سپهر مقیسه

گزارش تمرین دوم رایانش ابری

پاییز ۱۴۰۱

گام اول:

ایمیج مورد استفاده به این صورت ساخته میشود که یک فایل با نام Dockerfile ایجاد میکنیم و در ان دستور from ubuntu را مینویسم و سپس در همان صفحه یک cmd باز کرده و دستور . docker build را میزنیم تا این تصویر را بسازد

```
FROM ubuntu
RUN apt update
RUN apt install curl -y
```

```
ASUS@ASUS MINGW64 ~/PycharmProjects/pythonProject6
$ docker build -t ubuntuwithcurl:v0 .
[+] Building 32.5s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 93B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/ubuntu:latest
 => [1/3] FROM docker.io/library/ubuntu@sha256:27cb6e6ccef575a4698b66f5de06c7ecd61589132d5a91d098f7f3f9285415a9
 => CACHED [2/3] RUN apt update
 => [3/3] RUN apt install curl -y
 => exporting to image
 => => exporting layers
 => => writing image sha256:c0264a7bfeb01f75186b3e2a5a1a0aa5349cb79190f548cd2d17d043fb03ffc4
 => => naming to docker.io/library/ubuntuwithcurl:v0

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

ASUS@ASUS MINGW64 ~/PycharmProjects/pythonProject6
```

☐	first 0a4cedfea83a	latest	In use	1

برای pushکردن بر docker hubینگونه عمل میکنیم که نام imageرا به صورت yourdockerhub username/tag ذخیره میکنیم به صورت زیر:

sepehrmoghiseh / ubuntutest

و سپس ذخیره را میزنیم تا image را ذخیره کند .

حال برای تست کردن یک pull ایجاد میکنیم

```
G:\New folder (2)>docker pull sepehrmoghiseh/ubuntutest
Using default tag: latest
latest: Pulling from sepehrmoghiseh/ubuntutest
e96e057aae67: Downloading [==========================>
```

```
# curl aut.ac.ir
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://aut.ac.ir:443/">here</a>.</p>
</body></html>
#
```

```
ASUS@ASUS MINGW64 /g/New folder (2)
$ docker images
REPOSITORY                  TAG       IMAGE ID       CREATED        SIZE
first                       latest    0a4cedfea83a   35 hours ago   77.8MB
sepehrmoghiseh/ubuntutest   latest    b9db0844ba2c   5 weeks ago    77.8MB

ASUS@ASUS MINGW64 /g/New folder (2)
```

گام دوم :

کد سرور توسعه داده شده به این صورت است :

```python
import requests
from flask import Flask, request

dic = {}
dic['vs_currency'] = 'usd'

app = Flask(__name__)


@app.route('/getprice', methods=['get'])
def getAd():
    coin = request.form.get('coin')
    dic['ids'] = coin
    response=requests.get('https://api.coingecko.com/api/v3/coins/markets', params=dic).json()
    return_response={}
    return_response['name']=coin
    return_response['price']=response[0]['current_price']
    return return_response


if __name__ == "__main__":
    app.run(debug=True)
```

و پاسخ دریافتی:

GET http://127.0.0.1:5000/getprice

Params Authorization Headers (8) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary Graph(

	KEY	VALUE
☑	coin	bitcoin
	Key	Value

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "name": "bitcoin",
3     "price": 18212.5
4 }
```

است. حال با دستور docker pull redis ایمیج ردیس را بر روی سرور شخصی می آوریم

```
ASUS@ASUS MINGW64 ~
$ docker pull redis
Using default tag: latest
latest: Pulling from library/redis
025c56f98b67: Pull complete
060e65aed679: Pull complete
b95291e865b7: Pull complete
5d91e1f209b7: Pull complete
3b64ec7ffb7e: Pull complete
0fb27e19f260: Pull complete
Digest: sha256:4970a4bbd34f9072b56389e85185204dd07dc86ba74a1be44
Status: Downloaded newer image for redis:latest
docker.io/library/redis:latest
```

