

به نام خداوند جان و خرد

تمرین تحویلی دوم بخش اول
درس رایانش ابری

امیرفاضل کوزه گر کالجی

9931099

در ابتدا یک داکر فایل می نویسیم تا ایمج آلپاین را دریافت کرده و به آن قابلیت curl را اضافه کند.

```
FROM alpine:3.18

RUN apk update && apk add curl

CMD ["sh"]

~
~
~
~
~
~
~
~
```

حال ایمج ساخته شده را بیلد و آن را ران می کنیم تا از وجود curl در آن باخبر شویم.

```
mr_amirfazel@Tahmoore: /media/mr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker run alpine-fin
mr_amirfazel@Tahmoore: /media/mr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker run -it alpine-fin
/ # curl --help
Usage: curl [options...] <url>
-d, --data <data>          HTTP POST data
-f, --fail                 Fail fast with no output on HTTP errors
-h, --help <category>     Get help for commands
-i, --include               Include protocol response headers in the output
-o, --output <file>        Write to file instead of stdout
-O, --remote-name           Write output to a file named as the remote file
-s, --silent               Silent mode
-S, --upload-file <file>   Transfer local FILE to destination
-u, --user <user:password> Server user and password
-A, --user-agent <name>    Send User-Agent <name> to server
-v, --verbose              Make the operation more talkative
-V, --version              Show version number and quit

This is not the full help, this menu is stripped into categories.
Use "--help category" to get an overview of all categories.
For all options use the manual or "--help all".
/ #
```

در ادامه باید این ایمج را داخل داکر هاب آپلود کنیم. برای این منظور خواهیم داشت:

```
alpine:latest - 3 years ago - 3.5mb
mr_amirfazel@Tahmoore: /media/mr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker tag alpine-fin:latest amirfazel/alpine-fin:1.0
mr_amirfazel@Tahmoore: /media/mr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker push amirfazel/alpine-fin
mr_amirfazel@Tahmoore: /media/mr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker push amirfazel/alpine-fin:1.0
The push refers to repository [docker.io/amirfazel/alpine-fin]
97dc59f9d96e: Pushed
cc2447e1835a: Mounted from library/alpine
1.0: digest: sha256:8e016fe4b78dbaba90745cb5452a29ca577bbc37dec68987399d0b13a74057b5 size: 738
mr_amirfazel@Tahmoore: /media/mr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ |
```

نتیجه در داکر هاب به شکل زیر خواهد بود:

در ادامه این ایمج را از داکر هاب دریافت خواهیم کرد:

```
nr_amirfazel@Tahmoorees:/media/nr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker pull amirfazel/alpine-fin:1.0
1.0: Pulling from amirfazel/alpine-fin
Digest: sha256:8e010fe4b78dbba90745cb5452a29ca577bbc37dec68987399d0b13a74057b5
Status: Image is up to date for amirfazel/alpine-fin:1.0
nr_amirfazel@Tahmoorees:/media/nr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
amirfazel/alpine-fin 1.0             5f53069293a6   17 minutes ago 13.4MB
alpine-fin          latest          5f53069293a6   17 minutes ago 13.4MB
fazel/alp           latest          3f933a3f1be2   17 minutes ago 13.4MB
amirfazel/alpine_img 1.0             78b54d191161   22 hours ago   11.5MB
alpine_img2         latest          78b54d191161   22 hours ago   11.5MB
alpine_img          latest          d6f54ac470c2   22 hours ago   9.21MB
redis               latest          7f27d00cb8e0   6 days ago     138MB
<none>              <none>          8ca4d80f4f35   5 weeks ago     7.34MB
postgres            latest          f7d9a0d4223b   7 weeks ago     417MB
redis               <none>          0ec8ab59a35f   5 months ago   117MB
backend             1.0.0           3df62bee67fc   9 months ago   21.7MB
mysql               latest          b939d379d46e   9 months ago   514MB
diamol/base         latest          9fc3f74c0b53   2 years ago     7.15MB
diamol/ch03-web-ping latest          c748a03c9314   3 years ago     75.3MB
diamol/ch02-hello-diamol latest          387f84126dbc   3 years ago     5.55MB
nr_amirfazel@Tahmoorees:/media/nr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker run amirfazel/alpine-fin:1.0
nr_amirfazel@Tahmoorees:/media/nr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker run -it amirfazel/alpine-fin:1.0
/ # curl --version
curl 8.4.0 (x86_64-alpine-linux-musl) libcurl/8.4.0 OpenSSL/3.1.3 zlib/1.2.13 brotli/1.0.9 libidn2/2.3.4 nghttp2/1.57.0
Release Date: 2023-10-11
Protocols: dict file ftp ftps gopher gophers http https imap imaps mqtt pop3 pop3s rtsp smb smbs smtp smtps telnet tftp ws wss
Features: alt-svc AsynchDNS brotli HSTS HTTP2 HTTPS-proxy IDN IPV6 Largefile libz NTLM SSL threadsafe TLS-SRP UnixSockets
/ # exit
```

