

Docker Day –01

Setup docker in ubuntu: end to end docker process:

I have created one aws ec2 instance with instance type c7iflex.large --> availability zone -> eu-north-1b.

And connected to above server by git bash using below commands

```
ssh -i /c/Users/kasev/Downloads/aws_key_manju_mail.pem.pem ubuntu@16.171.36.240
```

Step 1: Install Docker on Ubuntu

Run these commands one by one in your EC2 terminal 

```
# Update your packages  
sudo apt update -y  
  
# Install Docker engine  
sudo apt install docker.io -y  
  
# Enable and start Docker service  
sudo systemctl enable docker  
sudo systemctl start docker  
  
# Check Docker version  
docker --version
```

 You should see output like:

```
Docker version 24.x.x, build ...
```

Step 2: Add your user to the Docker group

This lets you run docker commands without using sudo every time.

```
sudo usermod -aG docker $USER
```

Then **logout and login again**, or run:

```
newgrp docker
```

To verify:

```
docker ps
```

If it shows no errors → good to go 

Step 3: Create the Project

Create a simple directory for our app:

```
mkdir docker-flask-app  
cd docker-flask-app
```

Step 4: Create Flask App Files

1 Create app.py

```
nano app.py
```

Paste this code:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello():

    return "Hello Everyone, Thanks for watching this video!! please
like this video and subscribe my channel!!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

:wq!
```

```
ubuntu@ip-172-31-42-32:~/docker-flask-app$ cat app.py
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello():
    return " hello everyone, thanks for watching this video"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

2 Create requirements.txt

```
flask
```

3 Create Dockerfile

```
nano Dockerfile
```

Paste this:

```
FROM python:3.9-slim
WORKDIR /app
COPY . /app
RUN pip install -r requirements.txt
EXPOSE 8080
CMD ["python", "app.py"]
```

Save and exit.

Step 5: Build Docker Image

```
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker build -t flask-docker-app:1.0 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/
Sending build context to Docker daemon 4.096kB
Step 1/6 : FROM python:3.9-slim
--> 085da638e1b8
Step 2/6 : WORKDIR /app
--> Using cache
--> e9e1df6de85d
Step 3/6 : COPY . /app
--> Using cache
--> 9324c3892b32
Step 4/6 : RUN pip install -r requirements.txt
--> Using cache
--> 8b775014caea
Step 5/6 : EXPOSE 8080
--> Using cache
--> 97d85bed08f1
Step 6/6 : CMD ["python", "app.py"]
--> Using cache
--> 0e12cb0f7dc9
Successfully built 0e12cb0f7dc9
Successfully tagged flask-docker-app:1.0
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
flask-docker-app  1.0          0e12cb0f7dc9  16 minutes ago  133MB
<none>          <none>       03f8bf9ebddc  38 minutes ago  133MB
python           3.9-slim    085da638e1b8  2 days ago    122MB
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker run -d -p 8080:5000 flask-docker-app:1.0
docker: invalid reference format
Run 'docker run --help' for more information
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker run -d -p 8080:5000 flask-docker-app:1.0
928e1d9b30307ab6d28305dd3b0bbe91817206d24f125e88186748242f7795f0
ubuntu@ip-172-31-33-209:~/docker-flask-app$ |
```

docker build -t flask-docker-app:1.0 .

Step 6: Verify the Image

```
docker images
```

You should see `flask-docker-app:1.0` in the list.

What Happened

- Docker pulled the `Python 3.9 base image`.
- Installed `Flask` and its dependencies (`click`, `werkzeug`, `jinja2`, etc.) inside the image.
- Successfully completed the `RUN pip install -r requirements.txt` step.
- Finally executed the `CMD ["python", "app.py"]` instruction – meaning when you run the container, it'll start your Flask app automatically.
- The image `flask-docker-app:1.0` is now ready.

The `warning about “running pip as root”` is harmless inside containers – you can safely ignore it since containers are isolated environments.

Step 7: Run Container

```
docker run -d -p 8080:5000 flask-docker-app:1.0 -- to create and start  
the new container from the flask-docker-app image
```

or

```
docker run -p hostport:containerport <imagename>
```

Now your container is running!