```
$ docker run -i redis
1:C 14 Dec 2022 18:41:38.119 # oO0OoO0OoO0Oo Redis is starting oO
1:C 14 Dec 2022 18:41:38.119 # Redis version=7.0.6, bits=64, comm
dified=0, pid=1, just started
1:C 14 Dec 2022 18:41:38.119 # Warning: no config file specified,
ult config. In order to specify a config file use redis-server /p
nf
1:M 14 Dec 2022 18:41:38.119 * monotonic clock: POSIX clock_getti
1:M 14 Dec 2022 18:41:38.120 * Running mode=standalone, port=6379
1:M 14 Dec 2022 18:41:38.120 # Server initialized
1:M 14 Dec 2022 18:41:38.120 # WARNING Memory overcommit must be
t it, a background save or replication may fail under low memory
g disabled, it can can also cause failures without low memory con
ps://github.com/jemalloc/jemalloc/issues/1328. To fix this issue
mit_memory = 1' to /etc/sysctl.conf and then reboot or run the co
m.overcommit_memory=1' for this to take effect.
```

```
/bin/sh: 7: redis[cli: not found
# redis-cli
127.0.0.1:6379> set a 5
OK
127.0.0.1:6379> get a
"5"
127.0.0.1:6379>
```

و یک دستور را ران میکنیم به صورت

```
ASUS@ASUS MINGW64 /d/cc
$ docker volume create redis-persist
redis-persist
```

حال با دستور

ولوم را ساخته و با این دستور

```
ASUS@ASUS MINGW64 /d/cc
$ docker run -d -p 6379:6379 --name redis-persist -v my-volume:/data redis
90c92818101969a0295de7e7335a1e5fe2e8eca5b97ecf78728de720abd58a98
```

یک کانتینر به کمک ولوم ساخته شده بالا می آوریم

سپس اگر اینجا دستور get را بزنیم متوجه میشویم که مقدار را ذخیره کرده است

```
ASUS@ASUS MINGW64 /d/cc
$ docker run -d -p 6379:6379 --name rediscontainer -v redis-persist:/data redis
c69cdbe4265442c9993a299982ed73946e19ff006a5969a561bbeb5d3a5d1f42

ASUS@ASUS MINGW64 /d/cc
$ docker exec -it rediscontainer sh
# redis-cli
127.0.0.1:6379> get a
"5"
127.0.0.1:6379>
```

سپس با این دستور شبکه را میسازیم

```
ASUS@ASUS MINGW64 /d/cc
$ docker network create bit-redis
f414813a86fdb60d8dd5a475c8b9b294acf8f72151abd00da05b9843fe9c9825

ASUS@ASUS MINGW64 /d/cc
$ docker network ls
NETWORK ID     NAME        DRIVER    SCOPE
f414813a86fd   bit-redis   bridge    local
3c78c26d1058   bridge      bridge    local
16675b711286   host        host      local
76a05ec98053   none        null      local
```

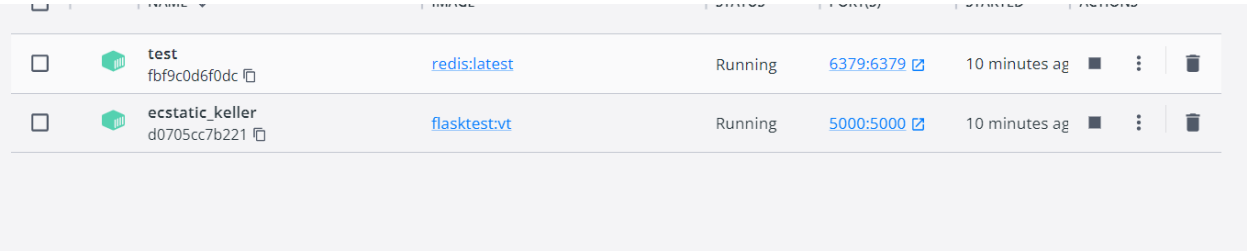
در این مرحله ایمیج را پوش میکنیم

