بسم الله الرحمن الرحیم

سید امیرمهدی میرشریفی

۹۸۳۱۱۰۵

تکلیف دو درس رایانش ابری

دکتر جوادی

گام اول

در این بخش ابتدا می بایست یک داکر فایل نوشته شود و سپس build گرفته شود تا image مورد نظر به دست آید. در ابتدا داکر فایل زیر را آماده می کنیم:

Dockerfile
```
1   FROM ubuntu:22.04
2   RUN apt-get update && apt-get install -y \
3   curl
```

ابتدا از طریق دستور From ایمیج بیسی که میخواهید در image خود استفاده کنید می نویسید. این ایمیج میتواند سیستم عامل یا فریم ورکی مبتنی بر یک سیستم عامل خاص یا هر چیز دیگری که به عنوان بیس نیاز دارید باشد. در این جا از ایمیج ابونتو با ورژن مشخص شده استفاده میکنیم که در داکر هاب موجود است.
دستور RUN ، دستوراتی که هستند که موقع build اجرا میشوند. برای این که بتوانیم curl را که در ابتدا روی سیستم عامل موجود نیست را بدون نیاز به دخالت کاربر نصب کنیم ، میبایست این کامند هایی که خود کاربر نیاز است وارد کند را موقع ساختن ایمیج انجام دهیم. از این رو ابتدا apt را آپدیت میکنیم تا لیست پکیج ها و فایل های که مناسب نصب هستند آپلود شوند و سپس از طریق کامند apt-get ، کرل را دریافت میکنیم. پس از این که داکر فایل آماده شد آن را با دستور زیر میسازیم :

```
PS C:\Users\sam\Desktop\courses\term1401-B\cloud computing\HW\2\1> docker build .  -t ubuntu-curl
[+] Building 92.0s (7/7) FINISHED
 => [internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 124B                                                                0.0s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/ubuntu:22.04                                     1.9s
 => [1/3] FROM docker.io/library/ubuntu:22.04@sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a6d21   0.0s
 => CACHED [2/3] RUN apt update                                                                     0.0s
 => [3/3] RUN apt-get update && apt-get install -y curl                                            89.2s
 => exporting to image                                                                              0.7s
```

با استفاده از t - میتوانیم نام ایمیج را انتخاب کنیم که این نام برای ایمیج ساخته شده ubuntu-curl است.
اکنون این ایمیج آماده است که با استفاده از کامند زیر قابل ملاحظه است :

```
Docker images
```

زمان push کردن ایمیج باید به یک نکته توجه کنیم . از آنجایی که اسمی که برای ایمیج در نظر گرفتیم احتمالا با خیلی از اسامی که دیگران در نظر گرفته اند یکی است پس با یک استانداردی که عرف است و آن این است که ابتدای اسم یوزرنیم و سپس یک اسلش را قرار میدهیم و سپس آن را پوش میکنیم.
این کار را با استفاده از کامند زیر انجام میدهیم:

```
docker push sammirsharifi/ubuntu-curl:latest
```

که در نتیجه آن ایمیج در حساب کاربری داکرهاب قابل مشاهده است:

TAG

latest

Last pushed **an hour ago** by sammirsharifi

DIGEST	OS/ARCH	SCANNED	LAST PULL
9188dc517305	linux/amd64	---	an hour ago

اکنون این ایمیج با دستور زیر قابل دسترس است:

```
docker pull sammirsharifi/ubuntu-curl:latest
```

پس از اجرای این دستور و نصب این ایمیج میتوانید آن را در لیست ایمیج ها مشاهده کنید.

```
C:\Users\sam>docker images
REPOSITORY                 TAG       IMAGE ID       CREATED             SIZE
sammirsharifi/ubuntu-curl  latest    5cbafccbaa1a   About an hour ago   129MB
```

اکنون میشود به طریق زیر کانتینر مورد نظر را اجرا کنیم و همچنین از ابتدا کرل نصب است و دیگر نیاز به نصب نیست. اکنون با ارسال دستور کرل و به url مورد نظر نتیجه زیر را مشاهده کرد:

گام دوم

در این بخش هدف برآن است که یک سرور را به وجود بیاوریم که با دریافت یک لینک بلند ، کوتاه شده آن را برگرداند ، همچنین اگر قبلا لینک بلند دریافتی را کوتاه کرده بودیم ، می بایست استعلام آنرا از پایگاه داده ای که برای افزایش سرعت از حافظه اصلی استفاده میکند بگیریم و نتیجه آماده را برگردانیم.

بنا بر این ابتدا با استفاده از ایمیج ردیس شرایط را برای این پایگاه داده فراهم میکنیم.

در فایل اصلی یک پوشه است به نام redis که در آن تنظیمات مربوط به ساخت ایمیج ردیس قرار دارد:

```
redis > Dockerfile
1   FROM redis
2   COPY redis.conf /usr/local/etc/redis/redis.conf
3   CMD [ "redis-server", "/usr/local/etc/redis/redis.conf" ]
4   EXPOSE 6380
```

همچنین تنظیمات مربوط به آن در فایل کانفیگ قرار گرفته است.

