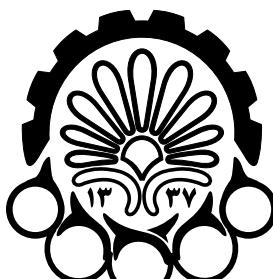


به نام خدا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

رایانش ابری

گزارش پروژه پایانی

داکر و کوبرنتیز

استاد درس:

آقای دکتر جوادی

نویسنده‌گان:

رادین شایانفر

ماهان احمدوند



گام دوم

محتویات Dockerfile ساخته شده به شکل زیر می باشد:

```
Cloud-Computing-Project — vim dockerfile — 79x25

FROM python:3.9-alpine AS build
#
RUN python -m venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"

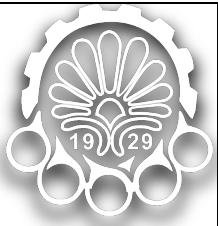
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

FROM python:3.9-alpine AS target
WORKDIR /app
COPY --from=build /opt/venv /opt/venv
ENV PATH="/opt/venv/bin:$PATH"

ENV ENV_FILE=/env/.env

COPY /app .
CMD ["python", "app.py"]
-
```

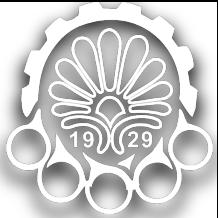
برای ساخت این ایمیج، از `multistage build` استفاده می کنیم. به این صورت که ابتدا در یک ایمیج با عنوان `build`، کتابخانه های مورد نیاز اجرای برنامه را نصب کرده، سپس با کپی فایل های کتابخانه در ایمیج دوم (با عنوان `target`)، از این ایمیج برای اجرای برنامه استفاده می کنیم. به طور دقیق تر، ابتدا با استفاده از `From`، `Base Image` را مشخص کرده ایم که `python3.9-alpine` می باشد، سپس با استفاده از دستور `WORKDIR` مشخص کرده ایم که دایرکتوری فعلی ایمیجی که می خواهیم بسازیم کجا باشد، که `/app` را مشخص کرده ایم و در صورتی که وجود نداشته باشد خود به خود ساخته می شود، سپس فایل `requirements.txt` را که در `/app` با استفاده از `COPY`، کپی می کنیم، این فایل را بصورت جدا کپی کرده ایم، زیرا می دانیم که همواره کتابخانه های موردنیاز پروژه کمتر از کد و بقیه محتویات چهار تغییر می شوند، درنتیجه این کپی را بصورت جدا انجام داده ایم که برای مراحل بعدی `cache` شود، سپس با استفاده از دستور `RUN` و نوشتن `pip install -no-cache-dir -r requirements.txt`، کتابخانه های موردنیاز را نصب کرده ایم. سپس فایل های این کتابخانه های



نصب شده، در اینجی دوم کپی شده و درنهایت نیز کل محتویاتی که در پوششی /app وجود دارند را همگی با استفاده از دستور COPY، کپی کرده‌ایم. سپس با استفاده از CMD، دستوری که می‌خواهیم اجرا شود را که python app.py می‌باشد را مشخص می‌کنیم.

کد مربوط به سرور به شکل زیر می‌باشد:

```
1 import os
2 import redis
3 from hashlib import sha256
4 from dotenv import load_dotenv
5 from flask import Flask, jsonify, request, redirect
6
7 load_dotenv(os.getenv('ENV_FILE', '.env'))
8
9 app = Flask(__name__)
10 rdb = redis.Redis(host=os.getenv('REDIS_HOST'), port=os.getenv('REDIS_PORT'), db=os.getenv('REDIS_DB'), password=os.getenv('REDIS_SECRET'))
11
12 @app.route('/shortener', methods=['POST'])
13 def shortener():
14     # Getting post url
15     url = request.form['u']
16     # Shortening url using sha hash
17     hashed = sha256(url.encode()).hexdigest()
18     shortened = hashed[:5]
19
20     rdb.set(shortened, url, ex=int(os.getenv('URL_EX')))
21
22     return jsonify({
23         "shortened": shortened
24     })
25
26 @app.route('/<shortened>', methods=['GET'])
27 def redirector(shortened):
28     url = rdb.get(shortened)
29     if url is None:
30         return jsonify({
31             "status": 404,
32             "message": "Requested URL not found."
33         }), 404
34
35     return redirect(url.decode(), code=302)
36
37 if __name__ == '__main__':
38     app.run(host=os.getenv('HOST'), port=os.getenv('PORT'))
```



حال می خواهیم با استفاده از Dockerfile ساخته شده ایمیج موردنظر را بسازیم:

```
radin@Radin-Laptop:~/cc/project
FLASK_DEBUG=1 ... × radin@Radin-Lapt... × radin@Radin-Lapt... × radin@Radin-Lapt... × radin@Radin-Lapt...
(venv) radin > ~/cc/project ⌘ master ⌘ docker build -t ccp:1.0 +
Sending build context to Docker daemon 19.19MB
Step 1/7 : FROM python:3.9-alpine
--> 8c034acf745b
Step 2/7 : WORKDIR /app
--> Using cache
--> 4916a426405f
Step 3/7 : COPY requirements.txt .
--> Using cache
--> 10b41a18a0fb
Step 4/7 : RUN pip install --no-cache-dir -r requirements.txt
--> Running in ef2834e38702
Collecting click==8.0.3
--> Downloading click-8.0.3-py3-none-any.whl (97 kB)
Collecting Deprecat...
--> Downloading Deprecat...
Collecting Flask==2.0.2
--> Downloading Flask-2.0.2-py3-none-any.whl (95 kB)
Collecting itsdangerous==2.0.1
--> Downloading itsdangerous-2.0.1-py3-none-any.whl (18 kB)
Collecting Jinja2==3.0.3
--> Downloading Jinja2-3.0.3-py3-none-any.whl (133 kB)
Collecting MarkupSafe==2.0.1
--> Downloading MarkupSafe-2.0.1-cp39-cp39-musllinux_1_1_x86_64.whl (30 kB)
Collecting packaging==21.3
--> Downloading packaging-21.3-py3-none-any.whl (40 kB)
Collecting pyparsing==3.0.6
--> Downloading pyparsing-3.0.6-py3-none-any.whl (97 kB)
Collecting python-dotenv==0.19.2
--> Downloading python_dotenv-0.19.2-py2.py3-none-any.whl (17 kB)
```

```
radin@Radin-Laptop:~/cc/project
FLASK_DEBUG=1 ... × radin@Radin-Lapt... × radin@Radin-Lapt... × radin@Radin-Lapt... × radin@Radin-Lapt...
Downloading python_dotenv-0.19.2-py2.py3-none-any.whl (17 kB)
Collecting redis==4.1.1
--> Downloading redis-4.1.1-py3-none-any.whl (173 kB)
Collecting Werkzeug==2.0.2
--> Downloading Werkzeug-2.0.2-py3-none-any.whl (288 kB)
Collecting wrapt==1.13.3
--> Downloading wrapt-1.13.3-cp39-cp39-musllinux_1_1_x86_64.whl (80 kB)
Installing collected packages: wrapt, pyparsing, MarkupSafe, Werkzeug, packaging, Jinja2, itsdangerous, Deprecat...
--> click, redis, python-dotenv, Flask
Successfully installed Deprecat...
--> click-8.0.3
Collecting packaging==21.3
--> Downloading packaging-21.3-py3-none-any.whl (40 kB)
Collecting pyparsing==3.0.6
--> Downloading pyparsing-3.0.6-py3-none-any.whl (97 kB)
Collecting python-dotenv==0.19.2
--> Downloading python_dotenv-0.19.2-py2.py3-none-any.whl (17 kB)

WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the '/usr/local/bin/python -m pip install --upgrade pip' command.

Removing intermediate container ef2834e38702
--> c0bf6dc37941
Step 5/7 : ENV ENV_FILE=/env/.env
--> Running in 35f693b59ac5
Removing intermediate container 35f693b59ac5
--> 7cad09c4595c
Step 6/7 : COPY .
--> 0e5f6d7ed15f
Step 7/7 : CMD ["python", "app.py"]
--> Running in 00f4eba0c19c
Removing intermediate container 00f4eba0c19c
--> 5a0d49198441
Successfully built 5a0d49198441
Successfully tagged ccp:1.0
(venv) radin > ~/cc/project ⌘ master ⌘
```