در این عکس قابل مشاهده است که

1. ایمج را از داکر هاب دریافت می کنیم
2. آن را اجرا و لیست ایمج ها را مشاهده می کنیم
3. از وجود دستور curl در آن اطمینان می یابیم

حال دستور curl را روی وبسایت گوگل اجرا خواهیم کرد:

```
Run 'docker container COMMAND --help' for more information on a command.
mr_amirfazel@Tahmoores:~/media/mr_amirfazel/New Volume/University/Semester-7/Cloud Computing/assignments/2/step1$ docker run -it amirfazel/alpine-fn:1.0
/ # curl google.com
<!DOCTYPE html>
<html lang=en>
  <meta charset=utf-8>
  <meta name=viewport content="initial-scale=1, minimum-scale=1, width=device-width">
  <title>Error 403 (Forbidden)!!1</title>
  <style>
    *{margin:0;padding:0}html,code{font:15px/22px arial,sans-serif}html{background:#fff;color:#222;padding:15px}body{margin:7% auto 0;max-width:390px;min-height:180px;padding:30px 0 15px}* > body{background:url(//www.google.com/images/errors/robot.png) 100% 5px no-repeat;padding-right:205px}p{margin:11px 0 22px;overflow:hidden}ins{color:#777;text-decoration:none}a img{border:0}@media screen and (max-width:772px){body{background:none;margin-top:0;max-width:none;padding-right:0}}#logo{background:url(//www.google.com/images/branding/googlelogo/1x/googlelogo_color_150x54dp.png) no-repeat;margin-left:-5px}@media only screen and (min-resolution:192dpi){#logo{background:url(//www.google.com/images/branding/googlelogo/2x/googlelogo_color_150x54dp.png) no-repeat 0% 0%/100% 100%;-moz-border-image:url(//www.google.com/images/branding/googlelogo/2x/googlelogo_color_150x54dp.png) 0}}@media only screen and (-webkit-min-device-pixel-ratio:2){#logo{background:url(//www.google.com/images/branding/googlelogo/2x/googlelogo_color_150x54dp.png) no-repeat;-webkit-background-size:100% 100%}}#logo{display:inline-block;height:54px;width:150px}
  </style>
  <a href=//www.google.com><span id=logo aria-label=Google></span></a>
  <p><b>403.</b><ins>That's an error.</ins>
  <p>Your client does not have permission to get URL <code></code> from this server. <ins>That's all we know.</ins>
  #
```

نتیجه اجرای دستور curl روی وبسایت google.com طبق عکس بالا قابل مشاهده می باشد.

گام دوم

در ابتدا ایمیج مربوط به redis را ران می کنیم.

```
mr_amirfazel@Tahmoores:~$ docker run --name redis-weather -p 6379:6379 -d redis:4.0
Unable to find image 'redis:4.0' locally
4.0: Pulling from library/redis
54fec2fa59d0: Pull complete
9c94e11103d9: Pull complete
04ab1bfc453f: Pull complete
7988789e1fb7: Pull complete
8ce1bab2086c: Pull complete
40e134f79af1: Pull complete
Digest: sha256:2e03fdd159f4a08d2165ca1c92adde438ae4e3e6b0f74322ce013a78ee81c88d
Status: Downloaded newer image for redis:4.0
b59759177d4b5255615d9920244162a1a2817450416281053933b2ca0935acc5
```

تا از وجود redis در برنامه مان مطمئن باشیم.

در ادامه برنامه را به زبان پایتون و با فریمورک flask توسعه می دهیم و در آن از redis که توسط داکر آماده شده بهره می بریم.

قسمتی از api سرور نوشته شده در شکل زیر قابل مشاهده می باشد:

```
9
10 EXPIRATION_TIME = os.getenv('EXPIRATION_TIME')
11 PORT = os.getenv('PORT')
12 CITY = os.getenv('CITY')
13 API_KEY = os.getenv('API_KEY')
14
15 redis_client = RedisHandler()
16
17 app = Flask(__name__)
18 CORS(app)
19
20
21 @app.route('/ping', methods=['GET'])
22 def pong():
23     return 'pong', 200
24
25 @app.route('/api/weather/', methods=['GET'])
26 def get_temperature_info(city = CITY):
27     cached_value = redis_client.redis_get(city)
28     result = {}
29
30     if cached_value is not None:
31         result = cached_value
32     else:
33         try:
34             weather_data = get_city_temperature_data(API_
35 except Exception as e:
36             return jsonify({'error': e.__str__()}), 401
37
38
39         redis_client.redis_cache(city, EXPIRATION_TIME, w
40         result = weather_data.to_json()
41
42     return result, 200
43
44 if __name__ == '__main__':
45     from argparse import ArgumentParser
46
47     parser = ArgumentParser()
48     parser.add_argument('-p', '--port', type=int, default
49     args = parser.parse_args()
50     port = args.port
51     app.run(host='0.0.0.0', port=port)
```

