

#014 Volumes

Introduction

this is part 14 from the journey it's a long journey(360 day) so go please check previous parts , and if you need to walk in the journey with me please make sure to follow because I may post more than once in 1 Day but surely I will post daily at least one 😊.

And I will cover lot of tools as we move on.

Download app_014

```
(base) in ~  
> cd Documents/DevOpsJourney/app_013/  
(base) (master)in ~/Documents/DevOpsJourney/app_013  
> 1  
Dockerfile app.py requirements.txt  
(base) (master)in ~/Documents/DevOpsJourney/app_013  
>
```

if you follow [part 9](#)

```
cd location/DevOpsJourney/app_014/
```

replace location with where you put the DevOpsJourney

if your new go to [part 9](#) and do same steps will download old lecture files and new one.

The Problem

so the problem with our Docker right now is every time we need to make a change for our code we need to rebuild image , is time and space waste , and not a pragmatic way to do.

Let's see the problem

first let's build our app

```
docker image build -t app_014 .
```

```
/bin/bash
(base) (master)in ~/Documents/DevOpsJourney/app_014
> docker image build -t app_014 .
Sending build context to Docker daemon  4.608kB
Step 1/8 : FROM python:3.9-rc-alpine
--> a206803e6cb1
Step 2/8 : RUN mkdir /app
--> Using cache
--> cccdc7a8e00
Step 3/8 : WORKDIR /app
--> Using cache
--> 1c43ee4da9d4
Step 4/8 : COPY requirements.txt requirements.txt
--> Using cache
--> 30fcc80c614c
Step 5/8 : RUN pip install -r requirements.txt
--> Using cache
--> 4cb299d53b91
Step 6/8 : COPY . .
--> 721c9adcc96a
Step 7/8 : LABEL maintainer="Omar ElKhatib"
--> Running in 9c9e045649ae
Removing intermediate container 9c9e045649ae
--> 6c027fad6903
Step 8/8 : CMD python app.py
--> Running in 9c9c8668b84f
Removing intermediate container 9c9c8668b84f
--> af274e2c7588
Successfully built af274e2c7588
Successfully tagged app_014:latest
(base) (master)in ~/Documents/DevOpsJourney/app_014
>
```

and let's run our app

```
docker run -it app_014
```

```
(base) (master)in ~/Documents/DevOpsJourney/app_014
> docker run -it app_014
Hello lets save a number to a file
```

our app will print a message and save a number to a file.txt
let's take a look inside our app

```
1 app.py
print("Hello lets save a number to a file")

f = open('file.txt', 'w')
nb = 12
f.write(str(nb)+"\n")
f.close()

~
~
```

it will save number 12 to a file.txt

let's take a look inside our app folder , as we know docker will create an app folder inside his own image because we asked to do this , check [#009 Dockerfile](#) so to access this app folder using interactive shell

```
(base) (master)in ~/Documents/DevOpsJourney/app_014
> docker run -it app_014 sh
/app # ls
Dockerfile      app.py          file.txt        requirements.txt
/app # cat file.txt
12
/app #
```

```
docker run -it app_014 sh
```

inside the interactive shell

```
cat file.txt
```

we can see that 12 is stored

Time to make a change

let's change 12 in our app.py to 666
first exit our interactive shell

```
/app # exit
(base) (master)in ~/Documents/DevOpsJourney/app_014
>
```

```
exit
```

I use vim as my text editor(actually as complete IDE ;D)

let's take a look again inside the container

```
(base) (master)in ~/Documents/DevOpsJourney/app_014
> docker run -it app_014 sh
/app # ls
Dockerfile      app.py          file.txt        requirements.txt
/app # cat file.txt
12
/app #
```

```
docker run -it app_014 sh
```

inside the interactive shell

```
cat file.txt
```

we can see that isn't changed

The fix

we can fix our problem using -v (volume)

```
docker run -it --rm --name app_014 -v $PWD:/app app_014
```

