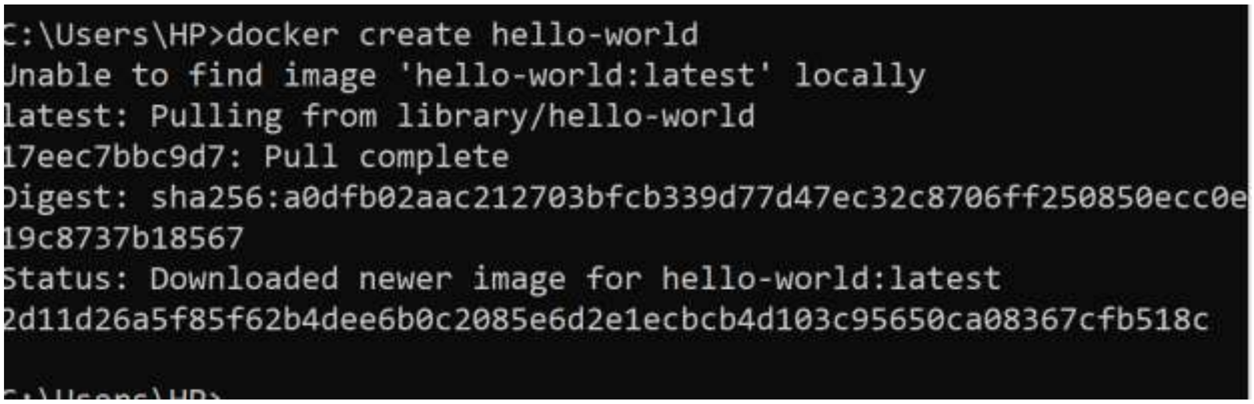
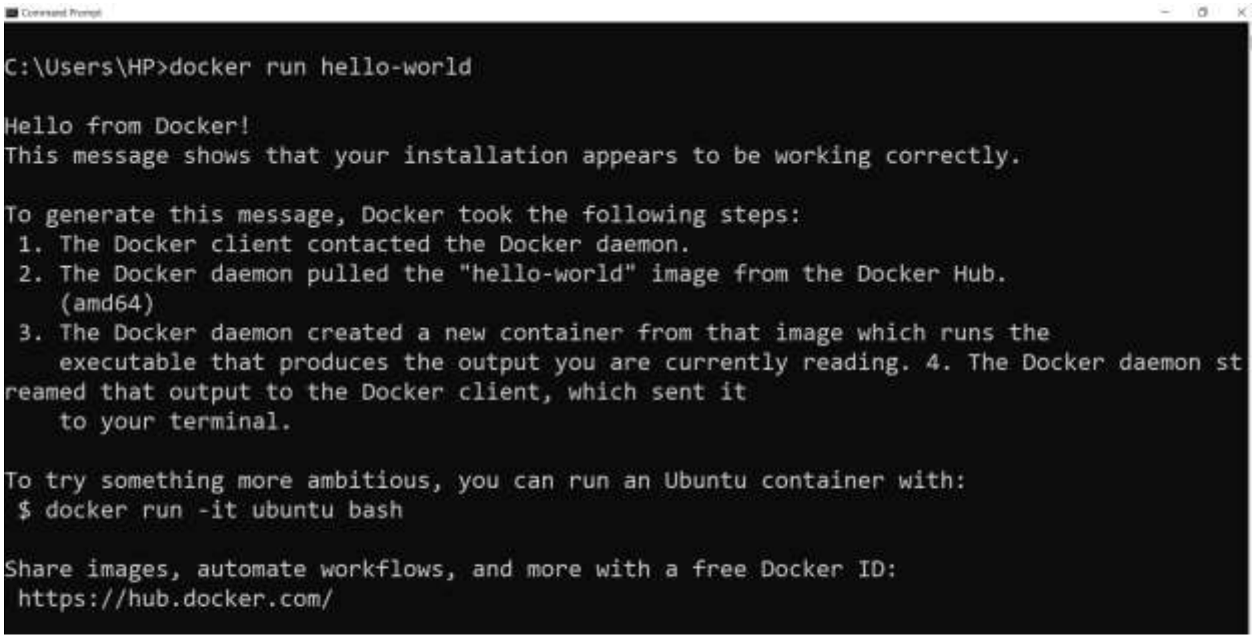
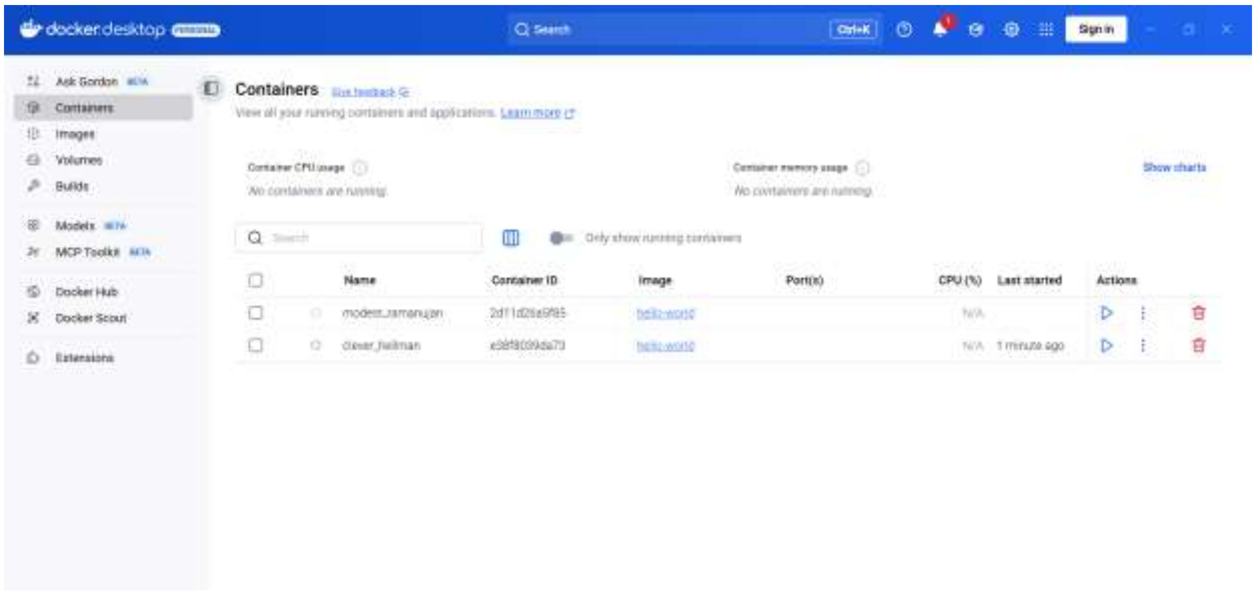
Container Management Commands



```
:\Users\HP>docker create hello-world
Jnable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:a0dfb02aac212703bfcb339d77d47ec32c8706ff250850ecc0e
19c8737b18567
Status: Downloaded newer image for hello-world:latest
2d11d26a5f85f62b4dee6b0c2085e6d2e1ecbcb4d103c95650ca08367cfb518c
```



```
C:\Users\HP>docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading. 4. The Docker daemon st
reamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/
```



Containers Give feedback

View all your running containers and applications. Learn more

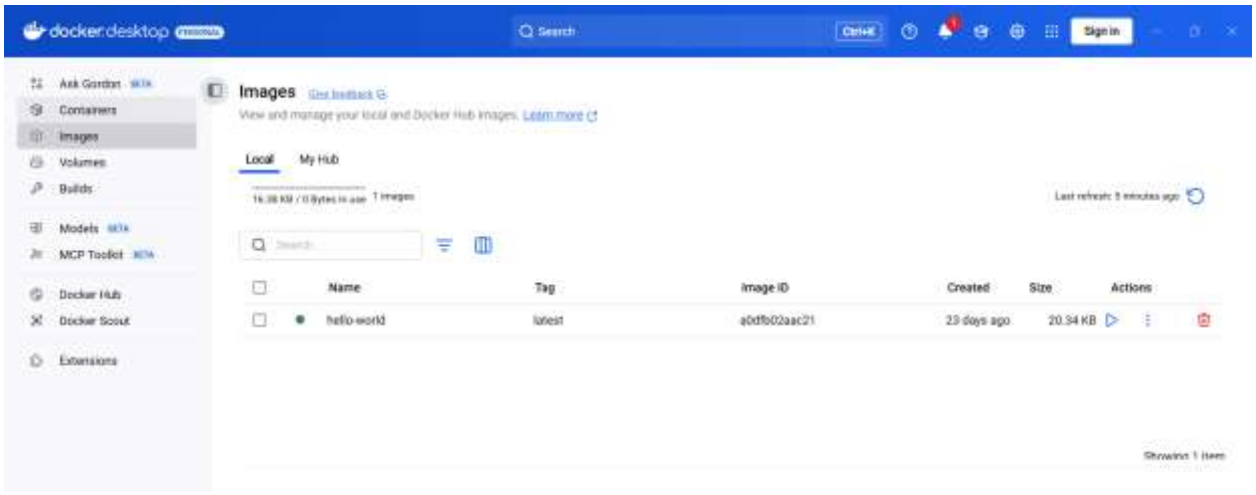
Container CPU usage
No containers are running.

Container memory usage
No containers are running.

Show charts

Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last started
modest_ramanujan	2d11d26a9f85	hello-world		N/A	
clever_hellman	e58f8039da73	hello-world		N/A	1 minute ago



Images Give feedback

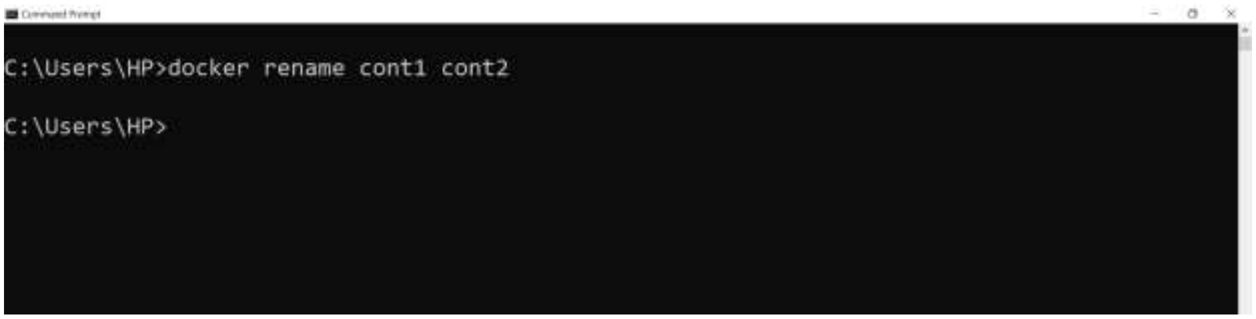
View and manage your local and Docker Hub images. Learn more

16.38 KB / 0 Bytes in use 1 images

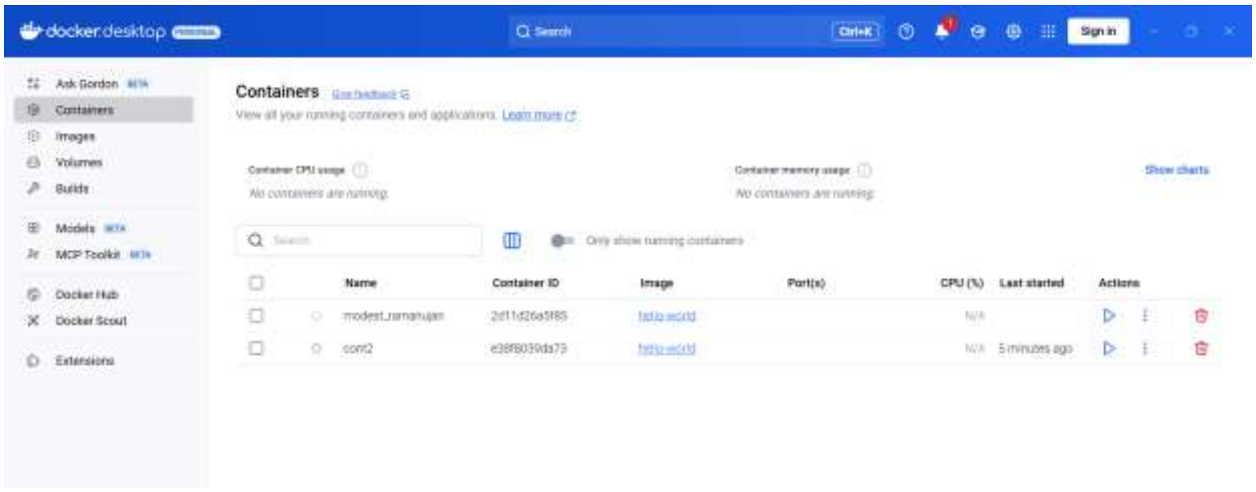
Last refresh: 5 minutes ago

	Name	Tag	Image ID	Created	Size
●	hello-world	latest	a0dfb02aac21	23 days ago	20.34 KB

Showing 1 item



```
C:\Users\HP>docker rename cont1 cont2

C:\Users\HP>
```



Containers Give feedback

View all your running containers and applications. Learn more

Container CPU usage
No containers are running.

Container memory usage
No containers are running.

Show charts

Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
modest_ramanujan	2d11d26a5f85	hello-world		N/A		
cont2	e38f8039da73	hello-world		N/A	5 minutes ago	

Since Docker requires memory ≤ memory-swap, you should update both

This sets:

- Memory = **512 MB**
- MemorySwap = **1 GB**

In Docker, **memory swap** is the combined limit of: memory (RAM) + swap space

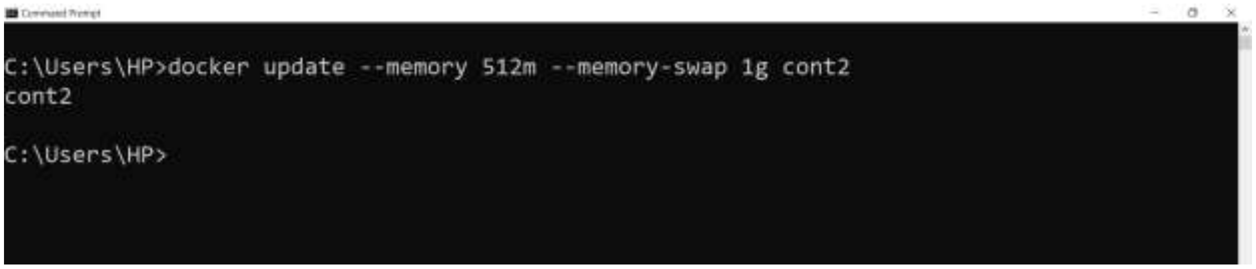
- **`--memory` (or `-m`)** → maximum RAM the container can use.
- **`--memory-swap`** → maximum RAM + swap space the container can use.

Swap is a portion of the disk used as "virtual memory" when RAM is full. It's much slower than RAM, but it prevents processes from being killed immediately when they exceed physical memory.

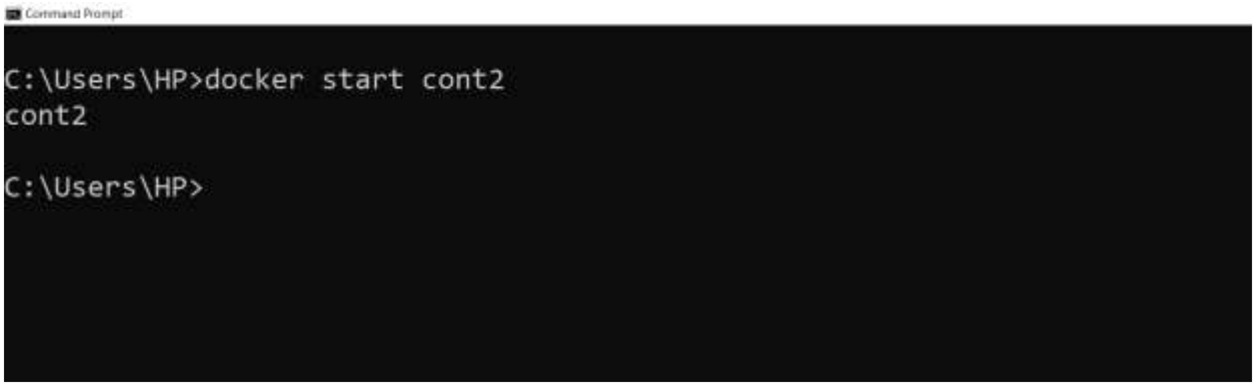
1. **If you set only `--memory`:**

   By default, `--memory-swap` is set to **2 × memory**.

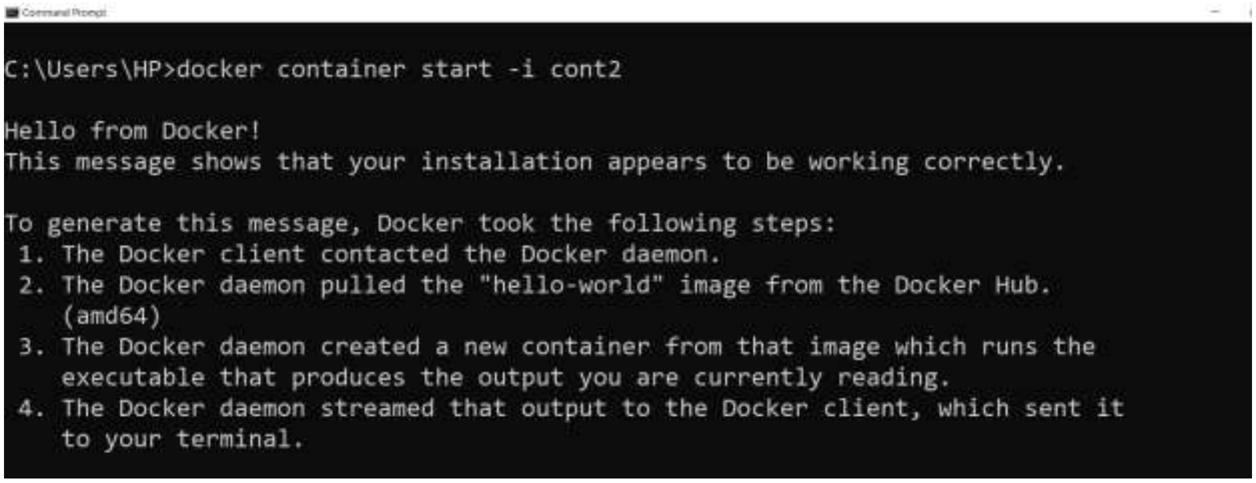
   Example: `--memory=1g` → container can use **1 GB RAM + 1 GB swap = 2 GB total**.



