

به نام خدا

عنوان : اجرای application در داکر

انجام دهنده : زهرا منصوری

شماره دانشجویی : ۹۶۱۵۰۹۱۰۴۸

استاد : استاد فروغی

## فهرست

۳.....	مقدمه
۴.....	توضیح برنامه
۵.....	Docker File
۶.....	Build an image
۶.....	Docker images خروجی
۷.....	image اجرای
۷.....	docker ps خروجی
۸.....	logs -f خروجی

## مقدمه

داکر یک پلتفرم متن باز است که بر مبنای سیستم عامل لینوکس راه اندازی شده و ابزاری است که می تواند فرایند ایجاد، پیاده سازی و اجرای برنامه ها را با استفاده از Container ها بسیار ساده کند.

در مقابل ماشین های مجازی، Container ها قرار دارند. آن ها می توانند جایگزین مناسبی برای ماشین های مجازی باشند. Container ها محیط های اجرایی را جدا کرده و هسته سیستم عامل را به اشتراک می گذارند. Container ها نسبت به ماشین های مجازی از منابع کمتری استفاده می کنند.

Container این امکان را برای توسعه دهندگان فراهم می کند تا بسته کاملی از برنامه های خود همراه تمامی بخش های مورد نیاز آن ایجاد کرده و آن را در قالب یک بسته واحد ارسال کنند.

برای ساخت یک برنامه داکر و همچنین کار با داکر باید از کامپوننت های مختلف استفاده کنیم. در ادامه این کامپوننت ها را در قالب اجرای یک application به زبان python در داکر معرفی و بررسی می کنیم.

## توضیح برنامه

برنامه زیر به زبان پایتون نوشته شده و آمار مبتلایین و فوت شدگان بر اثر ویروس کرونا را به تفکیک هر کشور گرفته و در جدول نمایش می دهد. کتابخانه های استفاده شده در این برنامه عبارتند از : requests , texttable , beautifulsoup4 .

در این برنامه ابتدا یک `requests.get` به آدرس :

[“https://www.worldometers.info/coronavirus/countries-where-coronavirus-has-spread / “](https://www.worldometers.info/coronavirus/countries-where-coronavirus-has-spread/)

می‌زنیم و از فایل html دریافت شده اطلاعات نام کشور، تعداد مبتلایان و فوت‌شدگان و قاره را به دست می‌آوریم و در نهایت آنها را در قالب یک جدول در خروجی نمایش می‌دهیم.

```

# By Zahra Mansouri
# importing modules
import requests
from bs4 import BeautifulSoup
import texttable as tt

# URL for scrapping data
url = 'https://www.worldometers.info/coronavirus/countries-where-coronavirus-has-spread/'

# get URL html
page = requests.get(url)
soup = BeautifulSoup(page.text, 'html.parser')

data = []

# soup.find_all('td') will scrape every
# element in the url's table
data_iterator = iter(soup.find_all('td'))

while True:
    try:
        country = next(data_iterator).text
        confirmed = next(data_iterator).text
        deaths = next(data_iterator).text
        continent = next(data_iterator).text

        data.append((
            country,
            confirmed,
            deaths,
            continent
        ))
    except StopIteration:
        break

table = tt.Texttable()
table.add_rows(['Country ', ' Number of cases ', ' Deaths ', ' Continent ']) + data
table.set_cols_align(["c", "c", "c", "c"])
print(table.draw())

```

"main.py" 40 lines, 968 characters

شکل ۱

## Docker File

هر container داکر به وسیله یک فایل داکر شروع به کار می کند. Dockerfile ها در واقع فایل های تنظیمات داکر هستند که با استفاده از آن ها می توانیم به داکر بگوییم که یک container را چگونه بالا بیاورد و تنظیم کند.

در اولین خط از این فایل، base image را برای Docker مشخص می کنیم. چون برنامه ما به زبان python3.8 نوشته شده است پس لازم است آن را به صورت زیر تعریف کنیم :

FROM python:3.8

سپس فایل اصلی برنامه را اضافه می کنیم و چون فایل اصلی برنامه و DockerFile در یک directory قرار دارند از نقطه به عنوان نشان directory جاری استفاده می کنیم.