حال باید برنامه خود را داکرایز کنیم بدین منظور در اولین گام، یک داکر فایل برای برنامه مان می نویسیم که به شکل زیر خواهد بود:

```
weather > step2 > Dockerfile
1 FROM python:3.10-slim
2
3 WORKDIR /app
4
5 COPY app.py requirements.txt /app/
6 COPY handlers/ /app/handlers/
7
8 RUN pip install --no-cache-dir -r requirements.txt
9
10 ENV API_KEY=5fadda87e2msh91fbec44114e41dp147ddejsn949782727d43
11 ENV EXPIRATION_TIME=300
12 ENV PORT=8080
13 ENV CITY=tehran
14
15 EXPOSE $PORT
16
17 CMD ["python", "app.py"]
18
```

در ادامه با دستور `docker build` ایمیج برنامه خود را می سازیم و این ایمیج قابل مشاهده خواهد بود:

```
creating py_w... done
● root@srv9958975399:~/cc_ass2/pyWeather/step2# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
pyw latest f3813b7f79fd 18 hours ago 143MB
redis latest 961dda256baa 13 days ago 138MB
redis 4.0 191c4017dcdd 3 years ago 89.3MB
○ root@srv9958975399:~/cc_ass2/pyWeather/step2#
```

در ادامه دو ایمجی که داریم را ران می کنیم تا برنامه مان به طور کامل و توسط داکر اجرا شود. نکات قابل توجه:

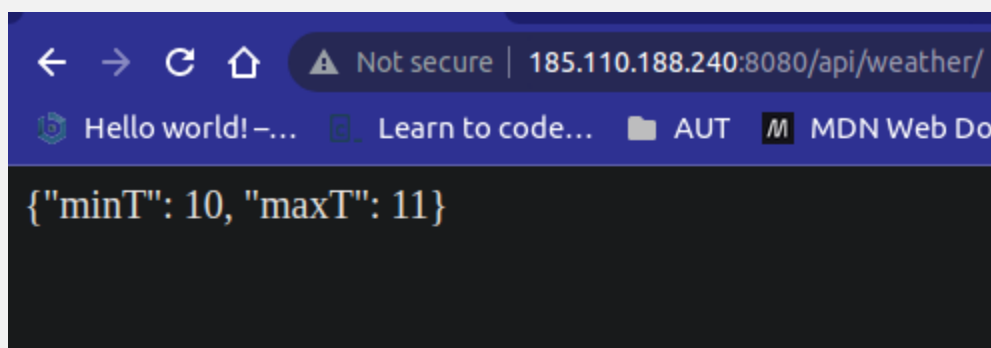
- یک شبکه برای دو کانتینر ایجاد می کنیم و آنها را به آن شبکه متصل می کنیم. نام شبکه weather می باشد.
- در دستور کار، خواسته شده تا برای ذخیره داده های ردیس از یک volume استفاده شود تا در صورت توقف و اجرای کانتینر ردیس، اطلاعات ذخیره شده حفظ شوند.
- دو مورد بالا در همین اجرای کانتینر ها مشاهده می شوند.

```

root@srv9958975399:~/cc_ass2/pyWeather/step2# docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
pyw latest f3813b7f79fd 18 hours ago 143MB
redis latest 961dda256baa 13 days ago 138MB
redis 4.0 191c4017dcdd 3 years ago 89.3MB
root@srv9958975399:~/cc_ass2/pyWeather/step2# docker network ls
NETWORK ID NAME DRIVER SCOPE
fd90d99ec469 bridge bridge local
f20d722a03b4 host host local
92e3ec65d4f9 none null local
de1fc4b8952d weather bridge local
root@srv9958975399:~/cc_ass2/pyWeather/step2# pwd
/root/cc_ass2/pyWeather/step2
root@srv9958975399:~/cc_ass2/pyWeather/step2# docker run -d -p 6379:6379 --network weather -v /root/cc_ass2/pyWeather/step2:/data redis:4.0
f79196ad96628b4a861ced98e046d50b507394b2fc23e816a1561cac7395324e
root@srv9958975399:~/cc_ass2/pyWeather/step2# docker run -d -p 8080:8080 --network weather pyw
9c56c316255501f2ac49e33b54812ab790f82a6c9ce17c2b6b0a54b7ce114b4e
root@srv9958975399:~/cc_ass2/pyWeather/step2# docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
9c56c3162555 pyw "python app.py" 4 seconds ago Up 3 seconds 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp quizzical_chandrasekhar
f79196ad9662 redis:4.0 "docker-entrypoint.s..." 31 seconds ago Up 30 seconds 0.0.0.0:6379->6379/tcp, :::6379->6379/tcp funny_blackwell
root@srv9958975399:~/cc_ass2/pyWeather/step2#