در ادامه میبایست یک سرور بالا بیاوریم که خواسته هایی که در بالا گفتیم را برآورده کند . همچنین برای اتصال سرور با ردیس ، نیاز است که علاوه بر اضافه کردن کتابخانه مخصوص ردیس ، یک شبکه نیز بسازیم تا این دو کانتینر در آن بتوانند ارتباط داشته باشند. در ادامه به توضیح همه این موارد می پردازیم.

درابتدا به سراغ فایل server میرویم که درون آن فایل config قرار دارد. این فایل شامل تنظیم هایی است که کاربر میتواند به صورت دلخواه آن را تغییر دهد که این تنظیمات آدرس سروری است که برای کوتاه کردن url از آن استفاده میکنیم و همچنین مدت زمان اعتبار در ردیس :

```
server > config > development.yaml
1  api:
2    url: https://api.apilayer.com/short_url/hash
3  redis:
4    EX: 300
```

در ادامه در پوشه src یک پوشه به نام redis است که در واقع تنظیمات شی ردیسی است که میخواهیم از آن در سرورمان استفاده کنیم:

```
const { createClient } = require('redis')

const {
  REDIS_SERVICE_HOST: host,
  REDIS_SERVICE_PORT: port,
  REDIS_PASSWORD: password,
} = process.env
const redisOptions = {
  url: `redis://:${password}@${host}:${port}`
}
const client = createClient(redisOptions)

client.on('error', () => {})
client.on('connect', (e) => {
  console.log('Redis connection established.')
  module.exports.connected = true
})

client.connect().catch(console.log)
Object.assign(module.exports, { client })
```

همانطور که مشاهده میکنید اطلاعاتی از قبیل هاست ردیس ، شماره پورت و رمز عبور از طریق متغیر های محیطی که تعریف شده است به شی پاس داده میشود و یک شی از این پایگاه داده ساخته میشود که کلید ارتباط با سرور ردیس است .

در ادامه در فایل index.js تنظیمات سرور قرار دارد که قابل مشاهده است . همچین داکر فایل ایمیج سرور نیز به این صورت است:

server > Dockerfile

```
1
2   FROM node:alpine3.16
3   WORKDIR /app
4   COPY package*.json ./
5   RUN npm install --registry=https://npm.iranrepo.ir/
6   COPY . .
7   CMD [ "npm", "start" ]
8   EXPOSE 80
```

در آخر از طریق یک فایل شل اسکریپت دستورات لازم برای بیلد کردن ایمیج ها و تشکیل شبکه و ران کردن کانتینتر ها را ست میکنیم:

```
BASEDIR="$( cd -- "$(dirname "$0")" >/dev/null 2>&1 ; pwd -P )"
REDIS_CONTAINER_NAME="node-redis"
SERVER_CONTAINER_NAME="node-url-shortener"

cd "${BASEDIR}"/redis || exit
docker build . -t sam/redis

cd "${BASEDIR}"/server || exit
docker build . -t sam/node-url-shortener

if ! (docker network ls | grep node-url-shortener); then
  docker network create --subnet=172.20.0.0/16 node-url-shortener
fi

if (docker container ls -a -f NAME="$SERVER_CONTAINER_NAME" | grep "$SERVER_CONTAINER_NAME"); then
  docker container stop "$SERVER_CONTAINER_NAME"
  docker container rm "$SERVER_CONTAINER_NAME"
fi

if (docker container ls -a -f NAME="$REDIS_CONTAINER_NAME" | grep "$REDIS_CONTAINER_NAME"); then
  docker container stop "$REDIS_CONTAINER_NAME"
  docker container rm "$REDIS_CONTAINER_NAME"
fi

docker run -it --name "$REDIS_CONTAINER_NAME" --net node-url-shortener --ip 172.20.0.2 \
  -v "${BASEDIR}"/redis/data:/data -d sam/redis
docker run -it --name "$SERVER_CONTAINER_NAME" --net node-url-shortener --ip 172.20.0.3 \
  -d -p 3000:80 sam/node-url-shortener
```

در همین فایل است که با استفاده از دستور v- میتوان با استفاده از تکنیک volume اطلاعات ردیس را ذخیره کرد تا درصورت پایین آمدن کانتینتر ردیس ، اطلاعات از دست نرود.

همچنین با استفاده از کد زیر ایمیج سرور را روی داکر هاب آپلود خواهیم کرد:

push sammirsharifi/node-url-shortener

بخش سوم

این بخش با توجه به تحریم ها نا تمام ماند و صرفا فایل های یمل تهیه شد که این مسئله با یکی از تدریسیار محترم در میان گذاشته شد. فایل ها همراه بقیه فایل ها آپلود شده اند.