


گزارش تمرین دوم مبانی رایانش ابری سپهر نوعی ۹۹۳۱۰۶۳


فاز 0:

بخش ۱) ارسال ایمیج روی داکرهاب:

```
(sepehr@sepehr) - [/media/.../Term 8/Cloud/Homeworks/HW2]
$ docker push sepehrnoey/hw2:v1
The push refers to repository [docker.io/sepehrnoey/hw2]
52cef481024f: Pushed
192b95c806e3: Pushed
eecec6ae1e8b: Pushed
3a79d5412acd: Pushed
4857056bad11: Pushed
69141b6c4721: Pushed
0bbac9765c1f: Pushed
4c9c2b9681ab: Pushed
d4fc045c9e3a: Pushed
v1: digest: sha256:a899cf7f58d572b52c60eb8cb089dfc7c6d6b86decb055a7e34783960f9acee5 size: 2203
```



sepehrnoey/hw2 

Updated 11 minutes ago

This repository does not have a description 

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
 v1		Image	10 minutes ago	11 minutes ago

[See all](#)

لیست image های موجود در سیستم:

```
(sepehr@sepehr) - [/media/.../Term 8/Cloud/Homeworks/HW2]
$ docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
hw2	v1	47e35b334e12	19 minutes ago	146MB
sepehrnoey/hw2	v1	47e35b334e12	19 minutes ago	146MB
<none>	<none>	d15a5e57b117	21 minutes ago	146MB
python	3.9.19-alpine	8f4429cefc37	2 weeks ago	48.2MB
rabbitmq	3-management	10e413ab292a	6 weeks ago	250MB

دریافت image ساخته شده از داکرهاب:

```
(sepehr@sepehr) - [/media/.../Term 8/Cloud/Homeworks/HW2]
$ docker image pull sepehrnoey/hw2:v1
v1: Pulling from sepehrnoey/hw2
Digest: sha256:a899cf7f58d572b52c60eb8cb089dfc7c6d6b86dec055a7e34783960f9acee5
Status: Image is up to date for sepehrnoey/hw2:v1
docker.io/sepehrnoey/hw2:v1
```

ساخت کانتینر از image دریافت شده:

```
(sepehr@sepehr) - [/media/.../Term 8/Cloud/Homeworks/HW2]
$ docker build -t sepehrnoey/hw2:v1 .
Sending build context to Docker daemon 1.248MB
Step 1/6 : FROM python:3.9.19-alpine
--> 8f4429cefc37
Step 2/6 : WORKDIR /app
--> Using cache
--> 809cd6431008
Step 3/6 : COPY cc_hw2_hello.py /app/
--> Using cache
--> 5c03cd70de8d
Step 4/6 : COPY movies.json /app/
--> Using cache
--> d937b9d15e29
Step 5/6 : RUN pip install numpy
--> Using cache
--> f371cca234cc
Step 6/6 : CMD [ "python", "cc_hw2_hello.py" ]
--> Using cache
--> 47e35b334e12
Successfully built 47e35b334e12
Successfully tagged sepehrnoey/hw2:v1
```

نمایش اجرای کانتینر:

```

(sepehr@sepehr) - [/media/.../Term 8/Cloud/Homeworks/HW2]
$ docker run sepehrnoey/hw2:v1
Warm regards from the Cloud Computing TAs! Welcome to the Docker project.

We want to simply sort the elements in an array using NumPy in cc_hw2_hello.py
The array:
[[ 3  7  1]
 [10  3  2]
 [ 5  6  7]]

Sort the whole array:
[ 1  2  3  3  5  6  7  7 10]

Sort along each row:
[[ 1  3  7]
 [ 2  3 10]
 [ 5  6  7]]