```

طبق تصویر زیر می توان دریافت که ارتباط بین برنامه و ردیس ایجاد شده و نتایج، مطابق انتظار ما هستند:



```

{"minT": 10, "maxT": 11}

```

داده های موجود در ردیس، در یک فایل با فرمت db ذخیره می شوند:

```
backup.db
1 REDIS00080 redis-ver4.0.140
2 redis-bits000ctimeV`e0used-memC
3 000aof-preamble00005t;~0$
```

```
root@srv9958975399:~/cc_ass2/pyWeather/step2# docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
9c56c3162555  pyw       "python app.py"         10 minutes ago Up 10 minutes 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp  quizzical_chandrasekhar
f79196ad9662  redis:4.0 "docker-entrypoint.s..." 11 minutes ago Up 11 minutes 0.0.0.0:6379->6379/tcp, :::6379->6379/tcp  funny_blackwell

root@srv9958975399:~/cc_ass2/pyWeather/step2# docker inspect 9c
[
  {
    "Id": "9c56c316255501f2ac49e33b54812ab790f82a6c9ce17c2b6b0a54b7ce114b4e",
    "Created": "2023-11-24T14:50:36.03156415Z",
    "Path": "python",
    "Args": [
      "app.py"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 38494,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-11-24T14:50:36.249803013Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:f3813b7f79fdb1f57e1c897988b5f7f480ff2d88a4578744532b57dd3cd95d4",
    "ResolvConfPath": "/var/lib/docker/containers/9c56c316255501f2ac49e33b54812ab790f82a6c9ce17c2b6b0a54b7ce114b4e/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/9c56c316255501f2ac49e33b54812ab790f82a6c9ce17c2b6b0a54b7ce114b4e/hostname",
    "HostsPath": "/var/lib/docker/containers/9c56c316255501f2ac49e33b54812ab790f82a6c9ce17c2b6b0a54b7ce114b4e/hosts",
    "LogPath": "/var/lib/docker/containers/9c56c316255501f2ac49e33b54812ab790f82a6c9ce17c2b6b0a54b7ce114b4e/9c56c316255501f2ac49e33b54812ab790f82a6c9ce17c2b6b0a54b7ce114b4e-json.log",
    "Name": "/quizzical_chandrasekhar",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "docker-default",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,
      "ContainerIDFile": "",
      "LogConfig": {
        "Type": "json-file",
        "Config": {}
      },
      "NetworkMode": "weather",
      "PortBindings": {
        "8080/tcp": [
          {
            "HostIp": "",
            "HostPort": "8080"
          }
        ]
      },
      "RestartPolicy": {
        "Name": "no"
      }
    }
  }
]
```

با نوشتن
دستور
docker
inspect
اطلاعات
کانتینر برنامه
خود را رصد
می کنیم:

حال دستور
docker
stats
را مشاهده می

کنیم تا اطلاعاتی را در مورد منابع مصرفی کانتینر هایمان مشاهده کنیم:

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
9c56c3162555	quizzical_chandrasekhar	0.01%	27.96MiB / 957.5MiB	2.92%	14.1kB / 7.27kB	2MB / 172kB	1
f79196ad9662	funny_blackwell	0.10%	2.457MiB / 957.5MiB	0.26%	6.28kB / 4.78kB	1.45MB / 0B	4

در این بخش، ایميج ساخته شده را در داکر هاب، پوش می کنیم. طبق عکس زیر:

```

root@srv9958975399:~/cc_ass2/pyWeather/step2# docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
pyw           latest    f3813b7f79fd   18 hours ago   143MB
redis         latest    961dda256baa   2 weeks ago    138MB
redis         4.0       191c4017dcdd   3 years ago     89.3MB
root@srv9958975399:~/cc_ass2/pyWeather/step2# docker tag pyw amirfazel/py_weather:1.0
root@srv9958975399:~/cc_ass2/pyWeather/step2# docker push amirfazel/py_weather:1.0
The push refers to repository [docker.io/amirfazel/py_weather]
001e70d21f09: Pushed
6f5bbe8aaa9: Pushed
d36ff43bf8ac: Pushed
60c917be404a: Pushed
c5a1d9cc2ba7: Mounted from library/python
f54f78a46f40: Mounted from library/python
fb35fdbdd2d7: Mounted from library/python
9dd8dac9def3: Mounted from library/python
92770f546e06: Mounted from library/redis
1.0: digest: sha256:05c2cc9b1fb146500c16103f7ce88719b9f7c4e5be41a5a76e96ed78a729d0ab size: 2202
root@srv9958975399:~/cc_ass2/pyWeather/step2#
  
```

و با مراجعه به سایت داکر هاب، می توانیم دو ایميج ساخته شده در این تمرین را مشاهده کنیم:

amirfazel / py_weather Contains: Image Last pushed: a minute ago	Inactive ☆ 0 📦 0 🌐 Public
amirfazel / alpine-fin Contains: Image Last pushed: 16 days ago	Inactive ☆ 0 📦 2 🌐 Public

