Keeping a single application running with `volume mount`

There is often a need to share data between different containers. You can distinguish three typical situations in which it may arise:

1. we run the same container multiple times, and we want to ensure continuity of its operation. For example, run #2 uses the results of run #1 (the data from run #1 does not disappear, but is transferred for the next run)
2. we have several containers running in parallel, and we want them to exchange files among themselves.
3. we have several containers running in sequence and we want them to pass the results of operations to each other.

In the following example, we will show how to ensure the continuity of a single application using the **volume mount** mechanism.

Our application

In our example, we will use the application *app_1.py* implementing a simple data frame transformation:

```python
import pandas as pd
import random
import os

print('... App 1 Started ...\n')

script_dir = os.path.abspath( os.path.dirname( __file__ ) )

input_path = 'data/input_1.csv'

if os.path.isfile(input_path):
    df = pd.read_csv(input_path, header = None)
else:
    df = pd.DataFrame([[1, 32, 10], [3, 4, 315]])
    df.to_csv(input_path, index = False, header = False)

df = pd.read_csv(input_path, header = None)
print('Original df:')
print(df)

n = random.randint(1,5)

print('\nRandom number: ', n)

df_out = n*df

output_path = os.path.join(script_dir, 'data/output_1.csv')
df_out.to_csv(input_path, header=False, index=False)
df_out.to_csv(output_path, header=False, index=False)

print("")
print('Transformed df:')
print(df_out)

print('\n... App 1 Completed ...\n')
```

Note the exports to csv file at the end of the application: we pass the calculation result not only to the `data/output_1.csv` file, but also to the `data/input_1.csv` input file. As a result, subsequent runs, with the data continuity mechanism working properly, should use the results of previous runs.

Creating an image

We create the image based on the Dockerfile:

```
#syntax=docker/dockerfile:1

FROM python:3.8-slim-buster

WORKDIR /app

COPY requirements.txt requirements.txt
RUN pip3 install -r requirements.txt

COPY . .

CMD ["python3","app_1.py"]
```

We use the command to do this:

`docker build -t app1_vm:v1 .`.

Creating a shared volume

The *volume mount* functionality allows you to create a local but docker-managed shared volume. Let's create such a volume using the command:

`docker volume create vm_app1`.

You can now verify, for example using Docker Desktop, that it does not contain any data.

Sequentially launching containers using the volume mount mechanism.

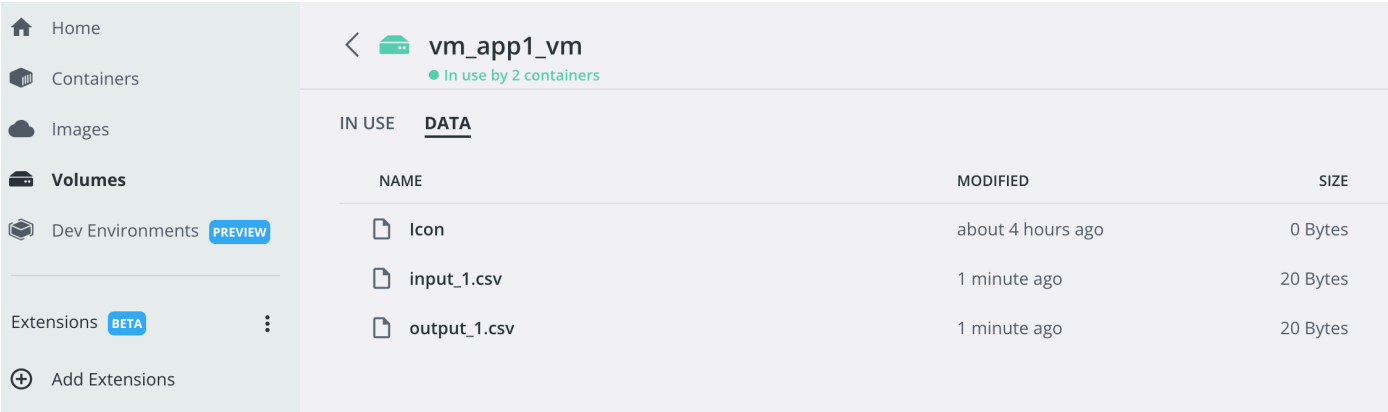
To run a container **with the option to synchronize data with a Docker volume**, at the command line type:

`docker run -ti -v vm_app1_vm:/app/data app1_vm:v1`.

The result is not just a running script:

```
(base) wodecki@iMac-iMac 3.4. Pojedyncza aplikacja z volume mount % docker run -ti -v vm_app1_vm:/app/data app1_vm:v1
... App 1 Started ...

Original df:
   0   1    2
0  1  32   10
1  3   4  315

Random number:  2

Transformed df:
   0   1    2
0  2  64   20
1  6   8  630

... App 1 Completed ...
```

... but also new data in our volume:



Subsequent container runs use the results of the previous ones:

```
(base) wodecki@iMac-iMac 3.4. Pojedyncza aplikacja z volume mount % docker run -ti -v vm_app1_vm:/app/data app1_vm:v1
... App 1 Started ...

Original df:
   0   1    2
0  1  32   10
1  3   4  315

Random number:  2

Transformed df:
   0   1    2
0  2  64   20
1  6   8  630

... App 1 Completed ...

(base) wodecki@iMac-iMac 3.4. Pojedyncza aplikacja z volume mount % docker run -ti -v vm_app1_vm:/app/data app1_vm:v1
... App 1 Started ...

Original df:
   0   1    2
0  2  64   20
1  6   8  630

Random number:  4

Transformed df:
    0    1     2
0   8  256    80
1  24   32  2520

... App 1 Completed ...

(base) wodecki@iMac-iMac 3.4. Pojedyncza aplikacja z volume mount % docker run -ti -v vm_app1_vm:/app/data app1_vm:v1
... App 1 Started ...

Original df:
    0    1     2
0   8  256    80
1  24   32  2520

Random number:  5

Transformed df:
     0     1      2
0   40  1280    400
1  120   160  12600

... App 1 Completed ...
```

Use cases

In particular, you can use the above mechanism in the following situations:

Production

I want to ensure the continuity of my application.

... so that subsequent container runs with my application can use the results of previous runs

I use docker volume for this purpose.

I want different containers running in parallel to exchange data.

I use docker volume for this purpose.

I want to store the data used by my containers with a cloud service provider.

I use docker volume for this purpose.

Useful resources

A very good presentation of the *bind mount* and *volume mount* methods can be found in the official docker documentation available here.