گام اول)

پروژه با فریم‌ورک Node.js توسعه داده شده است.

فایل‌های مربوط به کد سرور که در پوشه src قرار دارد:

index.js: فایل اصلی سرور

db.js: اتصال به پایگاه داده MongoDB

model.js: توصیف Scheme پایگاه داده:

```
const URLSchema = new mongoose.Schema({
    longUrl: String,
    urlCode: String,
    creationDate: String,
    expirationDate: String
})
```

shorten.js: کوتاه‌سازی لینک (متد POST)

redirect.js: بازگردانی لینک اصلی از روی لینک کوتاه‌شده (متد GET)

config.js: فایل مربوط به تنظیمات که پارامترهای آن از روی environment variable ها مقدارهی می‌شود.:

src > JS config.js > ...

```js
var config = {}

config.port = process.env.PORT
config.expire = process.env.EXPIRE
config.db_uri = process.env.DB
config.db_user = process.env.DB_USER
config.db_pass = process.env.DB_PASS

export default config
```

گام دوم)

برای استفاده از multistage build، دو مرحله‌ی زیر در نظر گرفته شده است:

۱- دانلود dependency های پروژه (پروژه Node.js نیازی به کامپایل یا build ندارد و دریافت dependency ها برای اجرای پروژه کافیست).

۲- کپی کردن dependency ها از مرحله قبل، انتقال source code ها و اجرای پروژه

```
Dockerfile > FROM
FROM node:alpine AS dependencies
WORKDIR /usr/src/app
COPY package.json ./
RUN npm install

FROM node:alpine AS run
WORKDIR /usr/src/app
COPY --from=dependencies /usr/src/app/node_modules ./node_modules
COPY package.json ./
COPY ./src ./src
EXPOSE 8080
CMD npm start
```

بیلد و ارسال image به داکرهاب:

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ docker build -t short .
```

```
Successfully tagged short:latest
```

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ docker tag short:latest hzaredar/short:latest
hossein@Hossein-Laptop:~/Desktop/Cloud$ docker push hzaredar/short:latest
The push refers to repository [docker.io/hzaredar/short]
```

hzaredar / **short**

This repository does not have a description

Last pushed: 2 minutes ago

Tags and Scans

VULNERABILITY SCANNING - **DISABLED**
Enable

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
latest		2 minutes ago	2 minutes ago

گام سوم)

فایل‌های YAML نوشته شده:

- mongo_deploy.yaml: توصیف deployment پایگاه داده
- mongo_service.yaml: توصیف سرویس تعریف شده بر روی پایگاه داده
- mongo_secret.yaml: تنظیم نام کاربری و گذرواژه مورد نیاز پایگاه داده

- persist_vol.yaml: توصیف persisent volume
- Persist_claim.yaml: توصیف درخواست یک volume که از آن برای ذخیره دائمی داده‌های پایگاه داده استفاده می‌شود

- short_deploy.yaml: توصیف deployment سرور کوتاه‌کننده لینک
- short_service.yaml: توصیف سرویسی که بر روی pod های سرور قرار می‌گیرد
- short_config.yaml: تنظیمات مربوط به پورت سرور، زمان انقضای لینک‌ها و آدرس سرور پایگاه داده

برای سادگی کار، فایل start.sh نوشته شده است که با اجرای آن پادهای قبلی حذف شده پادهای جدید اجرا می‌شوند. فایل create_vol.sh هم برای اجرای volume می‌باشد.

تعداد پادهای پایگاه داده:

تعداد Pod های در نظر گرفته‌شده برای پایگاه داده ۱ می‌باشد. می‌توان با تنظیم StatefulSet این تعداد را بیشتر از ۱ نیز در نظر گرفت و در آن صورت، داده‌های قرار گرفته بر روی این پادها با یکدیگر Synch خواهد شد. با این حال، در جهت استفاده کمتر از حافظه، از ایجاد redundacy صرف نظر شده و فقط ۱ پاد را اختصاص داده‌ایم.

Deployment ها:

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ kubectl get deployments
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
mongo    1/1     1            1           5m52s
short    2/2     2            2           5m49s
```

Pods ها:

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
mongo-d5ccd4ddd-lnlrn     1/1     Running   0          6m20s
short-6c7bdc89c8-mkjpf    1/1     Running   0          6m17s
short-6c7bdc89c8-nnln5    1/1     Running   0          6m17s
```

Secret ها:

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ kubectl get secret
NAME                  TYPE                                  DATA   AGE
default-token-s94w7   kubernetes.io/service-account-token   3      47d
mongo-chart-token-9wbln   kubernetes.io/service-account-token   3      92s
mongo-secret          kubernetes.io/basic-auth              2      6m53s
```

ConfigMap ها:

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ kubectl get cm
NAME                 DATA   AGE
kube-root-ca.crt     1      47d
short-config         3      8m21s
```