Sort along each column:
[[ 3  3  1]
 [ 5  6  2]
 [10  7  7]]

```

بخش ۲) تفاوت RUN و CMD در زمان اجرای دستور آن است. دستور RUN برای اجرای یکسری دستورات مورد نیاز در هنگام build کردن کانتینر است (docker build), مانند pip install numpy که یک کتابخانه مورد نیاز را برای اجرای بعدی نصب می کند. در حالی که CMD برای تعیین دستوراتی که باید به محض بالا آمدن کانتینر (docker run) اجرا شوند می باشد, مانند python cc_hw2_hello.py

بخش ۳)

دلیل تفاوت حجم این دو image: در فرایند ساخت یک کانتینر, instruction ها به صورت لایه ای اجرا می شوند و روی لایه قبلی به صورت stack قرار می گیرند. و حجم نهایی کانتینر, مجموع حجم این لایه ها می باشد. در واقع, دستور rm در داکر, تنها instruction آن, یعنی دستور چیزی که باید در آن لایه حذف شود را ذخیره می کند.

راه حل آن: استفاده از multi-stage build
این تکنیک به ما کمک می کند که یکسری فایل هارا در یک temporary stage بسازیم ولی در image انتهایی وجود نداشته باشند, که در نهایت باعث کاهش حجم image ما می شود. در واقع, در این تکنیک, فرایند build را به چند stage مجزا از هم که هر کدام می توانند base image مخصوص خودشان را داشته باشند تقسیم می شود.
استفاده از آن به این صورت است که هر stage جدید با وجود دستور FROM مشخص می شود. و در هر stage میتوانیم مشخص کنیم که چه چیزهایی را از استیج های قبلی نیاز داریم. مانند قطعه کد زیر که در image نهایی, ابزارها وجود ندارند و فقط باینری های اپلیکیشن وجود خواهند داشت.

```

10
11 # Stage 1: Build environment (large image with build tools)
12 FROM python:3.9-slim AS builder
13
14 WORKDIR /app
15
16 COPY requirements.txt .
17 RUN pip install -r requirements.txt
18
19 COPY . .
20 RUN python setup.py build
21
22 # Stage 2: Final image (small image with application binaries)
23 FROM python:3.9-slim
24
25 WORKDIR /app
26
27 COPY --from=builder /app/dist .
28
29 # Final image contains only the application binaries, not build tools
30

```

بخش ۴) برای بازیابی یک فایل که در یک لایه وجود داشته ولی در لایه های بعدی حذف شده است، می توان به ترتیب زیر عمل کرد:

۱- در ابتدا image مد نظر را با دستور save داکر، به صورت فایل tar ذخیره می کنیم:

```

(sepehr@sepehr) - [/media/.../Term 8/Cloud/Homeworks/HW2]
$ docker save -o absh-image.tar abazshoushtari/cc-ta-hw2docker-ctf

```

۲- در ادامه، این فایل tar را در یک فولدر دلخواه extract میکنیم:

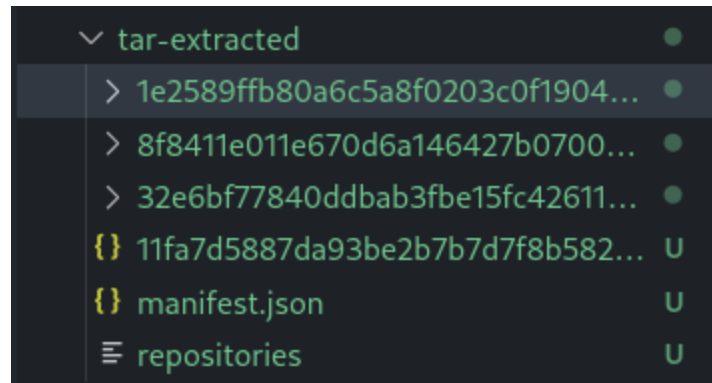
```

(sepehr@sepehr) - [/media/.../Term 8/Cloud/Homeworks/HW2]
$ mkdir tar-extracted

(sepehr@sepehr) - [/media/.../Term 8/Cloud/Homeworks/HW2]
$ tar -xf absh-image.tar -C tar-extracted

