
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>  
Date:   Thu Feb 16 22:31:15 2023 +0530  
  
    test  
  
commit 6b8126a68423c9b17dff6facdb561fe6fb1a9148  
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>  
Date:   Thu Feb 16 22:23:50 2023 +0530  
  
    committed1  
  
commit 94c403f41b620276057c1482f7019219f2127762  
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>  
Date:   Thu Feb 16 21:45:21 2023 +0530  
  
    Committed  
  
commit 782c7ad809240ddb7e3c95478b1e270db6d74f6e  
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>  
Date:   Thu Feb 16 21:15:34 2023 +0530  
  
    committed  
  
commit f4004f6b8ce544a566bf7640236bfce47deb6ba9  
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>  
Date:   Tue Feb 14 12:02:17 2023 +0530  
  
    committed
```


Git Merge

The "git merge" command. The git merge command is used to merge the branches. The syntax for the git merge command is as: \$ git merge <query>

```
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/merge (main)
$ git merge b12
Updating 6090477..0e70627
Fast-forward
 1 | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 1

JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/merge (main)
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/he
ro-vired (main)
$ git merge origin/main
Already up to date.
```

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

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (master)
$ git config --global user.name "jyothikatakam396"
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (m
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (master)
$ git config --global user.email "20a91a05f5@aec.edu.in"
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (master)
$ git clone https://github.com/jyothikatakam396/hero-vired.git
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text (m
aster)
$ git clone https://github.com/jyothikatakam396/hero-vired
.git
Cloning into 'hero-vired'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 12 (delta 0), reused 0 (delta 0), pack-reuse
d 0
Receiving objects: 100% (12/12), done.
```

Git fetch: git fetch is the command that tells your local git to retrieve the latest meta-data info from the original (yet doesn't do any file transferring. It's more like just checking to see if there are any changes available).

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~
$ cd hero-vired
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git fetch
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 293 bytes | 22.00 KiB/s, done.
From https://github.com/jyothikatakam396/hero-vired
   c06822a..22db8ce  main       -> origin/main
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ git log
commit 315738deff9a0735d792861644a691e7d48489a9 (HEAD -> main)
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 22:31:15 2023 +0530

    test

commit 6b8126a68423c9b17dff6facdb561fe6fb1a9148
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 22:23:50 2023 +0530

    committed1

commit 94c403f41b620276057c1482f7019219f2127762
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 21:45:21 2023 +0530

    Committed

commit 782c7ad809240ddb7e3c95478b1e270db6d74f6e
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 21:15:34 2023 +0530

    committed

commit f4004f6b8ce544a566bf7640236bfce47deb6ba9
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Tue Feb 14 12:02:17 2023 +0530

    committed
```

Git Push

The git push command is **used to upload local repository content to a remote repository**. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to git fetch, but whereas fetching imports commits to local branches, pushing exports commits to remote branches.

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/hero-vired (main)
$ git push
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), 313 bytes | 313.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/jyothikatakam396/hero-vired.git
   c06822a..22db8ce  main -> main
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/hero-vired (main)
$ git log
commit 22db8ce5e1b5e8f6d3525e11b52107266dd62422 (HEAD -> main, origin/main, origin/HEAD)
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 23:01:38 2023 +0530
```

test

```
commit c06822a006106172c8556d9ac342e02b9ecb3218
Author: jyothikatakam396 <84368383+jyothikatakam396@users.noreply.github.com>
Date: Thu Feb 16 22:05:34 2023 +0530
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 /d/git practice/git_text/hero-vired (main)
$ git log
commit 22db8ce5e1b5e8f6d3525e11b52107266dd62422 (HEAD -> main, origin/main, origin/HEAD)
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 23:01:38 2023 +0530
```

test

Git pull: The `git pull` command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows. The `git pull` command is actually a combination of two other commands, `git fetch` followed by `git merge`. In the first stage of operation `git pull` will execute a `git fetch` scoped to the local branch that `HEAD` is pointed at. Once the content is downloaded, `git pull` will enter a merge workflow. A new merge commit will be created and `HEAD` updated to point at the new commit.

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (m1)
$ git pull
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 7 (delta 1), reused 7 (delta 1), pack-reused 0
Unpacking objects: 100% (7/7), 571 bytes | 35.00 KiB/s, done.
From https://github.com/jyothikatakam396/hero-vired
  22db8ce..6f40182  main      -> origin/main
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=origin/<branch> m1
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (m1)
$ git log
commit 315738deff9a0735d792861644a691e7d48489a9 (HEAD -> m1, main)
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 22:31:15 2023 +0530

    test

commit 6b8126a68423c9b17dff6facdb561fe6fb1a9148
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 22:23:50 2023 +0530

    committed1

commit 94c403f41b620276057c1482f7019219f2127762
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 21:45:21 2023 +0530