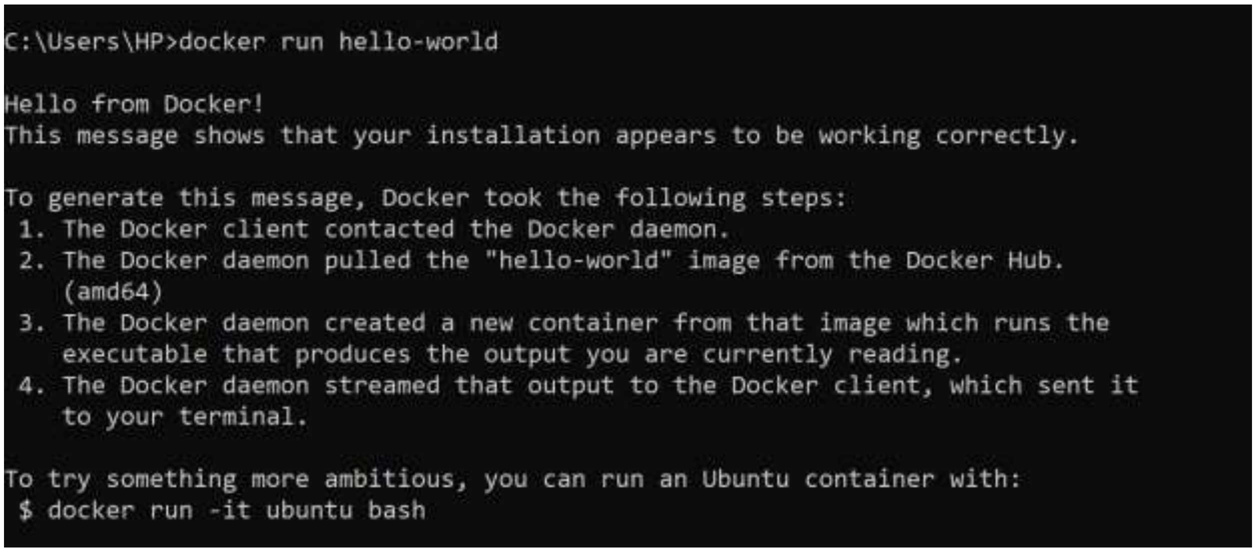
```
C:\Users\HP>docker update --memory 512m --memory-swap 1g cont2
cont2

C:\Users\HP>
```



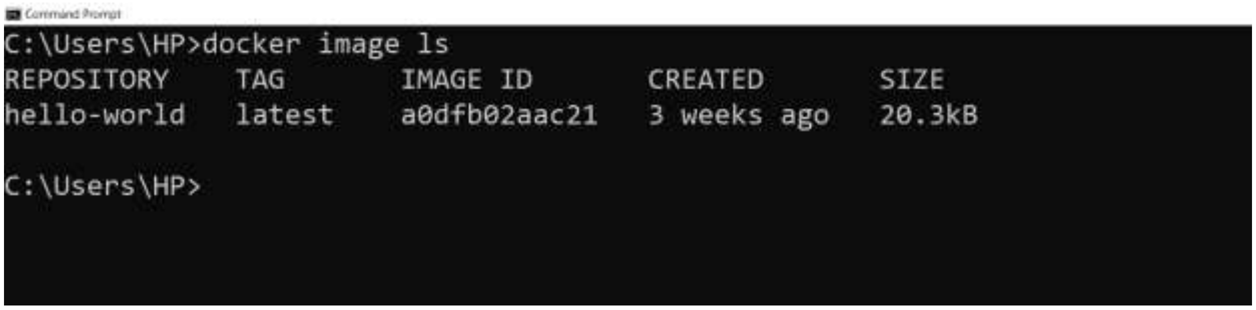
```
C:\Users\HP>docker start cont2
cont2

C:\Users\HP>
```



```
C:\Users\HP>docker container start -i cont2

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
```

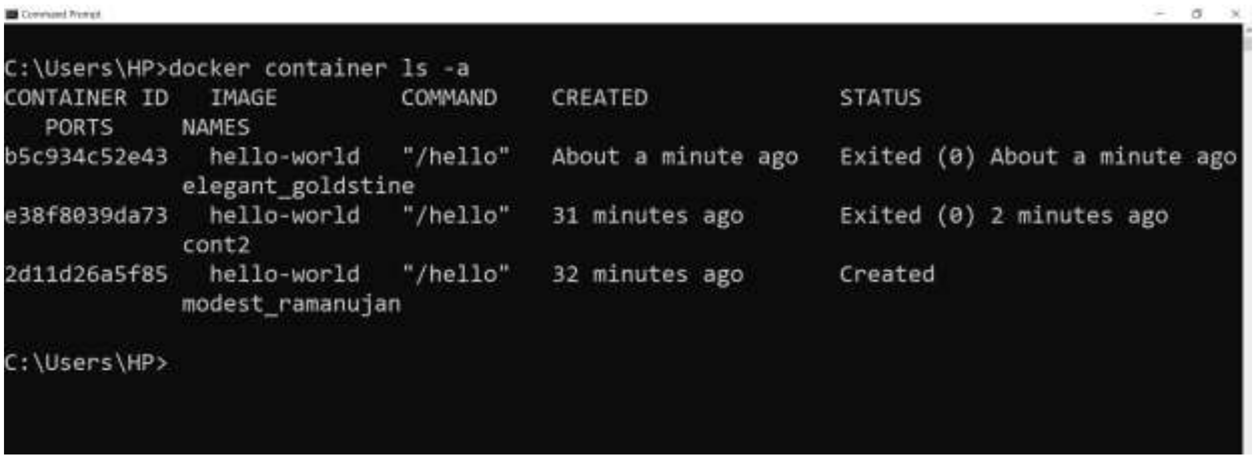


```
C:\Users\HP>docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash
```



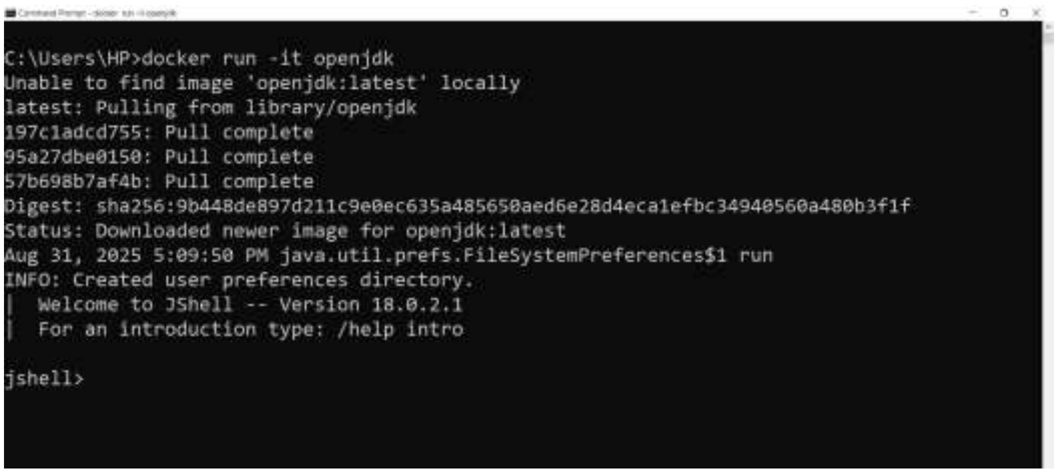
```
C:\Users\HP>docker image ls
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
hello-world   latest    a0dfb02aac21   3 weeks ago   20.3kB

C:\Users\HP>
```

-a shows both stopped and running containers

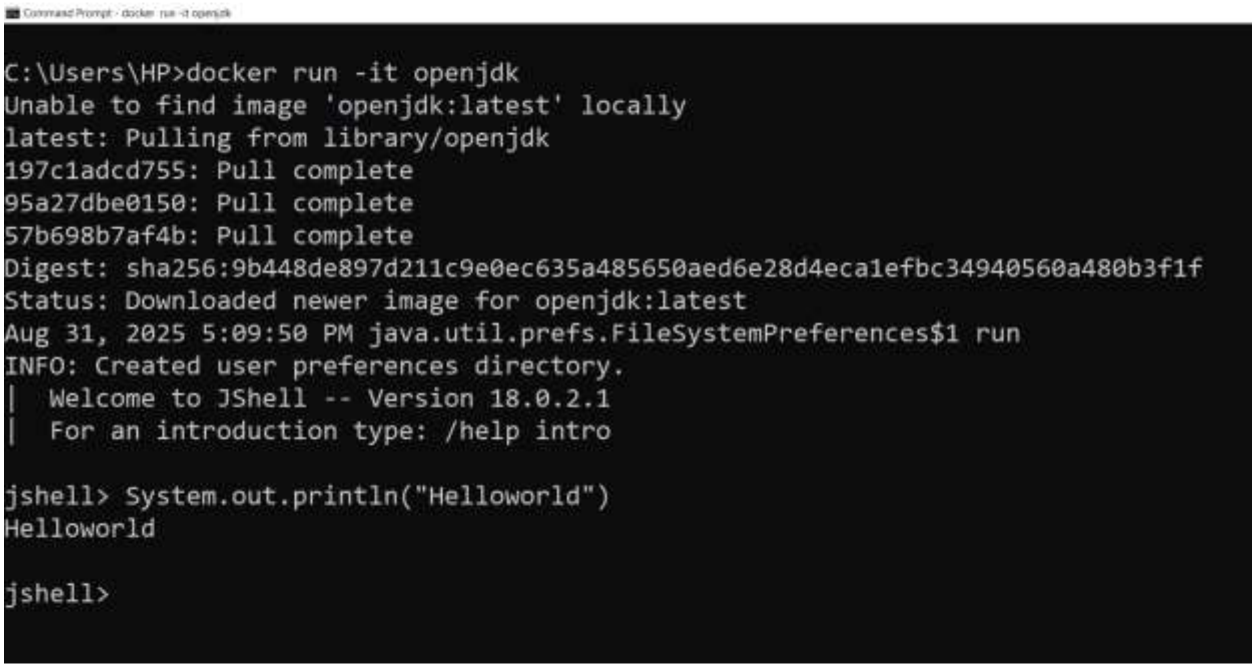


```
C:\Users\HP>docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED              STATUS
   PORTS     NAMES
b5c934c52e43   hello-world   "/hello"   About a minute ago   Exited (0) About a minute ago
             elegant_goldstine
e38f8039da73   hello-world   "/hello"   31 minutes ago       Exited (0) 2 minutes ago
             cont2
2d11d26a5f85   hello-world   "/hello"   32 minutes ago       Created
             modest_ramanujan

C:\Users\HP>
```

Running containers interactively allows you to run commands inside the container if it supports it. We can use the openjdk image. This allows us to execute java commands line by line in a Java shell

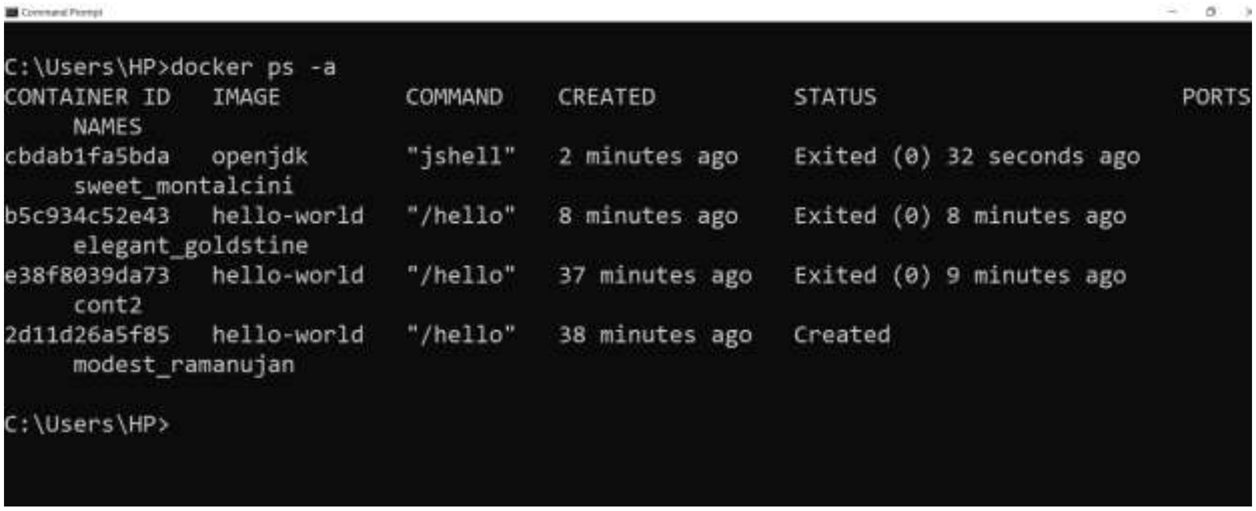


```
C:\Users\HP>docker run -it openjdk
Unable to find image 'openjdk:latest' locally
latest: Pulling from library/openjdk
197c1adcd755: Pull complete
95a27dbe0150: Pull complete
57b698b7af4b: Pull complete
Digest: sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a480b3f1f
Status: Downloaded newer image for openjdk:latest
Aug 31, 2025 5:09:50 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
|  Welcome to JShell -- Version 18.0.2.1
|  For an introduction type: /help intro

jshell>
```

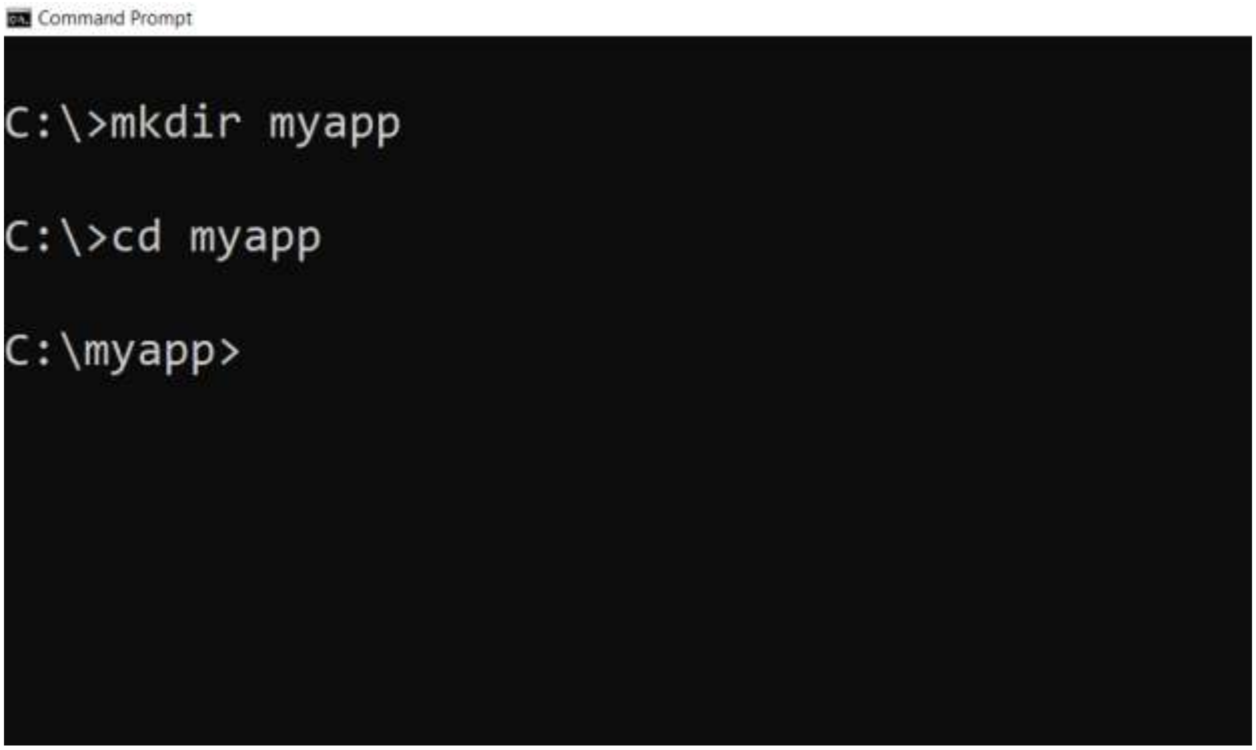


```
C:\Users\HP>docker run -it openjdk
Unable to find image 'openjdk:latest' locally
latest: Pulling from library/openjdk
197c1adcd755: Pull complete
95a27dbe0150: Pull complete
57b698b7af4b: Pull complete
Digest: sha256:9b448de897d211c9e0ec635a485650aed6e28d4eca1efbc34940560a480b3f1f
Status: Downloaded newer image for openjdk:latest
Aug 31, 2025 5:09:50 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
|  Welcome to JShell -- Version 18.0.2.1
|  For an introduction type: /help intro

jshell> System.out.println("Helloworld")
Helloworld

jshell>
```

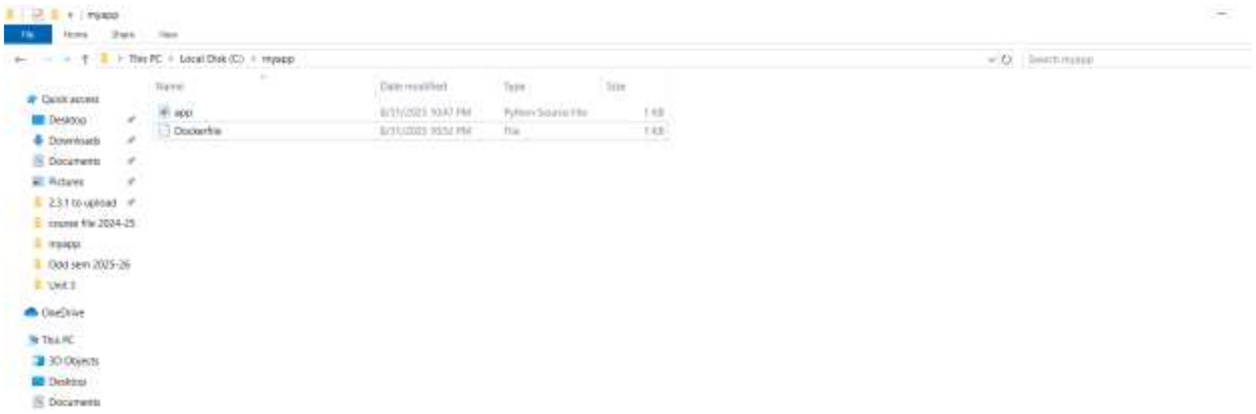
Press ctrl+d to stop jshell



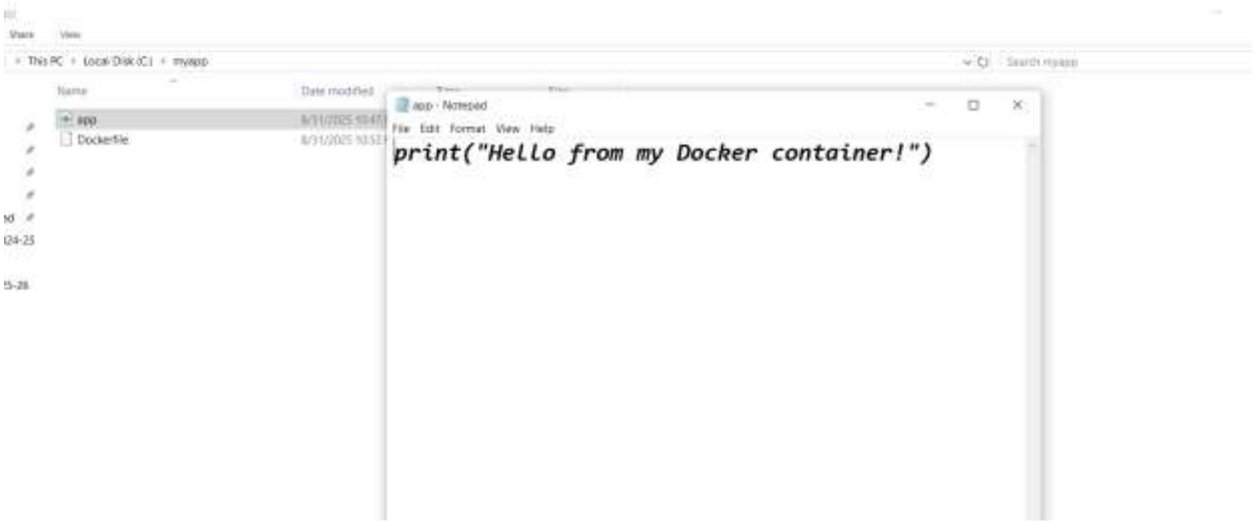
```
C:\Users\HP>docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                       PORTS
     NAMES
cbdab1fa5bda   openjdk       "jshell"   2 minutes ago    Exited (0) 32 seconds ago
     sweet_montalcini
b5c934c52e43   hello-world   "/hello"   8 minutes ago    Exited (0) 8 minutes ago
     elegant_goldstine
e38f8039da73   hello-world   "/hello"   37 minutes ago   Exited (0) 9 minutes ago
     cont2
2d11d26a5f85   hello-world   "/hello"   38 minutes ago   Created
     modest_ramanujan

C:\Users\HP>
```