Check with:

```
docker ps
```

Step 7: Test Your App

Go to your AWS instance's **Public IP**, open in browser:

<http://<your-public-ip>:8080>



You should see:

Hello, Docker! 🚀 Flask app running inside AWS EC2 container.

the above content we can edit in `app.py` file.

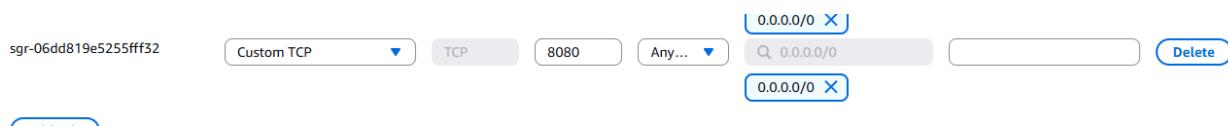
⚠️ If You Don't See It:

Make sure your **EC2 Security Group** allows **Inbound Rule: Port 8080 (TCP)**.

To fix:

- Go to **AWS Console** → **EC2** → **Security Groups** → **Inbound Rules** → **Edit**
- Add:
 - Type: Custom TCP
 - Port Range: 8080
 - Source: 0.0.0.0/0

Then reload your browser.





Congratulations!

You just completed:

Docker Build → Run → Deploy → Access via Public IP on AWS EC2

NOTE: IF YOU MAKE ANY CHANGES IN APP.PY WE SHOULD RUN BELOW COMMANDS

2. Rebuild your Docker image (so the fixed app.py is used):

```
docker build -t flask-docker-app:1.0 .
```

3. Run the container again:

```
docker run -d -p 8080:5000 flask-docker-app:1.0 -- before run this  
command pls stop existed containers or change port numbers on above  
command.
```

```
Host server port no is 8080, container port no is 5000  
8080:500
```

Finally, Flask app is now **successfully deployed and running** inside the Docker container, and you can access it through the browser (e.g., <http://<your-EC2-public-IP>:8080>).

Docker day -02

Step 8: push the image to docker hub (registry)

```
docker tag <local-image-name>:<tag> <dockerhub-username>/<repository-  
name>:<tag>
```

```
docker tag flask-docker-app:1.0 kmuthyal/flask-docker-app:1.0
```

```
docker push kasevaddi/myapp:latest
```

```

ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker login
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker login

USING WEB-BASED LOGIN

Info - To sign in with credentials on the command line, use 'docker login -u <username>'

Your one-time device confirmation code is: RGZK-GVGM
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate

Waiting for authentication in the browser...
^CLogin canceled
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker login -u kmuthyal

Info - A Personal Access Token (PAT) can be used instead.
To create a PAT, visit https://app.docker.com/settings

Password:
WARNING! Your credentials are stored unencrypted in '/home/ubuntu/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker tag flask-docker-app:1.0 kmuthyal/flask-docker-app:1.0
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker tag
docker: 'docker tag' requires 2 arguments

Usage: docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

Run 'docker tag --help' for more information
ubuntu@ip-172-31-33-209:~/docker-flask-app$ docker push kmuthyal/flask-docker-app:1.0
The push refers to repository [docker.io/kmuthyal/flask-docker-app]
538496566941: Pushed ✓
3a7c0c1b74c7: Pushed ✓
05a7e960f516: Pushed ✓
c8f6b54339a8: Mounted from library/python
298992e09a03: Mounted from library/python
4f237755fbae: Mounted from library/python
d7c97cb6f1fe: Mounted from library/python
1.0: digest: sha256:03ba5f7c4cde603ff2d483ee27aede1526d22df10b3a4fa483e2d0d7bae3240d size: 1783
ubuntu@ip-172-31-33-209:~/docker-flask-app$ ...

```

The screenshot shows the Docker Hub 'My Hub' interface. The left sidebar shows the user's profile (kmuthyal) and navigation links for Repositories, Hardened Images, Collaborations, Settings, Billing, Usage, Pulls, and Storage. The main area is titled 'Repositories' and displays a list of repositories under the 'kmuthyal' namespace. The repository 'kmuthyal/flask-docker-app' is highlighted with a pink oval. The table columns are Name, Last Pushed, Contains, Visibility, and Scout. The repository details are as follows:

Name	Last Pushed	Contains	Visibility	Scout
kmuthyal/flask-docker-app	11 minutes ago	IMAGE	Public	Inactive
kmuthyal/my-first-docker-image	about 2 months ago	IMAGE	Public	Inactive
kmuthyal/app1	over 1 year ago	IMAGE	Public	Inactive
kmuthyal/app2	over 1 year ago	IMAGE	Public	Inactive
kmuthyal/cart	over 2 years ago	IMAGE	Public	Inactive

Topic: how to pull and use the existed image from docker-hub

Perfect 🎉 — since you've already pushed our Docker image to **Docker Hub**, pulling and running it on a **new Ubuntu server** is very straightforward. Let's go step-by-step.

