به نام خدا

گزارش تمرین دوم ابری

kubernetes vs docker

روژینا کاشفی – ۹۸۳۱۱۱۸

گام اول

1 - ابتدا یک docker file مطابق زیر می‌نویسیم.

```
FROM alpine:3.14
RUN apk update && apk add curl
```

2 - سپس یک ایمیج می‌سازیم، دستور ساخت ایمیج برابر است با :

Docker build -t alphineimage .

```
rojina@Rojinakashefis-MacBook-Pro CC2 % docker build -t alphineimage .
[+] Building 81.1s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 89B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/alpine:3.14
 => [auth] library/alpine:pull token for registry-1.docker.io
 => [1/2] FROM docker.io/library/alpine:3.14@sha256:4c869a63e1b7c0722fed1e402a6466610327c3b83bdddb94bd94fb71da7f638a
 => => resolve docker.io/library/alpine:3.14@sha256:4c869a63e1b7c0722fed1e402a6466610327c3b83bdddb94bd94fb71da7f638a
 => => sha256:4c869a63e1b7c0722fed1e402a6466610327c3b83bdddb94bd94fb71da7f638a 1.64kB / 1.64kB
 => => sha256:f3ee74f7e495cb90740ffc620224f3e2a3647d8b87089b3d85494297a29ec065 528B / 528B
 => => sha256:275902a1839de6461d1dd0c7a488f7b888d7039fc11a495b71dead25e9e188ac 1.49kB / 1.49kB
 => => sha256:90cda3b7c32511829cd5ae9074240c36c34507085acb6e55b96f993740d2be93 2.72MB / 2.72MB
 => => extracting sha256:90cda3b7c32511829cd5ae9074240c36c34507085acb6e55b96f993740d2be93
 => [2/2] RUN apk update && apk add curl
 => exporting to image
 => => exporting layers
```

3 - لیست ایمیج‌های موجود جهت مطمین شدن از ساخت کامل را مشاهده می‌کنیم:

Docker image ls

```
rojina@Rojinakashefis-MacBook-Pro CC2 % docker image ls
REPOSITORY                                 TAG             IMAGE ID       CREATED          SIZE
alphineimage                               latest          44bc4fc5c908   2 minutes ago    9.57MB
```

4 - سپس یک تگ از ایمیج ساخته شده درست می‌کنیم تا آنرا در داکرهاب پوش کنیم.

Docker tag alphineimage rojinakashefi/alphineimage:latest

Docker tag alphineimage rojinakashefi/alphineimage

```
rojina@Rojinakashefis-MacBook-Pro CC2 % docker tag alphineimage rojinakashefi/alphineimage
```

5 - سپس تگ ساخته شده را در لیست ایمیج‌ها نیز مشاهده می‌کنیم:

Docker image ls

```
alphineimage                                                  latest
                          44bc4fc5c908   44 hours ago    9.57MB
rojinakashefi/alphineimage                                    latest
                          44bc4fc5c908   44 hours ago    9.57MB
```

6 - سپس ایمیج ساخته شده را در داکرهاب پوش می‌کنیم:

Docker image push rojinakashefi/aplhineimage:latest

```
rojina@Rojinakashefis-MacBook-Pro CC2 % docker image push rojinakashefi/alphineimage
Using default tag: latest
The push refers to repository [docker.io/rojinakashefi/alphineimage]
ba6234ea76bb: Layer already exists
28c3ef001483: Layer already exists
latest: digest: sha256:1164d74ac637449d3f75c7307f545491a888251787cda2f036f21fe971bb7850 size: 738
```

7 - مشاهده می‌کنیم ایمیج ساخته شده در داکرهاب اضافه شده است.

8 - سپس ایمیج ساخته شده را با دستور pull از داکرهاب دریافت می‌کنیم.

Docker pull rojinakashefi/alphineimage:latest

Docker pull rojinakashefi/alphineimage

```
rojina@Rojinakashefis-MacBook-Pro CC2 % docker pull rojinakashefi/alphineimage
Using default tag: latest
latest: Pulling from rojinakashefi/alphineimage
Digest: sha256:1164d74ac637449d3f75c7307f545491a888251787cda2f036f21fe971bb7850
Status: Image is up to date for rojinakashefi/alphineimage:latest
docker.io/rojinakashefi/alphineimage:latest
```

9 - سپس یک کانتینتر با حالت interactive با استفاده از ایمیج پول شده می‌سازیم و دستور curl را اجرا می‌کنیم.