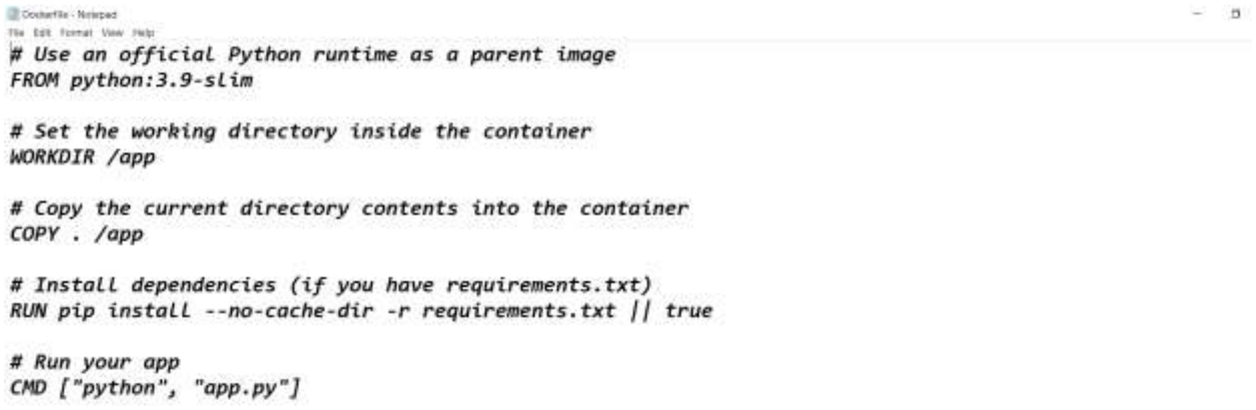
```
C:\>mkdir myapp

C:\>cd myapp

C:\myapp>
```



Name	Date modified	Type	Size
app	8/31/2025 10:47 PM	Python Source File	1 KB
Dockerfile	8/31/2025 10:52 PM	File	1 KB



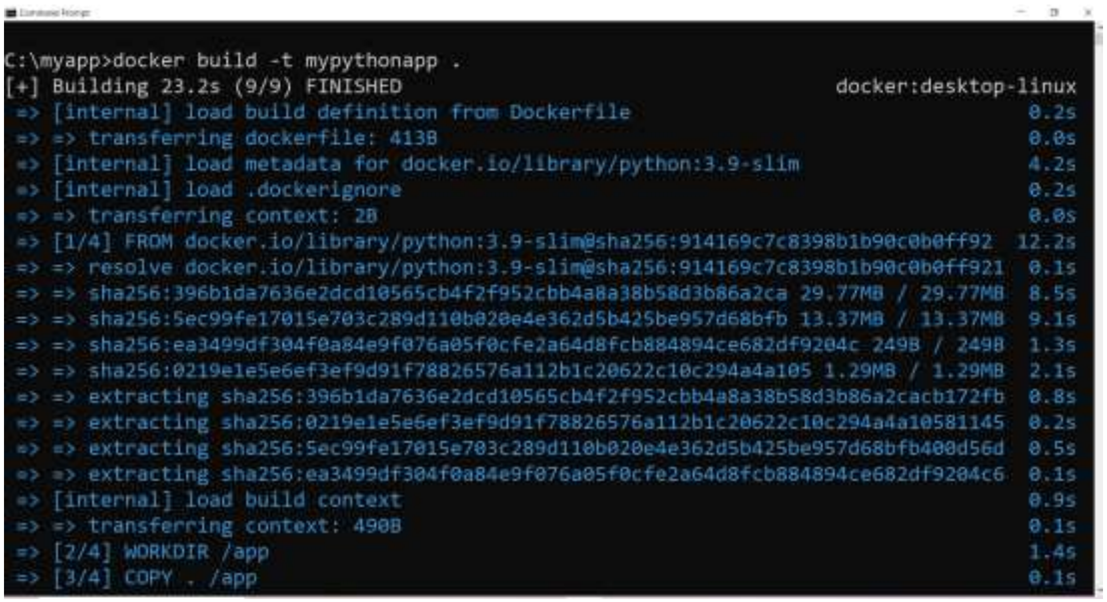
```
print("Hello from my Docker container!")
```



```
# Use an official Python runtime as a parent image
FROM python:3.9-slim

# Set the working directory inside the container
WORKDIR /app

# Copy the current directory contents into the container
COPY . /app

# Install dependencies (if you have requirements.txt)
RUN pip install --no-cache-dir -r requirements.txt || true

# Run your app
CMD ["python", "app.py"]
```

If it shows Dockerfile.txt, rename it:

ren C:\myapp\Dockerfile.txt Dockerfile



```
C:\myapp>docker build -t mypythonapp .
[+] Building 23.2s (9/9) FINISHED                                      docker:desktop-linux
 => [internal] load build definition from Dockerfile                                   0.2s
 => => transferring dockerfile: 413B                                                   0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                     4.2s
 => [internal] load .dockerignore                                                      0.2s
 => => transferring context: 2B                                                        0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff92  12.2s
 => => resolve docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921  0.1s
 => => sha256:396b1da7636e2dcd10565cb4f2f952cbb4a8a38b58d3b86a2ca 29.77MB / 29.77MB  8.5s
 => => sha256:5ec99fe17015e703c289d110b020e4e362d5b425be957d68bfb 13.37MB / 13.37MB  9.1s
 => => sha256:ea3499df304f0a84e9f076a05f0cfe2a64d8fcb884894ce682df9204c 249B / 249B  1.3s
 => => sha256:0219e1e5e6ef3ef9d91f78826576a112b1c20622c10c294a4a105 1.29MB / 1.29MB  2.1s
 => => extracting sha256:396b1da7636e2dcd10565cb4f2f952cbb4a8a38b58d3b86a2cacb172fb  0.8s
 => => extracting sha256:0219e1e5e6ef3ef9d91f78826576a112b1c20622c10c294a4a10581145  0.2s
 => => extracting sha256:5ec99fe17015e703c289d110b020e4e362d5b425be957d68bfb400d56d  0.5s
 => => extracting sha256:ea3499df304f0a84e9f076a05f0cfe2a64d8fcb884894ce682df9204c6  0.1s
 => [internal] load build context                                                      0.9s
 => => transferring context: 490B                                                      0.1s
 => [2/4] WORKDIR /app                                                                 1.4s
 => [3/4] COPY . /app                                                                  0.1s
```

1. **docker build**
   This tells Docker to **build an image** from a Dockerfile.
2. **-t mypythonapp**

   -t (or --tag) gives a **name** (and optionally a tag) to your image.

   Here, mypythonapp is the name of your image.

   If you don't give a tag like mypythonapp:1.0, Docker defaults to :latest.
   So this image will actually be called:

   mypythonapp:latest

3. **. (dot at the end)**

   This means **current directory**.

   Docker will look inside the current folder for a file named Dockerfile.

   It also includes everything in that directory as the **build context** (so it can copy files into the image).

Suppose you are inside C:\myapp and you have:

```
C:\myapp
├── Dockerfile
├── app.py
└── requirements.txt
```

When you run:

```
docker build -t mypythonapp .
```

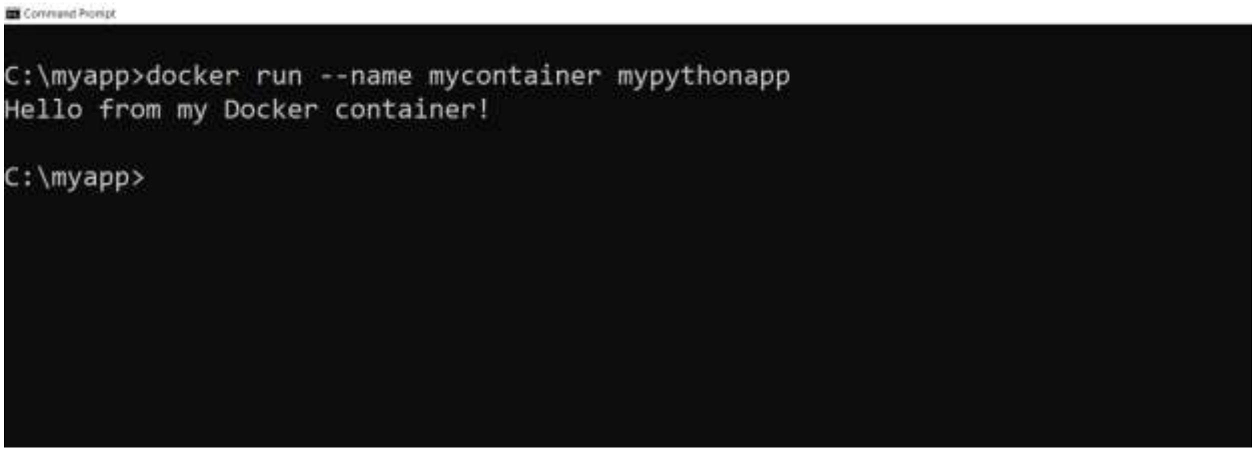
- Docker reads the **Dockerfile** in C:\myapp.
- It follows the instructions (e.g., copy files, install dependencies).
- It creates a new image named **mypythonapp:latest**.

After build, you can check your image with:

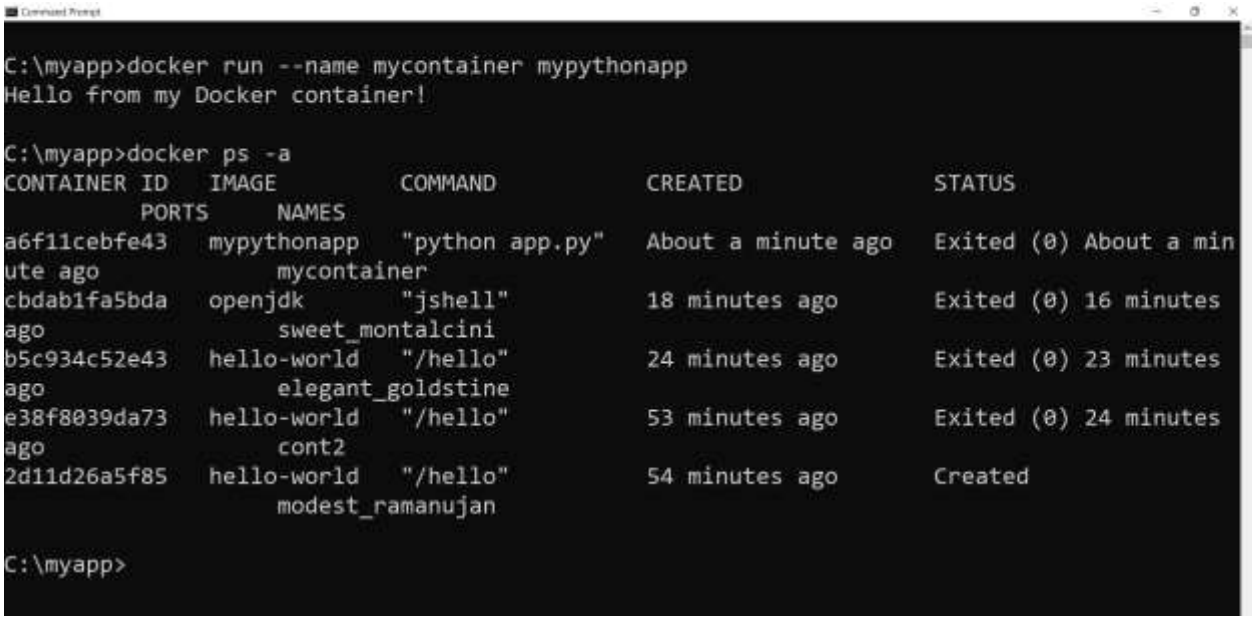
```
docker images
```

You should see something like:

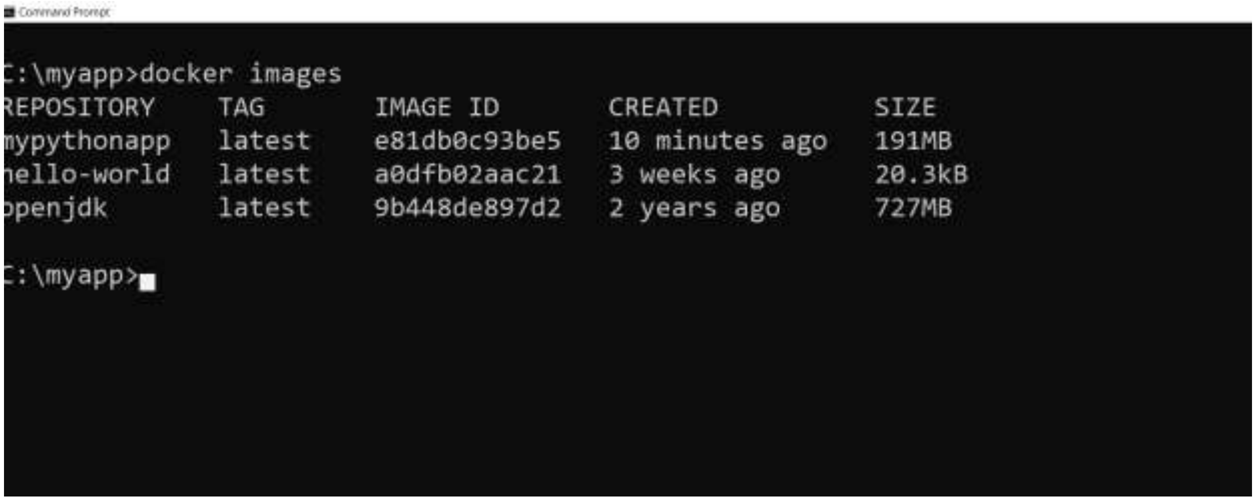
```
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
mypythonapp     latest    abcd1234efgh   5 seconds ago    120MB
```



```
C:\myapp>docker run --name mycontainer mypythonapp
Hello from my Docker container!

C:\myapp>
```

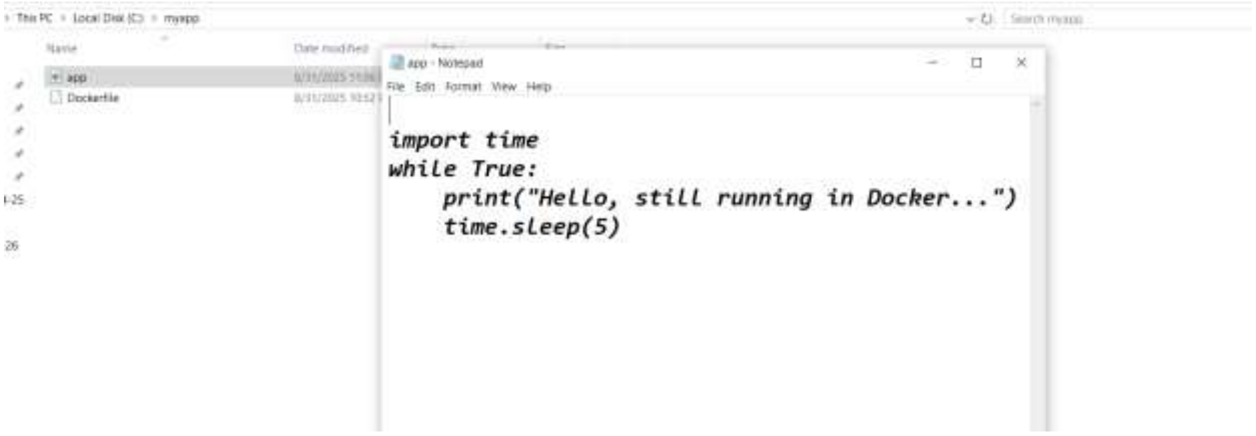


```
C:\myapp>docker run --name mycontainer mypythonapp
Hello from my Docker container!

C:\myapp>docker ps -a
CONTAINER ID   IMAGE         COMMAND           CREATED              STATUS
          PORTS     NAMES
a6f11cebfe43   mypythonapp   "python app.py"   About a minute ago   Exited (0) About a min
ute ago             mycontainer
cbdab1fa5bda   openjdk       "jshell"          18 minutes ago       Exited (0) 16 minutes
ago                 sweet_montalcini
b5c934c52e43   hello-world   "/hello"          24 minutes ago       Exited (0) 23 minutes
ago                 elegant_goldstine
e38f8039da73   hello-world   "/hello"          53 minutes ago       Exited (0) 24 minutes
ago                 cont2
2d11d26a5f85   hello-world   "/hello"          54 minutes ago       Created
                    modest_ramanujan

C:\myapp>
```

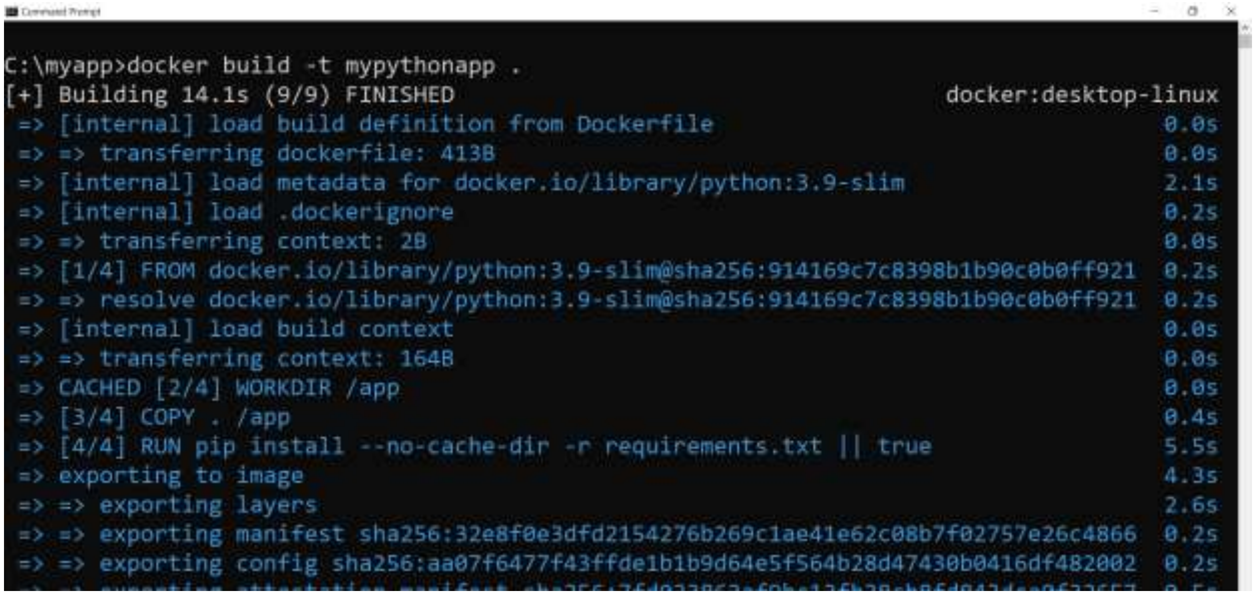


```
:\myapp>docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
mypythonapp   latest    e81db0c93be5   10 minutes ago   191MB
ello-world    latest    a0dfb02aac21   3 weeks ago      20.3kB
penjdk        latest    9b448de897d2   2 years ago      727MB

:\myapp>
```

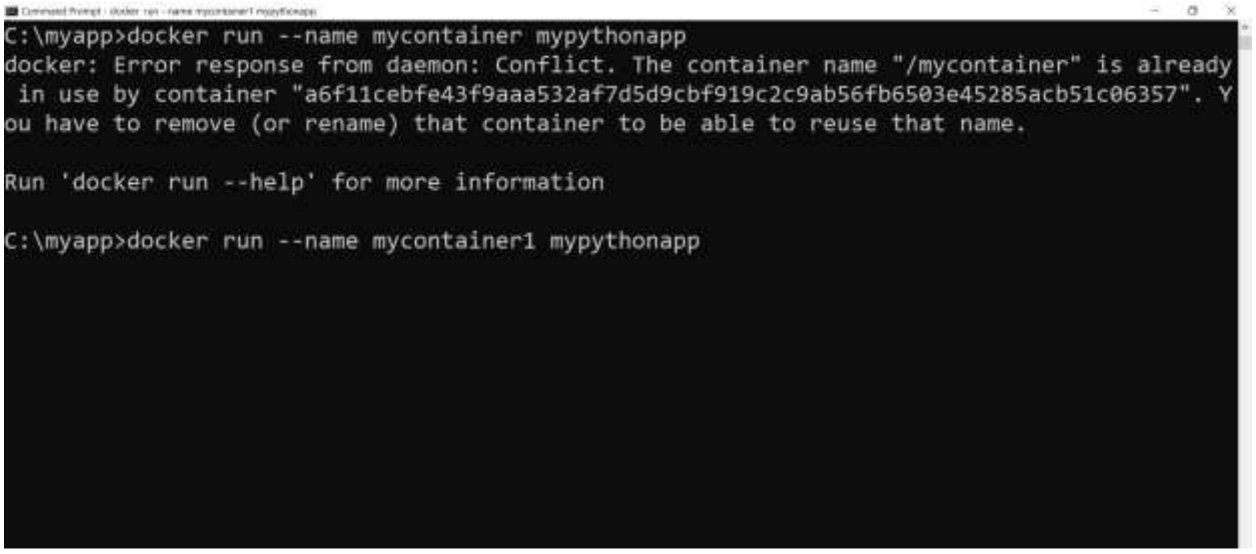
If you want it to stay running, instead of just printing once, modify **app.py** like this:



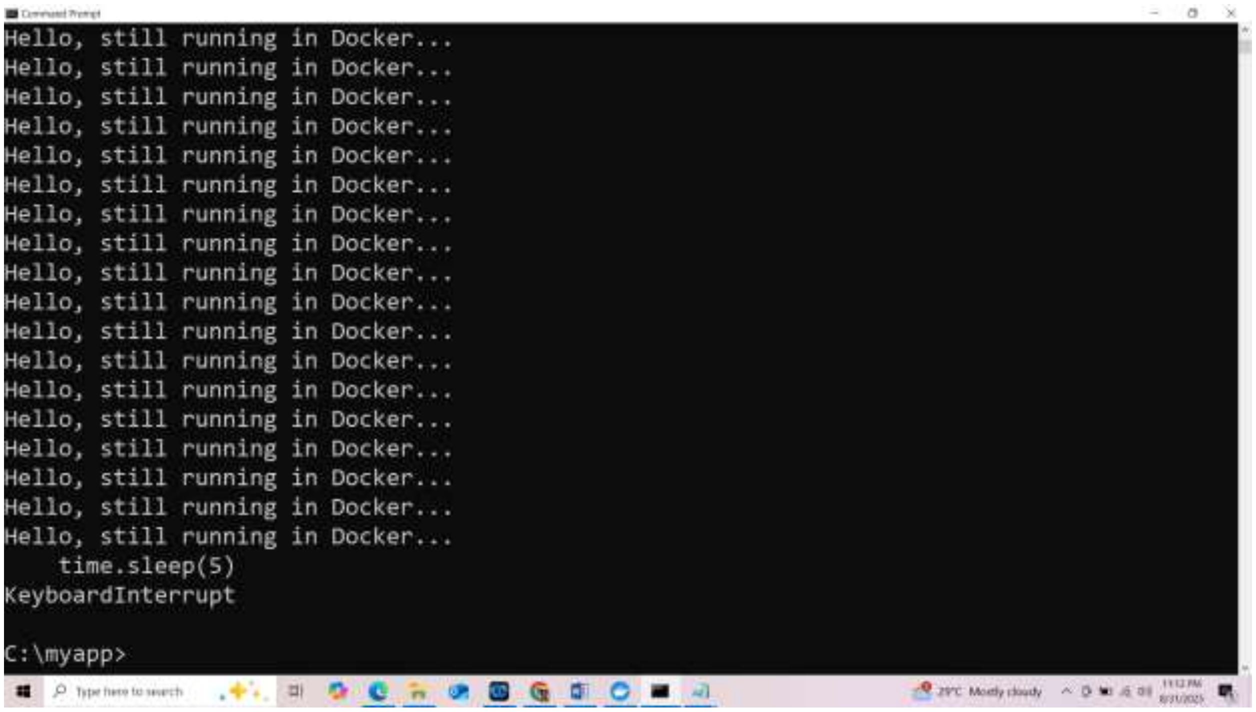
```
import time
while True:
    print("Hello, still running in Docker...")
    time.sleep(5)
```



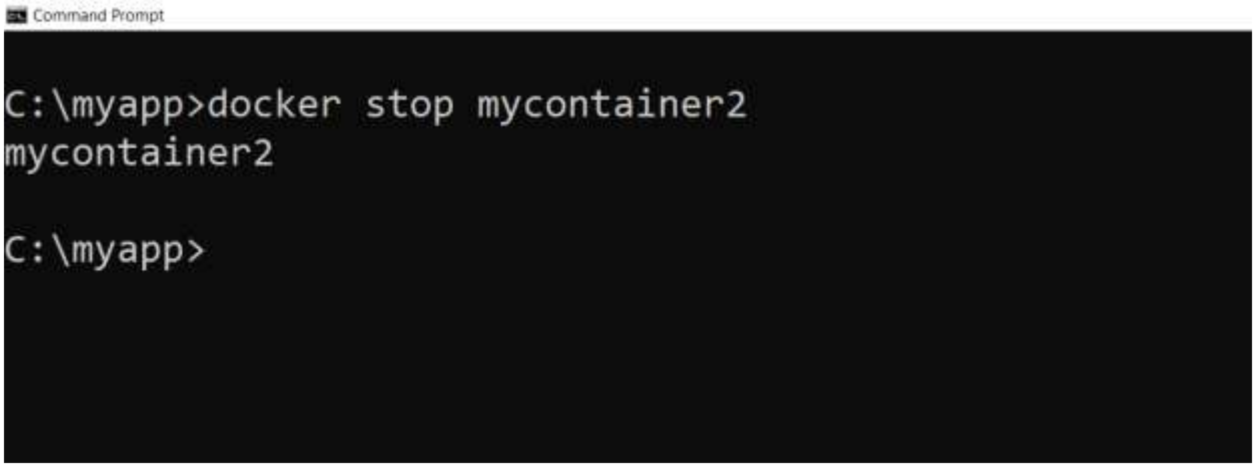
```
C:\myapp>docker build -t mypythonapp .
[+] Building 14.1s (9/9) FINISHED                                        docker:desktop-linux
 => [internal] load build definition from Dockerfile                                     0.0s
 => => transferring dockerfile: 413B                                                     0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                       2.1s
 => [internal] load .dockerignore                                                        0.2s
 => => transferring context: 2B                                                          0.0s
 => [1/4] FROM docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921      0.2s
 => => resolve docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921      0.2s
 => [internal] load build context                                                        0.0s
 => => transferring context: 164B                                                        0.0s
 => CACHED [2/4] WORKDIR /app                                                            0.0s
 => [3/4] COPY . /app                                                                    0.4s
 => [4/4] RUN pip install --no-cache-dir -r requirements.txt || true                     5.5s
 => exporting to image                                                                   4.3s
 => => exporting layers                                                                  2.6s
 => => exporting manifest sha256:32e8f0e3dfd2154276b269c1ae41e62c08b7f02757e26c4866      0.2s
 => => exporting config sha256:aa07f6477f43ffde1b1b9d64e5f564b28d47430b0416df482002      0.2s
```



```
C:\myapp>docker run --name mycontainer mypythonapp
docker: Error response from daemon: Conflict. The container name "/mycontainer" is already
 in use by container "a6f11cebfe43f9aaa532af7d5d9cbf919c2c9ab56fb6503e45285acb51c06357". Y
ou have to remove (or rename) that container to be able to reuse that name.

Run 'docker run --help' for more information

C:\myapp>docker run --name mycontainer1 mypythonapp
```



```
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
Hello, still running in Docker...
    time.sleep(5)
KeyboardInterrupt

C:\myapp>
```



```
C:\myapp>docker stop mycontainer2
mycontainer2

C:\myapp>
```

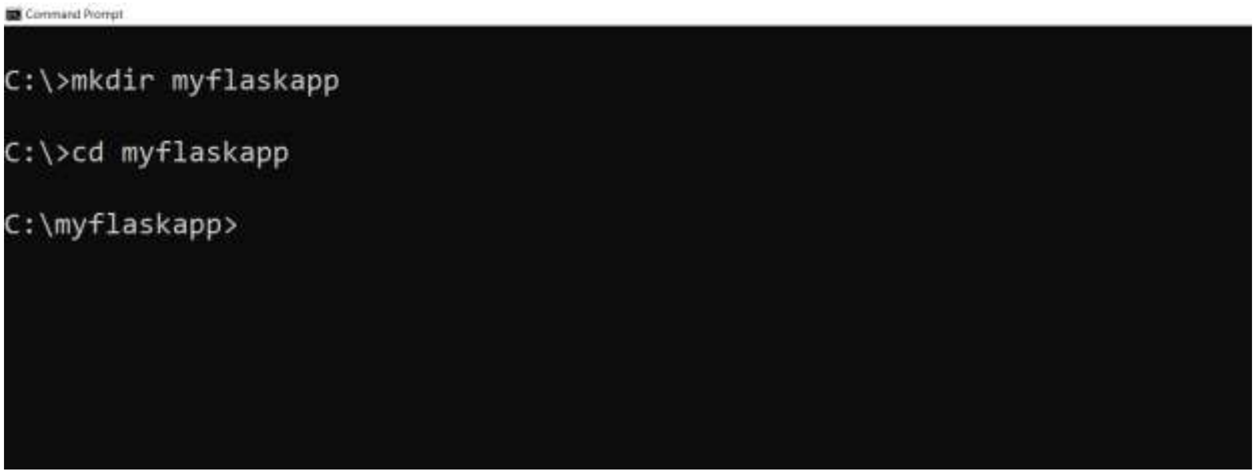
Simple working Flask example

Create your project files

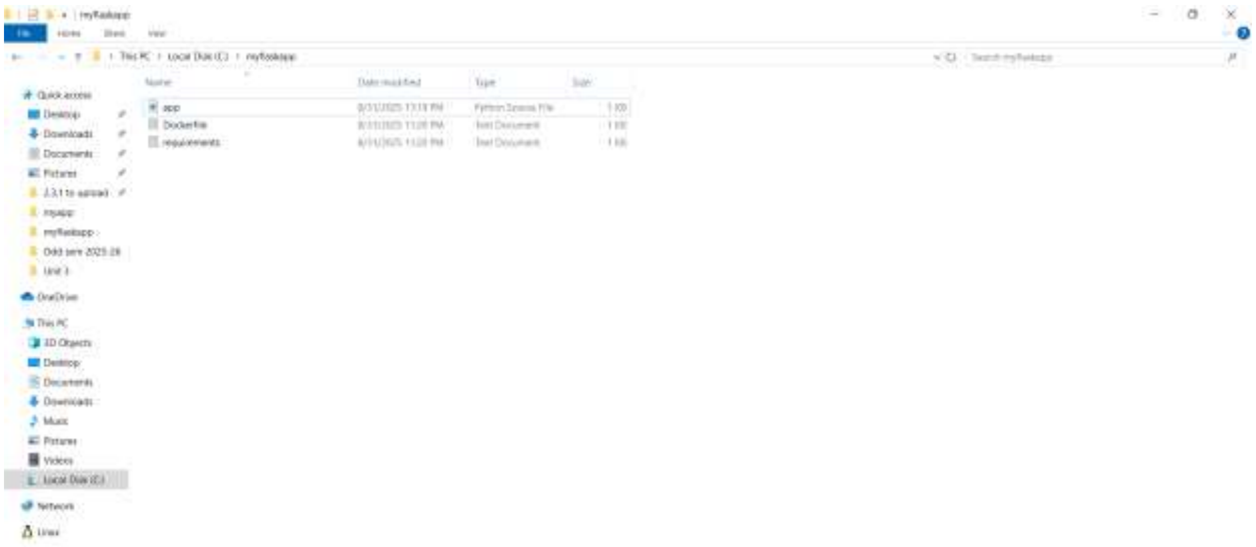
App.py

Requirements.txt

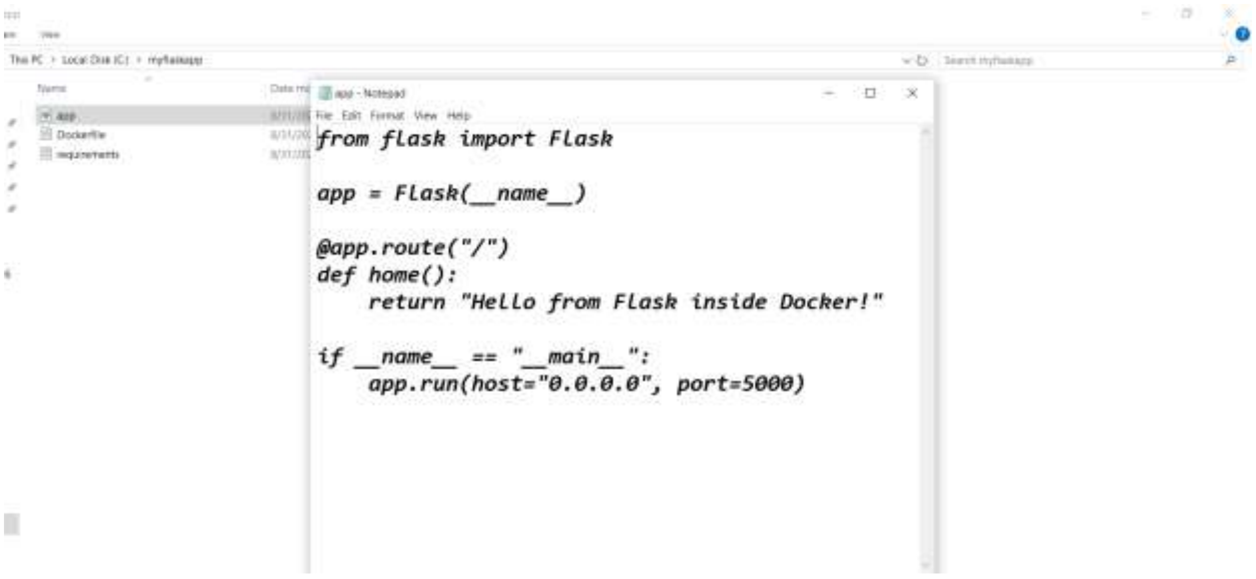
Dockerfile



```
C:\>mkdir myflaskapp

C:\>cd myflaskapp

C:\myflaskapp>
```



Name	Date modified	Type	Size
app	8/31/2025 11:19 PM	Python Source File	1 KB
Dockerfile	8/31/2025 11:20 PM	Text Document	1 KB
requirements	8/31/2025 11:20 PM	Text Document	1 KB



```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def home():
    return "Hello from Flask inside Docker!"

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

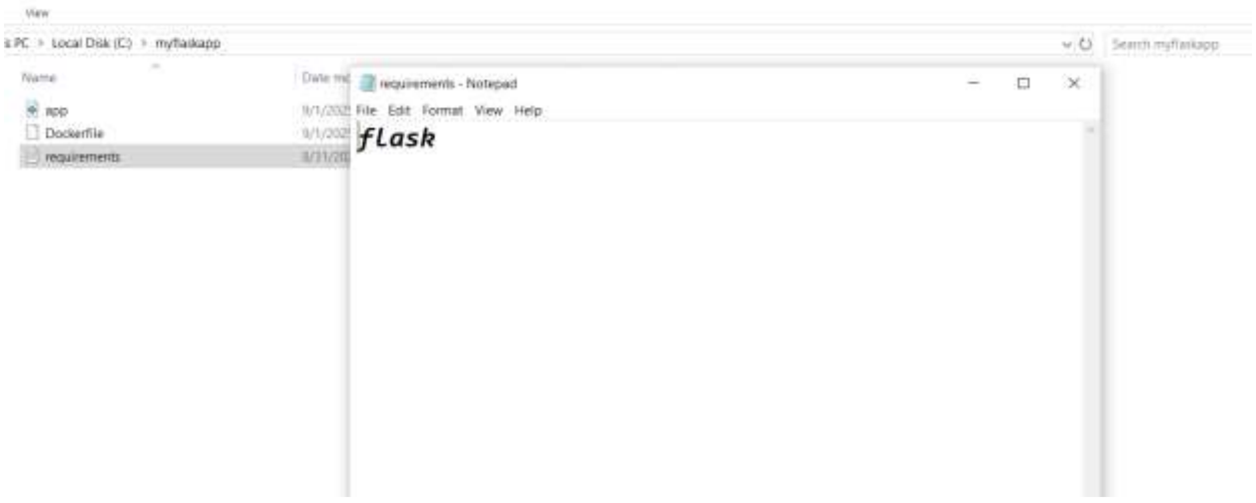
Flask(__name__) creates a tiny web app.

@app.route("/") defines one URL (the homepage) that returns text.

host="0.0.0.0" makes the app listen on **all** interfaces inside the container. This is critical and Docker can't forward traffic to 127.0.0.1 (loopback only).

port=5000 is the port **inside** the container where Flask listens.

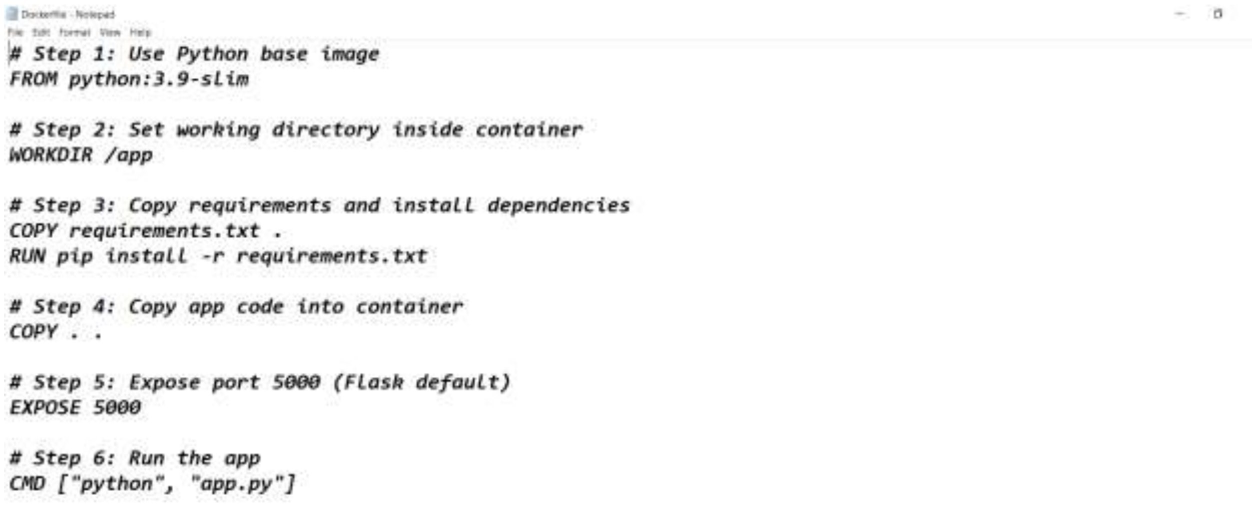
A list of Python packages your app needs. Docker will pip install these during the image build.



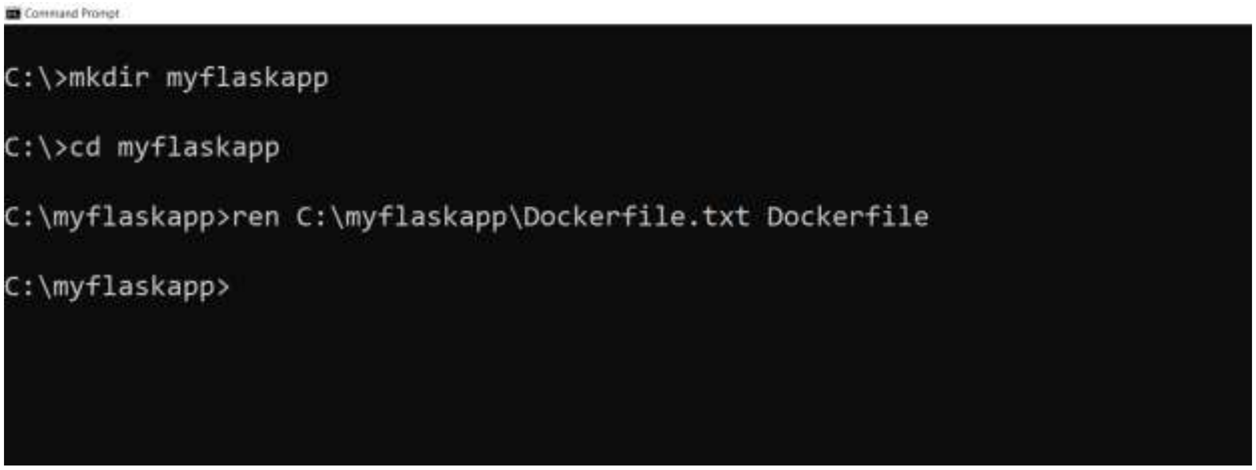
```
flask
```

Dockerfile

- ren C:\myflaskapp\Dockerfile.txt Dockerfile
- Docker builds in **layers**. If only your app code changes (not the dependencies), Docker will reuse the cached layer where dependencies were installed, so builds are much faster.
- Note: `EXPOSE 5000` is **documentation** inside the image, it does **not** publish the port to your host. Port publishing happens when you run the container with `-p`.