⌚ Step-by-step: Pull and run your image on a new server

Step 1 — Install Docker (if not installed yet)

If your new Ubuntu server doesn't have Docker installed, run:

```
sudo apt update  
sudo apt install -y docker.io  
sudo systemctl start docker  
sudo systemctl enable docker
```

You can verify Docker is running:

```
docker --version
```

Step : Add your user to the Docker group

This lets you run docker commands without using sudo every time.

```
sudo usermod -aG docker $USER
```

Then **logout and login again**, or run:

```
newgrp docker
```

Step 2 — Log in to Docker Hub

Use your Docker Hub credentials to log in so the new server can pull private or public images:

```
docker login -u kmuthyal (username of dockerhub)  
Passwd: password of above user
```

It will prompt for your **Docker Hub username** and **password (or access token)**.

Step 3 — Pull your image

Now, pull your image from Docker Hub:

```
docker pull <your-dockerhub-username>/<image-name>:<tag>
```

Example:

```
docker pull kmuthyal/flask-docker-app:1.0
```

This downloads the image layers to your new server.

You can verify it's there:

```
docker images
```

Step 4 — Run the container

Now run the Flask app container and expose it on your desired port:

```
docker run -d -p 8080:5000 --name flask-app kmuthyal/flask-docker-app:1.0
```

Explanation:

- `-d` → run in detached mode
- `-p 8080:5000` → map host port 8080 to container's internal port 5000 (Flask default)
- `--name flask-app` → container name
- `kmuthyal/flask-docker-app:1.0` → image from Docker Hub

Step 5 — Verify it's running

```
docker ps
```

You should see something like:

CONTAINER ID	IMAGE	COMMAND	STATUS
abcd1234efgh	kmuthyal/flask-docker-app:1.0	"python app.py"	Up 5 seconds
	0.0.0.0:8080->5000/tcp	flask-app	

Step: please enable inbound rules (tcp and 8080 port) in aws server.
Before access in browser

Step 6 — Access in browser

Now open your Flask app in the browser using your new server's **public IP**:

<http://<your-server-public-ip>:8080>

You should see your Flask app page (e.g., “Hello World!”).

Docker day –03

Docker is a lightweight process to create containers we should use docker for deploy and run the application inside the containers.

Topic: what is virtualization and containerization?

What is Virtualization?

Virtualization is a technology that allows you to run multiple **virtual machines (VMs)** on a single **physical machine**.

Each VM has its **own operating system (OS)**, kernel, libraries, and applications — behaving like a separate computer.

Example:

- You can run **Windows**, **Linux**, and **Ubuntu** virtual machines on one physical server.
- Tools: **VMware**, **VirtualBox**, **Hyper-V**, **KVM**

What is Containerization?

Containerization is a lightweight alternative to virtualization.

It packages applications and their dependencies into **containers** that share the **same host OS kernel** but run in isolated environments.

Tools: **Docker**, **Podman**, **Kubernetes**

How it works:

- Containers run on a **container runtime** (like Docker Engine).
- They share the host OS kernel but are isolated using Linux features (namespaces, cgroups).
- Containers start quickly and use less memory than VMs.