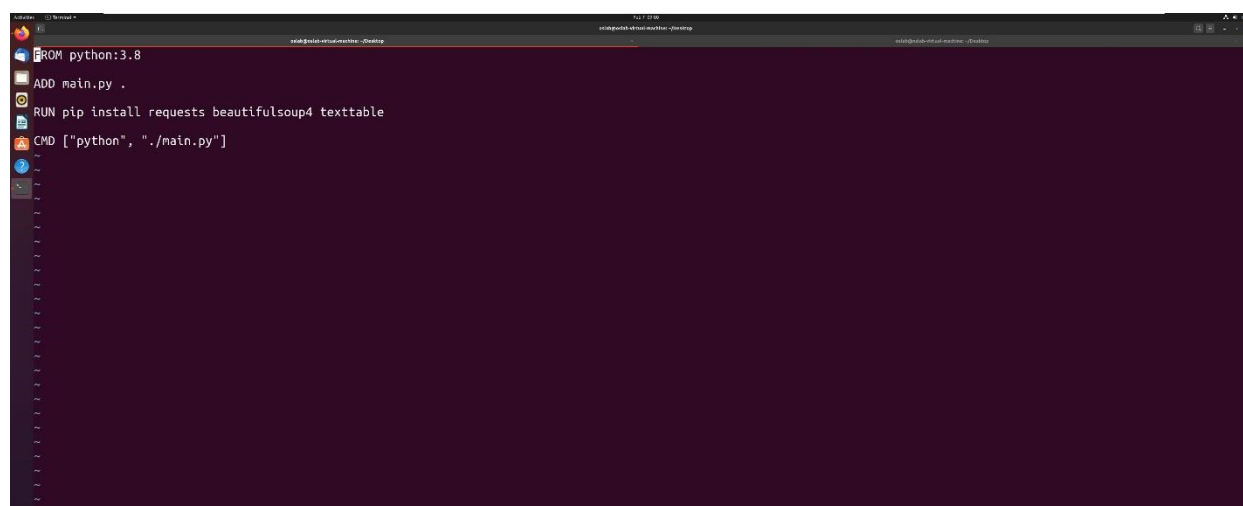
ADD main.py .

در مرحله بعد نیازمندی های برنامه را ( شامل همه کتابخانه هایی که در برنامه استفاده شده است ) با دستور RUN تعریف می کنیم. این دستور همه module های گفته شده را داخل image نصب می کند.

RUN pip install requests beautifulsoup4 texttable

با استفاده از دستور زیر نیز فرمان اجرایی را هنگام اجرای image تعریف می کنیم.

CMD ["python", "./main.py"]



شکل ۲

## Build an image

پس از ساخت DockerFile با استفاده از دستور `docker build` ، یک image می سازیم . Image یک فایل قابل حمل و شامل یک سری دستورالعمل است که مشخص می کند Container کدام کامپوننت های نرم افزاری را اجرا کند و اینکه چگونه آن را اجرا کند.

```
Build an image from a Dockerfile
oslab@oslab-virtual-machine:~/Desktop$ docker build -t python-coronavirus .
Sending build context to Docker daemon 16.9kB
Step 1/4 : FROM python:3.8
3.8: Pulling from library/python
b9a857cbf04d: Pull complete
d557ee2b540b: Pull complete
3b9ca4f00c2e: Pull complete
667fd949ed93: Pull complete
4ad46e8a1e5: Pull complete
381aea9d4031: Pull complete
8a9e78e1993b: Pull complete
9eff4c8a677: Pull complete
8f8516dd3328: Pull complete
Digest: sha256:c3bdfae4d2877b697a844cd9dc1469e2fb631bf8f12dd34870a4fd75a95b0166
Status: Downloaded newer image for python:3.8
----> 40251af0bd62
Step 2/4 : ADD main.py .
----> 8268b7f9f481
Step 3/4 : RUN pip install requests beautifulsoup4 texttable
----> Running in 96039acc56c
Collecting requests
  Downloading requests-2.25.1-py2.py3-none-any.whl (61 kB)
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.9.3-py3-none-any.whl (115 kB)
Collecting texttable
  Downloading texttable-1.6.3-py2.py3-none-any.whl (10 kB)
Collecting soupsieve>1.2
  Downloading soupsieve-2.1-py3-none-any.whl (32 kB)
Collecting certifi<=2017.4.17
  Downloading certifi-2020.12.5-py2.py3-none-any.whl (147 kB)
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.3-py2.py3-none-any.whl (137 kB)
Collecting idna<3,>=2.5
  Downloading idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting chardet<5,>=3.0.2
  Downloading chardet-4.0.0-py2.py3-none-any.whl (178 kB)
Installing collected packages: urllib3, soupsieve, idna, chardet, certifi, texttable, requests, beautifulsoup4
Successfully installed beautifulsoup4-4.9.3 certifi-2020.12.5 chardet-4.0.0 idna-2.10 requests-2.25.1 soupsieve-2.1 texttable-1.6.3 urllib3-1.26.3
Removing intermediate container 96039acc56c
----> 244a38393951
Step 4/4 : CMD ["python", ".", "/main.py"]
----> Running in 0ee4123f8c51
Removing intermediate container 0ee4123f8c51
----> 97c15b82f0b4
Successfully built 97c15b82f0b4
Successfully tagged python-coronavirus:latest
```

