بسمه تعالی

تمرین دوم مبانی رایانش ابری

علیرضا زارع زین آبادی 9931022

ترم پاییز 1402

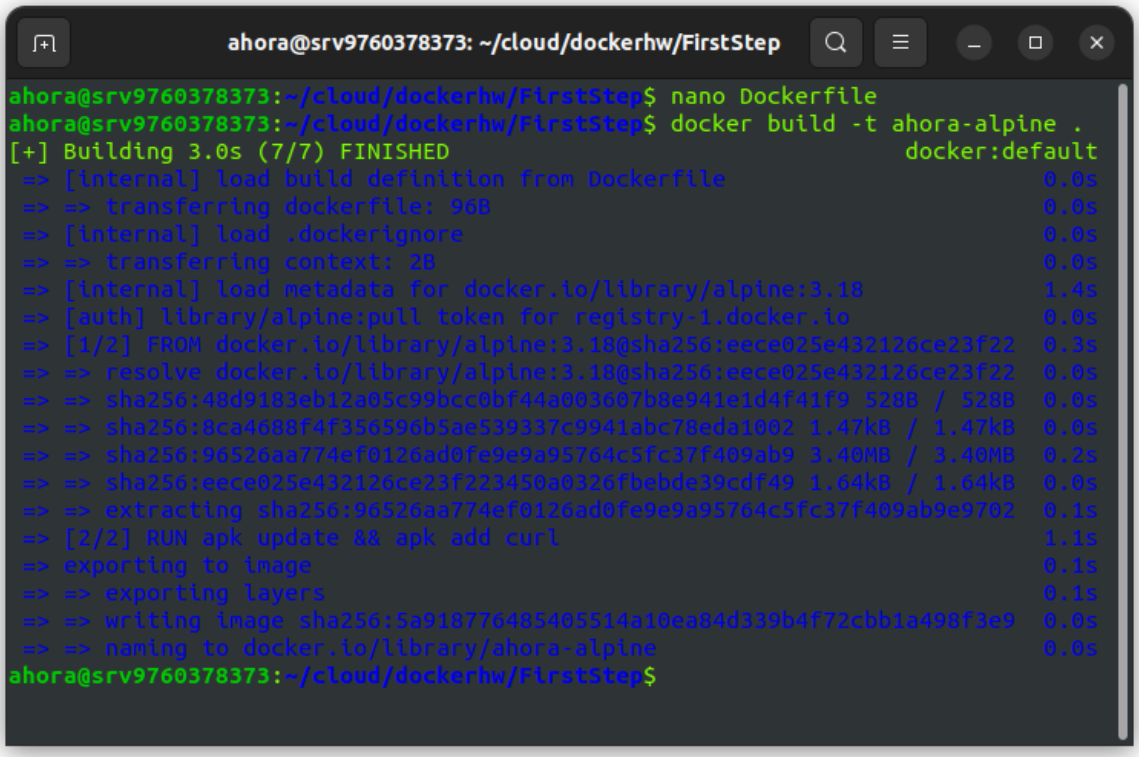
قسمت اول تمرین (داکر):

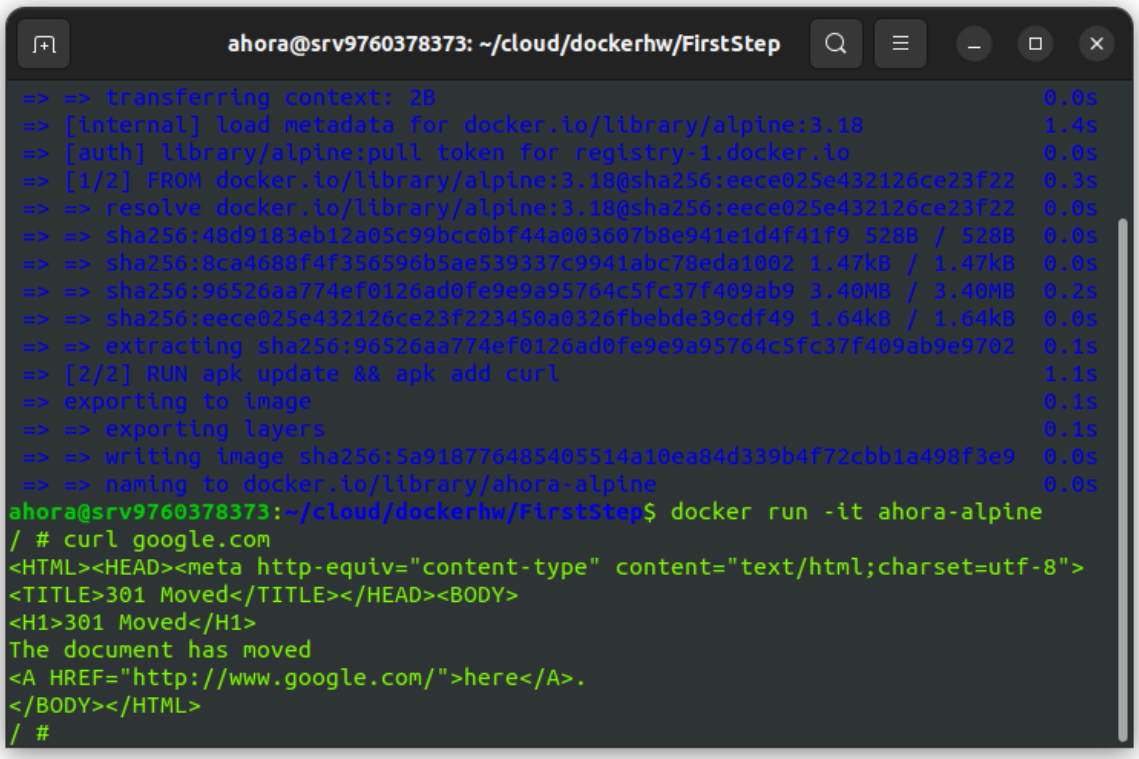
بخش اول:

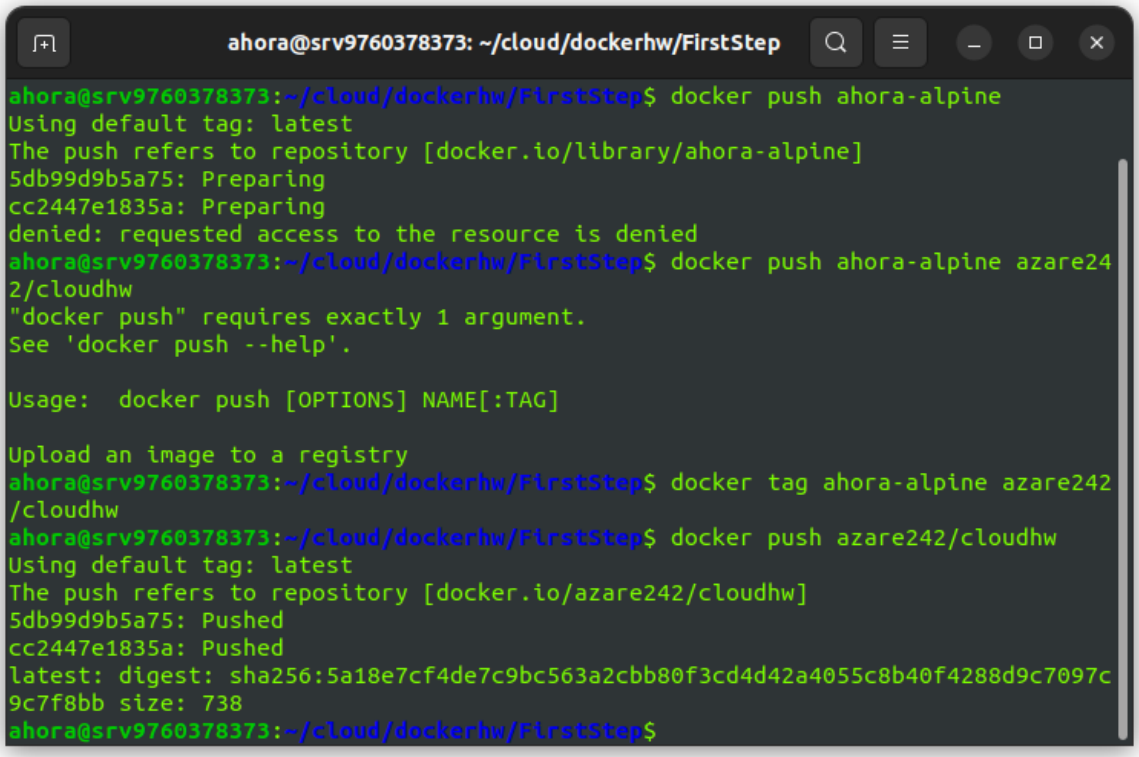
Alpine + cURL

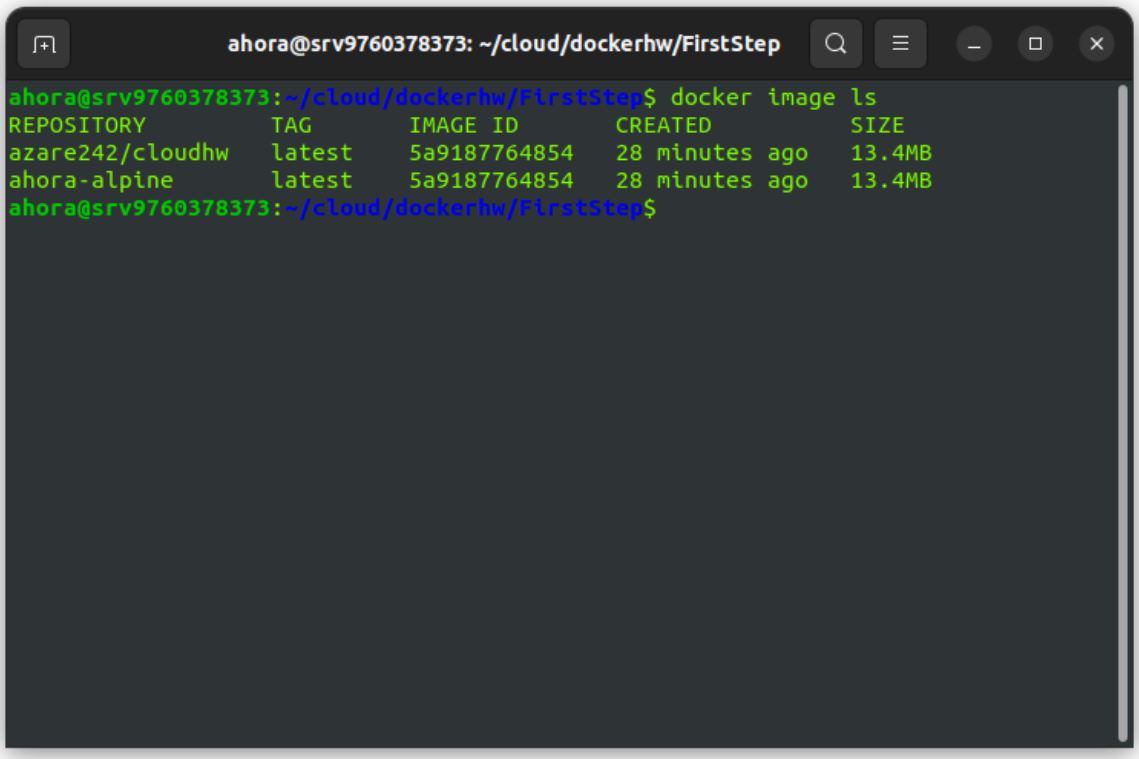
با بیلد کردن این داکر فایل به یک ایمیج الپاین دارای cURL میرسیم.