```
$ docker build -t flasktest:vt .
[+] Building 3.6s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 32B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/python:3.11.0a6-alpine3.15
 => [internal] load build context
 => => transferring context: 226.09kB
 => [1/5] FROM docker.io/library/python:3.11.0a6-alpine3.15@sha256:6c215dcdebd4c2d15e5e5fb8307f913c34af918b1b5e5f166d620c871281d2cc
 => CACHED [2/5] COPY requirements.txt .
 => CACHED [3/5] RUN pip install -r requirements.txt
 => [4/5] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:d3fe08a49aaa784ebaceab86d7b3b7a184ab882acbc2902a4f353aecd554f866
 => => naming to docker.io/library/flasktest:vt

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them

ASUS@ASUS MINGW64 ~/PycharmProjects/pythonProject6
$ docker run --name  test  -p 6379:6379 -v redis-persist:/data --net bit-redis -d redis
fbf9c0d6f0dc4c23ec65c5b4655edce5dd0e341c4e1a9ec488a8c67fc360fcd1

ASUS@ASUS MINGW64 ~/PycharmProjects/pythonProject6
$ docker run -it  -p 5000:5000 --net bit-redis flasktest:vt
 * Serving Flask app 'main'
 * 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://127.0.0.1:5000
 * Running on http://172.18.0.3:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 915-048-030
172.18.0.1 - - [17/Dec/2022 12:56:47] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [17/Dec/2022 12:56:48] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [17/Dec/2022 12:56:52] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [17/Dec/2022 13:02:30] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [17/Dec/2022 13:02:32] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [17/Dec/2022 13:04:07] "GET / HTTP/1.1" 200 -
172.18.0.1 - - [17/Dec/2022 13:04:13] "GET / HTTP/1.1" 200 -
```

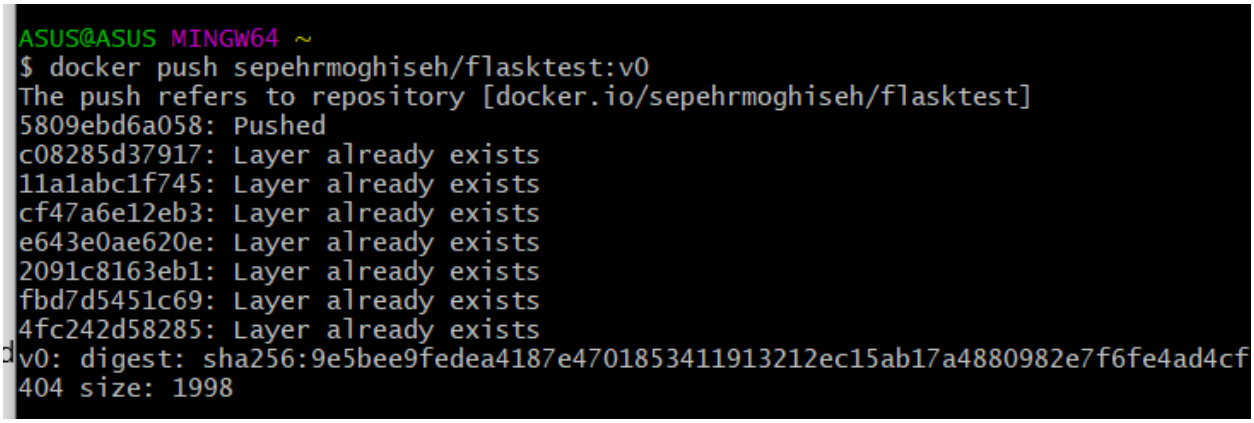
نتیجه آن به این صورت است

داکرفایل نیز به این صورت است:

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

LABEL maintainer="Sepehr Moghiseh  <Sepehrmoghiseh@aut.ac.ir>"

WORKDIR .

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

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



در شکل های زیر موارد خواسته شده در صورت پروژه اورده شده است.