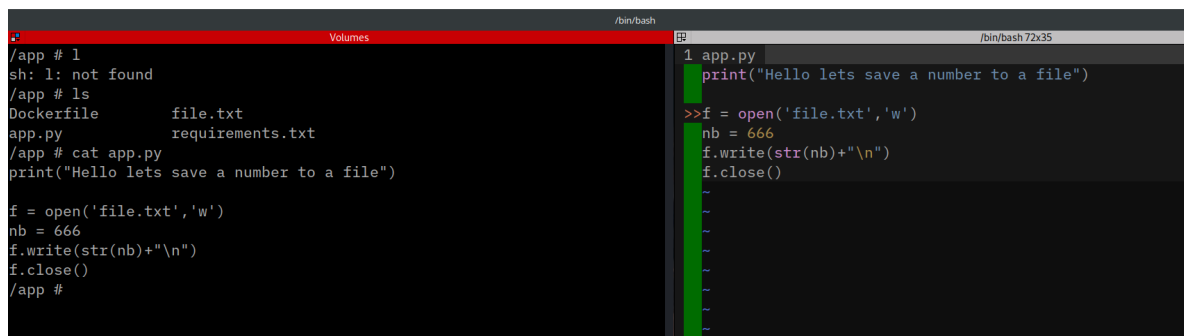
\$PWD is a default variable in linux it store the home url
in our case , we use in docker alpine which is a linux based distro , /app is the
directory we make in Dockerfile to store our app
again let's take a look inside

```
docker run -it --rm --name app_014 -v $PWD:/app app_014 sh
```

inside the interactive shell

```
cat app.py
```

as we see we got 666 without rebuild our image!



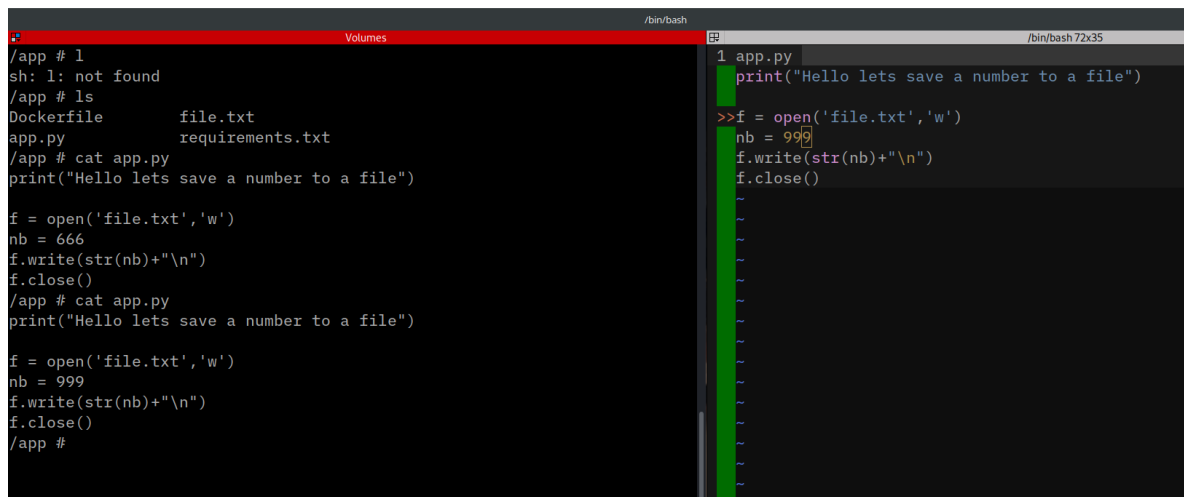
The screenshot shows a terminal window with two panes. The left pane shows the output of 'ls' and 'cat app.py' commands. The right pane shows the execution of a Python script that writes the number 666 to 'file.txt'.

```
/app # ls
sh: 1: not found
/app # ls
Dockerfile      file.txt
app.py          requirements.txt
/app # cat app.py
print("Hello lets save a number to a file")

f = open('file.txt','w')
nb = 666
f.write(str(nb)+"\n")
f.close()
/app #
```

```
1 app.py
print("Hello lets save a number to a file")
>>f = open('file.txt','w')
nb = 666
f.write(str(nb)+"\n")
f.close()
~
~
~
~
~
~
```

let's try 999



The image shows a Docker container environment with two panes. The left pane is a terminal window titled 'Volumes' with a '/bin/bash' prompt. It shows a sequence of commands: listing files, viewing 'app.py', and running it. The script writes the number 666 to 'file.txt'. The right pane is a code editor titled 'app.py' with a '/bin/bash 72x35' prompt. It shows the same Python script, but the variable 'nb' is now set to 999, and the file 'file.txt' is being updated with this new value. The terminal output on the left shows the script being run again, which would result in the file containing 999.

```
/app # l
sh: l: not found
/app # ls
Dockerfile      file.txt
app.py          requirements.txt
/app # cat app.py
print("Hello lets save a number to a file")

f = open('file.txt','w')
nb = 666
f.write(str(nb)+"\n")
f.close()
/app # cat app.py
print("Hello lets save a number to a file")

f = open('file.txt','w')
nb = 999
f.write(str(nb)+"\n")
f.close()
/app #
```

```
1 app.py
print("Hello lets save a number to a file")

>>f = open('file.txt','w')
nb = 999
f.write(str(nb)+"\n")
f.close()
```

see as soon I change my code is got updated inside the image with out rebuild the image.