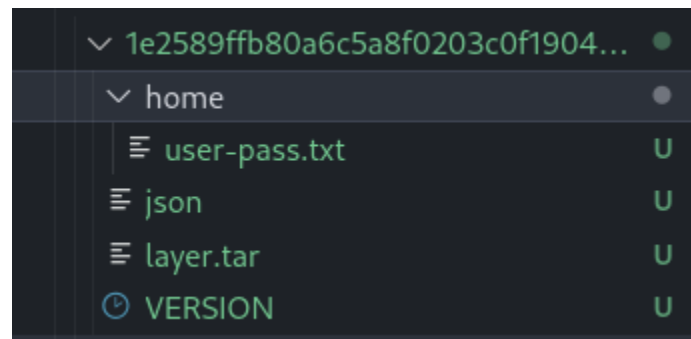
```

پس از انجام این مرحله، میبینیم که محتوای image ما به صورت زیر است:



با بررسی فایل manifest.json میتوانیم ترتیب layer ها را بررسی کنیم که در فیلد Layers در manifest.json قابل مشاهده است، دقت شود که هر instruction در داکر فایل ما یک layer جدید ساخته است و ما به محتوای دومین لایه نیاز داریم (....1e2589):

۳- در مرحله سوم، در فولدر layer مد نظرمان باید به دنبال این فایل باشیم. فایل layer.tar در این فولدر را extract میکنیم.

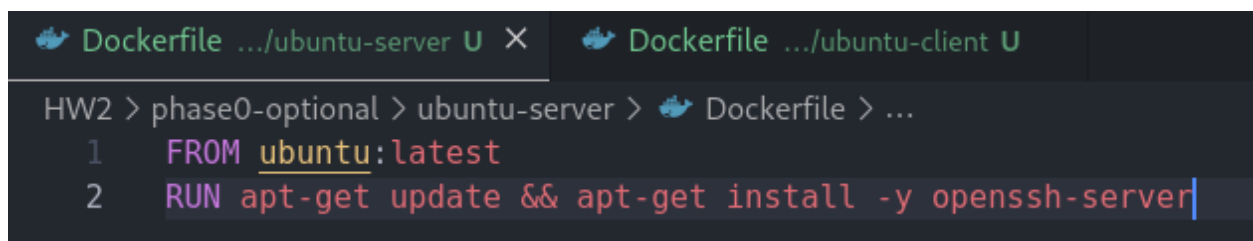


۴- فایل user-pass.txt در محتوای extract شده قابل مشاهده می باشد. محتوای آن به صورت زیر است:

```
1 username: ali_abolfazl
2 password: ajab-passwordie.hamash-gooshte
3
4 # See the following links:
5 # https://stackoverflow.com/questions/46697980/are-there-ways-to-access-files-removed-from-intermediate-layer-after-docker-1-10
6 # https://stackoverflow.com/questions/40575752/docker-extracting-a-layer-from-a-image
7
```

بخش امتیازی) برای انجام این بخش، ابتدا دو container به عنوان کلاینت و سرور بالا می آوریم که روی کانتینر کلاینت openssh-client نصب شده است و روی کانتینر سرور، openssh-server نصب شده است:

داکر فایل سرور:



داکر فایل کلاينت:

```
HW2 > phase0-optional > ubuntu-client > Dockerfile > ...
1 FROM ubuntu:latest
2 RUN apt-get update && apt-get install -y openssh-client
3
```

این کانتینر ها را build می کنیم و سپس به کانتینر سرور interactive terminal میزنیم:

```
(sepehr@sepehr) - [ /media/.../Homeworks/HW2/phase0-optional/ubuntu-server ]
$ docker run -it --name sshserver ubuntu-ssh-server
root@962ae003f2dc:/# nano
```

برای کانفیگ کردن ssh نیاز داریم که یک فایل را ادیت کنیم, برای اینکار ابتدا nano را نصب میکنیم تا راحت تر بتوانیم فایل ها را ادیت کنیم:

```
root@962ae003f2dc:/# apt install nano
Reading package lists... Done
```

سپس, فایل sshd_config در فولدر etc/ssh/ را ادیت می کنیم و در آن PermitRootLogin را به yes تغییر می دهیم:

```
root@962ae003f2dc:/# nano /etc/ssh/sshd_config
```

```
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

همچنین پسورد root را در این کانتینر عوض میکنیم:

```
root@962ae003f2dc:/# passwd root
New password:
Retype new password:
passwd: password updated successfully
```

سپس سرویس ssh را استارت میکنیم و می توانیم از این کانتینر خارج شویم:

```

root@962ae003f2dc:/# service ssh start
* Starting OpenBSD Secure Shell server sshd
root@962ae003f2dc:/# service --status-all
[ - ]   dbus
[ ? ]   hwclock.sh
[ - ]   procs
[ + ]   ssh
root@962ae003f2dc:/# exit
exit