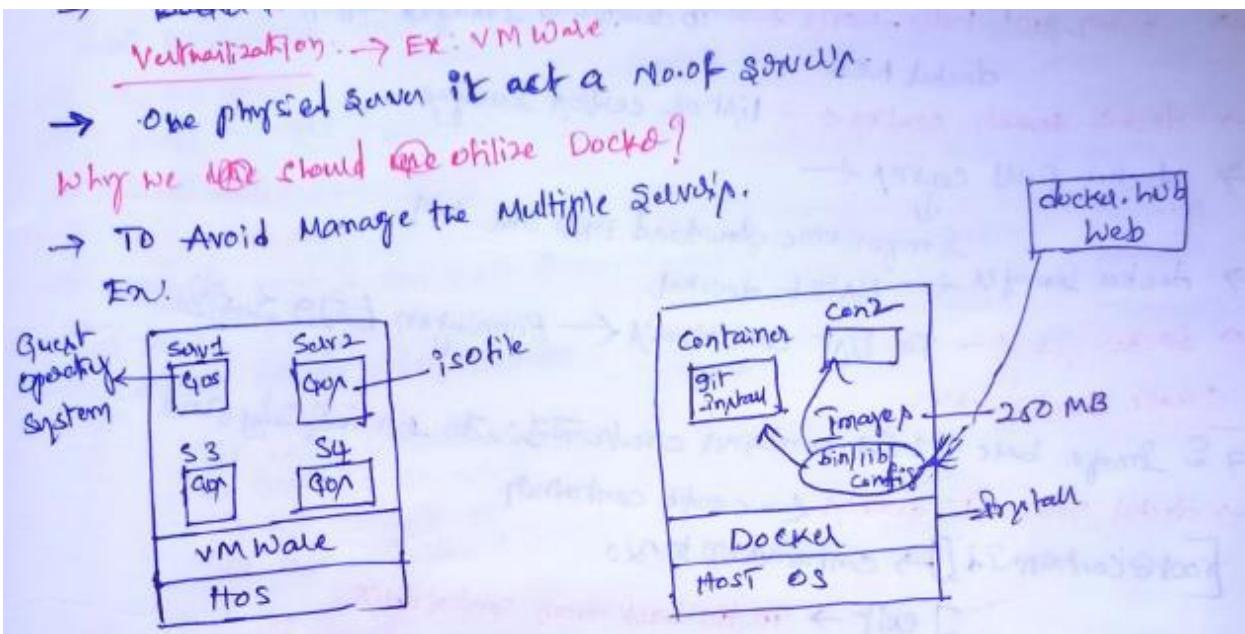
Difference Between Virtualization and Containerization

Feature	Virtualization	Containerization
Definition	Runs multiple OS instances on a single host	Runs multiple isolated apps on the same OS kernel
Abstraction Level	Hardware level (via hypervisor)	OS level (via container runtime)
Component	Virtual Machine (VM)	Container
Guest OS	Each VM has its own OS	All containers share the host OS kernel
Startup Time	Slow (minutes)	Fast (seconds)
Resource Usage	Heavy (each VM needs OS, memory, disk)	Lightweight (shares kernel, fewer resources)
Isolation	Strong (full OS isolation)	Moderate (process-level isolation)
Use Case	Running multiple different OS environments	Deploying microservices, fast CI/CD pipelines
Examples	VMware, VirtualBox, KVM	Docker, Podman, Kubernetes

In Simple Terms

- **Virtualization = multiple computers (VMs) on one machine.**
- **Containerization = multiple apps (containers) on one OS.**

Diff between virtualization and containerization?



Above scenario:

If we utilize the docker directly we can download the images from docker hub

After running the image container will be creating

We can create multiple containers using image

It will take time 1 or 2 seconds to create the containers from image

In containerization booting process is very easy

Maintain and integration is simple

Once install the docker – need to run below commands:

1st command of docker ---> docker info --> to list info of all containers and images,

Docker images - list images

Docker search centos – to list centos images

docker pull eclipse/centos – centos image download into our LM

docker images – to list all images

Container creation:

We should remember that create the container using what image(centos/mysql-57-centos7)

docker run -itd eclipse/centos – to just create the container from image

docker run -it eclipse/centos /bin/bash -- to stop the container with exit command

docker exec -it --user root 17ab0cd9b1cd /bin/bash

docker run -it eclipse/centos /bin/bash

```
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker attach 32d246d7a684
bash-4.2$ export PS1="[\u@\h \W]\$ "
[mysql@32d246d7a684 opt]$ pwd
```