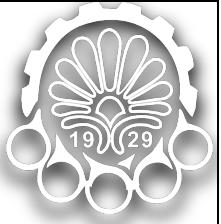
حال در داکرها ب لاجین کرده و ایمیج ساخته شده را بر روی داکرها ب ارسال می کنیم:

```
root@vps1637187394: ~/cc
FLASK_DEBU... radin@Radin... radin@Radin... radin@Radin... radin@Radin... radin@vps16... radin@Radin...
--> d22369eb9ad0
Successfully built d22369eb9ad0
Successfully tagged ccp:1.0
root@vps1637187394:~/cc# git login -u radinshayanfar
git: 'login' is not a git command. See 'git --help'.

The most similar command is
    column
root@vps1637187394:~/cc# docker login -u radinshayanfar
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@vps1637187394:~/cc# docker tag ccp:1.0 radinshayanfar/ccp:1.0
root@vps1637187394:~/cc# docker image ls
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
ccp                 1.0      d22369eb9ad0  58 seconds ago  60.8MB
radinshayanfar/ccp  1.0      d22369eb9ad0  58 seconds ago  60.8MB
<none>              <none>   2f03a45c1a98  4 minutes ago   60.8MB
python               3.9-alpine  e49e2f1d4108  17 hours ago   48.4MB
tgcopybot           1.2      02997737b330  6 weeks ago    204MB
radinshayanfar/tgcopybot  1.2  02997737b330  6 weeks ago    204MB
radinshayanfar/tgcopybot  latest   02997737b330  6 weeks ago    204MB
python               3.9-slim-bullseye ae3c4906b72c  6 weeks ago    122MB
hello-world          latest   feb5d9fea6a5   3 months ago   13.3kB
root@vps1637187394:~/cc# docker push radinshayanfar/ccp
Using default tag: latest
The push refers to repository [docker.io/radinshayanfar/ccp]
```

```
root@vps1637187394: ~/cc
FLASK_DEBU... radin@Radin... radin@Radin... radin@Radin... radin@Radin... root@vps16... radin@Radin...
Login Succeeded
root@vps1637187394:~/cc# docker tag ccp:1.0 radinshayanfar/ccp:1.0
root@vps1637187394:~/cc# docker image ls
REPOSITORY          TAG      IMAGE ID   CREATED        SIZE
ccp                 1.0      d22369eb9ad0  58 seconds ago  60.8MB
radinshayanfar/ccp  1.0      d22369eb9ad0  58 seconds ago  60.8MB
<none>              <none>   2f03a45c1a98  4 minutes ago   60.8MB
python               3.9-alpine  e49e2f1d4108  17 hours ago   48.4MB
tgcopybot           1.2      02997737b330  6 weeks ago    204MB
radinshayanfar/tgcopybot  1.2  02997737b330  6 weeks ago    204MB
radinshayanfar/tgcopybot  latest   02997737b330  6 weeks ago    204MB
python               3.9-slim-bullseye ae3c4906b72c  6 weeks ago    122MB
hello-world          latest   feb5d9fea6a5   3 months ago   13.3kB
root@vps1637187394:~/cc# docker push radinshayanfar/ccp
Using default tag: latest
The push refers to repository [docker.io/radinshayanfar/ccp]
tag does not exist: radinshayanfar/ccp:latest
root@vps1637187394:~/cc# docker push radinshayanfar/ccp:1.0
The push refers to repository [docker.io/radinshayanfar/ccp]
c0418e7fe56f: Pushed
912237bc6d0b: Pushed
c39868bdf211: Pushed
9ecb6129ff2e: Pushed
3a8f965bcc9c: Mounted from library/python
953ae22fd98f: Mounted from library/python
7c47301c002d: Mounted from library/python
f57e81d89e60: Mounted from library/python
8d3ac3489996: Mounted from library/python
1.0: digest: sha256:12fed7886d9caf93806a2c5b9a602a9d88c94332a51626e5cf0541af084ff699 size: 2200
root@vps1637187394:~/cc#
```



حال مشاهده می کنیم که ایمیج ساخته شده به درستی ارسال شده است:

The screenshot shows the Docker Hub interface for the repository `radinshayanfar / ccp`. The repository was updated a few seconds ago and is marked as "Not Scanned". It has 0 stars and 1 download. The repository is public.

General Tab:

- Advanced Image Management:** View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions. [View preview](#)
- radinshayanfar / ccp:** This repository does not have a description.
- Last pushed:** a minute ago
- Docker commands:** To push a new tag to this repository.
`docker push radinshayanfar/ccp:tagname`

Tags and Scans: This repository contains 1 tag(s).

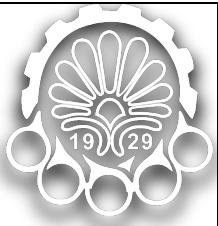
TAG	OS	PULLED	PUSHED
1.0		a minute ago	a minute ago

[See all](#)

VULNERABILITY SCANNING - DISABLED [Enable](#)

Automated Builds: Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating. Available with Pro, Team and Business subscriptions. [Upgrade to Pro](#) [Learn more](#)

Readme: Repository description is empty. Click [here](#) to edit.



گام سوم

حال زمان آن رسیده است که با نوشتن فایل‌های دیپلوبینت کوبرنتیز، پروژه‌ی خود را بر روی `minikube` بالا بیاوریم.

ConfigMap (۱)

برای انجام این کار ابتدا یک کامپوننت `ConfigMap` تعریف کرده‌ایم تا بتوانیم پورت سرور، زمان انقضای آدرس‌های ایجاد شده و آدرس سرور دیتابیس از آن خوانده شود:

```
● ● ● k8s — vim cm.yaml — 80x24
apiVersion: v1
data:
  .env: |
    HOST=0.0.0.0
    PORT=8080
    URL_EX=60

    REDIS_HOST=redis-service
    REDIS_PORT=6379
    REDIS_DB=0
kind: ConfigMap
metadata:
  creationTimestamp: null
  name: shortener-config
~
~
~
~
~
~
~
~
cm.yaml" 14L, 218B
```

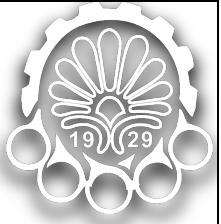


```
mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get Configmap
NAME          DATA   AGE
app-config     1      44d
config-map     1      44d
config-map.yaml 1      44d
kube-root-ca.crt 1      45d
shortener-config 1      40h
mahanahmadvand@Mahans-MacBook-Pro k8s %
```

Secret (۲)

حال نوبت به تعریف کامپوننت **Secret** رسیده است که وظیفه ذخیره سازی اسم و رمز عبور دیتابیس را بر عهده دارد. از آن جایی که این اطلاعات، مخصوصاً رمز عبور، جزو اطلاعات حساس هستند باید آنها را در **Secret** ذخیره نمود.