```

در ادامه برای اتصال، نیاز داریم که آدرس آبی کانتینر سرور را بدانیم که می‌توانیم به شکل زیر به دست بیاوریم:

```

(sepehr@sepehr) - [/media/.../Homeworks/HW2/phase0-optional/ubuntu-server]
$ docker inspect sshserver | grep IPAddress
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.3",
    "IPAddress": "172.17.0.3",

```

کار ما با کانتینر سرور تمام شده است، حالا کانتینر کلاینت را run می‌کنیم:

```

(sepehr@sepehr) - [/media/.../Homeworks/HW2/phase0-optional/ubuntu-client]
$ docker run -it --name sshclient ubuntu-ssh-client
root@401f94aef104:/# ssh root@172.17.0.3

```

در این کانتینر به کانتینر سرور ssh می‌زنیم و پسورد را وارد می‌کنیم و می‌بینیم که به درستی متصل شده است:

```

root@401f94aef104:/# ssh root@172.17.0.3
The authenticity of host '172.17.0.3 (172.17.0.3)' can't be established.
ED25519 key fingerprint is SHA256:C5FSj3APEQpubd92fKPDRh1i0UbS2tA0rKF54iok5A.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.17.0.3' (ED25519) to the list of known hosts.
root@172.17.0.3's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.6.9-amd64 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

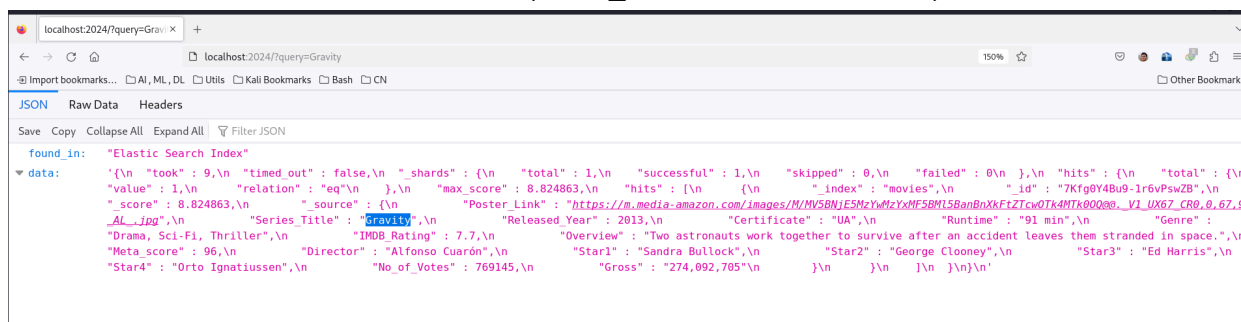
```

پایان فاز صفر

فاز 1:

اسکرین شات های کوئری های مختلف:

- جستجوی یک کوئری که در فایل json موجود است: فیلم Gravity جستجو شده است و همانطور که مشخص است, توسط elastic search پیدا شده است (found_in: Elastic Search Index)



- حال که یکبار این عبارت جستجو شده است و پیدا شده است, همین عبارت در کد به redis اضافه می شود تا دوباره به elastic یا api برای این عبارت در جستجوهای بعدی مراجعه نشود. پس دوباره همین عبارت را جستجو می کنیم:

بخش امتیازی) از kibana نیز استفاده شده است:

```
51
52 kibana:
53   image: kibana:8.2.2
54   container_name: mykib
55   environment:
56     - "ELASTICSEARCH_URL=http://elasticsearch:9200"
57     - xpack.security.enabled=false
58     - xpack.security.enrollment.enabled=false
59     # - xpack.security.encryptionKey.
60   ports:
61     - "5601:5601"
62   depends_on:
63     - elasticsearch
64   networks:
65     - searchnet
66   expose:
67     - 5601
68
```

Index Management - Elastic

localhost:5601/app/management/data/index_management/indices

elastic

Find apps, content, and more. Ex: Discover

Stack Management Index Management

Index Management

[Indices](#) Data Streams Index Templates Component Templates

Update your Elasticsearch indices individually or in bulk. [Learn more.](#) ☐ Include rollout in

Search

<input type="checkbox"/>	Name	Health	Status	Primaries	Replicas	Docs
<input type="checkbox"/>	movies	● yellow	open	1	1	999

Rows per page: 10