شکل ۳

## خروجی Docker images

```
oslab@oslab-virtual-machine:~/Desktop$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
python-coronavirus   latest             97c15b82f0b4        5 minutes ago      893MB
python               3.8                40251af0bd62        5 days ago         883MB
hello-world          latest             bf756fb1ae65        13 months ago      13.3kB
```

شکل ۴

## اجرای image

Docker run در واقع نوعی دستور است که container را راه اندازی می کند.

```
oslab@oslab-virtual-machine:~/Desktop$ docker run --name reports python-coronavirus
```

Country	Number of cases	Deaths	Continent
United States	27,532,602	473,735	North America
India	10,831,279	155,078	Asia
Brazil	9,504,996	231,234	South America
Russia	3,967,281	76,661	Europe
United Kingdom	3,945,680	112,465	Europe
France	3,317,333	78,794	Europe
Spain	2,971,914	61,386	Europe
Italy	2,636,738	91,273	Europe
Turkey	2,524,786	26,685	Asia
Germany	2,287,200	61,998	Europe
Colombia	2,151,207	55,693	South America
Argentina	1,976,689	49,110	South America
Mexico	1,926,080	165,786	North America
Poland	1,550,255	39,087	Europe
South Africa	1,473,700	46,180	Africa
Iran	1,466,435	58,469	Asia
Ukraine	1,244,849	23,597	Europe
Peru	1,180,478	42,121	South America
Indonesia	1,157,837	31,556	Asia
Czech Republic (Czechia)	1,034,975	17,235	Europe
Netherlands	1,005,760	14,403	Europe
Canada	883,627	20,756	North America

شکل ۵

## خروجی docker ps

این دستور لیست container های روی سرویس دهنده ی داکر شما را نمایش می دهد. با اضافه کردن -l به انتهای این دستور آخرین container راه اندازی شده در هر حالتی که باشد را به شما نمایش می دهد.

```
oslab@oslab-virtual-machine:~/Desktop$ docker ps -l
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5c2ef8047239	python-coronavirus	python ./main.py	About a minute ago	Exited (0) About a minute ago		reports

شکل ۶

## خروجی -f logs

```
oslab@oslab-virtual-machine:~/Desktop$ docker logs -f reports
```

Country	Number of cases	Deaths	Continent
United States	27,532,682	473,735	North America
India	10,831,279	155,078	Asia
Brazil	9,504,996	231,234	South America
Russia	3,967,281	76,661	Europe
United Kingdom	3,945,680	112,465	Europe
France	3,317,333	78,794	Europe
Spain	2,971,914	61,386	Europe
Italy	2,636,738	91,273	Europe
Turkey	2,524,786	26,685	Asia
Germany	2,287,200	61,998	Europe
Colombia	2,151,207	55,693	South America
Argentina	1,976,689	49,110	South America
Mexico	1,926,080	165,786	North America
Poland	1,550,255	39,087	Europe
South Africa	1,473,700	46,180	Africa
Iran	1,466,435	58,469	Asia
Ukraine	1,244,849	23,597	Europe
Peru	1,180,478	42,121	South America
Indonesia	1,157,837	31,556	Asia
Czech Republic (Czechia)	1,034,975	17,235	Europe
Netherlands	1,005,760	14,403	Europe

شکل ۷