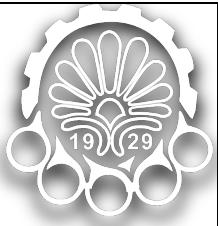
```
apiVersion: v1
kind: Secret
metadata:
  creationTimestamp: null
  name: redis-secret
data:
  REDIS_PASSWORD: dmVyeXNlY3JldHBhc3N3b3Jk
  "secret.yaml" 7L, 134B
```



```
██████████ k8s — -zsh — 80x24
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get secret
NAME          TYPE        DATA   AGE
default-token-km5cp  kubernetes.io/service-account-token  3       45d
redis-secret    Opaque      1       38h
mahanahmadvand@Mahans-MacBook-Pro k8s % ]
```

Persistent Volume & Persistent Volume Claim (۳)

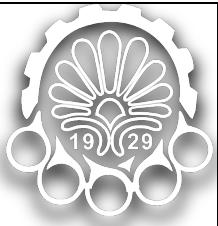
حال در این مرحله به منظور پایدار ماندن اطلاعات دیتابیس در صورت بروز مشکل برای پادهای مربوطه، لازم است تا برای آن **Persistent Volume** تعریف کنیم و در ادامه ساخت **Persistent Volume Claim** برای استفاده از آن است.



```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: redis
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 100Mi
  hostPath:
    path: /data/redis/

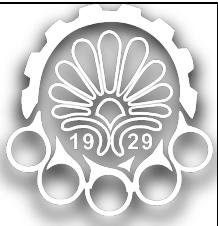
```

```
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS    CLAIM
STORAGECLASS   REASON     AGE
app        100Mi      RWO          Retain          Bound    default/app-pvc
manual           44d
redis       100Mi      RWO          Retain          Bound    default/redis-pvc
manual           39h
mahanahmadvand@Mahans-MacBook-Pro k8s %
```



```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redis-pvc
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
```

```
[mahanahmadvand@Mahans-MacBook-Pro k8s %] kubectl get pvc  
NAME      STATUS  VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS  AGE  
app-pvc   Bound    app      100Mi     RWO          manual        44d  
redis-pvc Bound    redis    100Mi     RWO          manual        39h  
[mahanahmadvand@Mahans-MacBook-Pro k8s %]
```

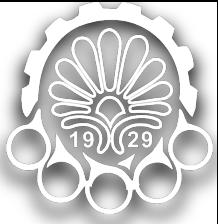


redis-deployment (۴)

حال در این مرحله یک توصیف دیپلومینت که وظیفه آماده سازی و نگهداری از دیتابیس را برعهده دارد، همچنین رمز عبور تعریف شده در Secret را نیز به این دیپلومینت اضافه کرده‌ایم، همچنین ساخته شده در مرحله قبل را نیز براین دیپلومینت سوار کرده‌ایم.

```
● ● ●   k8s — vim redis-deployment.yaml — 75x42

apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: redis
    name: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  strategy: { }
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: redis
    spec:
      containers:
        - image: redis:6.2.6-alpine
          name: redis
          env:
            - name: REDIS_PASSWORD
              valueFrom:
                secretKeyRef:
                  key: REDIS_PASSWORD
                  name: redis-secret
            command: [ "redis-server" ]
            args: [ "--requirepass", $(REDIS_PASSWORD) ]
          ports:
            - containerPort: 6379
          resources: { }
          volumeMounts:
            - mountPath: /data
              name: data
      volumes:
        - name: data
          persistentVolumeClaim:
            claimName: redis-pvc
status: { }
```

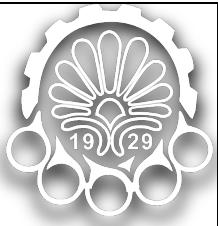


همانطور که مطابق بالا مشاهده می‌کنیم، تعداد پادها یا **replica** را برابر ۱ قرار داده‌ایم، زیرا برای دسترسی به دیتابیس فقط به یک پاد نیاز داریم، همچنین درصورت تعریف کردن بیش از یک پاد در واقع باید کنتل شود که دیتاها ریپلیکاهای مختلف باید با هم سینک شوند در واقع زمانی که یک درخواست برای یک پاد مشخص می‌رود و در آن دیتای ذخیره می‌شود، در واقع پادها دیگر نیز باید خبر دارد شوند و دیتای جدید به پادها دیگر هم منتقل شود و در واقع باید پادها دیگر نیز آپدیت شوند.

```
● ● ● k8s — -zsh — 112x26
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get deployment
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
awesome-shortener-redis   1/1     1           1           43h
awesome-shortener-shortener 2/2     2           2           43h
redis           1/1     1           1           12m
shortener       2/2     2           2           12m
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
awesome-shortener-redis-79477dd8-65x8m   1/1     Running   0          11m
awesome-shortener-shortener-78bc98d658-n4zbn 1/1     Running   0          11m
awesome-shortener-shortener-78bc98d658-nw7pg 1/1     Running   0          11m
net-check        1/1     Running   3 (7h26m ago) 4d2h
redis-f77865589-smj8t      1/1     Running   0          11m
shortener-64ccc75d94-fpc6d      1/1     Running   0          11m
shortener-64ccc75d94-jms99      1/1     Running   0          11m
mahanahmadvand@Mahans-MacBook-Pro k8s % ]
```

redis-service (۵)

حال برای دسترسی به این دیتابیس یک **Service** نیاز داریم که با استفاده از آن بتوانیم ارتباط پروژه و دیتابیس را برقرار کنیم.



```
● ○ ● k8s — vim redis-service.yaml — 79x26

apiVersion: v1
kind: Service
metadata:
  name: redis-service
spec:
  selector:
    app: redis
  ports:
    - protocol: TCP
      port: 6379
      targetPort: 6379

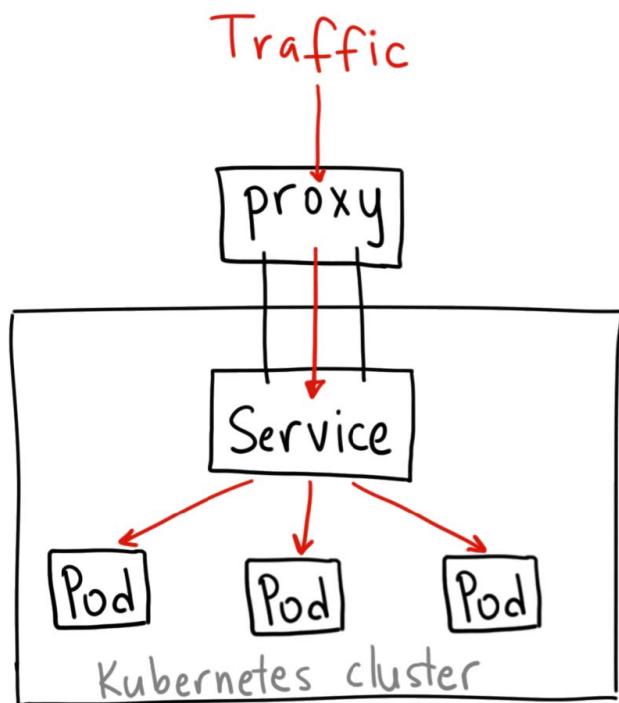
redis-service.yaml" 11L, 163B
```

```
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get services
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)  AGE
app-service    ClusterIP  10.106.60.156 <none>       8080/TCP  48d
awesome-shortener-redis-service  ClusterIP  10.99.213.194 <none>       6379/TCP  43h
awesome-shortener-shortener-service ClusterIP  10.106.208.245 <none>       80/TCP   43h
kubernetes     ClusterIP  10.96.0.1    <none>       443/TCP  49d
redis-service   ClusterIP  10.96.152.66  <none>       6379/TCP  18m
shortener-service ClusterIP  10.107.219.158 <none>       80/TCP   18m
mahanahmadvand@Mahans-MacBook-Pro k8s % ]
```