```
# Step 1: Use Python base image
FROM python:3.9-slim

# Step 2: Set working directory inside container
WORKDIR /app

# Step 3: Copy requirements and install dependencies
COPY requirements.txt .
RUN pip install -r requirements.txt

# Step 4: Copy app code into container
COPY . .

# Step 5: Expose port 5000 (Flask default)
EXPOSE 5000

# Step 6: Run the app
CMD ["python", "app.py"]
```



```
C:\>mkdir myflaskapp

C:\>cd myflaskapp

C:\myflaskapp>ren C:\myflaskapp\Dockerfile.txt Dockerfile

C:\myflaskapp>
```

Build the docker image

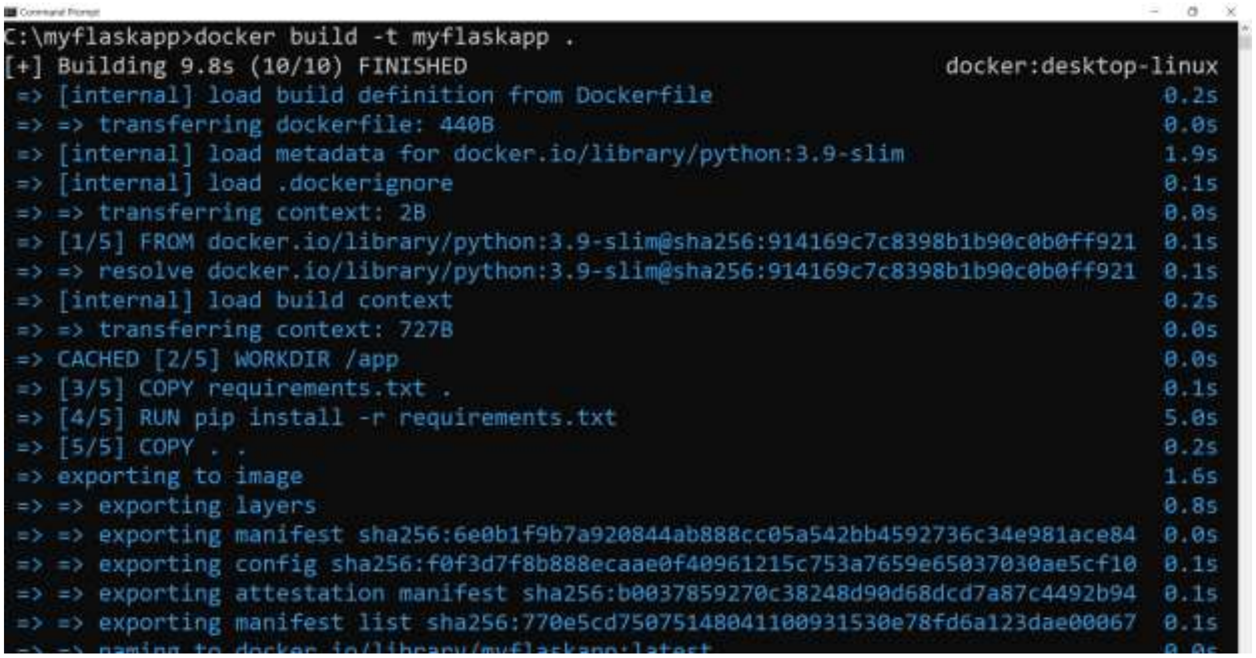
docker build tells Docker to construct an image using the instructions in Dockerfile.

-t myflaskapp tags (names) the resulting image as mypflaskapp:latest.

(dot) is the **build context** = “send everything in the current folder to the Docker daemon so it can COPY files into the image.”

1. Pull python:3.9-slim if you don’t have it.
2. Create /app and set it as the working directory.
3. Copy requirements.txt and run pip install.
4. Copy your source code.
5. Record EXPOSE 5000.
6. Set the default command to python app.py.

Result: a **portable image** that contains everything your app needs to run.



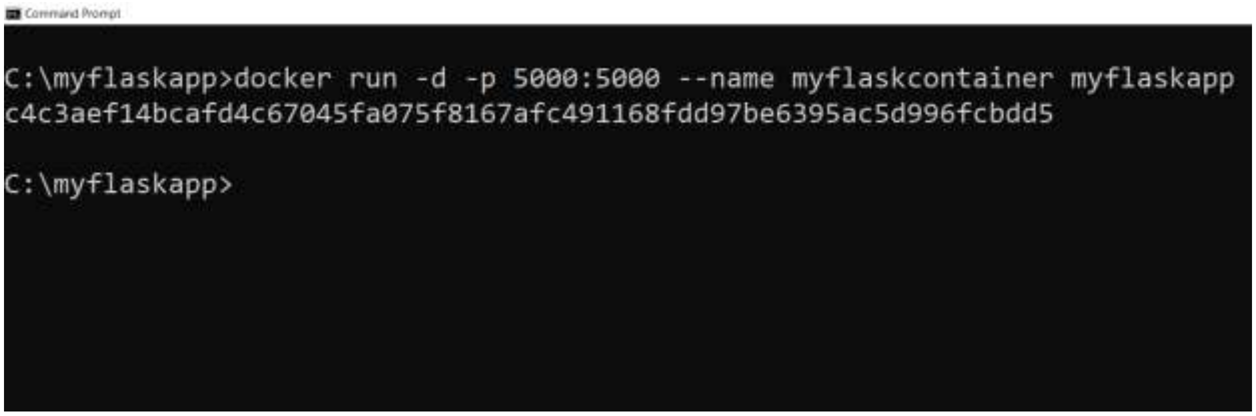
```
C:\myflaskapp>docker build -t myflaskapp .
[+] Building 9.8s (10/10) FINISHED                                        docker:desktop-linux
 => [internal] load build definition from Dockerfile                                      0.2s
 => => transferring dockerfile: 440B                                                      0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                        1.9s
 => [internal] load .dockerignore                                                         0.1s
 => => transferring context: 2B                                                           0.0s
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921  0.1s
 => => resolve docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921  0.1s
 => [internal] load build context                                                         0.2s
 => => transferring context: 727B                                                         0.0s
 => CACHED [2/5] WORKDIR /app                                                             0.0s
 => [3/5] COPY requirements.txt .                                                         0.1s
 => [4/5] RUN pip install -r requirements.txt                                             5.0s
 => [5/5] COPY . .                                                                        0.2s
 => exporting to image                                                                    1.6s
 => => exporting layers                                                                   0.8s
 => => exporting manifest sha256:6e0b1f9b7a920844ab888cc05a542bb4592736c34e981ace84  0.0s
 => => exporting config sha256:f0f3d7f8b888ecaae0f40961215c753a7659e65037030ae5cf10  0.1s
 => => exporting attestation manifest sha256:b0037859270c38248d90d68dcd7a87c4492b94  0.1s
 => => exporting manifest list sha256:770e5cd7507514804110093153e78fd6a123dae00067  0.1s
 => => naming to docker.io/library/myflaskapp:latest                                      0.0s
```

Run the container in detached mode

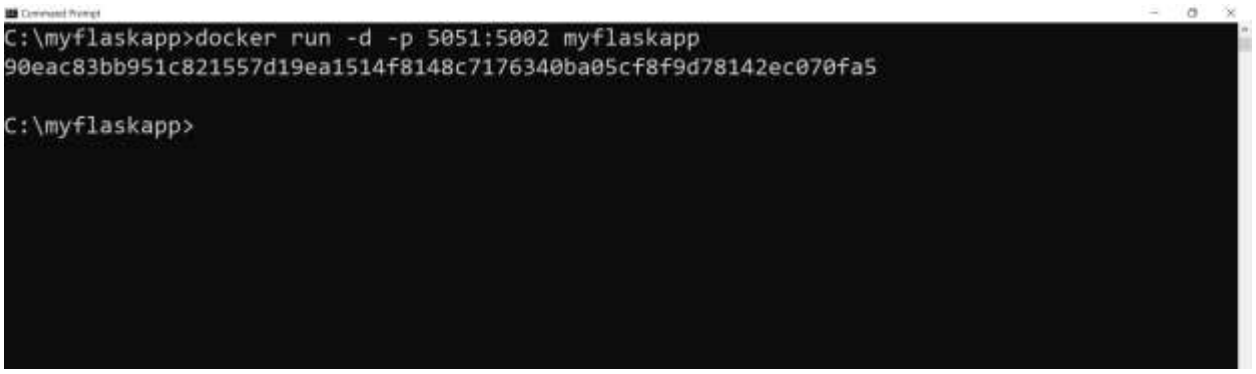
A **container** is a *running instance* of the image.

-p 5000:5000 maps **host port 5000 → container port 5000**
(format: HOST:CONTAINER). So when you open http://localhost:5000 on your machine, Docker forwards that traffic into the container’s port 5000 where Flask is listening.

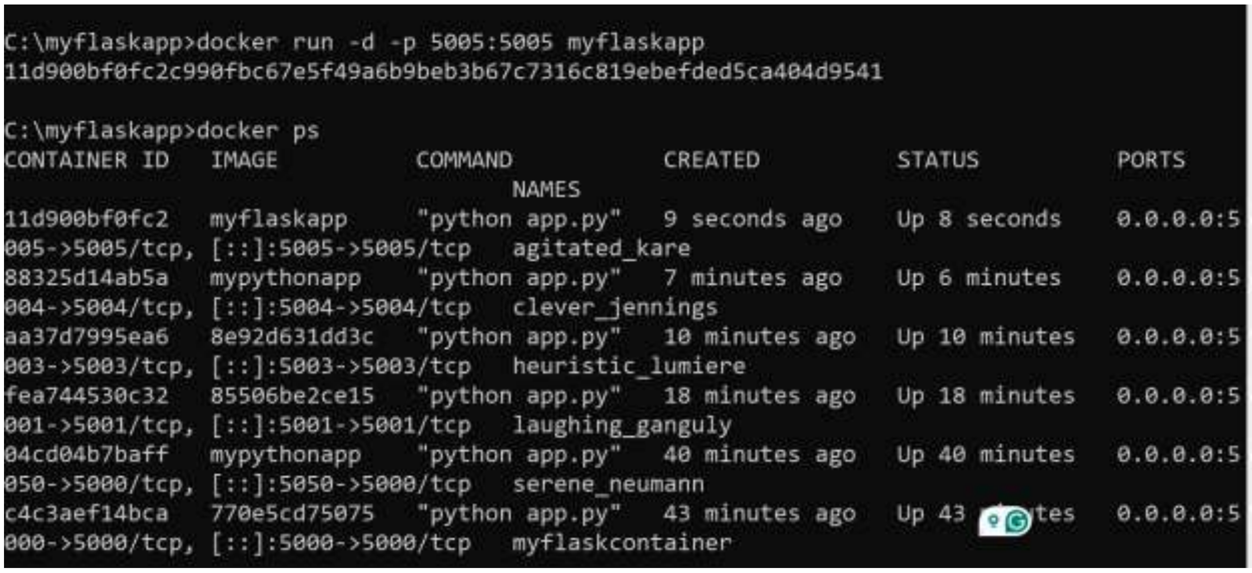
Docker starts the container and executes python app.py (from CMD).



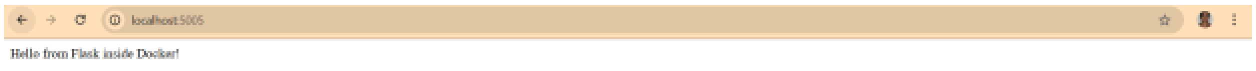
```
C:\myflaskapp>docker run -d -p 5000:5000 --name myflaskcontainer myflaskapp
c4c3aef14bcafd4c67045fa075f8167afc491168fdd97be6395ac5d996fcbdd5

C:\myflaskapp>
```



```
C:\myflaskapp>docker run -d -p 5051:5002 myflaskapp
90eac83bb951c821557d19ea1514f8148c7176340ba05cf8f9d78142ec070fa5

C:\myflaskapp>
```

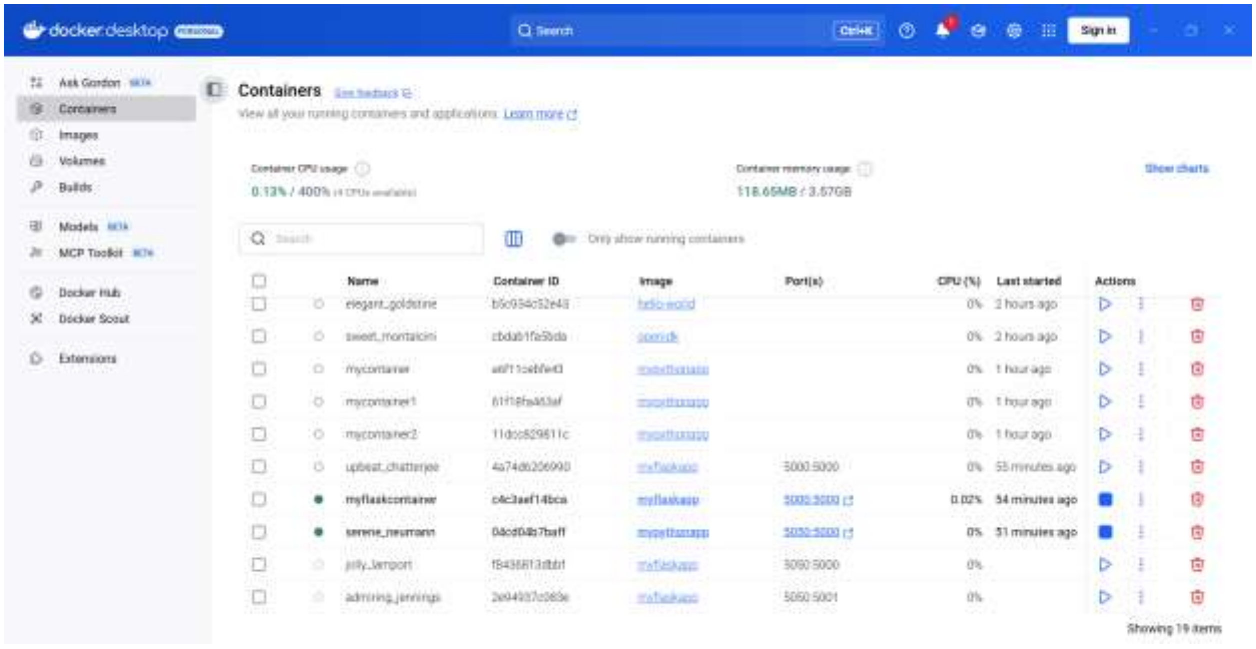


```
C:\myflaskapp>docker run -d -p 5005:5005 myflaskapp
11d900bf0fc2c990fbc67e5f49a6b9beb3b67c7316c819ebefded5ca404d9541