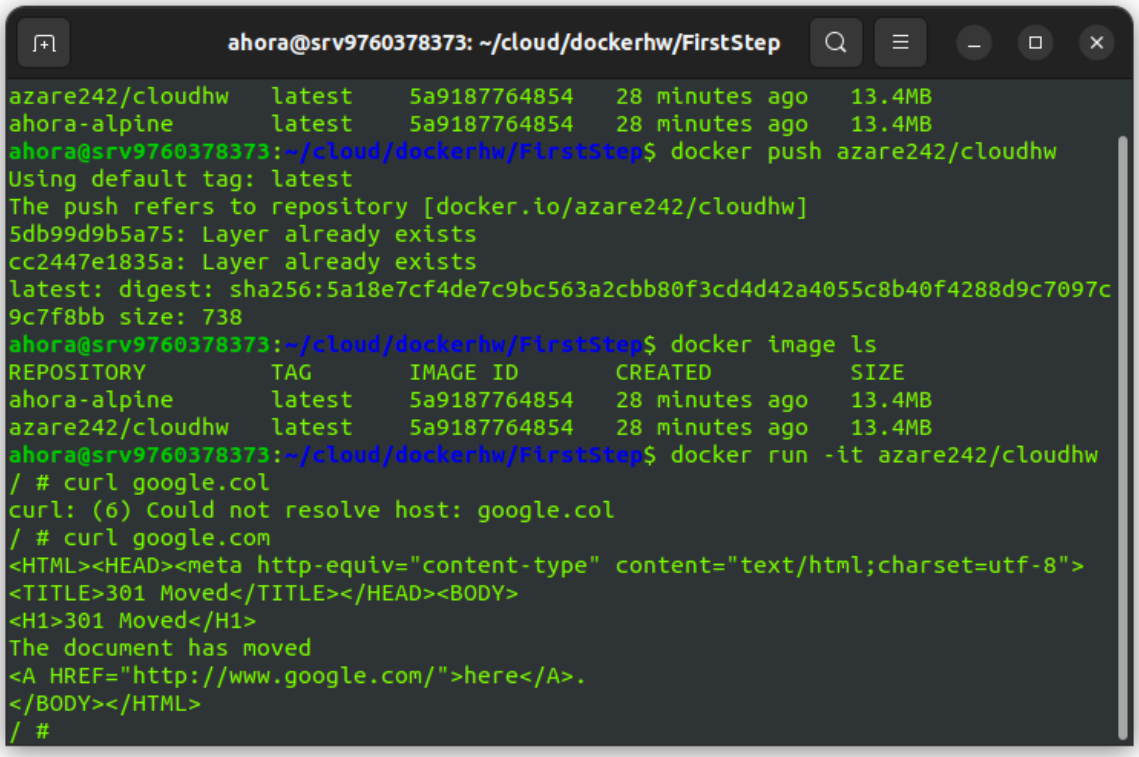
اسکرین شات های مراحل ساخت؛ تست؛ و قراردهی روی داکرهاب از صفحه بعد.