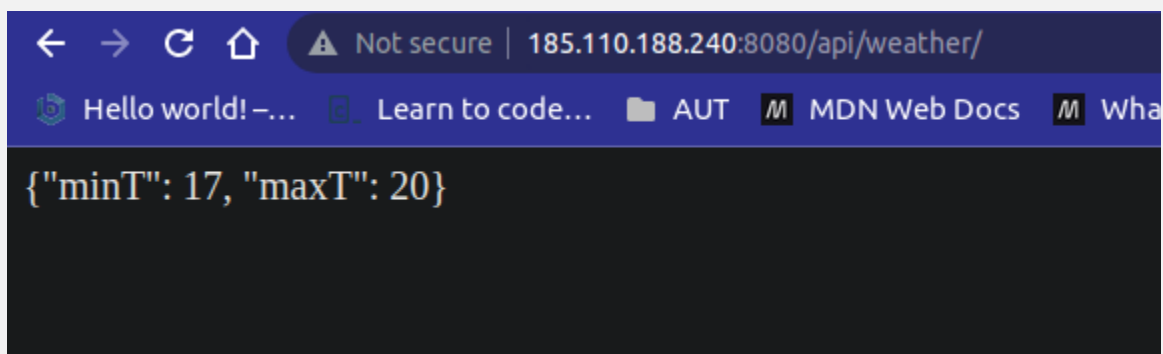
در مرحله آخر اجرای تمامی کانتینر هایمان، ایجاد شبکه ها و volume ها را بر عهده docker-compose می گذاریم. برای این کار، صرفا نیاز است تا یک فایل docker-compose.yml بنویسیم. طبق عکس زیر، این فایل بدین صورت نوشته شده است:

```
pyWeather > step2 > docker-compose.yml
1  version: '3'
2
3  services:
4    flask-app:
5      build:
6        context: ./
7      image: pyw
8      container_name: py_w
9      ports:
10       - "8080:8080"
11     networks:
12       - weather_network
13     depends_on:
14       - redis
15     environment:
16       - API_KEY=5fadda87e2msh91fbec44114e41dp147ddejsn949782727d43
17       - EXPIRATION_TIME=300
18       - PORT=8080
19       - CITY=venice
20
21
22     redis:
23       image: redis:4.0
24       container_name: redis_w
25       ports:
26         - "6379:6379"
27       networks:
28         - weather_network
29       volumes:
30         - redis-data:/data
31
32
33
34     volumes:
35       redis-data:
36
37     networks:
38       weather_network:
39
```

حال با اجرای دستور `docker-compose up -d` کانتینر ها را، اجرا می کنیم:

```
root@srv9958975399:~/cc_ass2/pyWeather/step2# docker-compose up -d
Creating network "step2_weather_network" with the default driver
Creating redis_w ... done
Creating py_w ... done
root@srv9958975399:~/cc_ass2/pyWeather/step2#
```

برای اطمینان از اجرای صحیح برنامه به سراغ وب اپلیکیشن موجود رفته و آن را اجرا می کنیم:



بخش دوم

گزارش اول

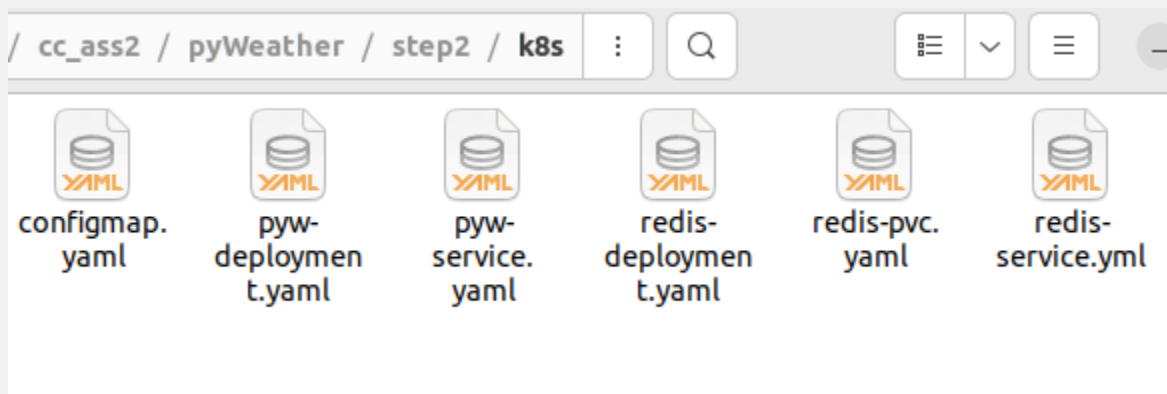
ابتدا minikube را نصب و آن را اجرا می کنیم:

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get nodes
NAME          STATUS    ROLES          AGE   VERSION
minikube      Ready    control-plane  9h    v1.28.3
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$
```

گزارش دوم

فایل های مرتبط با کوبر را ایجاد می کنیم. این فایل ها عبارت اند از:



در ادامه این فایل ها را در کلاستر کوبرنتیز اعمال می کنیم:

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl apply -f k8s/
configmap/py-weather-config unchanged
deployment.apps/py-weather-deployment unchanged
service/py-weather-service unchanged
deployment.apps/redis-deployment unchanged
persistentvolumeclaim/redis-pvc unchanged
service/redis-service unchanged
```

حال، وضعیت پاد های در حال اجرا را با اجرای دستور `kubectl get po` مشاهده می کنیم:

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get po
NAME                                READY   STATUS              RESTARTS   AGE
py-weather-deployment-69ff7cb956-xn6tt 0/1     ImagePullBackOff    0           31m
py-weather-deployment-d9d78bd7f-8j5xc  1/1     Running             0           35m
py-weather-deployment-d9d78bd7f-l4mlz  1/1     Running             0           35m
py-weather-deployment-d9d78bd7f-x9m29  1/1     Running             0           37m
redis-deployment-5b6659d5f-hbdrh       1/1     Running             0           45m
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$
```

همانطور که در تصویر مشخص است، سه پاد اجرایی از برنامه خود و یک پاد اجرایی از ردیس در اختیار داریم.

برای مشاهده deployment و service ها نیز دستور مرتبط با آنها را اجرا می کنیم. داریم:

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get deployment
NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
py-weather-deployment              3/3     1             3           46m
redis-deployment                   1/1     1             1           46m
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$
```

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get svc
NAME                                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes                          ClusterIP    10.96.0.1        <none>        443/TCP    9h
py-weather-service                  ClusterIP    10.103.147.110   <none>        8080/TCP   69m
redis-service                       ClusterIP    10.96.105.123    <none>        6379/TCP   9h
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$
```

A. نام پاد ها بر اساس نامی که در `metdata` فایل `pyw-deployment.yaml` مشخص کرده ایم، در کنار یک شناسه یا هَش تولید شده توسط خود کوبرنتیز، نام گذاری می شوند.

B. با اجرای دستور `kubectl get po -o wide` می توانیم آدرس IP تک تک پاد ها را مشاهده کنیم:

```
tahmoo@zodiac: ~/Desktop/cc_ass2/pyWeather/step2$ kubectl get po -o wide
NAME                                READY    STATUS              RESTARTS   AGE    IP             NODE             NOMINATED NODE   READINESS GATES
py-weather-deployment-69ff7cb956-xn6tt 0/1      ImagePullBackOff    0          38m    10.244.0.30    minikube         <none>            <none>
py-weather-deployment-d9d78bd7f-8j5xc 1/1      Running             0          43m    10.244.0.29    minikube         <none>            <none>
py-weather-deployment-d9d78bd7f-l4nlz 1/1      Running             0          43m    10.244.0.28    minikube         <none>            <none>
py-weather-deployment-d9d78bd7f-x9m29 1/1      Running             0          44m    10.244.0.27    minikube         <none>            <none>
redis-deployment-5b6659d5f-hbdrh       1/1      Running             0          52m    10.244.0.17    minikube         <none>            <none>
```

همانطور که مشخص می‌شود IP پاد ها با یکدیگر متفاوت است و هنگام از دست رفتن یک پاد، آدرس IP پاد دیگر در دسترس نیست و نمی‌توان از آن استفاده کرد. برای همین از سرویس استفاده می‌کنیم.

C. تعیین IP ثابت و یکتا توسط سرویس هایی از نوع clusterIP اتفاق می‌افتد. سرویس ها به طور پیشفرض از این نوع هستند. برای همین نوع آن را تعیین نمی‌کنیم.

D. در فایل pyw-service.yaml، مقدار port و targetPort را برابر با 8080 تنظیم کرده ایم:

```
pyw-deployment.yaml  x  pyw-service.yaml  x
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: py-weather-service
5 spec:
6   selector:
7     app: py-weather
8   ports:
9     - protocol: TCP
10       port: 8080
11       targetPort: 8080
```

همین مقدار را می‌توانیم با نوشتن دستور `kubectl get svc/py-weather-service -o yaml`، در محیط کوبر مشاهده کنیم:

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get svc/py-weather-service -o yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Service","metadata":{"annotations":{},"name":"py-weather-service","namespace":"default"},"spec":{"ports":[{"port":8080,"protocol":"TCP","targetPort":8080}],"selector":{"app":"py-weather"}}}
  creationTimestamp: "2023-11-30T06:46:14Z"
  name: py-weather-service
  namespace: default
  resourceVersion: "1679"
  uid: d9351191-db89-44a3-8f7c-6c7bfbf89a82
spec:
  clusterIP: 10.103.147.110
  clusterIPs:
  - 10.103.147.110
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - port: 8080
    protocol: TCP
    targetPort: 8080
  selector:
    app: py-weather
  sessionAffinity: None
  type: ClusterIP
status:
  loadBalancer: {}
```