C:\myflaskapp>docker ps
CONTAINER ID   IMAGE          COMMAND           CREATED           STATUS          PORTS
                              NAMES
11d900bf0fc2   myflaskapp     "python app.py"   9 seconds ago     Up 8 seconds    0.0.0.0:5
005->5005/tcp, [::]:5005->5005/tcp   agitated_kare
88325d14ab5a   mypythonapp    "python app.py"   7 minutes ago     Up 6 minutes    0.0.0.0:5
004->5004/tcp, [::]:5004->5004/tcp   clever_jennings
aa37d7995ea6   8e92d631dd3c   "python app.py"   10 minutes ago    Up 10 minutes   0.0.0.0:5
003->5003/tcp, [::]:5003->5003/tcp   heuristic_lumiere
fea744530c32   85506be2ce15   "python app.py"   18 minutes ago    Up 18 minutes   0.0.0.0:5
001->5001/tcp, [::]:5001->5001/tcp   laughing_ganguly
04cd04b7baff   mypythonapp    "python app.py"   40 minutes ago    Up 40 minutes   0.0.0.0:5
050->5000/tcp, [::]:5050->5000/tcp   serene_neumann
c4c3aef14bca   770e5cd75075   "python app.py"   43 minutes ago    Up 43 [illegible]tes   0.0.0.0:5
000->5000/tcp, [::]:5000->5000/tcp   myflaskcontainer
```



localhost:5005

Hello from Flask inside Docker!



Containers

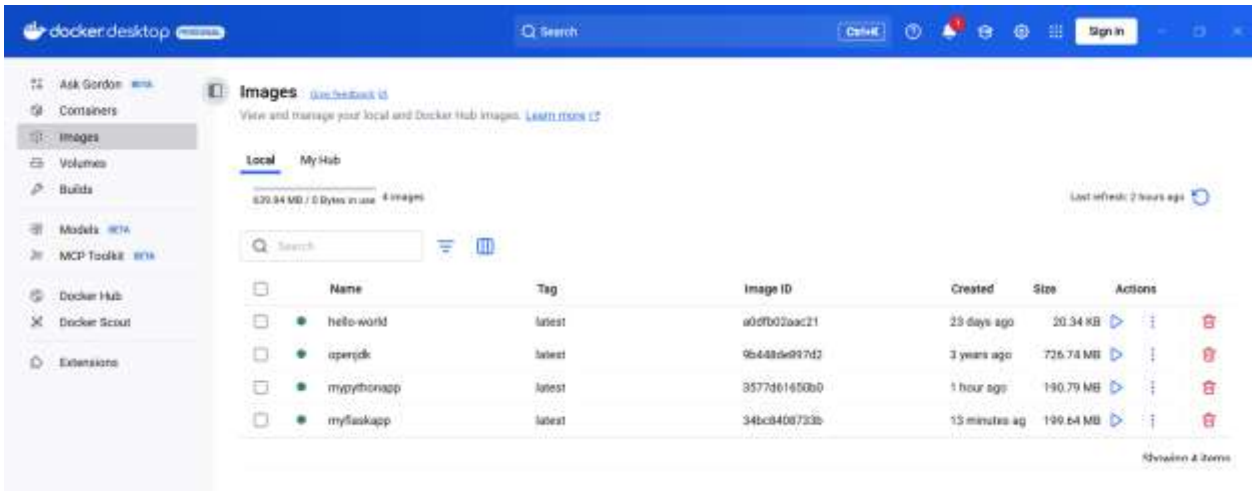
View all your running containers and applications.

Container CPU usage
0.13% / 400% (4 CPUs available)

Container memory usage
118.65MB / 3.57GB

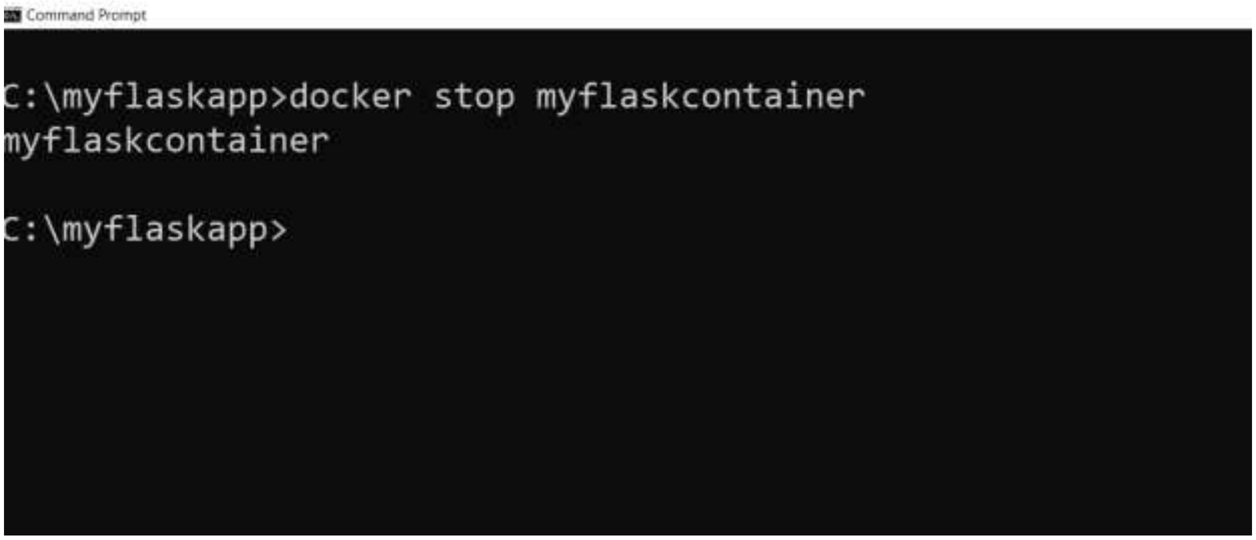
Name	Container ID	Image	Port(s)	CPU (%)	Last started
elegant_goldstine	b5c934c52e43	hello-world		0%	2 hours ago
sweet_montalcini	cbdab1fa5bda	openjdk		0%	2 hours ago
mycontainer	a6f11cebfe43	mypythonapp		0%	1 hour ago
mycontainer1	61f18fa463af	mypythonapp		0%	1 hour ago
mycontainer2	11dcc829611c	mypythonapp		0%	1 hour ago
upbeat_chatterjee	4a74d6206990	myflaskapp	5000:5000	0%	55 minutes ago
myflaskcontainer	c4c3aef14bca	myflaskapp	5000:5000	0.02%	54 minutes ago
serene_neumann	0dcd04b7baff	mypythonapp	5050:5000	0%	51 minutes ago
jolly_lamport	f8436813dbbf	myflaskapp	5050:5000	0%	
admiring_jennings	2e94937c083e	myflaskapp	5050:5001	0%	

Showing 19 items

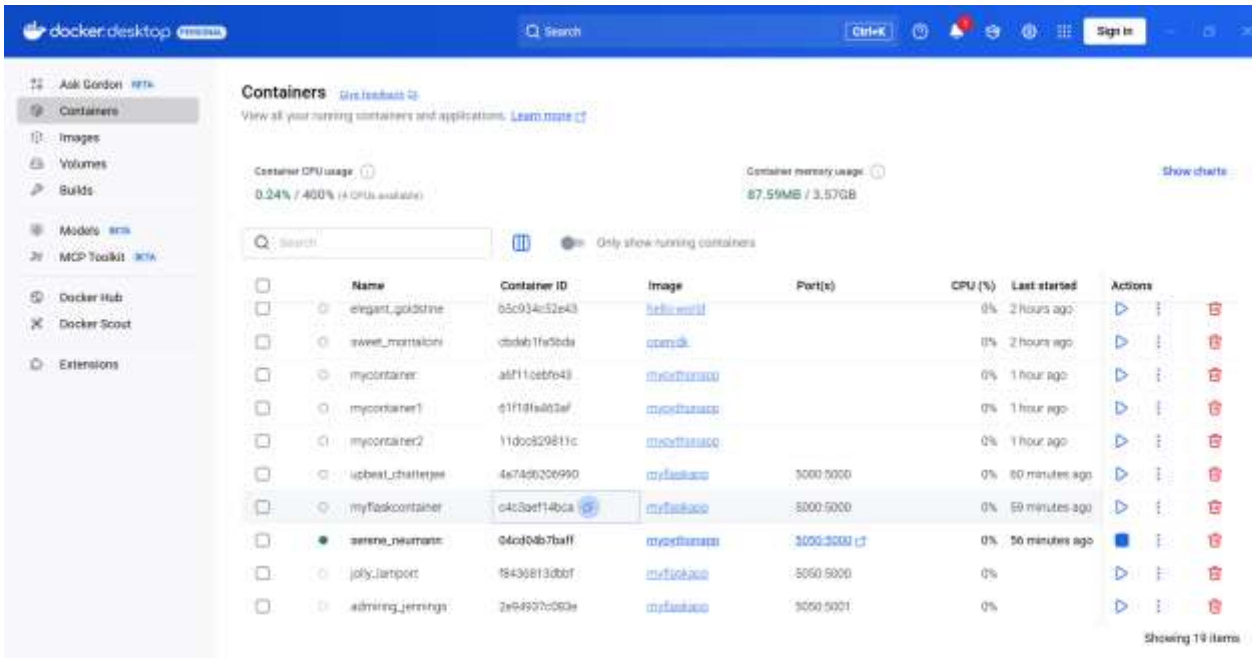


Name	Tag	Image ID	Created	Size	Actions
hello-world	latest	a0dfb02aac21	23 days ago	20.34 KB	
openjdk	latest	9b448de997d2	3 years ago	726.74 MB	
mypythonapp	latest	3577d61650b0	1 hour ago	190.79 MB	
myflaskapp	latest	34bc8408733b	13 minutes ag	199.64 MB	

Showing 4 items



```
C:\myflaskapp>docker stop myflaskcontainer
myflaskcontainer

C:\myflaskapp>
```



Containers

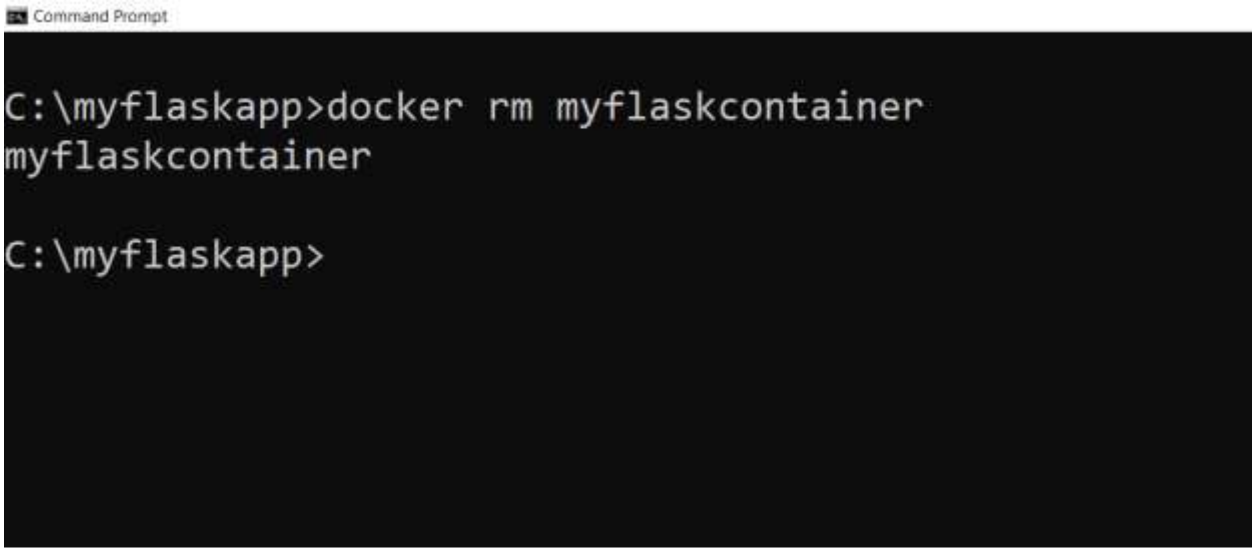
View all your running containers and applications. Learn more

Container CPU usage: 0.24% / 400% (4 CPUs available)

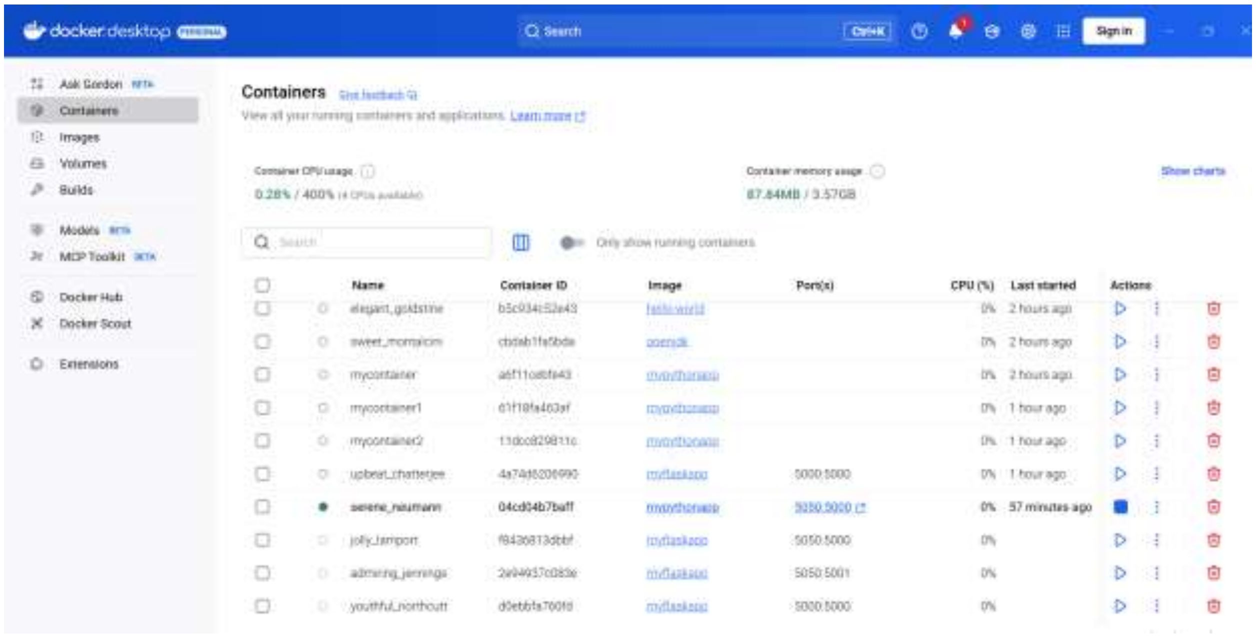
Container memory usage: 87.59MB / 3.57GB

Name	Container ID	Image	Port(s)	CPU (%)	Last started
elegant_goldstine	b5c934c52e43	hello-world		0%	2 hours ago
sweet_montalcini	dbdab1fa5bda	openjdk		0%	2 hours ago
mycontainer	a6f11cebfe43	mypythonapp		0%	1 hour ago
mycontainer1	61f1dfa863af	mypythonapp		0%	1 hour ago
mycontainer2	11dcc829811c	mypythonapp		0%	1 hour ago
upbeat_chatterjee	4a74d6206990	myflaskapp	5000:5000	0%	60 minutes ago
myflaskcontainer	c4c3aef14bca	myflaskapp	5000:5000	0%	59 minutes ago
serene_neumann	04cd04b7baff	mypythonapp	5050:5000	0%	56 minutes ago
jolly_lamport	f8436813dbbf	myflaskapp	5050:5000	0%	
admiring_jennings	2e94937c080e	myflaskapp	5050:5001	0%	

Showing 19 items



```
C:\myflaskapp>docker rm myflaskcontainer
myflaskcontainer

C:\myflaskapp>
```



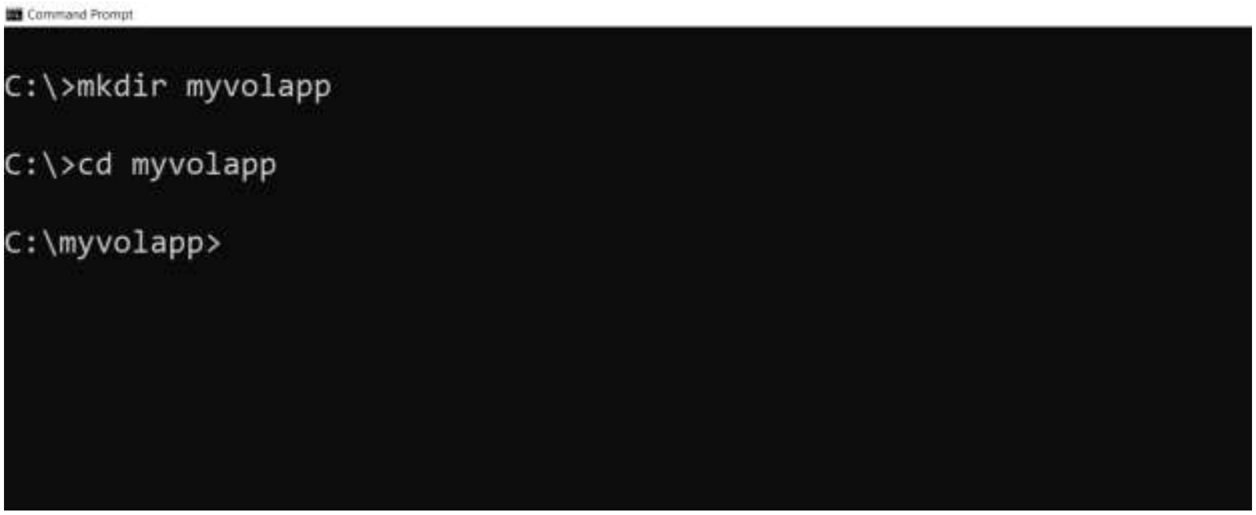
Container CPU usage
0.28% / 400% (4 CPUs available)

Container memory usage
87.84MB / 3.57GB

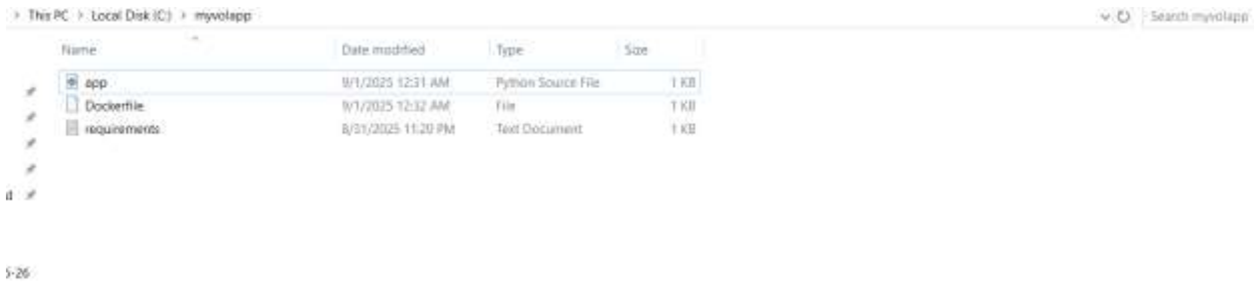
Name	Container ID	Image	Port(s)	CPU (%)	Last started
elegant_goldstine	b5c934c52e43	hello-world		0%	2 hours ago
sweet_mormalcm	cbdab1fa5bda	openjdk		0%	2 hours ago
mycontainer	a6f11ca6fe43	mypythonapp		0%	2 hours ago
mycontainer1	61f18fa463af	mypythonapp		0%	1 hour ago
mycontainer2	11dcc829811c	mypythonapp		0%	1 hour ago
upbeat_chatterjee	4a74d6206990	myflaskapp	5000:5000	0%	1 hour ago
serene_neumann	04cd04b7baff	mypythonapp	5050:5000	0%	57 minutes ago
jolly_lamport	f8436813dbbf	myflaskapp	5050:5000	0%	
admiring_jennings	2e94937c083e	myflaskapp	5050:5001	0%	
youthful_northcutt	d0ebbfa760fd	myflaskapp	5000:5000	0%	

A **volume** in Docker is a way to persist data outside the container's writable layer.

- Containers are temporary → if you delete a container, its data is lost.
- Volumes allow you to **store data on your host machine** so that even if the container is removed, the data persists.

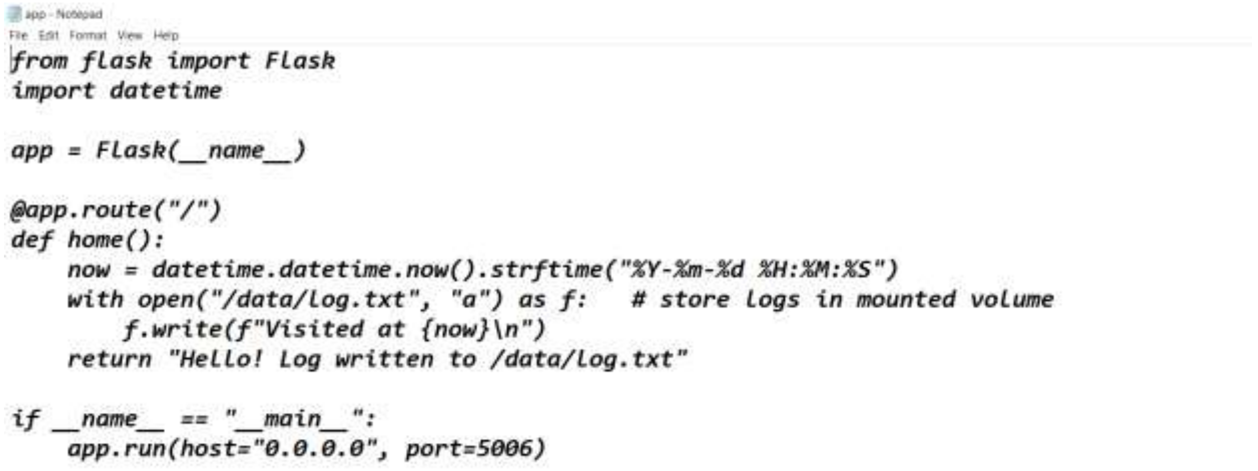


```
C:\>mkdir myvolapp

C:\>cd myvolapp

C:\myvolapp>
```

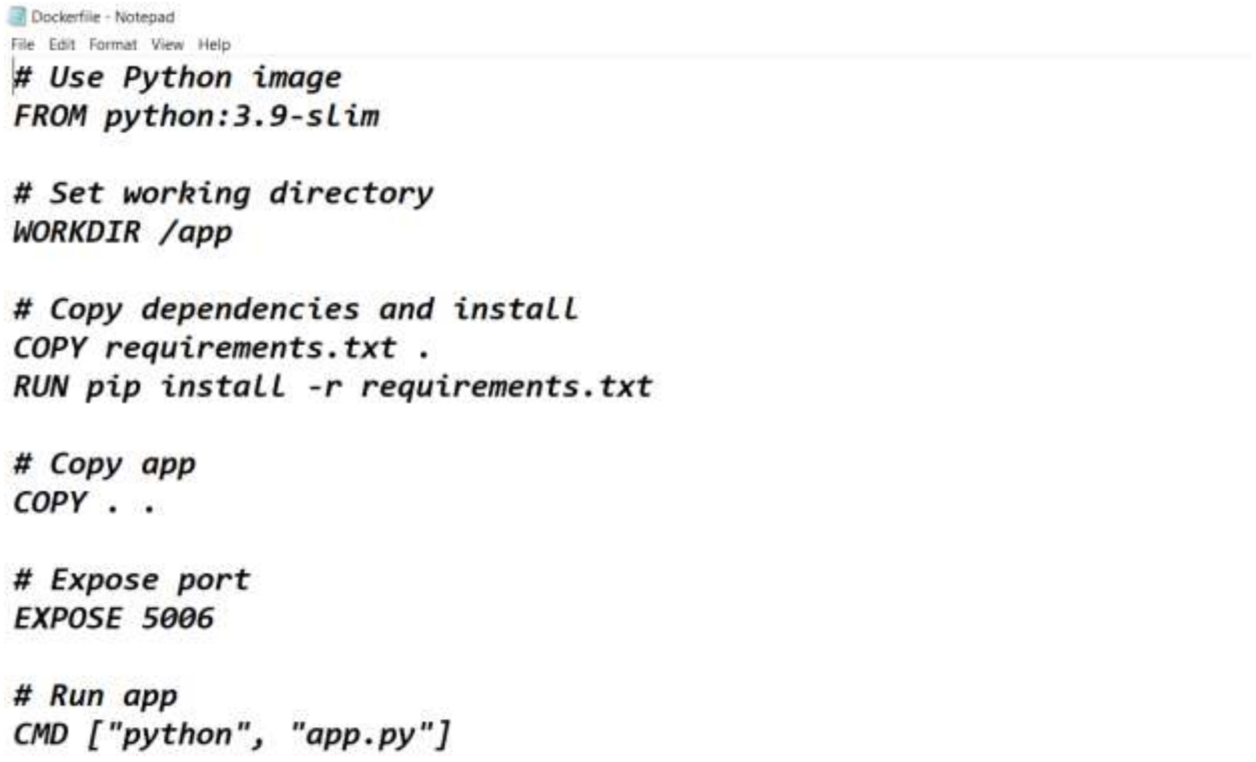


This PC > Local Disk (C:) > myvolapp

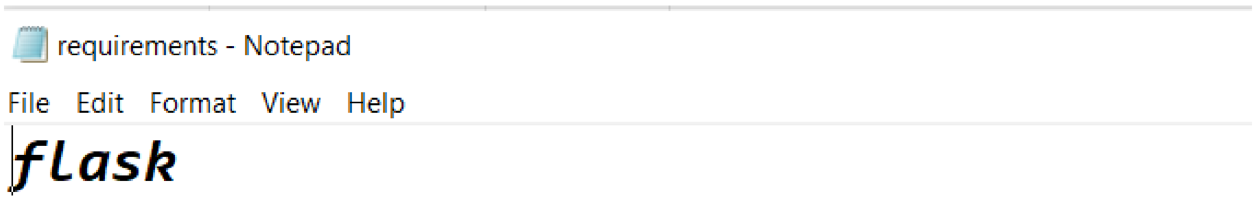
Name	Date modified	Type	Size
app	9/1/2025 12:31 AM	Python Source File	1 KB
Dockerfile	9/1/2025 12:32 AM	File	1 KB
requirements	8/31/2025 11:20 PM	Text Document	1 KB