Docker run –name alphinecurl -it rojinakashefi/alphineimage:latest

Curl google.com

می‌توانیم به صورت زیر نیز اجرا کنیم

docker run -it --name alphinecurl --rm rojinakashefi/alphineimage curl google.com

```
rojina@Rojinakashefis-MacBook-Pro CC2 % docker run --name alphinecurl -it rojinakashefi/alphineimage
/ # curl google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
/ #
```

برای مشاهده container ها در حالت که در حال اجرا هستند دستورdocker ps و برای مشاهده تمامی کانتینرها چه در گذشته در حال اجرا یا چه الان در حال اجرا بوده‌اند از دستور docker ps -a استفاده می‌کنیم.

گام دوم

1 - ابتدا ایمیج redis را با استفاده از دستور docker pull redis از داکرهاب دریافت می‌کنیم.

```
rojina@Rojinakashefis-MacBook-Pro step2 % docker pull redis
Using default tag: latest
latest: Pulling from library/redis
6064e7e5b6af: Pull complete
f6bd55a0e6ff: Pull complete
7525aa3c12b6: Pull complete
e64bc40eb9ab: Pull complete
2d63145a0c4b: Pull complete
ca0411fb9ec8: Pull complete
Digest: sha256:fdaa0102e0c66802845aa5c961cb89a091a188056811802383660cd9e10889da
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
```

2- سپس با دستور docker image ls مشاهده می‌کنیم درست دریافت انجام شده.

3- سپس با دستور docker inspect redis پورت مربوطه به ردیس را بدست می‌اوریم.

و همچنین اطلاعات مربوط به ایمیج سرور را می‌توانیم با docker inspect coinprice بدست بیاوریم.

4- سپس volume و network مورد نظر را با دستورات

Docker network create mynet
Docker volume create myvol

درست می‌کنیم.

5- سپس یک کانتیر از ردیس با دستور زیر بالا میاوریم.

docker run -d -p 6379:6379 -v myvol:/data --network mynet --name myredis redis

```
rojina@Rojinakashefis-MacBook-Pro step2 % docker run -d -p 6379:6379 -v myvol:/data --network mynet --name myredis redis
992d90f7d600e42f7be92055f89a390feda8c3751978a042ad1f8032e3d3d920
rojina@Rojinakashefis-MacBook-Pro step2 % docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED         STATUS         PORTS                    NAMES
992d90f7d600   redis     "docker-entrypoint.s…"   6 seconds ago   Up 5 seconds   0.0.0.0:6379->6379/tcp   myredis
```

6- یک ایمیج با استفاده از داکر فایل نوشته شده برای برنامه می‌سازیم و یک کانتینر از ان بالا می‌اوریم.

Docker build -t coinprice .
Docker image ls
Dokcer run -it –name getprice –network mynet -p 4000:4000 --rm coinprice

```
FROM python:3.11.0a6-alpine3.15

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD python app.py
```

```
rojina@Rojinakashefis-MacBook-Pro step2 % docker build -t coinpric
e .
[+] Building 3.6s (11/11) FINISHED
 => [internal] load build definition from Dockerfile         0.0s
 => => transferring dockerfile: 180B                         0.0s
 => [internal] load .dockerignore                            0.0s
 => => transferring context: 2B                              0.0s
 => [internal] load metadata for docker.io/library/python:3  3.3s
 => [auth] library/python:pull token for registry-1.docker.  0.0s
 => [1/5] FROM docker.io/library/python:3.11.0a6-alpine3.15  0.0s
 => [internal] load build context                            0.2s
 => => transferring context: 223.32kB                        0.1s
 => 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
 => CACHED [5/5] COPY . .                                    0.0s
 => exporting to image                                       0.0s
 => => exporting layers                                      0.0s
 => => writing image sha256:39fabdc7f534bf98e788dee54008d54  0.0s
 => => naming to docker.io/library/coinprice                 0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
rojina@Rojinakashefis-MacBook-Pro step2 % docker image ls
REPOSITORY                                  TAG                                                 IMAGE ID       CREATED
        SIZE
coinprice                                   latest                                              39fabdc7f534   20 hour
```

```
rojina@Rojinakashefis-MacBook-Pro step2 % docker run -it --name getprice --network mynet -p 4000:4000 --rm coinprice
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:4000
 * Running on http://172.18.0.3:4000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 965-007-313
172.18.0.1 - - [20/Dec/2022 07:03:47] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [20/Dec/2022 07:03:47] "GET /favicon.ico HTTP/1.1" 404 -
```

7- وارد کش ردیس می‌شویم.