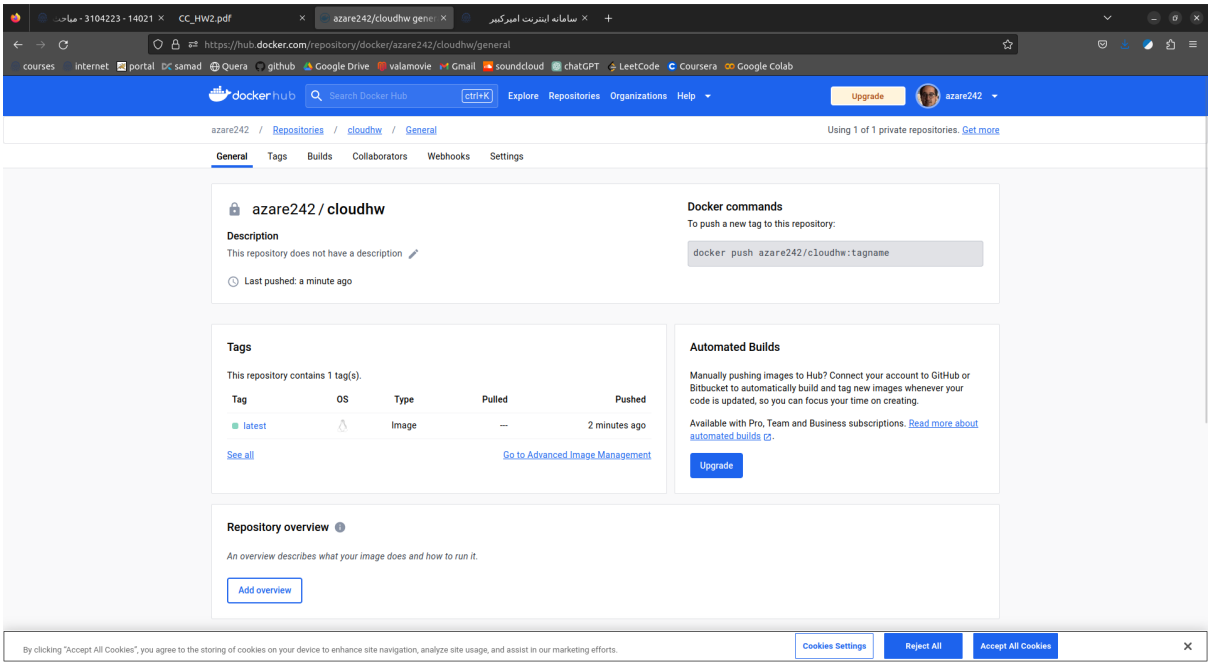

ahora@srv9760378373: ~/cloud/dockerhw/FirstStep
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ nano Dockerfile
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker build -t ahora-alpine .
[+] Building 3.0s (7/7) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 96B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/alpine:3.18 1.4s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> [1/2] FROM docker.io/library/alpine:3.18@sha256:eece025e432126ce23f22 0.3s
=> => resolve docker.io/library/alpine:3.18@sha256:eece025e432126ce23f22 0.0s
=> => sha256:48d9183eb12a05c99bcc0bf44a003607b8e941e1d4f41f9 528B / 528B 0.0s
=> => sha256:8ca4688f4f356596b5ae539337c9941abc78eda1002 1.47kB / 1.47kB 0.0s
=> => sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9 3.40MB / 3.40MB 0.2s
=> => sha256:eece025e432126ce23f223450a0326fbebde39cdf49 1.64kB / 1.64kB 0.0s
=> => extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e9702 0.1s
=> [2/2] RUN apk update && apk add curl 1.1s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:5a918776485405514a10ea84d339b4f72cbb1a498f3e9 0.0s
=> => naming to docker.io/library/ahora-alpine 0.0s
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$


ahora@srv9760378373: ~/cloud/dockerhw/FirstStep
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/alpine:3.18 1.4s
=> [auth] library/alpine:pull token for registry-1.docker.io 0.0s
=> [1/2] FROM docker.io/library/alpine:3.18@sha256:eece025e432126ce23f22 0.3s
=> => resolve docker.io/library/alpine:3.18@sha256:eece025e432126ce23f22 0.0s
=> => sha256:48d9183eb12a05c99bcc0bf44a003607b8e941e1d4f41f9 528B / 528B 0.0s
=> => sha256:8ca4688f4f356596b5ae539337c9941abc78eda1002 1.47kB / 1.47kB 0.0s
=> => sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9 3.40MB / 3.40MB 0.2s
=> => sha256:eece025e432126ce23f223450a0326fbebde39cdf49 1.64kB / 1.64kB 0.0s
=> => extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e9702 0.1s
=> [2/2] RUN apk update && apk add curl 1.1s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:5a918776485405514a10ea84d339b4f72cbb1a498f3e9 0.0s
=> => naming to docker.io/library/ahora-alpine 0.0s
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker run -it ahora-alpine
/ # curl google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
/ #


ahora@srv9760378373: ~/cloud/dockerhw/FirstStep
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker push ahora-alpine
Using default tag: latest
The push refers to repository [docker.io/library/ahora-alpine]
5db99d9b5a75: Preparing
cc2447e1835a: Preparing
denied: requested access to the resource is denied
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker push ahora-alpine azare24
2/cloudhw
"docker push" requires exactly 1 argument.
See 'docker push --help'.
Usage: docker push [OPTIONS] NAME[:TAG]
Upload an image to a registry
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker tag ahora-alpine azare242
/cloudhw
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker push azare242/cloudhw
Using default tag: latest
The push refers to repository [docker.io/azare242/cloudhw]
5db99d9b5a75: Pushed
cc2447e1835a: Pushed
latest: digest: sha256:5a18e7cf4de7c9bc563a2cbb80f3cd4d42a4055c8b40f4288d9c7097c
9c7f8bb size: 738
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$


ahora@srv9760378373: ~/cloud/dockerhw/FirstStep
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
azare242/cloudhw latest 5a9187764854 28 minutes ago 13.4MB
ahora-alpine latest 5a9187764854 28 minutes ago 13.4MB
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$