```python
from flask import Flask
import datetime

app = Flask(__name__)

@app.route("/")
def home():
    now = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open("/data/log.txt", "a") as f:   # store logs in mounted volume
        f.write(f"Visited at {now}\n")
    return "Hello! Log written to /data/log.txt"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5006)
```

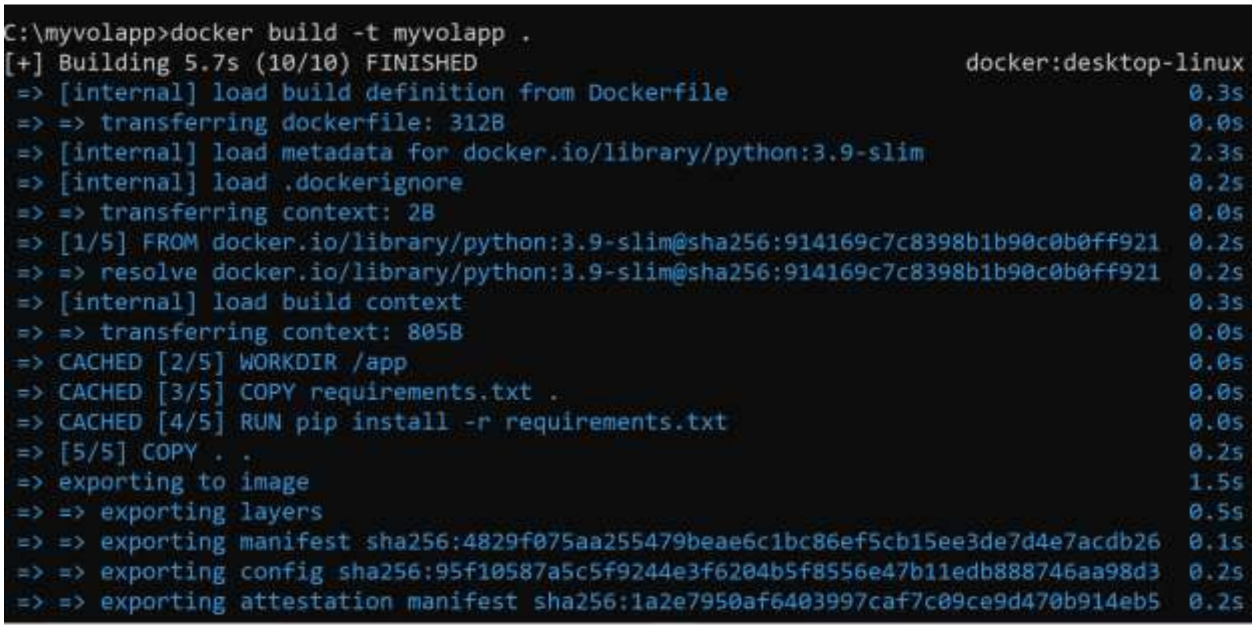


```
# Use Python image
FROM python:3.9-slim

# Set working directory
WORKDIR /app

# Copy dependencies and install
COPY requirements.txt .
RUN pip install -r requirements.txt

# Copy app
COPY . .

# Expose port
EXPOSE 5006

# Run app
CMD ["python", "app.py"]
```



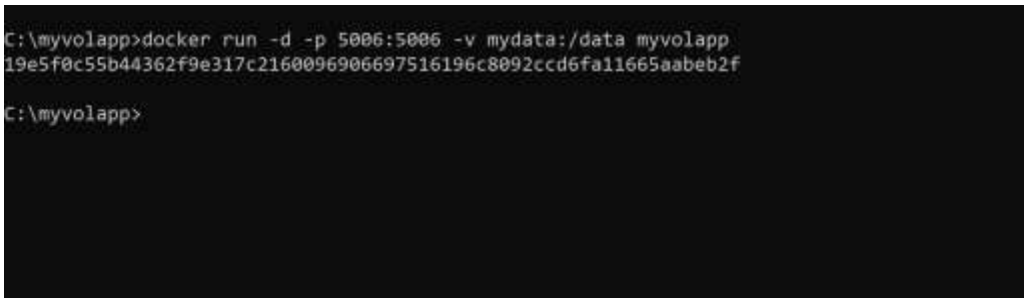
```
flask
```

Build the image



```
C:\myvolapp>docker build -t myvolapp .
[+] Building 5.7s (10/10) FINISHED                                     docker:desktop-linux
 => [internal] load build definition from Dockerfile                                   0.3s
 => => transferring dockerfile: 312B                                                   0.0s
 => [internal] load metadata for docker.io/library/python:3.9-slim                     2.3s
 => [internal] load .dockerignore                                                      0.2s
 => => transferring context: 2B                                                        0.0s
 => [1/5] FROM docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921  0.2s
 => => resolve docker.io/library/python:3.9-slim@sha256:914169c7c8398b1b90c0b0ff921  0.2s
 => [internal] load build context                                                      0.3s
 => => transferring context: 805B                                                      0.0s
 => CACHED [2/5] WORKDIR /app                                                          0.0s
 => CACHED [3/5] COPY requirements.txt .                                               0.0s
 => CACHED [4/5] RUN pip install -r requirements.txt                                   0.0s
 => [5/5] COPY . .                                                                     0.2s
 => exporting to image                                                                 1.5s
 => => exporting layers                                                                0.5s
 => => exporting manifest sha256:4829f075aa255479beae6c1bc86ef5cb15ee3de7d4e7acdb26  0.1s
 => => exporting config sha256:95f10587a5c5f9244e3f6204b5f8556e47b11edb888746aa98d3  0.2s
 => => exporting attestation manifest sha256:1a2e7950af6403997caf7c09ce9d470b914eb5  0.2s
```

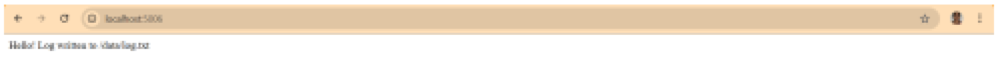
Run container with a volume



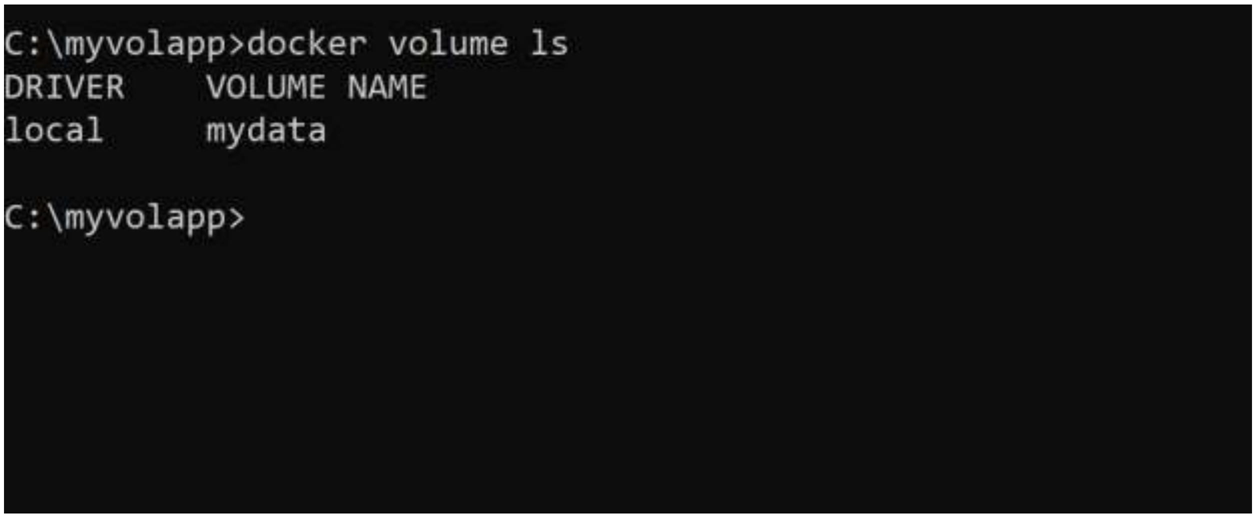
```
C:\myvolapp>docker run -d -p 5006:5006 -v mydata:/data myvolapp
19e5f0c55b44362f9e317c2160096906697516196c8092ccd6fa11665aabeb2f

C:\myvolapp>
```

Access the app

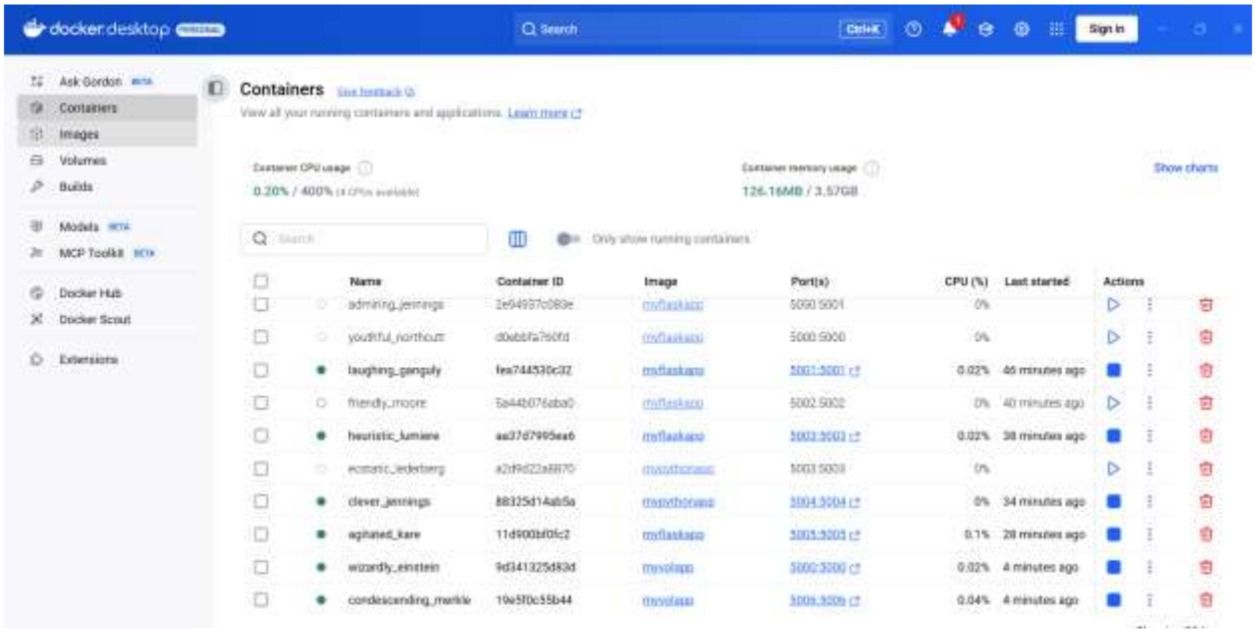
Each time you refresh, a new timestamp is written to `/data/log.txt` inside the container (which is actually stored in the volume).



```
localhost:5006
Hello! Log written to /data/log.txt
```



```
C:\myvolapp>docker volume ls
DRIVER    VOLUME NAME
local     mydata

C:\myvolapp>
```



Containers Give feedback

View all your running containers and applications. Learn more

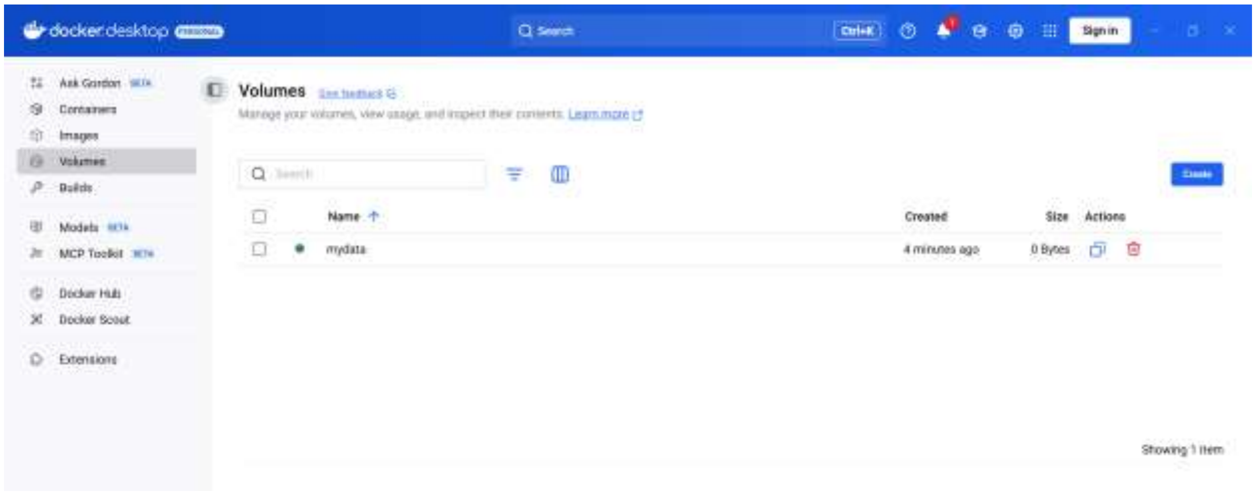
Container CPU usage
0.20% / 400% (4 CPUs available)

Container memory usage
126.16MB / 3.57GB

Show charts

Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
○	admiring_jennings	2e94937c083e	myflaskapp	5050:5001	0%		
○	youthful_northcutt	d0ebbfa760fd	myflaskapp	5000:5000	0%		
●	laughing_ganguly	fea744530c32	myflaskapp	5001:5001	0.02%	46 minutes ago	
○	friendly_moore	5a44b076aba0	myflaskapp	5002:5002	0%	40 minutes ago	
●	heuristic_lumiere	aa37d7995ea6	myflaskapp	5003:5003	0.02%	38 minutes ago	
○	ecstatic_lederberg	a2d9d22a8870	mypythonapp	5003:5003	0%		
●	clever_jennings	88325d14ab5a	mypythonapp	5004:5004	0%	34 minutes ago	
●	agitated_kare	11d900bf0fc2	myflaskapp	5005:5005	0.1%	28 minutes ago	
●	wizardly_einstein	9d341325d83d	myvolapp	5000:5000	0.02%	4 minutes ago	
●	condescending_merkle	19e5f0c55b44	myvolapp	5006:5006	0.04%	4 minutes ago	

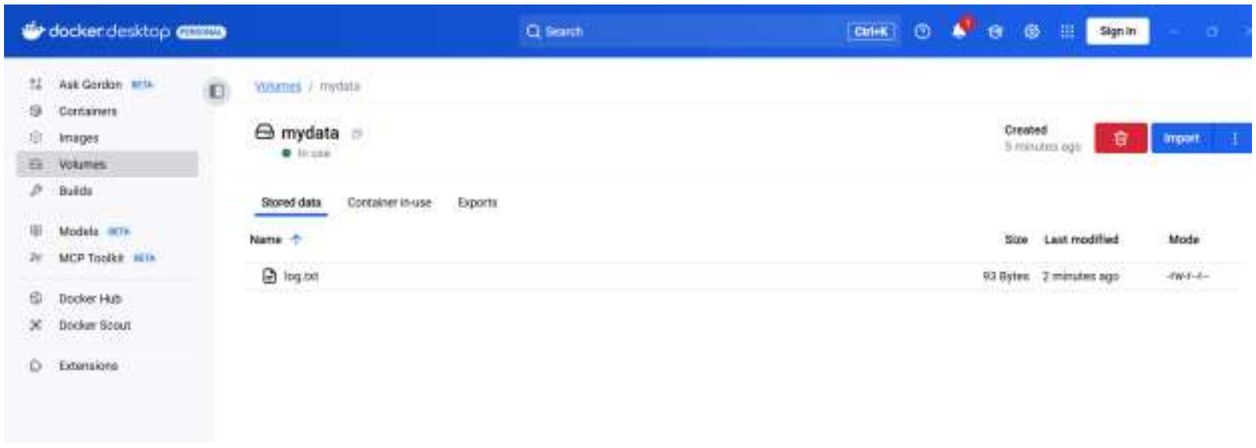


Volumes

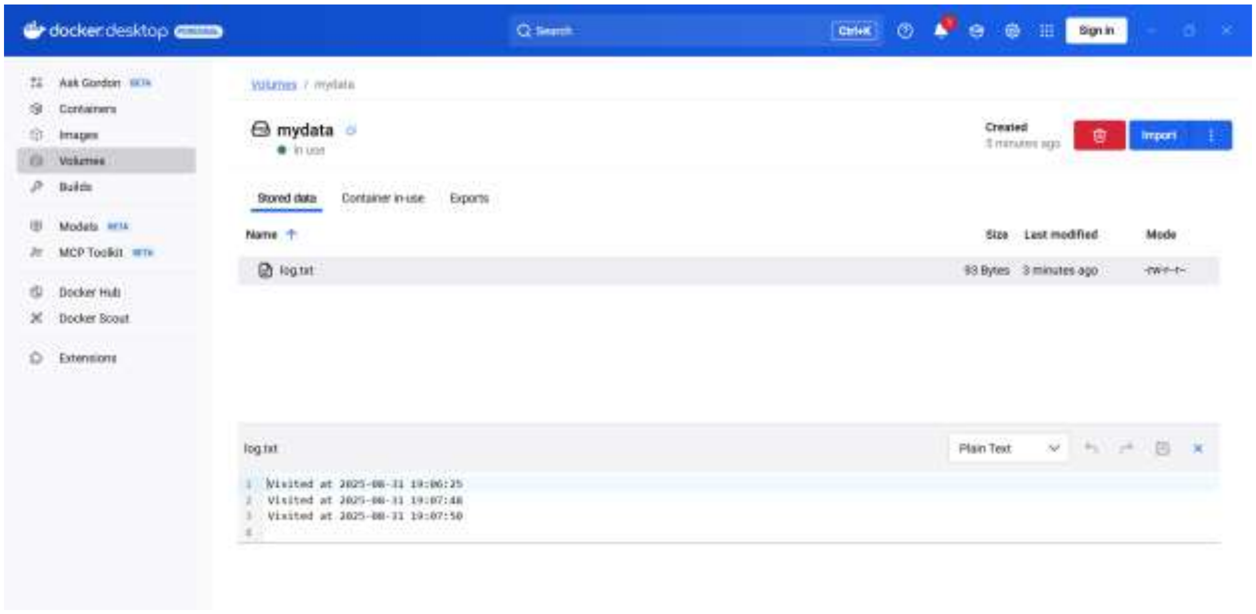
Manage your volumes, view usage, and inspect their contents.

Name	Created	Size	Actions
mydata	4 minutes ago	0 Bytes	

Showing 1 item



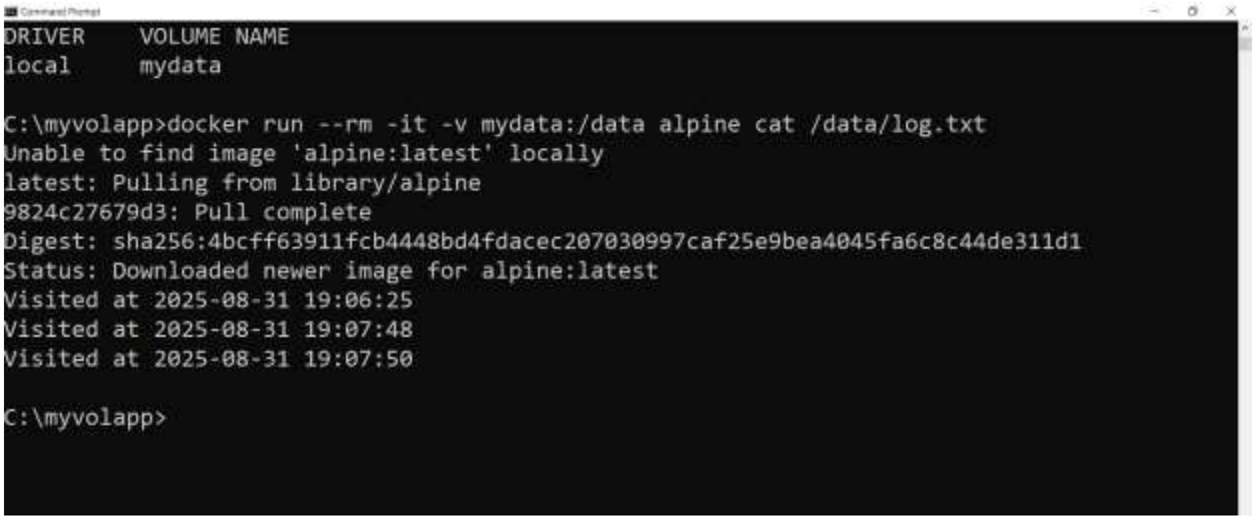
Name	Size	Last modified	Mode
log.txt	93 Bytes	2 minutes ago	-rw-r--r--



Name	Size	Last modified	Mode
log.txt	93 Bytes	3 minutes ago	-rw-r--r--

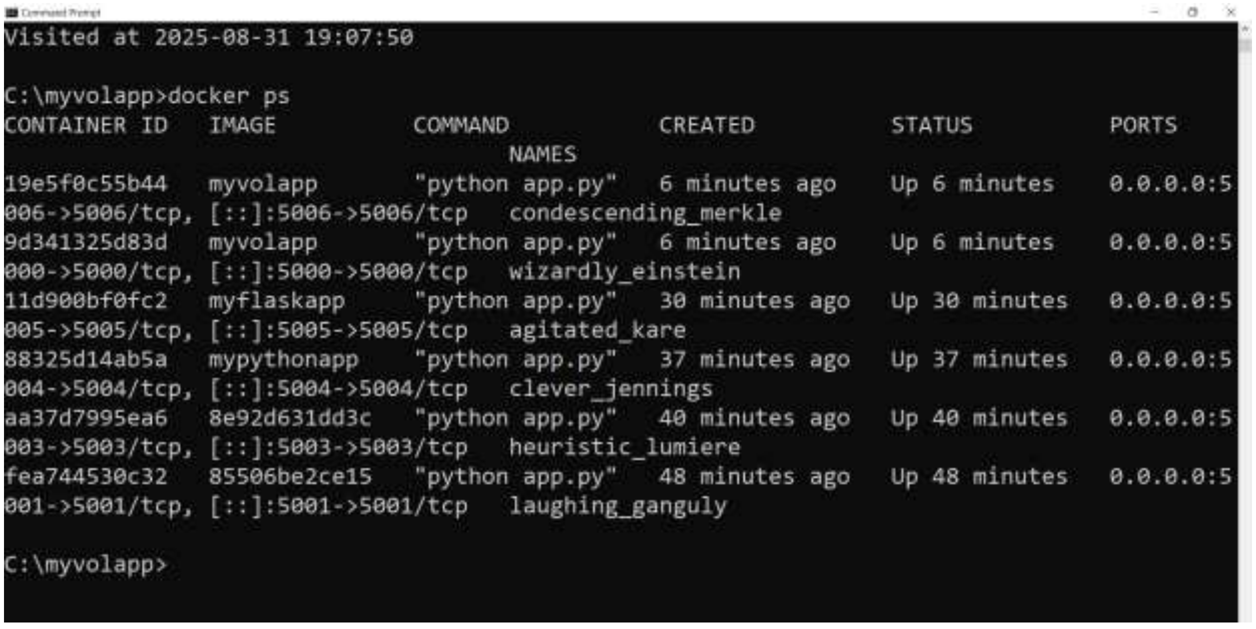
log.txt