Now we are inside the container – we can use this container as new server

It means we can install the packages,

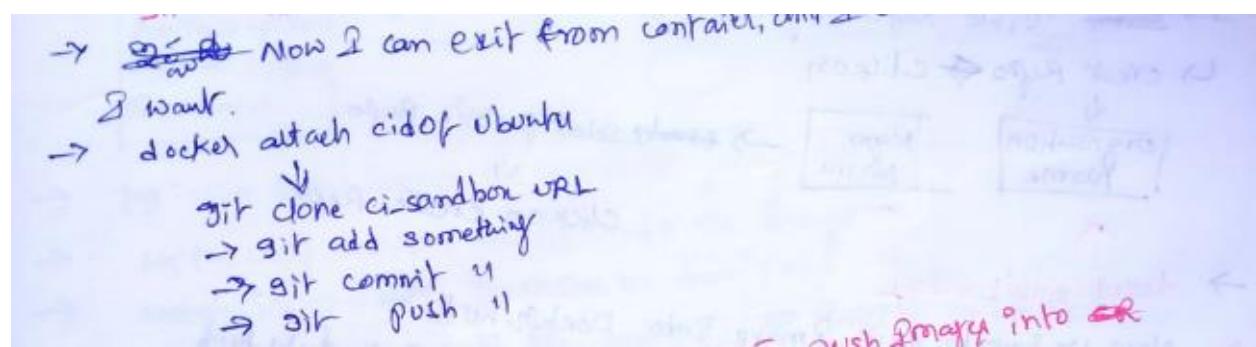
Example git, maven, sonarqube, jenkins etc .. we can clone git...

apt update && apt install -y git

git --version

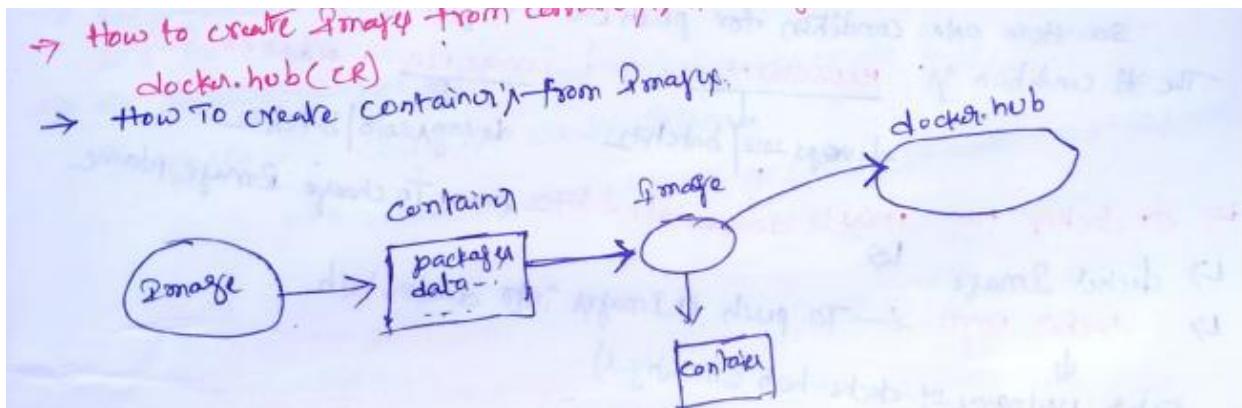
cat /etc/os-release

If you want add files or directories using git follow below



Docker day -04

How to create image from containers and how to push images to docker hub:



If see the above image it shows:

1.Create containers from image – installed git in this container

Docker run -itd centos or ubuntu – create containers based on image. Without pulling the image this cmd download the image from central repo and place into LM

Docker attach cid -- to login container

apt update

apt install git -y

```
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
kmuthyal/flask-docker-app  1.0      7d153c000721  5 days ago   133MB
flask-docker-app     1.0      7d153c000721  5 days ago   133MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker run -itd ubuntu
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
4b3ffd8ccb52: Pull complete
Digest: sha256:66460d557b25769b102175144d538d88219c077c678a49af4afca6fbfc1b525 ✓
Status: Downloaded newer image for ubuntu:latest
cf0b00038841ade1d4f310f8febe0bd28ec667f1ae21ae8a1b5e38e3c317c
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
kmuthyal/flask-docker-app  1.0      7d153c000721  5 days ago   133MB
flask-docker-app     1.0      7d153c000721  5 days ago   133MB
ubuntu              latest    97bed23a3497  5 weeks ago  78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS           NAMES
cf0b00038841        ubuntu              "/bin/bash"         16 seconds ago   Up 16 seconds           gallant_villani
b98f25be2e9b        flask-docker-app:1.0   "python app.py"    5 days ago       Exited (137) 41 hours ago   xenodochial_satoshi
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker attach cf0b00038841
root@cf0b00038841:/# touch kasevaddi
root@cf0b00038841:/# apt update && apt install git -y ✓
```

git -version

Git clone **docker-install-commands** -- **clone this repo**

2. image creation from container

docker commit b6e6e0fae29b -- (of image(centos/ubuntu) _ - here new image will create, using existed container (Docker images -- to list