در واقع با استفاده از سرویس می توانیم با پادهای موجود در cluster ارتباط برقرار کنیم، در واقع در این حالت که نوع سرویس را تعیین نکرده ایم به این معناست که از نوع ClusterIP استفاده می کنیم،



همچنین توجه شود که در این نوع سرویس در واقع یک آی پی به سرویس ما اختصاص می‌یابد و با استفاده از پروکسی می‌توانیم به آن سرویس درخواست بزنیم و توجه شود که پادها فقط از داخل نود در دسترس هستند.



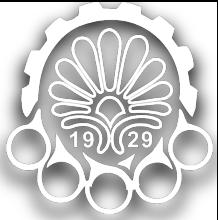
حال می‌خواهیم به یک نکته اشاره کنیم، هنگامی که یک سرویس ساخته می‌شود، این سرویس خود یک درست می‌کند که در واقع حاوی آدرس‌های آی پی پادهای ما می‌باشد:

```
[mahanahmadvand@Mahans-MacBook-Pro k8s %] kubectl get services
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
app-service    ClusterIP 10.106.60.156 <none>        8080/TCP   48d
awesome-shortener-redis-service  ClusterIP 10.99.213.194 <none>        6379/TCP   43h
awesome-shortener-shortener-service ClusterIP 10.106.208.245 <none>        80/TCP     43h
kubernetes     ClusterIP 10.96.0.1    <none>        443/TCP   49d
redis-service   ClusterIP 10.96.152.66 <none>        6379/TCP   19m
shortener-service ClusterIP 10.107.219.158 <none>        80/TCP     19m

[mahanahmadvand@Mahans-MacBook-Pro k8s %] kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
awesome-shortener-redis-79477ddd8-65x8m  1/1    Running   0          19m
awesome-shortener-shortener-78bc98d658-n4zbn 1/1    Running   0          19m
awesome-shortener-shortener-78bc98d658-nw7pg 1/1    Running   0          19m
net-check      1/1    Running   3 (7h34m ago) 4d2h
redis-f77865589-smj8t  1/1    Running   0          19m
shortener-64ccc75d94-fpc6d  1/1    Running   0          19m
shortener-64ccc75d94-jms99  1/1    Running   0          19m

[mahanahmadvand@Mahans-MacBook-Pro k8s %] kubectl get ep
NAME           ENDPOINTS   AGE
app-service    <none>        48d
awesome-shortener-redis-service  172.17.0.13:6379   43h
awesome-shortener-shortener-service 172.17.0.11:8080,172.17.0.15:8080 43h
kubernetes     192.168.49.2:8443   49d
redis-service   172.17.0.7:6379   19m
shortener-service 172.17.0.14:8080,172.17.0.9:8080 19m

[mahanahmadvand@Mahans-MacBook-Pro k8s %]
```



shortener-deployment (۶)

حال نوبت به نوشتن یک دیپلومینت رسیده است که وظیفه آماده سازی و نگهداری پادها را برای سرور اصلی عهده دارد. تعداد `replica` را برابر با ۲ تعیین کردہ‌ایم و همچنین در این قسمت `Secret` و `ConfigMap` ساخته شده را در اختیار پروژه قرار داده‌ایم، تا مقادیر لازم از طریق آن‌ها پر شود.

```
vim k8s — vim shortener-deployment.yaml — 86x44

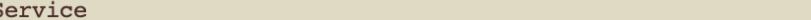
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: shortener
  name: shortener
spec:
  replicas: 2
  selector:
    matchLabels:
      app: shortener
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: shortener
    spec:
      containers:
        - image: radinshayanfar/ccp:1.1
          name: shortener
          resources:
            requests:
              cpu: "40m"
        ports:
          - containerPort: 8080
        env:
          - name: REDIS_SECRET
            valueFrom:
              secretKeyRef:
                key: REDIS_PASSWORD
                name: redis-secret
        volumeMounts:
          - mountPath: /env/.env
            subPath: .env
            name: config-map
            readOnly: true
        volumes:
          - name: config-map
            configMap:
              name: shortener-config
  status: {}
```



```
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
app-6d767887c7-5z2bs   0/1     Error      1 (10s ago) 16s
app-6d767887c7-dqfnp   1/1     Running    0          4s
app-6d767887c7-vz4zf   1/1     Running    1 (4s ago) 10s
app-7cf7c579c9-hsxgs   0/1     CrashLoopBackOff 52 (4m7s ago) 40h
curl              1/1     Running    2 (121m ago) 33d
redis-5db859f8bb-pbnsw 1/1     Running    0          16s
request-6bfbdcc5d87-2glnm 1/1     Terminating 1 (121m ago) 40h
request-6bfbdcc5d87-g5n4f 1/1     Terminating 1 (121m ago) 40h
request-c795d9b4f-k4wgs   1/1     Running    0          10s
request-c795d9b4f-wwpj    1/1     Running    0          16s
shortener-99b96f4-9f7jn   1/1     Running    0          13s
shortener-99b96f4-g9tvv    1/1     Running    0          16s
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get deployment
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
app         1/3       3           1           44d
redis       1/1       1           1           40h
request     2/2       2           2           44d
shortener   2/2       2           2           40h
mahanahmadvand@Mahans-MacBook-Pro k8s %
```

shortener-service (✓)

حال نیاز به یک سرویس داریم که با استفاده از آن بتوانیم به پرورش و در واقع سروری که توسعه داده ایم دسترسی داشته باشیم.