docker exec -it myredis redis-cli

یک ریکوست از سمت سرور میزنیم و مشاهده می‌کنیم که کلید اضافه شده است و زمان بیت کوین ذخیره شده به مرور زمان در حال کاهش هست و پس از ۵ دقیقه پاک می‌شود

```
rojina@Rojinakashefis-MacBook-Pro step2 % docker exec -it myredis redis-cli
127.0.0.1:6379> keys *
1) "bitcoin"
127.0.0.1:6379> get bitcoin
"16790.41"
127.0.0.1:6379> ttl bitcoin
(integer) 166
127.0.0.1:6379> ttl bitcoin
(integer) 13
127.0.0.1:6379> ttl bitcoin
(integer) 7
127.0.0.1:6379> ttl bitcoin
(integer) 5
127.0.0.1:6379> ttl bitcoin
(integer) -2
127.0.0.1:6379> get bitcoin
(nil)
```

سپس یک ریکوست دیگه می‌زنیم و آنرا ذخیره می‌کنیم و کانتیر ردیس را استاپ و پاک می‌کنیم و دوباره اجرا می‌کنیم تا مشاهده کنیم ایا مقدار ذخیره شده باقی مانده یا خیر که مشاهده می‌کنیم باقی مانده و بعد از ۵ دقیقه پاک می‌شود.

```
127.0.0.1:6379> get bitcoin
"16783.24"
127.0.0.1:6379> save
OK
127.0.0.1:6379> exit
rojina@Rojinakashefis-MacBook-Pro step2 % docker stop myredis
myredis
rojina@Rojinakashefis-MacBook-Pro step2 % docker rmi myredis
Error: No such image: myredis
rojina@Rojinakashefis-MacBook-Pro step2 % docker rm myredis
myredis
rojina@Rojinakashefis-MacBook-Pro step2 % docker exec -it myredis redis-cli
Error: No such container: myredis
rojina@Rojinakashefis-MacBook-Pro step2 % docker run -d -p 6379:6379 -v myvol:/data --network mynet --name myredis redis
5ca084610917a857d5048c2d4bef43e44a1735c784a51d6b39c5bcaab7c23802
rojina@Rojinakashefis-MacBook-Pro step2 % docker exec -it myredis redis-cli
127.0.0.1:6379> keys *
1) "bitcoin"
127.0.0.1:6379> get bitcoin
"16783.24"
127.0.0.1:6379> ttl bitcoin
(integer) 219
127.0.0.1:6379> ttl bitcoin
(integer) -2
127.0.0.1:6379> get bitcoin
(nil)
127.0.0.1:6379>
```

8- ایمیج اپ خود را در داکرهاب پوش می‌کنیم.

```
rojina@Rojinakashefis-MacBook-Pro step2 % docker tag coinprice rojinakashefi/coinprice:latest
rojina@Rojinakashefis-MacBook-Pro step2 % docker image ls
REPOSITORY                      TAG         IMAGE ID       CREATED
  SIZE
coinprice                       latest      39fabdc7f534   21 hours ago
  96.9MB
rojinakashefi/coinprice         latest      39fabdc7f534   21 hours ago
  96.9MB
```

```
rojina@Rojinakashefis-MacBook-Pro step2 % docker image push rojinakashefi/coinprice:latest
The push refers to repository [docker.io/rojinakashefi/coinprice]
609e5046db27: Pushed
edaa66f9a68c: Pushed
98dd87556967: Pushed
274534e4e856: Pushed
dde5e1423a9f: Mounted from library/python
50e7e093fcef: Pushed
4e1b79615f40: Mounted from library/python
3618203fbcc2: Pushed
4f4ce317c6bb: Mounted from library/python
latest: digest: sha256:048c7e3b7ed1910c9f12ef5ff5bd7242f214e065163d3d95d2d58bd69c0b1b13 size: 2204
```

rojinakashefi / **coinprice**
Contains: Image | Last pushed: a minute ago

Not Scanned   0   0   Public

9- با استفاده از دستور docker ps می‌توانیم لیست کانتینرهای موجود رامشاهده کنیم.

```
rojina@Rojinakashefis-MacBook-Pro step2 % docker ps
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                    NAMES
b9478c0c097a   redis       "docker-entrypoint.s…"   7 seconds ago    Up 6 seconds    0.0.0.0:6379->6379/tcp   myredis
0376bbd29f97   coinprice   "/bin/sh -c 'python …"   12 seconds ago   Up 12 seconds   0.0.0.0:4000->4000/tcp   getprice
```

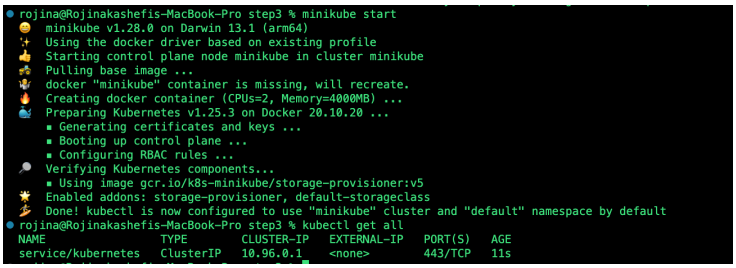
10- با استفاده از دستور docker stats منابع اختصاص یافته به هر کانتیر مشاهده می‌شود.