3. Docker tag imageid (above image id) kmuthyal/docker2025-kasevaddi -- to change re name of above image becoz while creatiing above there is no name. Using this image id we can rename the image name (custom name) kmuthyal/docker2025-kasevaddi

Docker images – above image can see

If we create containers from above image kmuthyal/docker2025-kasevaddi then inside that container git and repo should be there.

4. Docker run –it kmuthyal/docker2025-kasevaddi -- to login to the containr

So as per above process download new image ---> created container using new image (installed git and cloned repo inthis container) --> again created kmuthyal/docker2025-kasevaddi custom image --> again created container (verified git and cloned repo) whether data copied or not). It means we have copied data. Between isolated containers.

```

done.
root@cf0b00038841:/# git --version ✓
git version 2.43.0
root@cf0b00038841:# ls
bin boot dev etc home kasevaddi lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@cf0b00038841:# git clone https://github.com/KASEVADDI/docker-install-commands.git ✓
Cloning into 'docker-install-commands'...
remote: Enumerating objects: 44, done.
remote: Counting objects: 100% (44/44), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 44 (delta 15), reused 22 (delta 4), pack-reused 0 (from 0)
Receiving objects: 100% (44/44), 12.96 MiB | 30.44 MiB/s, done.
Resolving deltas: 100% (15/15), done.
root@cf0b00038841:# ls
bin boot dev docker-install-commands etc home kasevaddi lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@cf0b00038841:# read escape sequence
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cf0b00038841 ubuntu "/bin/bash" 4 minutes ago Up 4 minutes gallant_villani
b98f25be2e9b flask-docker-app:1.0 "python app.py" 5 days ago Exited (137) 41 hours ago xenodochial_satoshi
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
kmuthyal/flask-docker-app 1.0 7d153c000721 5 days ago 133MB
flask-docker-app 1.0 7d153c000721 5 days ago 133MB
ubuntu latest 97bed23a3497 5 weeks ago 78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker commit cf0b00038841 sha256:e88a72c773b2487166b091fbcba8247ca39ce7b6245ef889117392e04a226080
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
<none> <none> e88a72c773b2 4 seconds ago 236MB
flask-docker-app 1.0 7d153c000721 5 days ago 133MB
kmuthyal/flask-docker-app 1.0 7d153c000721 5 days ago 133MB
ubuntu latest 97bed23a3497 5 weeks ago 78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker tag e88a72c773b2 kmuthyal/docker2025-kasevaddi
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
kmuthyal/docker2025-kasevaddi latest e88a72c773b2 57 seconds ago 236MB
kmuthyal/flask-docker-app 1.0 7d153c000721 5 days ago 133MB
flask-docker-app 1.0 7d153c000721 5 days ago 133MB
ubuntu latest 97bed23a3497 5 weeks ago 78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker commit cf0b00038841 sha256:e88a72c773b2487166b091fbcba8247ca39ce7b6245ef889117392e04a226080
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
<none> <none> e88a72c773b2 4 seconds ago 236MB
flask-docker-app 1.0 7d153c000721 5 days ago 133MB
kmuthyal/flask-docker-app 1.0 7d153c000721 5 days ago 133MB
ubuntu latest 97bed23a3497 5 weeks ago 78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker tag e88a72c773b2 kmuthyal/docker2025-kasevaddi
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
kmuthyal/docker2025-kasevaddi latest e88a72c773b2 57 seconds ago 236MB
kmuthyal/flask-docker-app 1.0 7d153c000721 5 days ago 133MB
flask-docker-app 1.0 7d153c000721 5 days ago 133MB
ubuntu latest 97bed23a3497 5 weeks ago 78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker run -it kmuthyal/docker2025-kasevaddi
root@78aa4a961de7:~# git --version
git version 2.43.0
root@78aa4a961de7:~# ls
bin boot dev docker-install-commands etc home kasevaddi lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@78aa4a961de7:~# ...