```
ASUS@ASUS MINGW64 ~
$ docker push sepehrmoghiseh/flasktest:v0
The push refers to repository [docker.io/sepehrmoghiseh/flasktest]
5809ebd6a058: Pushed
c08285d37917: Layer already exists
11a1abc1f745: Layer already exists
cf47a6e12eb3: Layer already exists
e643e0ae620e: Layer already exists
2091c8163eb1: Layer already exists
fbd7d5451c69: Layer already exists
4fc242d58285: Layer already exists
v0: digest: sha256:9e5bee9fedea4187e4701853411913212ec15ab17a4880982e7f6fe4ad4cf
404 size: 1998
```

```
$ docker inspect flasktest:vt
[
    {
        "Id": "sha256:b64c9ea86d43ccf64ea5cdd4a1f2a07082fca504afeb7ad3876ea96b31
b9d546",
        "RepoTags": [
            "flasktest:vt",
            "sepehrmoghiseh/flasktest:v1",
            "sepehrmoghiseh/flasktest:v2",
            "sepehrmoghiseh/flasktesting:v1"
        ],
        "RepoDigests": [],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2022-12-17T16:14:50.9891457Z",
        "Container": "",
        "ContainerConfig": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": null,
            "Cmd": null,
            "Image": "",
            "Volumes": null,
            "WorkingDir": "",
            "Entrypoint": null,
            "OnBuild": null,
            "Labels": null
        },
        "DockerVersion": "",
        "Author": "",
        "Config": {
            "Hostname": "",
            "Domainname": "",
            "User": "",
            "AttachStdin": false,
            "AttachStdout": false,
            "AttachStderr": false,
            "Tty": false,
            "OpenStdin": false,
            "StdinOnce": false,
            "Env": [
```

```
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
8bf52419a703   flasktest:vt   "python main.py"         18 minutes ago   Up 10 seconds   0.0.0.0:5000->8000/tcp   heuristic_black
dad85a5e0f4d   redis          "docker-entrypoint.s…"   20 minutes ago   Up 20 minutes   0.0.0.0:6379->6379/tcp   test
```

```
MINGW64:/c/Users/ASUS
CONTAINER ID   NAME              CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O   PIDS
8bf52419a703   heuristic_black   0.76%     60.44MiB / 9.237GiB   0.64%     8.56kB / 3.64kB   0B / 0B     3
dad85a5e0f4d   test              0.18%     7.656MiB / 9.237GiB   0.08%     3.06kB / 1.19kB   0B / 0B     5
```

گام سوم

به این صورت config-map را ساخته

```
ASUS@ASUS MINGW64 ~/PycharmProjects/pythonProject6
$ kubectl apply -f flask-config.yaml
configmap/flask-config created

ASUS@ASUS MINGW64 ~/PycharmProjects/pythonProject6
$ kubectl get configmap flask-config -o yaml
apiVersion: v1
data:
  COIN: bitcoin
  PORT: "5000"
  TIMER: "300"
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"COIN":"bitcoin","PORT":"5000","TIMER":"300"},"kind":"ConfigMap","metadata":{"annotations":{},"creationTimestamp":"2022-12-18T08:20:56Z","name":"flas
lt","resourceVersion":"7337","uid":"2d00a9ca-4779-4f52-907e-b137beed4bd8"}}
  creationTimestamp: "2022-12-18T08:41:03Z"
  name: flask-config
  namespace: default
  resourceVersion: "8914"
  uid: 09ee76e1-b728-4948-bb0e-82f9e0769e27
```

```
NAME                DATA   AGE
flask-config        3      84m
kube-root-ca.crt    1      107m
```

```
$ kubectl get service
NAME            TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AG
E
flask-service   LoadBalancer   10.109.90.136   <pending>     5000:31798/TCP   16
h
kubernetes      ClusterIP      10.96.0.1       <none>        443/TCP          16
h
test            LoadBalancer   10.109.173.24   <pending>     6379:31553/TCP   16
h
```