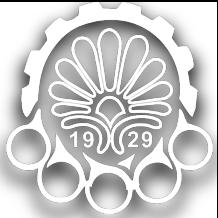


```
apiVersion: v1
kind: Service
metadata:
  name: shortener-service
spec:
  selector:
    app: shortener
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
```



```
● ● ● k8s -- zsh -- 104x27
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get services
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)      AGE
app-service   ClusterIP  10.106.60.156  <none>        8080/TCP    48d
awesome-shortener-redis-service ClusterIP  10.99.213.194  <none>        6379/TCP    43h
awesome-shortener-shortener-service ClusterIP  10.106.208.245  <none>        80/TCP      43h
kubernetes    ClusterIP  10.96.0.1       <none>        443/TCP    49d
redis-service  ClusterIP  10.96.152.66   <none>        6379/TCP    18m
shortener-service ClusterIP  10.107.219.158  <none>        80/TCP      18m
mahanahmadvand@Mahans-MacBook-Pro k8s % ]
```

```
● ● ● k8s -- zsh -- 104x27
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get services
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)      AGE
app-service   ClusterIP  10.106.60.156  <none>        8080/TCP    48d
awesome-shortener-redis-service ClusterIP  10.99.213.194  <none>        6379/TCP    43h
awesome-shortener-shortener-service ClusterIP  10.106.208.245  <none>        80/TCP      43h
kubernetes    ClusterIP  10.96.0.1       <none>        443/TCP    49d
redis-service  ClusterIP  10.96.152.66   <none>        6379/TCP    19m
shortener-service ClusterIP  10.107.219.158  <none>        80/TCP      19m
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
awesome-shortener-redis-79477ddd8-65x8m  1/1     Running   0          19m
awesome-shortener-shortener-78bc98d658-n4zbn 1/1     Running   0          19m
awesome-shortener-shortener-78bc98d658-nw7pg 1/1     Running   0          19m
net-check       1/1     Running   3 (7h34m ago) 4d2h
redis-f77865589-smj8t  1/1     Running   0          19m
shortener-64ccc75d94-fpc6d  1/1     Running   0          19m
shortener-64ccc75d94-jms99  1/1     Running   0          19m
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get ep
NAME          ENDPOINTS      AGE
app-service   <none>        48d
awesome-shortener-redis-service 172.17.0.13:6379  43h
awesome-shortener-shortener-service 172.17.0.11:8080,172.17.0.15:8080  43h
kubernetes    192.168.49.2:8443  49d
redis-service 172.17.0.7:6379  19m
shortener-service 172.17.0.14:8080,172.17.0.9:8080  19m
mahanahmadvand@Mahans-MacBook-Pro k8s % ]
```



تست کار کرد قسمت اصلی پروژه

حال در این قسمت می خواهیم کار کرد قسمت اصلی پروژه را تست کنیم.

برای این کار ابتدا از یک ایمیج آماده که در تمرین دوم ساخته ایم (ایمیج آماده alpine) که بر روی آن curl نصب شده است، استفاده می کنیم و ابتدا برای URL Shortening درخواست می دهیم و سپس چک می کنیم که درستی انجام شده باشد:

```
● ○ ● k8s — kubectl run -it net-check --image=radinshayanfar/alpine-c...
[mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl run -it net-check --image=radinshayanfar/alpine-curl:1.0 -- sh
If you don't see a command prompt, try pressing enter.
[/ # curl -r post shortener-service/shortener --form u=http://google.com ]
Warning: Invalid character is found in given range. A specified range MUST
Warning: have only digits in 'start'-'stop'. The server's response to thi
s
Warning: request is uncertain.
{"shortened":"aa223"}
[/ # curl -r get shortener-service/aa223
Warning: Invalid character is found in given range. A specified range MUST
Warning: have only digits in 'start'-'stop'. The server's response to thi
s
Warning: request is uncertain.
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to target URL: <a href="http://
google.com">http://google.com</a>. If not click the link./ # ]
```

نکته: توجه شود که برای عمل URL Shortening در واقع URL داده شده کاربر را hash می کنیم و پنج کاراکتر اول hash را به عنوان URL کوتاه شده استفاده می کنیم و در مرحله‌ی بعدی با استفاده از URL کوتاه شده درخواست می دهیم و مشاهده می کنیم که درخواست موفقیت آمیز بوده است.



پیاده سازی docker compose جهت خودکار سازی ایجاد منابع و وابستگی های موردنیاز پروژه و نهایتا build و اجرای آن

(۱) محتویات و توضیح مختصر docker compose پیاده سازی شده:

تمامی اپلیکیشن هایی را که dockerize می کنیم و برای آنها network، storage و ... در نظر می گیریم، تمامی این موارد از طریق docker cli اتفاق افتاده اند، ما در واقع بجای اینکه برای هر بار دیپللوی کردن اپلیکیشن موردنظر و اجرا کردن آن یک سری کامند داکر بزنیم، در واقع از docker compose استفاده می کنیم.

خط اول در واقع نشان دهنده ورژنی است که با آن docker compose را می خواهیم بنویسیم، این ورژن برای این است که اگر آپدیت شد و docker compose خود را با ورژن های جدیدتر اجرا می کنیم، docker compose بداند که با چه استانداردی این فایل را نوشته ایم.

در بخش services تک تک سرویس هایی که داریم را تعریف می کنیم، در اینجا یک اپلیکیشن app و یک redis داریم.

حال توصیف app به این صورت است که در واقع app از ایمیجی که از dockerfile ساخته شده است استفاده می کند.

حال در اینجا volume تعریف کرده ایم و نیز portmap نیز انجام داده ایم که با استفاده از ports این کار را انجام داده ایم ، که در واقع ports یک آرایه از ما می گیرد که در واقع پورت هایی هستند که باید پابلیش شوند به هاست که در واقع اینجا پورت 8080 را به پورت 8080 داخلی کانتینر پابلیش کرده ایم.

حال برای redis از ایمیج redis:6.2.6-alpine استفاده کرده ایم و همچنین با استفاده از verysecretpassword command پسورد تعیین کردیم و برای volume redis نیز تعیین کردیم.



حال با استفاده از دستور `docker-compose up` اپلیکیشن‌های خود را بالا می‌آوریم:

```
+| sudo docker-compose up      radin@Radin-Laptop:~/cc/project x   root@vps1637187394:~/cc x   radin@Radin-Laptop:~/cc/project x
(venv) radin@Radin-Laptop:~/cc/project> sudo docker-compose up
Recreating project_redis_1 ... done
Starting project_app_1 ... done
Attaching to project_app_1, project_redis_1
redis_1  | 1:C 19 Jan 2022 15:42:16.046 # 000000000000 Redis is starting o000o00000000
redis_1  | 1:C 19 Jan 2022 15:42:16.046 # Redis version=6.2.6, bits=64, commit=00000000, modified=0, pid=1, just started
redis_1  | 1:C 19 Jan 2022 15:42:16.046 # Configuration loaded
redis_1  | 1:M 19 Jan 2022 15:42:16.046 * monotonic clock: POSIX clock_gettime
redis_1  | 1:M 19 Jan 2022 15:42:16.047 * Running mode=standalone, port=6379.
redis_1  | 1:M 19 Jan 2022 15:42:16.047 # Server initialized
redis_1  | 1:M 19 Jan 2022 15:42:16.047 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory' = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
redis_1  | 1:M 19 Jan 2022 15:42:16.047 * Loading RDB produced by version 6.2.6
redis_1  | 1:M 19 Jan 2022 15:42:16.047 * RDB age 13 seconds
redis_1  | 1:M 19 Jan 2022 15:42:16.047 * RDB memory usage when created 0.79 Mb
redis_1  | 1:M 19 Jan 2022 15:42:16.047 # Done loading RDB, keys loaded: 0, keys expired: 0.
redis_1  | 1:M 19 Jan 2022 15:42:16.047 * DB loaded from disk: 0.000 seconds
redis_1  | 1:M 19 Jan 2022 15:42:16.047 * Ready to accept connections
app_1    | * Serving Flask app 'app' (lazy loading)
app_1    | * Environment: production
app_1    |     WARNING: This is a development server. Do not use it in a production deployment.
app_1    |     Use a production WSGI server instead.
app_1    | * Debug mode: off
app_1    | * Running on all addresses.
app_1    |     WARNING: This is a development server. Do not use it in a production deployment.
app_1    |     * Running on http://172.19.0.2:8080/ (Press CTRL+C to quit)
app_1    | 172.19.0.1 - - [19/Jan/2022 15:42:23] "POST /shortener HTTP/1.1" 200 -
app_1    | 172.19.0.1 - - [19/Jan/2022 15:42:45] "GET /d0e19 HTTP/1.1" 302 -
```