```

```

root@78aa4a961de7:~# read escape sequence
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
cf0b00038841 ubuntu "/bin/bash" 4 minutes ago Up 4 minutes gallant_villani
b98f25be2e9b flask-docker-app:1.0 "python app.py" 5 days ago Exited (137) 41 hours ago xenodochial_satoshi
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
kmuthyal/flask-docker-app 1.0 7d153c000721 5 days ago 133MB
flask-docker-app 1.0 7d153c000721 5 days ago 133MB
ubuntu latest 97bed23a3497 5 weeks ago 78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker commit cf0b00038841 sha256:e88a72c773b2487166b091fbcba8247ca39ce7b6245ef889117392e04a226080
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
<none> <none> e88a72c773b2 4 seconds ago 236MB
flask-docker-app 1.0 7d153c000721 5 days ago 133MB
kmuthyal/flask-docker-app 1.0 7d153c000721 5 days ago 133MB
ubuntu latest 97bed23a3497 5 weeks ago 78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker tag e88a72c773b2 kmuthyal/docker2025-kasevaddi
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
kmuthyal/docker2025-kasevaddi latest e88a72c773b2 57 seconds ago 236MB
kmuthyal/flask-docker-app 1.0 7d153c000721 5 days ago 133MB
flask-docker-app 1.0 7d153c000721 5 days ago 133MB
ubuntu latest 97bed23a3497 5 weeks ago 78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker run -it kmuthyal/docker2025-kasevaddi
root@78aa4a961de7:~# git --version
git version 2.43.0
root@78aa4a961de7:~# ls
bin boot dev docker-install-commands etc home kasevaddi lib lib64 media mnt opt proc root run sbin srv sys tmp usr var
root@78aa4a961de7:~# ...

```

How do push docker image from lm to docker hub:

Process:

1. Need to create docker hub account— search in web— hub.docker.com --> need to login with mail id,username

Once login to docker hub --> we can create repos like git gub

Process:

Click on home – repos – click on create repo pls check below image

Condition: docker image name and docker repo name must be same

Example; kmuthyal/docker2025-kasevaddi – this custom image name and docker repos name.

The screenshot shows the 'Create repository' page on Docker Hub. A pink circle highlights the 'Public' visibility option, which is selected. Another pink circle highlights the 'Create' button at the bottom right. A large pink oval encloses the 'Pushing images' section on the right, which contains CLI commands for pushing images and a note to replace 'tagname' with the desired image repository tag.

The screenshot shows the repository page for 'kmuthyal/docker2025-kasevaddi'. A pink circle highlights the repository name. A large pink oval encloses the 'Docker commands' section on the right, which contains the command 'docker push kmuthyal/docker2025-kasevaddi:tagname'. Below the repository name, it says 'No tags available' and 'This repository doesn't have any tagged images.'

Process to push image to docker hub:

To push exit from container:

Docker images

Docker login –u kmuthyal

Docker push imagename

```
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker login -u kmuthyal
Info - A Personal Access Token (PAT) can be used instead.
To create a PAT, visit https://app.docker.com/settings

Password:
Login Succeeded
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker images
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
kmuthyal/docker2025-kasevaddi    latest   e88a72c773b2  13 minutes ago  236MB
kmuthyal/flask-docker-app       1.0     7d153c000721  5 days ago   133MB
flask-docker-app               1.0     7d153c000721  5 days ago   133MB
ubuntu                  latest   97bed23a3497  5 weeks ago   78.1MB
ubuntu@ip-172-31-33-149:~/docker-flask-app$ docker push kmuthyal/docker2025-kasevaddi
Using default tag: latest
The push refers to repository [docker.io/kmuthyal/docker2025-kasevaddi]
a601141e36b2: Pushed
073ec47a8c22: Mounted from library/ubuntu
latest: digest: sha256:1e77bf4b47a1081b357cb4dcb00d44cd3bfc23cbfc7d655b5ff6d1549627504 size: 741
ubuntu@ip-172-31-33-149:~/docker-flask-app$
```