```
Visited at 2025-08-31 19:06:25
Visited at 2025-08-31 19:07:48
Visited at 2025-08-31 19:07:50
```



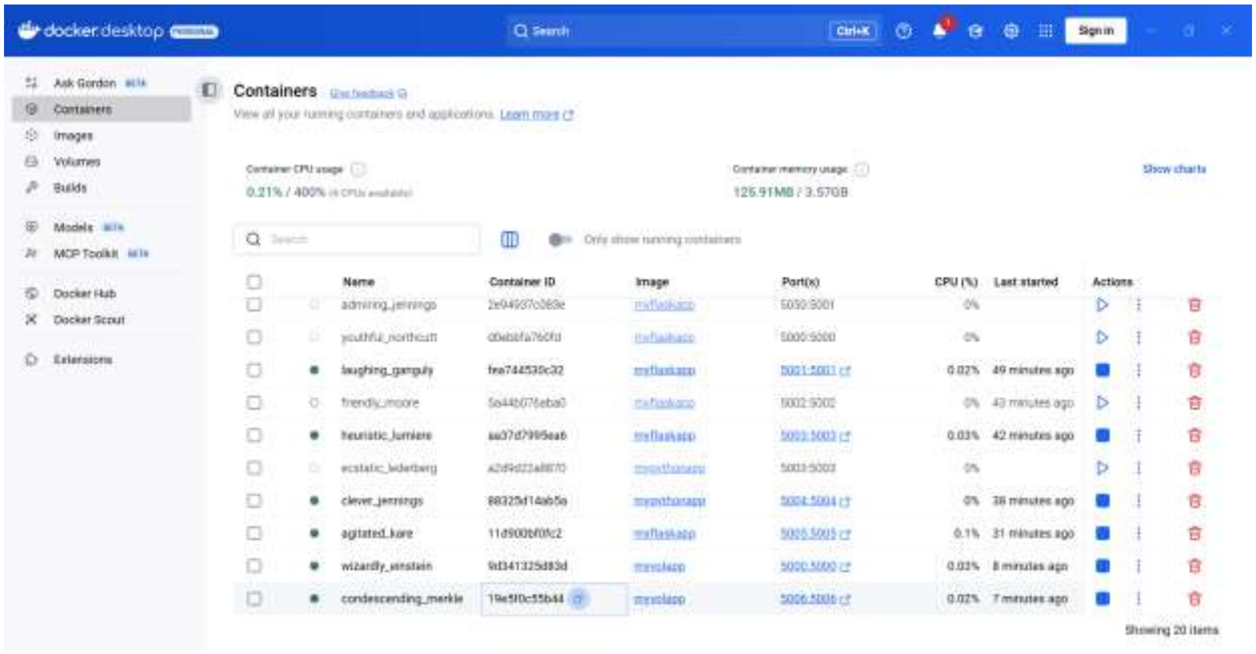
```
DRIVER    VOLUME NAME
local     mydata

C:\myvolapp>docker run --rm -it -v mydata:/data alpine cat /data/log.txt
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
9824c27679d3: Pull complete
Digest: sha256:4bcff63911fcb4448bd4fdacec207030997caf25e9bea4045fa6c8c44de311d1
Status: Downloaded newer image for alpine:latest
Visited at 2025-08-31 19:06:25
Visited at 2025-08-31 19:07:48
Visited at 2025-08-31 19:07:50

C:\myvolapp>
```

Stop & remove the container



```
Visited at 2025-08-31 19:07:50

C:\myvolapp>docker ps
CONTAINER ID   IMAGE          COMMAND           CREATED          STATUS          PORTS
                                    NAMES
19e5f0c55b44   myvolapp       "python app.py"   6 minutes ago    Up 6 minutes    0.0.0.0:5
006->5006/tcp, [::]:5006->5006/tcp   condescending_merkle
9d341325d83d   myvolapp       "python app.py"   6 minutes ago    Up 6 minutes    0.0.0.0:5
000->5000/tcp, [::]:5000->5000/tcp   wizardly_einstein
11d900bf0fc2   myflaskapp     "python app.py"   30 minutes ago   Up 30 minutes   0.0.0.0:5
005->5005/tcp, [::]:5005->5005/tcp   agitated_kare
88325d14ab5a   mypythonapp    "python app.py"   37 minutes ago   Up 37 minutes   0.0.0.0:5
004->5004/tcp, [::]:5004->5004/tcp   clever_jennings
aa37d7995ea6   8e92d631dd3c   "python app.py"   40 minutes ago   Up 40 minutes   0.0.0.0:5
003->5003/tcp, [::]:5003->5003/tcp   heuristic_lumiere
fea744530c32   85506be2ce15   "python app.py"   48 minutes ago   Up 48 minutes   0.0.0.0:5
001->5001/tcp, [::]:5001->5001/tcp   laughing_ganguly

C:\myvolapp>
```

Copy container id of myvolapp



Containers

View all your running containers and applications. Learn more

Container CPU usage
0.21% / 400% (4 CPUs available)

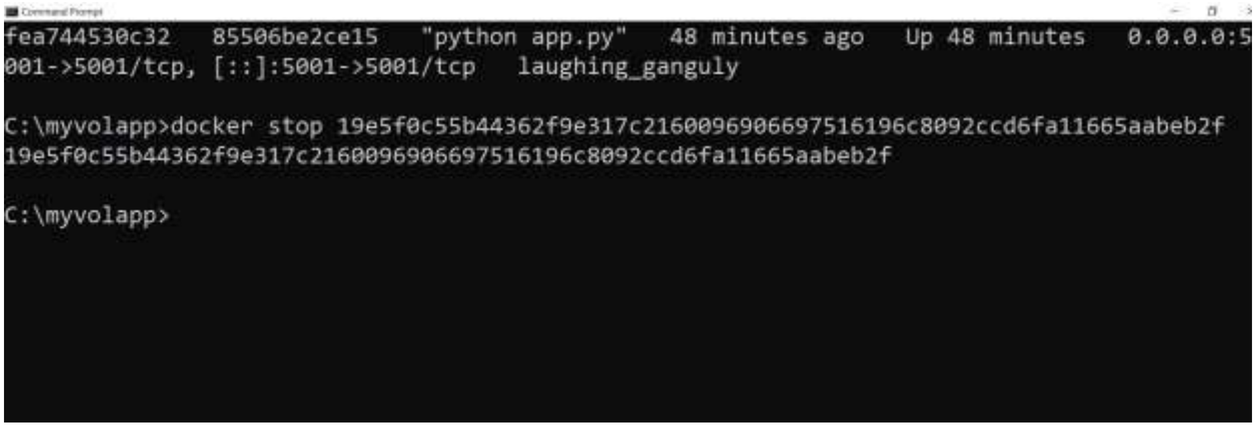
Container memory usage
125.91MB / 3.57GB

Show charts

Only show running containers

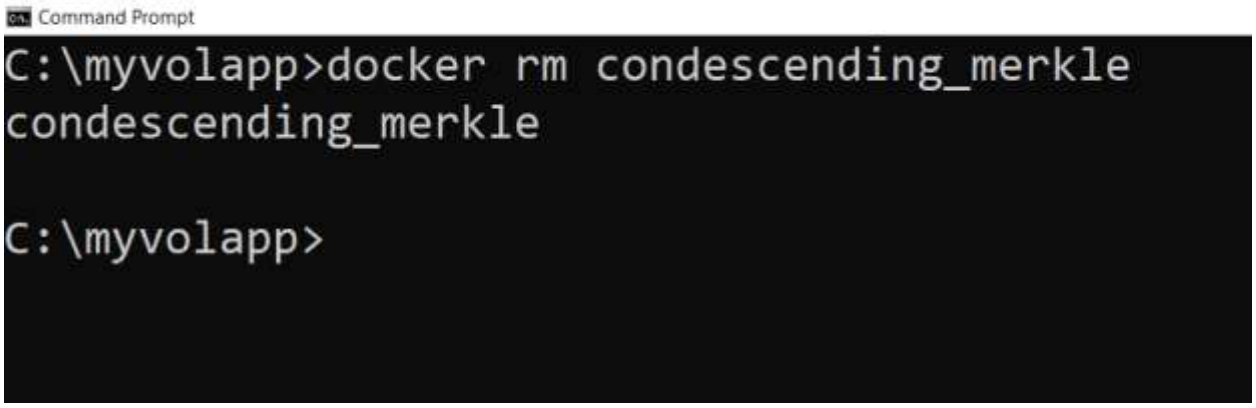
	Name	Container ID	Image	Port(s)	CPU (%)	Last started
	admiring_jennings	2e94937c083e	myflaskapp	5050:5001	0%	
	youthful_northcutt	d0ebbfa760fd	myflaskapp	5000:5000	0%	
●	laughing_ganguly	fea744530c32	myflaskapp	5001:5001	0.02%	49 minutes ago
	friendly_moore	5a44b076eba0	myflaskapp	5002:5002	0%	43 minutes ago
●	heuristic_lumiere	aa37d7995ea6	myflaskapp	5003:5003	0.03%	42 minutes ago
	ecstatic_lederberg	a2d9d22a8870	mypythonapp	5003:5003	0%	
●	clever_jennings	88325d14ab5a	mypythonapp	5004:5004	0%	38 minutes ago
●	agitated_kare	11d900bf0fc2	myflaskapp	5005:5005	0.1%	31 minutes ago
●	wizardly_einstein	9d341325d83d	myvolapp	5000:5000	0.03%	8 minutes ago
●	condescending_merkle	19e5f0c55b44	myvolapp	5006:5006	0.02%	7 minutes ago

Showing 20 items



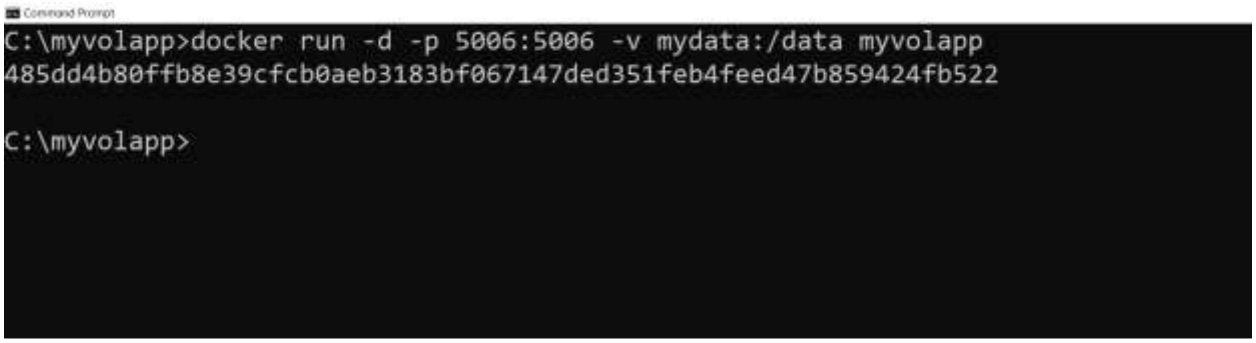
Command Prompt

```
fea744530c32   85506be2ce15   "python app.py"   48 minutes ago   Up 48 minutes   0.0.0.0:5
001->5001/tcp, [::]:5001->5001/tcp   laughing_ganguly

C:\myvolapp>docker stop 19e5f0c55b44362f9e317c2160096906697516196c8092ccd6fa11665aabeb2f
19e5f0c55b44362f9e317c2160096906697516196c8092ccd6fa11665aabeb2f

C:\myvolapp>
```

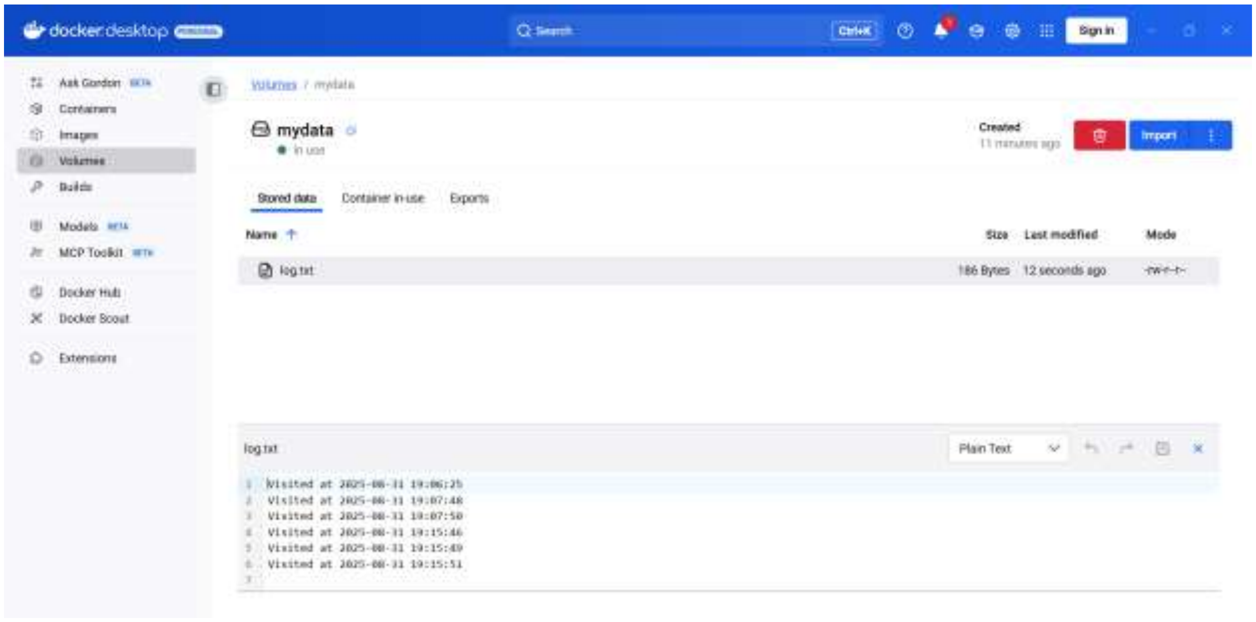


Command Prompt

```
C:\myvolapp>docker rm condescending_merkle
condescending_merkle

C:\myvolapp>
```



```
C:\myvolapp>docker run -d -p 5006:5006 -v mydata:/data myvolapp
485dd4b80ffb8e39cfcb0aeb3183bf067147ded351feb4feed47b859424fb522

C:\myvolapp>
```



Name	Size	Last modified	Mode
log.txt	186 Bytes	12 seconds ago	-rw-r--r--

log.txt

```
Visited at 2025-08-31 19:06:25
Visited at 2025-08-31 19:07:48
Visited at 2025-08-31 19:07:50
Visited at 2025-08-31 19:15:46
Visited at 2025-08-31 19:15:49
Visited at 2025-08-31 19:15:51
```