اجرای دیتابیس خود با استفاده از توصیف **stateful set** و جایگزین کردن آن با **deployment**

۱) دلیل استفاده از **stateful set** بجای **deployment**

در واقع **StatefulSet** یک کامپوننت در کوبرنیتیز می‌باشد که برای اپلیکیشن‌های **stateful** استفاده می‌شود، اپلیکیشن‌های **stateful**، اپلیکیشن‌هایی هستند که دیتا در خود ذخیره می‌کنند تا بتوانند حالت خود را ذخیره کنند و بتوانند حالت خود را دنبال کنند(**trace**) در مقابل اپلیکیشن‌های **stateless** هیچ حالتی از خود را ذخیره نمی‌کنند (**do not keep record of state of previous interactions**) و هر ریکوست یا **interaction** به صورت کاملاً **isolated** هندل می‌شود و با استفاده از اطلاعات خود درخواست فعلی هندل می‌شود.

به دلیل تفاوت این دو نوع اپلیکیشن‌ها، دیپلوی کردن این اپلیکیشن‌ها نیز با هم متفاوت است. اپلیکیشن‌های **stateless** توسط **deployment** دیپلوی می‌شوند و اپلیکیشن‌های **stateful** توسط **statefulset** دیپلوی می‌شوند.

در **deployment** پادها بصورت رندوم تولید می‌شوند و در انتهای آن‌ها **random hashes** وجود دارد و همچنین **identical** و **interchangeable** **load balancing** را برای هر پاد و برای هر ریکوست انجام می‌دهد و توجه شود که در واقع **scaling** بصورت رندوم بین پادها انجام می‌شود زیرا پادها **identical** هستند.



Deployment vs StatefulSet

The diagram illustrates a deployment setup. Three identical blue hexagonal icons, each representing a pod and labeled "my-app", are shown. They are all connected to a single central icon labeled "SVC" which contains a tree-like structure representing a service. This visualizes how a single service can load balance traffic to multiple identical pods.

- ▶ identical and interchangeable
- ▶ created in random order with random hashes
- ▶ one Service that load balances to any Pod

[SUBSCRIBE](#)

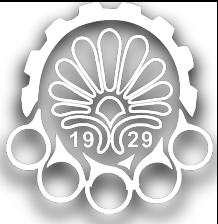
در `statefulset` نمی‌توان بصورت همزمان و به هر ترتیب دلخواهی پادها را `delete` یا `create` کرد و نمی‌توان پادها را بصورت رندوم آدرس‌دهی کرد و این امر به این دلیل است که پادها مختص نیستند، در واقع هر کدام `Identity` خود را دارند (این امر باعث تفاوت بین `statefulset` و `deployment` شده است).

Deployment vs StatefulSet

The diagram illustrates a deployment setup for MySQL. Three identical blue hexagonal icons, each representing a pod and labeled "mysql", are shown. They are all connected to a single central icon representing a service. A callout bubble next to the service icon says "more difficult".

- ▶ can't be created/deleted at same time
- ▶ can't be randomly addressed
- ▶ replica Pods are not identical
- Pod Identity

[SUBSCRIBE](#)



Pod Identity

- sticky identity for each pod

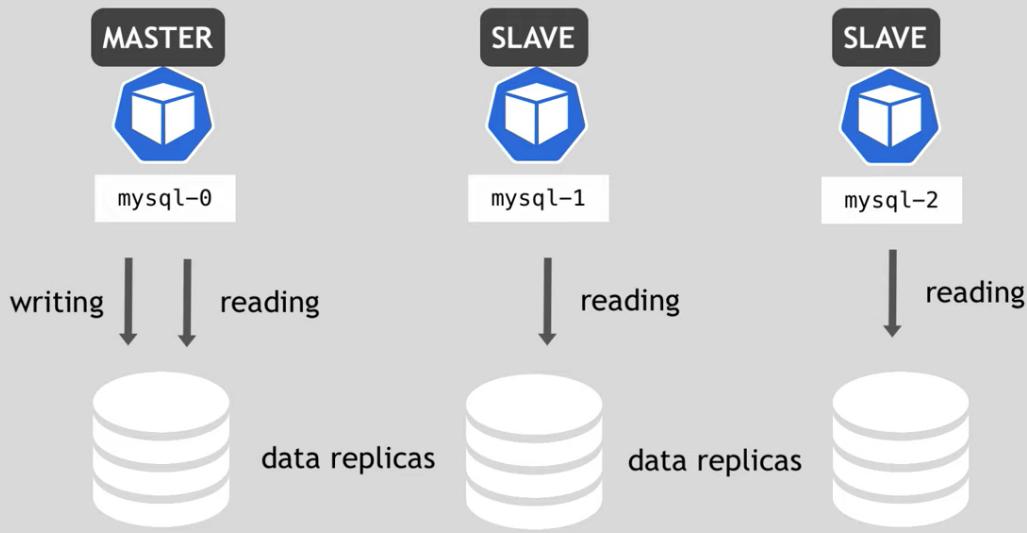


- created from **same specification**, but **not interchangeable!**

- persistent identifier across any re-scheduling

SUBSCRIBE

Scaling database applications



SUBSCRIBE

در این حالت در واقع یک پاد نقش master را دارد و پادهای دیگر همگی slave هستند و هم می‌تواند بخواند و هم می‌تواند بنویسد اما دیگر پادها که slave هستند فقط می‌توانند بخوانند زیرا در غیر اینصورت data inconsistency پیش می‌آید.

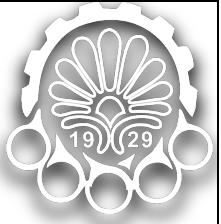


مطابق شکل قبل هر زمانی که master دیتای جدیدی write می‌کند، پادهای slave باید حواسشان باشد که دیتای خود را با master همگام کنند و synchronized باشند، همچنین توجه شود که این کار باعث می‌شود که دیگر دغدغه‌ی همگام بودن داده‌ها را نداشته باشیم (زیرا خودش این را هندل می‌کند)، همچنین توجه شود که هر پاد volume مجزای خود را داراست درصورتی که همگی از یک تعریف statefulset درست شده‌اند اما در deployment صرفاً یک volume برای تمامی پادها وجود دارد.