    Committed

commit 782c7ad809240ddb7e3c95478b1e270db6d74f6e
Author: jyothikatakam396 <20a91a05f5@aec.edu.in>
Date: Thu Feb 16 21:15:34 2023 +0530

    committed
```

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:

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 performs various actions on a file like searching a specified text and more.

The awk command is a Linux tool and programming language that allows users to process and manipulate data and produce formatted reports. The tool supports various operations for advanced text processing and facilitates expressing complex data selections.

 MINGW64:/c/Users/JYOTHI/hero-vired

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~  
$ cd hero-vired  
  
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)  
$ awk '{ print "printing"}'  
  
printing  
  
printing
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)  
$ vi s1  
  
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)  
$ aws '{print}' s1  
bash: aws: command not found  
  
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)  
$ awk '{print}' s1  
Name Dept  
jyo cse  
maha cse  
siri cse  
Bhanu It
```

Using Awk command

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ awk '{print}' s1
Name Dept
jyo cse
maha cse
siri cse
Bhanu It

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ awk '/cse/ {print}' s1
jyo cse
maha cse
siri cse

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ awk '{print $1}' s1
Name
jyo
maha
siri
Bhanu

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ awk '{print $NF}' s1
Dept
cse
cse
cse
It
```

Steps to follow bash scripting:

Step1) create the file with extension .sh.

Step 2)open the shell and write the script.

Step 3)save the code and run the code .

To run the run a code

Syntax: bash filename.sh

```
MINGW64:/c/Users/JYOTHI/hero-vired
for((i=2;i<=20;))
do
    for((j=i-1;j>=2;))
    do
        if [ `expr $i % $j` -ne 0 ] ; then
            prime=1
        else
            prime=0
            break
        fi
        j=`expr $j - 1`
    done
    if [ $prime -eq 1 ] ; then
        echo $i
    fi
    i=`expr $i + 1`
done
~
~
~
~
prime.sh[+] [unix] (05:29 01/01/1970) 17,5 All
-- INSERT --
```

```
JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ vi prime.sh

JYOTHI@DESKTOP-84J7GEV MINGW64 ~/hero-vired (main)
$ bash prime.sh
prime.sh: line 13: [: -eq: unary operator expected
3
5
7
11
13
17
19
```


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.

Image: Images are used to create containers. It uses a private container registry to share container images within the enterprise and also use public container registry to share container images with whole world.

Container: Containers are used to hold the entire package that is needed to run the application. We can say that the image is a template and the container is a copy of the template.

*These are the containers present in the docker desktop.

Steps to follow

*For setting up a container and run the ubuntu os,

→First we need to download the image of Ubuntu from docker hub using the command `docker pull ubuntu`.

→To create a container and execute the image use the command `docker run -it ubuntu`.

→To get an idea about the available update use `apt update` command.

**Download the ubuntu OS image from the docker hub.

The screenshot is divided into two main sections. The top section is a Windows Command Prompt window titled 'Command Prompt'. It shows the execution of the following commands and their outputs:

```
Microsoft Windows [Version 10.0.22000.1455]
(c) Microsoft Corporation. All rights reserved.

C:\Users\JY0THI>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
677076032cca: Downloading [=====>] 677076032cca: Pull complete
Digest: sha256:9a0bdde4188b896a372804be2384015e90e3f84906b750c1a53539b585fbb7f
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest

C:\Users\JY0THI>docker run -it ubuntu
root@4235679d74e2:/#
root@4235679d74e2:/# exit
exit

C:\Users\JY0THI>
```

The bottom section is the Docker Desktop application interface. The left sidebar shows navigation options: Containers, Images, Volumes, Dev Environments (BETA), Extensions, and Add Extensions. The main panel is titled 'Containers' and includes a toggle for 'Only show running containers' and a search bar. Below this is a table listing the containers:























	Name	Image	Status	Port(s)	Started	Actions
<input type="checkbox"/>	cool_mendel 4235679d74e2	ubuntu	Exited			
<input type="checkbox"/>	eager_haslett fa4b84c02a76	hello-world	Exited			
<input type="checkbox"/>	youthful_elbakyan f228c2724996	hello-world	Exited			
<input type="checkbox"/>	epic_elion cbc887a9a0e1	hello-world	Exited			

```
C:\Users\JY0THI>docker run -it ubuntu
root@65696ec551d9:/# 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/multiverse amd64 Packages [266 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [17.5 MB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [5557 B]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [807 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [860 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1792 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/restricted amd64 Packages [164 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1091 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [808 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [10.9 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1136 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 29s (893 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
5 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@65696ec551d9:/#
```

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"/>	 cool_mendel 4235679d74e2 	ubuntu	Exited			  
<input type="checkbox"/>	 eager_haslett fa4b84c02a76 	hello-world	Exited			  
<input type="checkbox"/>	 youthful_elbakyan f228c2724996 	hello-world	Exited			  
<input type="checkbox"/>	 epic_elion cbc887a9a0e1 	hello-world	Exited			  
<input type="checkbox"/>	 lucid_noether 65696ec551d9 	ubuntu	Running		3 minutes ago 