```
$ kubectl get deployment
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
flaskapi-deployment   2/2     2            2           16h
redis-deploy          1/1     1            1           16h
```

```
$ kubectl get pv
NAME            CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS   REASON   AGE
redis-pv-volume 2Gi        RWX            Retain           Bound    default/redis-pv-claim  manual                  47m
```

```
$ kubectl get pvc
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS   AGE
redis-pv-claim  Bound    redis-pv-volume  2Gi        RWX            manual         47m
```

```
$ kubectl get endpoints
NAME            ENDPOINTS                                   AGE
flask-service   172.17.0.3:5000,172.17.0.4:5000              113m
kubernetes      192.168.49.2:8443                           119m
test            172.17.0.5:6379                             39m
```

گام چهارم

این گونه اوبونتو را ران میکنیم :

```
$ kubectl run my-shellllllllllllll --rm -i --tty --image ubuntuwithcurl:v0 --command -- bash
If you don't see a command prompt, try pressing enter.
root@my-shellllllllllllll:/# curl aut.ac.ir
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https://aut.ac.ir:443/">here</a>.</p>
</body></html>
root@my-shellllllllllllll:/#
```

MINGW64:/c/Users/ASUS

سپس از مینیکیوب سرویس میگیریم:

```
$ minikube service flask-service
|-----------|---------------|-------------|---------------------------|
| NAMESPACE |     NAME      | TARGET PORT |           URL             |
|-----------|---------------|-------------|---------------------------|
| default   | flask-service |        5000 | http://192.168.49.2:31798 |
|-----------|---------------|-------------|---------------------------|
* Starting tunnel for service flask-service.
|-----------|---------------|-------------|-----------------------|
| NAMESPACE |     NAME      | TARGET PORT |          URL          |
|-----------|---------------|-------------|-----------------------|
| default   | flask-service |             | http://127.0.0.1:1665 |
|-----------|---------------|-------------|-----------------------|
* Opening service default/flask-service in default browser...
! Because you are using a Docker driver on windows, the terminal needs to be o
n to run it.
```

حال به ادرس دستور کرل میزنیم:

```
root@my-shellllllllllllll:/#  curl  http://192.168.49.2:31798
{
  "name": "bitcoin",
  "price": "16809.41"
}
root@my-shellllllllllllll:/#
```

```
root@my-shellllllllllllll:/# curl http://192.168.49.2:30228
{
  "name": "ethereum",
  "price": "1221.44"
}
root@my-shellllllllllllll:/#
```

برای مشاهده کردن توزیع بار پاد ها از kubectl log podname استفاده میکنیم:

```
$ kubectl logs flaskapi-deployment-5675c6fb86-n2ztb
 * Serving Flask app 'main'
 * Debug mode: on
←[31m←[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.←[0m
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.8:5000
←[33mPress CTRL+C to quit←[0m
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 846-909-524
172.17.0.1 - - [23/Dec/2022 09:24:16] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [23/Dec/2022 09:34:52] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [23/Dec/2022 15:15:00] "←[33mGET /favicon.ico HTTP/1.1←[0m" 404 -
172.17.0.1 - - [23/Dec/2022 15:15:48] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [23/Dec/2022 15:17:48] "GET / HTTP/1.1" 200 -
```

و پاد دوم به این صورت است :

```
$ kubectl logs flaskapi-deployment-5675c6fb86-qfs4s
 * Serving Flask app 'main'
 * Debug mode: on
←[31m←[1mWARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.←[0m
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.3:5000
←[33mPress CTRL+C to quit←[0m
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 119-427-072
172.17.0.1 - - [23/Dec/2022 09:24:17] "←[33mGET /favicon.ico HTTP/1.1←[0m" 404 -
172.17.0.1 - - [23/Dec/2022 09:24:43] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [23/Dec/2022 09:33:30] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [23/Dec/2022 09:33:51] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [23/Dec/2022 09:34:19] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [23/Dec/2022 15:14:59] "GET / HTTP/1.1" 200 -
```