۲) توصیف مورد استفاده برای ساخت Statefulset

توصیف مورد استفاده برای ساخت Statefulset :

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  creationTimestamp: null
  labels:
    app: redis
    name: redis
spec:
  serviceName: redis
  replicas: 3
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: redis
    spec:
      initContainers:
        - name: init-redis
          image: redis:6.2.6-bullseye
          env:
            - name: REDIS_PASSWORD
              valueFrom:
                secretKeyRef:
                  key: REDIS_PASSWORD
                  name: redis-secret
          command:
            - bash
            - '-c'
            - |
              set -ex
              # Generate redis server-id from pod ordinal index
              [[ "hostname" == <{0-5}>${ } ]] || exit 1
              ordinal=$(SHLVL=1 bash -c "echo \$HOSTNAME | awk '{print int($$NF/2)}'")
              # Copy appropriate conf.d files from config-map to emptyDir
              if [[ $ordinal -eq 0 ]]; then
                cp /mnt/config-map/master.conf /etc/redis.conf
              else
                cp /mnt/config-map/slave.conf /etc/redis.conf
              fi
              echo "requirepass ${REDIS_PASSWORD}" >> /etc/redis.conf
              echo "masterauth ${REDIS_PASSWORD}" >> /etc/redis.conf
      volumeMounts:
        - name: conf
          mountPath: /etc/
          subPath: redis.conf
        - name: config-map
          mountPath: /mnt/config-map
  containers:
    - image: redis:6.2.6-alpine
      name: redis
      command: [ "redis-server" ]
      args: [ "/etc/redis.conf" ]
```



توصیف مورد استفاده برای ساخت Statefulset موردنیاز برای ConfigMap

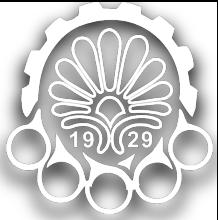
```
sts — vim cm.yaml — 80x32

apiVersion: v1
data:
  .env: |
    HOST=0.0.0.0
    PORT=8080
    URL_EX=60

  ■ REDIS_HOST=redis-0.redis
  ■ REDIS_PORT=6379
  ■ REDIS_DB=0
  master.conf: |
    bind 0.0.0.0
    protected-mode yes
    port 6379
    tcp-backlog 511
    timeout 0
    tcp-keepalive 300
    daemonize no
    supervised no
    pidfile /var/run/redis_6379.pid
    loglevel notice
    logfile ""
  slave.conf: |
    slaveof redis-0.redis 6379
kind: ConfigMap
metadata:
  creationTimestamp: null
  name: shortener-config
~
~
~
```



:Statefulset توصیف مورد استفاده برای ساخت Service موردنیاز برای



```
radin@Radin-Laptop:~/cc/project/k8s$ k apply -f redis-sts.yaml
nk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ k get pod
NAME READY STATUS RESTARTS AGE
net-check 1/1 Running 21 (3h19m ago) 38d
redis-0 1/1 Running 0 2s
nk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ k get pod
NAME READY STATUS RESTARTS AGE
net-check 1/1 Running 21 (3h20m ago) 38d
redis-0 1/1 Running 0 5s
redis-1 1/1 Running 0 6s
redis-2 0/1 Init:0/1 0 2s
nk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ k get pod
NAME READY STATUS RESTARTS AGE
net-check 1/1 Running 21 (3h20m ago) 38d
redis-0 1/1 Running 0 8s
redis-1 1/1 Running 0 11s
redis-2 1/1 Running 0 7s
nk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ k exec -it redis-0 -- redis-cli
Defaulted container "redis" out of: redis, init-redis (init)
127.0.0.1:6379> AUTH verysecretpassword
OK
127.0.0.1:6379> set name Mahan
OK
127.0.0.1:6379>
nk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ k exec -it redis-1 -- redis-cli
Defaulted container "redis" out of: redis, init-redis (init)
127.0.0.1:6379> AUTH verysecretpassword
OK
127.0.0.1:6379> get name
"\"Mahan\""
127.0.0.1:6379> 
```

حال مطابق شکل statefulset موردنظر را apply کرده‌ایم، سپس با استفاده از دستور kubectl get pods چک کرده‌ایم که پادها ساخته شده باشند، همانطور که مشاهده می‌کنیم پادها به درستی ساخته شده‌اند. پاد 0 نقش redis-0 و بقیه پادها نقش slave را دارند.

حال نوبت به آن رسیده است که data synchronization را چک کنیم و مطمئن شویم که statefulset موردنظر به درستی کار می‌کند. برای این کار به پاد 0 وصل شده‌ایم و مقدار redis-0 را در دیتابیس ذخیره کرده‌ایم، سپس به پاد 1 متصل شده‌ایم و مشاهده می‌کنیم که این مقدار در این پاد نیز وجود دارد و همگام سازی به درستی صورت گرفته است. حال دیگر کامپوننت‌های موردنیاز statefulset خود را apply می‌کنیم، سپس با استفاده از پاد net-URL Shortening check درخواست URL Shortening خود را می‌فرستیم و چک می‌کنیم که به درستی انجام شده باشد.



```
radin@Radin-Laptop:~/cc/project/k8s/sts
```

NAME	READY	STATUS	RESTARTS	AGE
net-check	1/1	Running	21 (3h19m ago)	30d
redis-0	1/1	Running	0	45s
redis-1	0/1	Init:0/1	0	2s

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ ls
```

```
cn.yaml redis-service.yaml redis-slave.yaml
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ vln cn.yaml
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ k exec -it redis-1 -- redis-cli
```

```
127.0.0.1:6379> AUTH verysecretpassword
```

```
127.0.0.1:6379> get aa223
```

```
(nil)
```

```
127.0.0.1:6379> get aa223
```

```
"http://google.com"
```

```
127.0.0.1:6379>
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$
```

حال مطابق شکل بعدی مشاهده می‌کنیم که مقدار URL کوتاه شده (پنج کاراکتر اول هش اصلی) در redis-1 slave یعنی نیز ذخیره شده است و این به این معناست که statefulset که موردنظر به درستی عمل می‌کند.

```
radin@Radin-Laptop:~/cc/project/k8s
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s
```

```
OK
```

```
127.0.0.1:6379> get name
```

```
"Mahan"
```

```
127.0.0.1:6379>
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ k apply -f cm.yaml
```

```
configmap/shortener-config configured
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s/sts$ cd ..
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s$ k apply -f shortener-deployment.yaml
```

```
deployment.apps/shortener created
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s$ k apply -f shortener-service.yaml
```

```
service/shortener-service created
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s$ k get pod
```

NAME	READY	STATUS	RESTARTS	AGE
net-check	1/1	Running	21 (3h23m ago)	38d
redis-0	1/1	Running	0	3m19s
redis-1	1/1	Running	0	3m17s
redis-2	1/1	Running	0	3m3s
shortener-8aff5fb797-qqsx	1/1	Running	0	10s
shortener-8aff5fb797-znsxf	1/1	Running	0	10s

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s$ k exec -it net-check -- sh
```

```
/ # curl -r post shortener-service/shortener --form u=http://google.com
```

```
Warning: Invalid character is found in given range. A specified range MUST
```

```
Warning: have only digits in 'start'-'stop'. The server's response to this
```

```
Warning: request is uncertain.
```

```
{"shortened": "aa223"}
```

```
/ #
```

```
/ #
```

```
/ # Google
```

```
shortener: not found
```

```
/ # http://google.com
```

```
shortener: not found
```

```
/ #
```

```
/ #
```

```
/ #
```

```
/ # Type a message
```

```
shortener: not found
```

```
/ #
```

```
command terminated with exit code 127
```

```
mk_user@Radin-Laptop:~/home/radin/cc/project/k8s$ k exec -it net-check -- sh
```

```
/ # curl -r get shortener-service/aa223
```

```
Warning: Invalid character is found in given range. A specified range MUST
```

```
Warning: have only digits in 'start'-'stop'. The server's response to this
```

```
Warning: request is uncertain.
```

```
{"shortened": "aa223"}
```

```
/ # curl -r get shortener-service/aa223
```

```
Warning: Invalid character is found in given range. A specified range MUST
```

```
Warning: have only digits in 'start'-'stop'. The server's response to this
```