سوال سوم:

چون ابتدا یک پاد برای ردیس ایجاد کرده بودیم، در این مرحله پاد ردیس را پاک می‌کنیم و طبق مشاهده، خواهیم دید که پس از مدتی یک پاد ردیس دیگر جایگزین آن می‌شود:

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get po
NAME                                READY   STATUS             RESTARTS   AGE
py-weather-deployment-69ff7cb956-xn6tt 0/1     ImagePullBackOff   0           56m
py-weather-deployment-d9d78bd7f-8j5xc 1/1     Running            0           61m
py-weather-deployment-d9d78bd7f-l4mlz 1/1     Running            0           61m
py-weather-deployment-d9d78bd7f-x9m29 1/1     Running            0           62m
redis-deployment-5b6659d5f-hbdrh       1/1     Running            0           70m
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl delete pods redis-deployment-5b6659d5f-hbdrh
pod "redis-deployment-5b6659d5f-hbdrh" deleted
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get po
NAME                                READY   STATUS             RESTARTS   AGE
py-weather-deployment-69ff7cb956-xn6tt 0/1     ImagePullBackOff   0           57m
py-weather-deployment-d9d78bd7f-8j5xc 1/1     Running            0           61m
py-weather-deployment-d9d78bd7f-l4mlz 1/1     Running            0           61m
py-weather-deployment-d9d78bd7f-x9m29 1/1     Running            0           62m
redis-deployment-5b6659d5f-ktl52       1/1     Running            0           9s
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$
```

گزارش سوم

A. در ادامه مرحله قبل، از پاد ردیس جدید یک لاگ میگیریم تا مشاهده کنیم تمامی مقادیر کش شده پاک شده اند.:

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get po
NAME                                READY    STATUS              RESTARTS   AGE
py-weather-deployment-69ff7cb956-xn6tt 0/1      ImagePullBackOff    0           56m
py-weather-deployment-d9d78bd7f-8j5xc 1/1      Running             0           61m
py-weather-deployment-d9d78bd7f-l4mlz 1/1      Running             0           61m
py-weather-deployment-d9d78bd7f-x9m29 1/1      Running             0           62m
redis-deployment-5b6659d5f-hbdrh       1/1      Running             0           70m
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl delete pods redis-deployment-5b6659d5f-hbdrh
pod "redis-deployment-5b6659d5f-hbdrh" deleted
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get po
NAME                                READY    STATUS              RESTARTS   AGE
py-weather-deployment-69ff7cb956-xn6tt 0/1      ImagePullBackOff    0           57m
py-weather-deployment-d9d78bd7f-8j5xc 1/1      Running             0           61m
py-weather-deployment-d9d78bd7f-l4mlz 1/1      Running             0           61m
py-weather-deployment-d9d78bd7f-x9m29 1/1      Running             0           62m
redis-deployment-5b6659d5f-ktl52       1/1      Running             0           9s
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl logs -f redis-deployment-5b6659d5f-ktl52
1:C 30 Nov 08:18:46.589 # o000o000o000o Redis is starting o000o000o000o
1:C 30 Nov 08:18:46.589 # Redis version=4.0.14, commit=00000000, modified=0, pid=1, just started
1:C 30 Nov 08:18:46.589 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 30 Nov 08:18:46.592 * Running mode=standalone, port=6379.
1:M 30 Nov 08:18:46.592 # Server initialized
1:M 30 Nov 08:18:46.592 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. This will create latency and memory usage issues with Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
1:M 30 Nov 08:18:46.593 * DB loaded from disk: 0.000 seconds
1:M 30 Nov 08:18:46.593 * Ready to accept connections
^C
```

فایل PVC را به شکل زیر می نویسیم:

```
pyw-deployment.yaml  x  redis-pvc.yaml  >
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redis-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl get pvc
NAME      STATUS    VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
redis-data Bound     pvc-367e742d-5a12-4352-ab13-57035d5595f4  1Gi        RWO            standard       17h
redis-pvc Bound     pvc-7868ee44-a2f0-4aed-bcbe-5536177f8c4e  1Gi        RWO            standard       9h
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$
```