11-

```
CONTAINER ID   NAME       CPU %   MEM USAGE / LIMIT     MEM %   NET I/O     BLOCK I/O    PIDS
b9478c0c097a   myredis    0.30%   2.527MiB / 7.667GiB   0.03%   726B / 0B   0B / 0B      5
0376bbd29f97   getprice   0.65%   55.61MiB / 7.667GiB   0.71%   946B / 0B   0B / 291kB   3
```

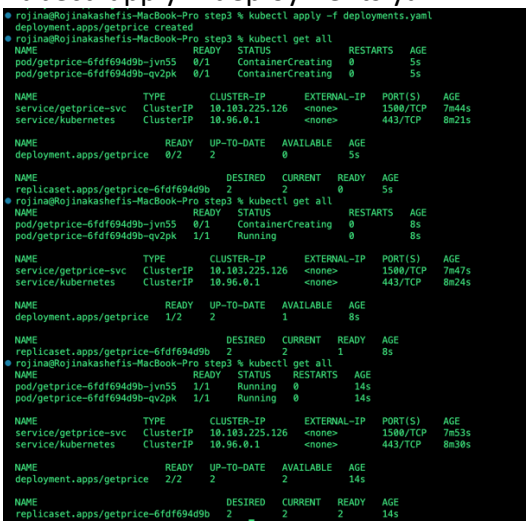
گام سوم

1- ابتدا دستور minikube start را اجرا می‌کنیم

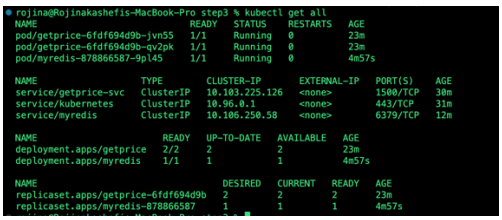


2- سپس تمامی فایل‌های yaml را اپلای می‌کنیم و با استفاده از kubectl get all وضعیت همه را مشاهده می‌کنیم، اگر فایلی به مشکل خورده بود می‌توانیم با استفاده از kubectl describe وضعیت را مشاهده کنیم یا اگه نیازی به تغییری داشت ابتدا انرا با kubectl delete پاک می‌کنیم و دوباره اپلای می‌کنیم.

Kubectl apply -f deployments.yaml



Kubectl get all



3- برای مشاهده endpointپاد ها از دستور زیر استفاده می‌کنیم:

Kubectl get pods -o wide

گام چهارم

برای ران کردن برنامه توسط کوبر دستور زیر را اجرا می‌کنیم.

Kubectl run -it –rm –image rojinakashefi/alphineimage alphinepod

```
rojina@Rojinakashefis-MacBook-Pro step3 % kubectl run -it --rm --image rojinakashefi/alphineimage alphinepod
If you don't see a command prompt, try pressing enter.
/ # curl getprice-svc:1500
{
  "host": "getprice-6fdf694d9b-jvn55",
  "name": "bitcoin",
  "price": "16810.73"
}
/ # curl getprice-svc:1500
{
  "host": "getprice-6fdf694d9b-qv2pk",
  "name": "bitcoin",
  "price": "16810.73"
}
/ #
```

مشاهده می‌کنیم هرسری هاست ها بین دو مقدار عوض می‌شوند که قابلیت load balancing را نشان می‌دهد. و اگر دستور kubectl get all را بزنیم یک پاد جدید جهت اجرا دستورات به اسم alphinepod اضافه شده است.

```
rojina@Rojinakashefis-MacBook-Pro step3 % kubectl get all
NAME                                READY   STATUS    RESTARTS   AGE
pod/alphinepod                      1/1     Running   0          5s
pod/getprice-6fdf694d9b-jvn55       1/1     Running   0          29m
pod/getprice-6fdf694d9b-qv2pk       1/1     Running   0          29m
pod/myredis-878866587-9pl45         1/1     Running   0          10m

NAME                    TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
service/getprice-svc    ClusterIP   10.103.225.126   <none>        1500/TCP   36m
service/kubernetes      ClusterIP   10.96.0.1        <none>        443/TCP    37m
service/myredis         ClusterIP   10.106.250.58    <none>        6379/TCP   18m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/getprice   2/2     2            2           29m
deployment.apps/myredis    1/1     1            1           10m

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/getprice-6fdf694d9b   2         2         2       29m
replicaset.apps/myredis-878866587     1         1         1       10m
rojina@Rojinakashefis-MacBook-Pro step3 %
```