ahora@srv9760378373: ~/cloud/dockerhw/FirstStep
azare242/cloudhw latest 5a9187764854 28 minutes ago 13.4MB
ahora-alpine latest 5a9187764854 28 minutes ago 13.4MB
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker push azare242/cloudhw
Using default tag: latest
The push refers to repository [docker.io/azare242/cloudhw]
5db99d9b5a75: Layer already exists
cc2447e1835a: Layer already exists
latest: digest: sha256:5a18e7cf4de7c9bc563a2cbb80f3cd4d42a4055c8b40f4288d9c7097c
9c7f8bb size: 738
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
ahora-alpine latest 5a9187764854 28 minutes ago 13.4MB
azare242/cloudhw latest 5a9187764854 28 minutes ago 13.4MB
ahora@srv9760378373:~/cloud/dockerhw/FirstStep$ docker run -it azare242/cloudhw
/ # curl google.col
curl: (6) Could not resolve host: google.col
/ # curl google.com
<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>301 Moved</TITLE></HEAD><BODY>
<H1>301 Moved</H1>
The document has moved
<A HREF="http://www.google.com/">here</A>.
</BODY></HTML>
/ #

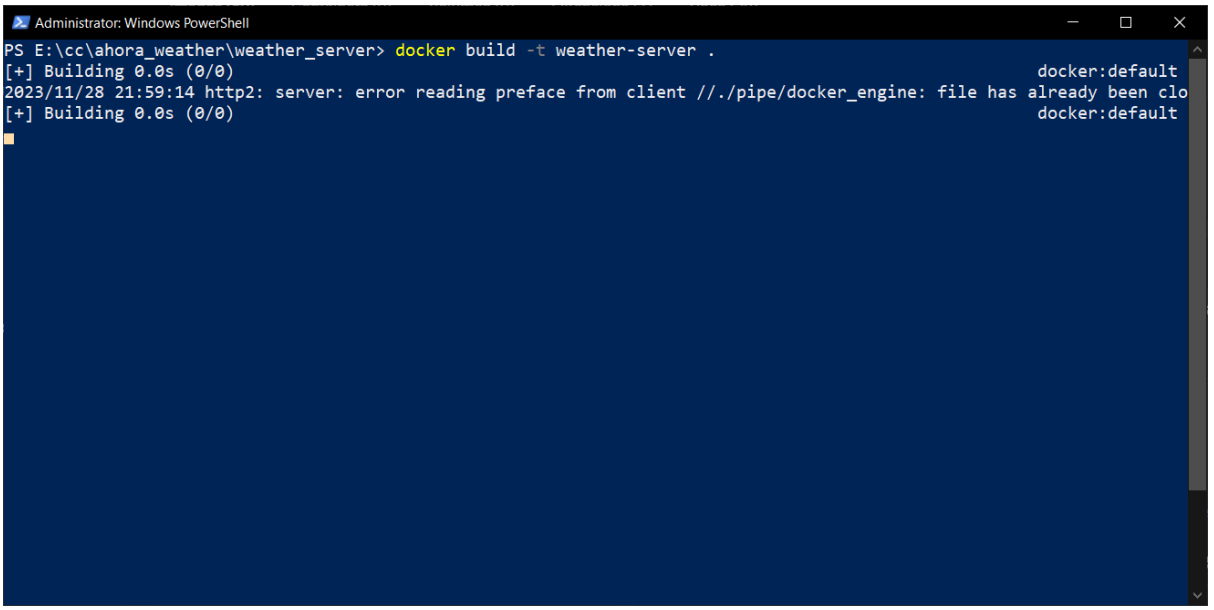

https://hub.docker.com/repository/docker/azare242/cloudhw/general
docker hub
Search Docker Hub
Explore
Repositories
Organizations
Help
Upgrade
azare242
azare242 / Repositories / cloudhw / General
Using 1 of 1 private repositories. Get more
General
Tags
Builds
Collaborators
Webhooks
Settings
azare242 / cloudhw
Description
This repository does not have a description
Last pushed: a minute ago
Docker commands
To push a new tag to this repository:
docker push azare242/cloudhw:tagname
Tags
This repository contains 1 tag(s).
Tag
OS
Type
Pulled
Pushed
latest
Image
---
2 minutes ago
See all
Go to Advanced Image Management
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. Read more about automated builds.
Upgrade
Repository overview
An overview describes what your image does and how to run it.
Add overview
By clicking "Accept All Cookies", you agree to the storing of cookies on your device to enhance site navigation, analyze site usage, and assist in our marketing efforts.
Cookies Settings
Reject All
Accept All Cookies