:Persistent Volume

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ kubectl get pvc
NAME              STATUS   VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mongo-vol-claim   Bound    mongo-vol   250Mi      RWO            standard       9h
```

ارتباط میان Pod ها و سرویس‌ها:

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ kubectl describe service/short-service
Name:                     short-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 appli=short
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.106.41.225
IPs:                      10.106.41.225
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30001/TCP
Endpoints:                172.17.0.6:8080,172.17.0.7:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

همانطور که مشاهده می‌کنیم، پادهای مربوط به سرور در آدرس‌های 172.17.0.6 و 172.17.0.7 در شبکه داخلی کلاستر قرار دارند و از پورت 8080 درخواست می‌پذیرند. سرویس در آدرس 10.106.41.255 قرار دارد و از طریق پورت 8080 درخواست‌ها را دریافت کرده به پادها منتقل می‌کند.

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ kubectl describe service/mongo-service
Name:                     mongo-service
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 appli=mongo
Type:                     ClusterIP
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.104.127.129
IPs:                      10.104.127.129
Port:                     <unset>  27017/TCP
TargetPort:               27017/TCP
Endpoints:                172.17.0.3:27017
Session Affinity:         None
Events:                   <none>
```

ارتباط میان پاد مونگو و سرویس متناظرش به شکل بالا می‌باشد.

اجرا:

ابتدا با استفاده از port forwarding، راهی برای ایجاد ارتباط به داخل کلاستر ایجاد می‌کنیم:

```
hossein@Hossein-Laptop:~/Desktop/Cloud$ kubectl port-forward service/short-service 8080
Forwarding from 127.0.0.1:8080 -> 8080
```

حال با ارسال دستور POST می‌توانیم از سرور کوتاه‌کننده لینک استفاده کنیم:

```
hossein@Hossein-Laptop:~$ curl --header "Content-Type: application/json" --request
 POST --data '{"url":"https://www.google.com"}' 'localhost:8080/shorten/url'
{"longUrl":"https://www.google.com","shortUrl":"localhost:8080/Z4QKlKMRg2","creati
onDate":"Wed Jan 26 2022 18:55:20 GMT+0000 (Coordinated Universal Time)","expirati
onDate":"Wed Jan 26 2022 18:57:20 GMT+0000 (Coordinated Universal Time)"}hossein@H
ossein-Laptop:~$
```

زمان expire شدن ۲ دقیقه قرار داده شده است.

با دستور GET لینک اصلی را دریافت کنیم:

```
hossein@Hossein-Laptop:~$ curl "localhost:8080/Z4QKlKMRg2"
Found. Redirecting to https://www.google.comhossein@Hossein-Laptop:~$
```

اگر لینک expire شده باشد:

```
hossein@Hossein-Laptop:~$ curl "localhost:8080/Z4QKlKMRg2"
"URL not found"hossein@Hossein-Laptop:~$
```

امتیازی‌ها:

HPA:

از ۳ معیار مختلف می‌توان برای AutoScaling استفاده کرد:

۱- متوسط میزان استفاده از CPU در پادها

۲- متوسط میزان اشغال رم در پادها

۳- وضعیت شبکه (متوسط تعداد Packet یا Request های دریافتی در پادها)

در اینجا ما از معیار اول استفاده کردیم چون می‌تواند ملاک ساده‌ای باشد از اینکه پادها چقدر تحت load قرار دارند:

k8s > ! short_hpa.yaml

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: short-hpa
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: short
  minReplicas: 2
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

هر گاه متوسط CPU Utilization بیشتر از ۵۰ باشد، پاد جدید ساخته می‌شود. حداقل تعداد پاد برابر ۲ و حداکثر آن برابر ۱۰ در نظر گرفته شده است.

برای آزمودن HPA، از test_hpa.sh استفاده می‌کنیم که ۴۰۰۰ درخواست GET به سرور ارسال می‌کند:

نتیجه بدین شکل خواهد بود:

```
hossein@Hossein-Laptop:~$ kubectl get hpa/short-hpa --watch
NAME        REFERENCE          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
short-hpa   Deployment/short   1%/50%    2         10        2          57s
short-hpa   Deployment/short   6%/50%    2         10        2          60s
short-hpa   Deployment/short   52%/50%   2         10        2          75s
short-hpa   Deployment/short   96%/50%   2         10        2          90s
short-hpa   Deployment/short   83%/50%   2         10        4          106s
short-hpa   Deployment/short   50%/50%   2         10        4          2m1s
short-hpa   Deployment/short   43%/50%   2         10        4          2m16s
short-hpa   Deployment/short   7%/50%    2         10        4          2m31s
short-hpa   Deployment/short   1%/50%    2         10        4          2m46s
short-hpa   Deployment/short   2%/50%    2         10        4          3m1s
short-hpa   Deployment/short   1%/50%    2         10        4          3m16s
```

با افزایش درخواست‌ها، تعداد پاد از ۲ به ۴ افزایش می‌یابد.