```
Warning: request is uncertain.
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 3.2 Final//EN">
```

```
<title>Redirecting...</title>
```

```
<h1>Redirecting...</h1>
```

```
<p>You should be redirected automatically to target URL: <a href="http://google.com">http://google.com</a>. If not click the link.</p>
```



ساختن یک کامپونت HPA در کلاستر کوبرنتیز به منظور انجام عملیات scaling

۱) به منظور انجام autoscaling توسط کوبرنتیز، نیاز به مشخص کردن معیاری که به کمک آن بار پادها و نیاز به scale up/down مشخص می شود داریم. در api نسخه یک، تنها معیار بهرهوری CPU برای پادها وجود داشت. اما در نسخه های بتای فعلی، می توان از معیارهای متفاوتی برای این کار استفاده کرد. این معیارها عبارتند از:

۱. containerResource: این معیار، مربوط به یک منبع مشخص (مانند cpu و memory) است که کانتینر در هر یک از پادها است. این معیار به شکل in built در کوبرنتیز وجود دارد و قابل استفاده است.

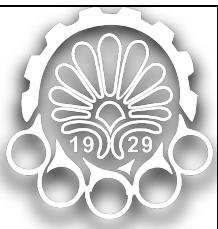
۲. object: این معیار مربوط به یک معیار سراسری است که به هیچ یک از object های کوبرنتیز مربوط نمی شود. به عنوان مثال، اندازه یک صفت در یک سرویس پیام رسانی ابری در خارج از کلاستر کوبرنتیز، یک معیار external است.

۳. hits-per-second: یک معیار مربوط به یک object در کلاستر کوبرنتیز (مثلا ingress) است.

۴. packets-per-second: این معیار، توصیف کننده هر یک از پادها در scale هدف فعلی (مثلا packets-per-second) است. در این معیار، مقادیر پیش از مقایسه با میزان هدف (حد آستانه مشخص شده برای scale کردن)، میانگین گرفته می شوند.

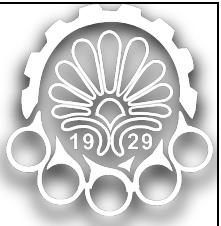
۵. resource: مانند containerResource کانتینر است. با این تفاوت که برای یک پاد (نه یک کانتینر) مشخص می شود.

در اینجا از معیار resource (و در حالت cpu) برای مقیاس کردن استفاده شده است. از بین موارد مطرح شده، مورد ۲، ۳ و ۴ به تنها یک کاربرد در دسترس نبودند. همچنین مورد ۱ و ۵ نیز، با توجه به اینکه در هر یک از پادها تنها یک کانتینر استفاده شده است، تفاوتی ندارد.



```
mahanahmadvand@Mahans-MacBook-Pro k8s % kubectl get hpa
NAME             REFERENCE           TARGETS          MINPODS   MAXPODS   REPLICAS   AGE
awesome-shortener-shortener   Deployment/awesome-shortener-shortener   <unknown>/50%   1          5          2          2d14h
shortener         Deployment/shortener        <unknown>/50%   1          5          2          2d19h
mahanahmadvand@Mahans-MacBook-Pro k8s %
```

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: shortener
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: shortener
  minReplicas: 1
  maxReplicas: 5
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
~
```



```
● ○ ● k8s — vim shortener-deployment.yaml — 87x45

apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: shortener
    name: shortener
spec:
  replicas: 2
  selector:
    matchLabels:
      app: shortener
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: shortener
    spec:
      containers:
        - image: radinshayanfar/ccp:1.1
          name: shortener
          resources:
            requests:
              cpu: "40m"
          ports:
            - containerPort: 8080
          env:
            - name: REDIS_SECRET
              valueFrom:
                secretKeyRef:
                  key: REDIS_PASSWORD
                  name: redis-secret
          volumeMounts:
            - mountPath: /env/.env
              subPath: .env
              name: config-map
              readOnly: true
        volumes:
          - name: config-map
            configMap:
              name: shortener-config
  status: {}

~
```

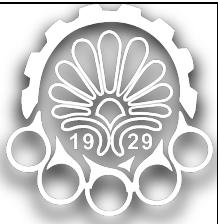


پیاده‌سازی **helm chart** جهت خودکارسازی ایجاد منابع

Helm، یک package manager برای کوبرنتیز است. در helm، یک چارت، در واقع یک پکیج است که در آن تمامی توصیف‌های لازم برای بالا آوردن یک برنامه و منابع مورد نیاز آن در کوبرنتیز، وجود دارد و با نصب آن چارت، این توصیف‌ها به کوبرنتیز ارسال شده و ایجاد می‌شوند. در شکل زیر، ساختار فایل‌های یک چارت آمده است.

```
wordpress/
  Chart.yaml      # A YAML file containing information about the chart
  LICENSE        # OPTIONAL: A plain text file containing the license for the chart
  README.md      # OPTIONAL: A human-readable README file
  values.yaml    # The default configuration values for this chart
  values.schema.json # OPTIONAL: A JSON Schema for imposing a structure on the values.yaml file
  charts/         # A directory containing any charts upon which this chart depends.
  crds/          # Custom Resource Definitions
  templates/      # A directory of templates that, when combined with values,
                  # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

در فایل Chart.yaml، اطلاعاتی کلی در مورد چارت (مانند نسخه و اطلاعات توسعه آن) وجود دارد. فایل values.yaml، برای مشخص کردن مقادیری است که به عنوان پارامتر قابل تغییر، در تولید توصیف منابع مورد نیاز استفاده می‌شود. در پوشه charts، پیش‌نیازهای چارت مورد نظر قرار می‌گیرند. در نهایت، مهم‌ترین بخش یک چارت، فایل‌های template آن هستند. این فایل‌ها، در واقع همان توصیفاتی است که به کوبرنتیز ارسال می‌شوند. با این تفاوت که به جای هارد کد کردن بخش‌های مختلف آن، به فرمت تمپلیت‌های Smarty، داخل {{ }} قرار می‌گیرند. به عنوان مثال، اگر در فایل values.yaml مقدار redisReplicas: 3 را داشته باشیم، در تمپلیت دیپلولویمنت ردیس، با نوشتن values.yaml replicas: {{ .Values.redisReplicas }} جایگزین می‌شود. همچنین این پارامترها در هنگام نصب چارت و با استفاده از آرگومان‌های CLI قابل مقداردهی هستند.



```
Cloud-Computing-Project — zsh — 80x24
[mahanahmadvand@Mahans-MacBook-Pro Cloud-Computing-Project % helm install awesome]
-shortener helm
NAME: awesome-shortener
LAST DEPLOYED: Sun Jan 23 23:15:53 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
mahanahmadvand@Mahans-MacBook-Pro Cloud-Computing-Project %
```

```
Cloud-Computing-Project — zsh — 116x24
[mahanahmadvand@Mahans-MacBook-Pro Cloud-Computing-Project % kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
awesome-shortener-redis-d455f8b64-k4mjp   1/1     Running   0          12m
awesome-shortener-shortener-765b599d7d-vlwk8   1/1     Running   0          12m
awesome-shortener-shortener-765b599d7d-zss8g   1/1     Running   0          12m
net-check      1/1     Running   2 (88m ago) 2d7h
redis-65ddf594df-mdz76   1/1     Running   0          12m
```

باتشکر

پیروز و پایدار باشید

رادین شایانفر

ماهان احمدوند