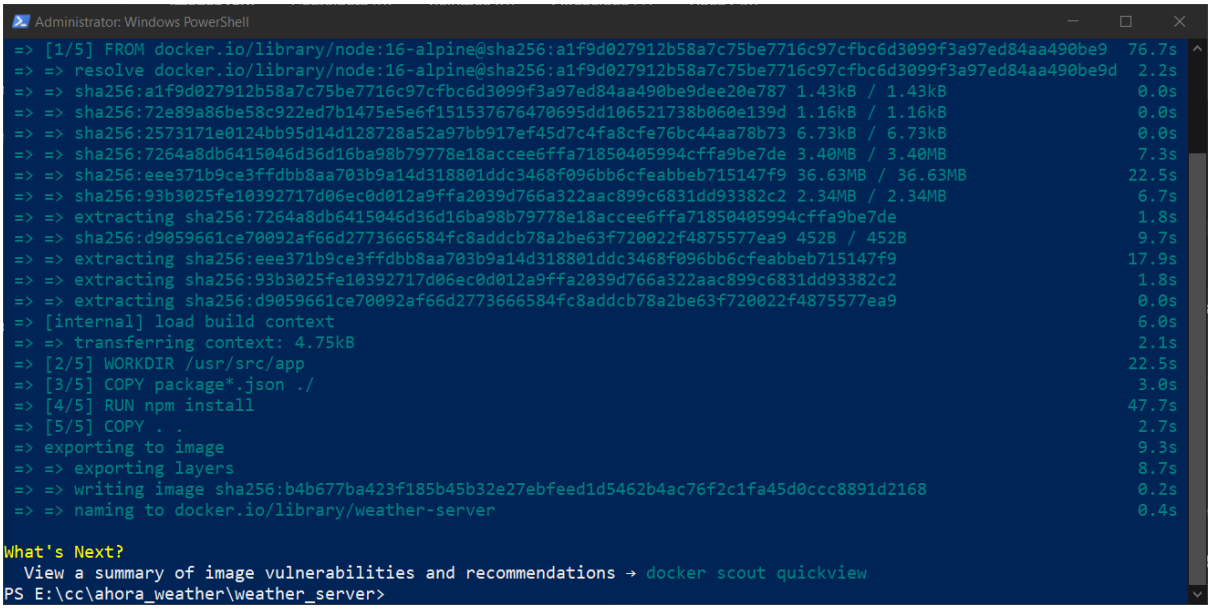
بخش دوم: سرور هواشناسی.
این سرور با استفاده از پکیج Express و محیط Nodejs با زبان جاوااسکریپت توسعه داده شد.
/api/v1/weather/<CITY NAME>
اندپوینتی است که اطلاعات آب و هوا را برمیگرداند.

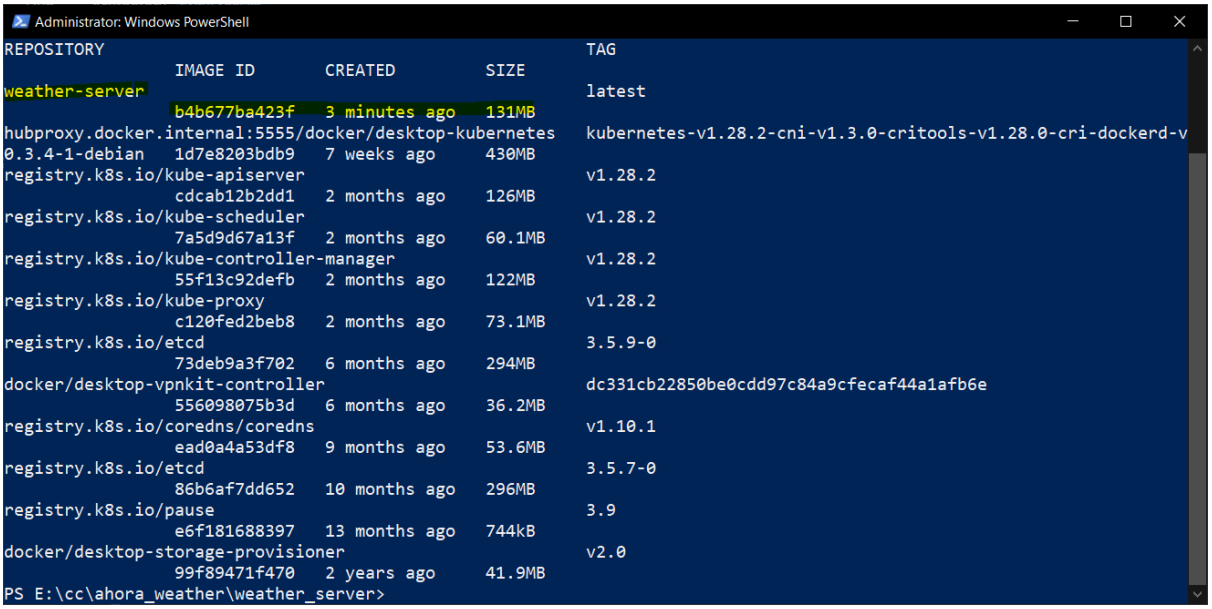
داکر فایل برای ساختن ایمیج آن:

```
FROM node:16-alpine as base

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 80

CMD ["npm", "start"]