گزارش چهارم

A. با اجرای دستوری تحت عنوان minikube addons list خروجی زیر را دریافت می‌کنیم:

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ minikube addons list
```

ADDON NAME	PROFILE	STATUS	MAINTAINER
ambassador	minikube	disabled	3rd party (Ambassador)
auto-pause	minikube	disabled	minikube
cloud-spanner	minikube	disabled	Google
csi-hostpath-driver	minikube	disabled	Kubernetes
dashboard	minikube	disabled	Kubernetes
default-storageclass	minikube	enabled ✓	Kubernetes
efk	minikube	disabled	3rd party (Elastic)
freshpod	minikube	disabled	Google
gcp-auth	minikube	disabled	Google
gvisor	minikube	disabled	minikube
headlamp	minikube	disabled	3rd party (kinvolk.io)
helm-tiller	minikube	disabled	3rd party (Helm)
inacel	minikube	disabled	3rd party (InAccel [info@inacel.com])
ingress	minikube	disabled	Kubernetes
ingress-dns	minikube	disabled	minikube
inspektor-gadget	minikube	disabled	3rd party (inspektor-gadget.io)
istio	minikube	disabled	3rd party (Istio)
istio-provisioner	minikube	disabled	3rd party (Istio)
kong	minikube	disabled	3rd party (Kong HQ)
kubeflow	minikube	disabled	3rd party
kubevirt	minikube	disabled	3rd party (KubeVirt)
logviewer	minikube	disabled	3rd party (unknown)
metallb	minikube	disabled	3rd party (MetalLB)
metrics-server	minikube	disabled	Kubernetes
nvidia-device-plugin	minikube	disabled	3rd party (NVIDIA)
nvidia-driver-installer	minikube	disabled	3rd party (Nvidia)
nvidia-gpu-device-plugin	minikube	disabled	3rd party (Nvidia)
olm	minikube	disabled	3rd party (Operator Framework)
pod-security-policy	minikube	disabled	3rd party (unknown)
portainer	minikube	disabled	3rd party (Portainer.io)
registry	minikube	disabled	minikube
registry-aliases	minikube	disabled	3rd party (unknown)
registry-creds	minikube	disabled	3rd party (UPMC Enterprises)
storage-provisioner	minikube	enabled ✓	minikube
storage-provisioner-gluster	minikube	disabled	3rd party (Gluster)
storage-provisioner-rancher	minikube	disabled	3rd party (Rancher)
volumesnapshots	minikube	disabled	Kubernetes

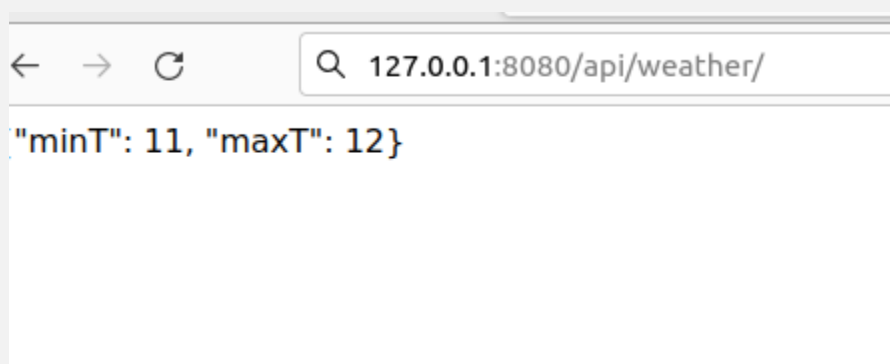
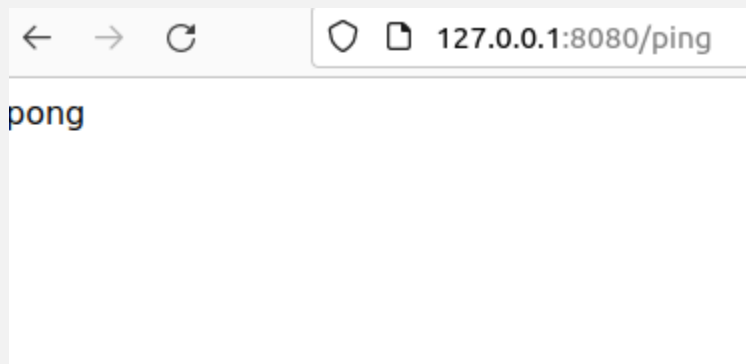
با توجه به تصویر بالا، مشاهده می‌شود که storage-provisioner فعال است پس می‌توان گفت که عملیات dynamic provisioning به طور پیش‌فرض فعال می‌شود.

B. خیر. نمی‌توان بیش از یک claim را به یک volume اختصاص داد. مزیت این مورد این است که دیتای پاد ها با یکدیگر قاطی نشده و روی هم تاثیر نمی‌گذارند. زیرا که هر پاد، فضای مخصوص به خودش را دارد.

سوال پنجم:

با استفاده از port-forward، از درون ماشین خودمان به سرویس های اجرا شده درون کلاستر ها، می‌توانیم درخواست بزنیم.

```
tahmoo@zodiac:~/Desktop/cc_ass2/pyWeather/step2$ kubectl port-forward svc/py-weather-service 8080:8080
Forwarding from 127.0.0.1:8080 -> 8080
Forwarding from [::1]:8080 -> 8080
```



گزارش پنجم

- A. طبق تصویر بالا، صحت اجرای برنامه را می‌توان نتیجه گرفت.
- B. در این مورد اما، با انجام port-forward درخواست های مان فقط به پاد اول ارسال می‌شوند و لود بالانس صورت نمی‌گیرد.