```

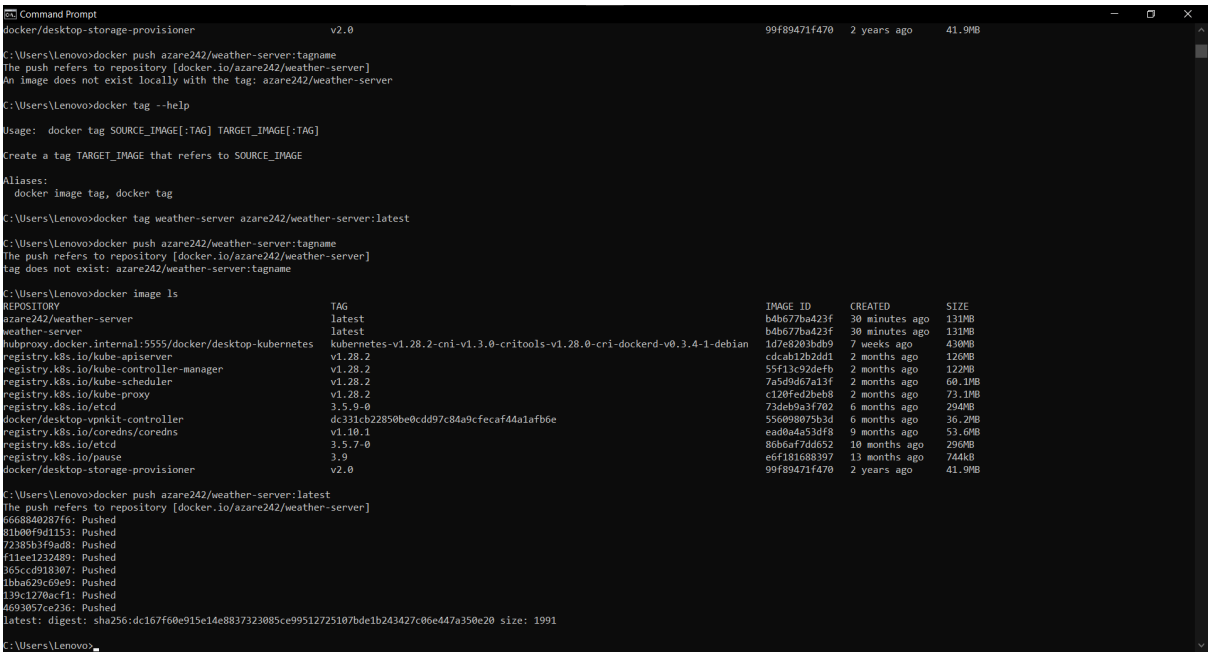
مراحل بیلد ایمیج:

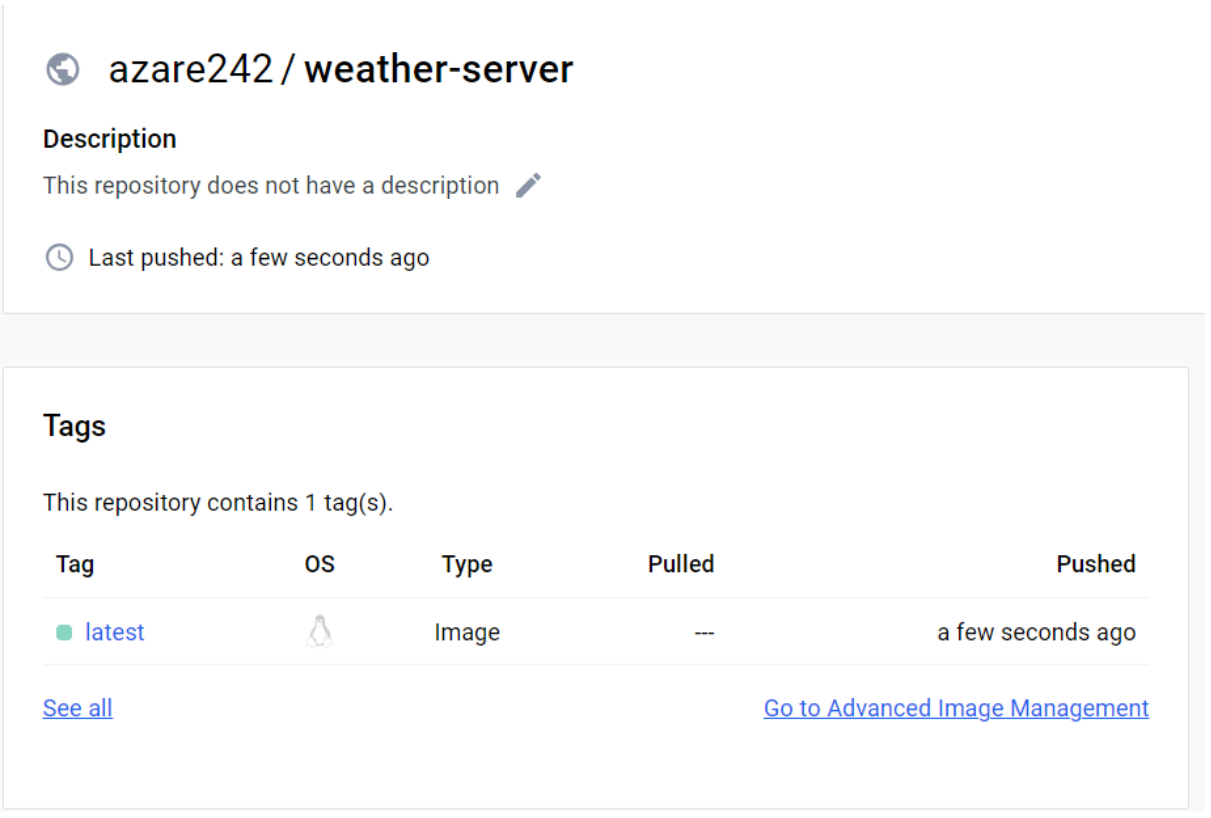




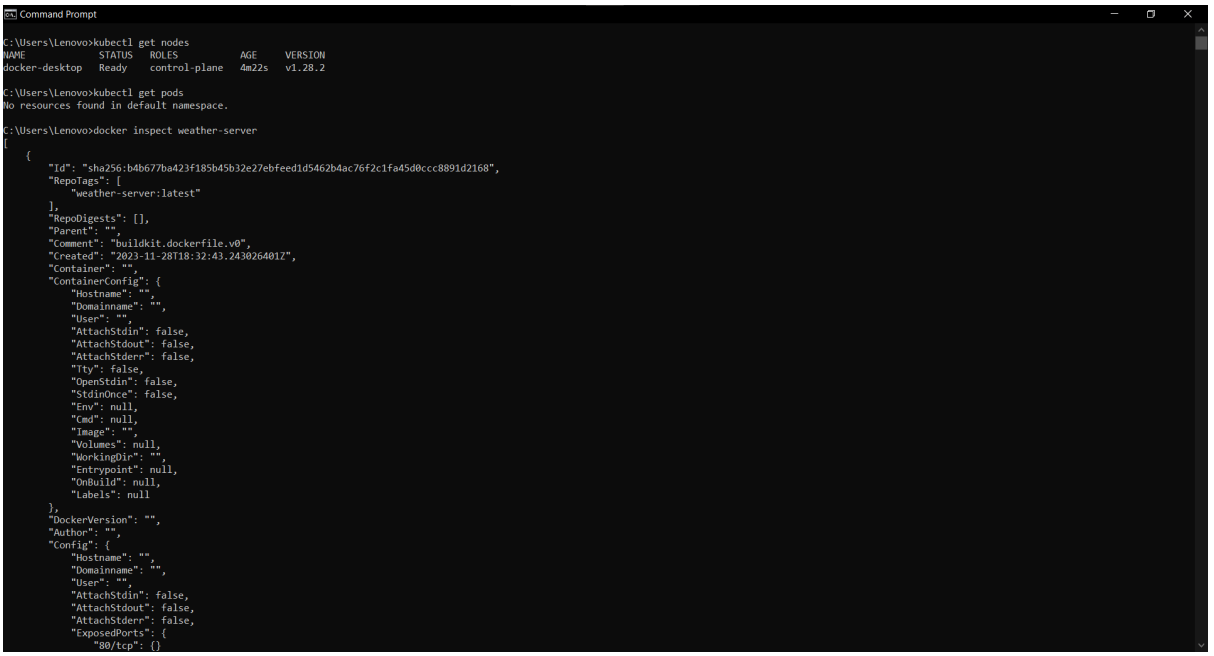

Administrator: Windows PowerShell
REPOSITORY TAG
IMAGE ID CREATED SIZE
weather-server latest
b4b677ba423f 3 minutes ago 131MB
hubproxy.docker.internal:5555/docker/desktop-kubernetes kubernetes-v1.28.2-cni-v1.3.0-critools-v1.28.0-cri-dockerd-v
0.3.4-1-debian 1d7e8203bdb9 7 weeks ago 430MB
registry.k8s.io/kube-apiserver v1.28.2
cdcab12b2dd1 2 months ago 126MB
registry.k8s.io/kube-scheduler v1.28.2
7a5d9d67a13f 2 months ago 60.1MB
registry.k8s.io/kube-controller-manager v1.28.2
55f13c92defb 2 months ago 122MB
registry.k8s.io/kube-proxy v1.28.2
c120fed2beb8 2 months ago 73.1MB
registry.k8s.io/etcd 3.5.9-0
73deb9a3f702 6 months ago 294MB
docker/desktop-vpnkit-controller dc331cb22850be0cdd97c84a9cfecaf44a1afb6e
556098075b3d 6 months ago 36.2MB
registry.k8s.io/coredns/coredns v1.10.1
ead0a4a53df8 9 months ago 53.6MB
registry.k8s.io/etcd 3.5.7-0
86b6af7dd652 10 months ago 296MB
registry.k8s.io/pause 3.9
e6f181688397 13 months ago 744kB
docker/desktop-storage-provisioner v2.0
99f89471f470 2 years ago 41.9MB
PS E:\cc\ahora_weather\weather_server>

قراردهی ایمیج روی داکر هاب:





همچنین inspect ایمیج:



داکر کامپوز:

```yaml
version: '3'
services:
  app:
    image: "azare242/weather-server:latest"
    ports:
      - "9999:80"
    depends_on:
      - redis
    environment:
      - PORT=80
      - REDIS_HOST=redis
      - REDIS_PORT=6379
      - WEATHER_URL=https://api.api-ninjas.com/v1/weather?city=
      - WEATHER_AUTH=YbdgZf59lSo6VSJ6BaklYQ==xghcOBjyCR8UbPPf
      - REDIS_PAIR_TTL=300
  redis:
    image: "redis:7.2-alpine"
    ports:
    - "8989:6379"
    volumes:
      - redis-data:/data
volumes:
  redis-data:

```

از ردیس 7.2 با بیس ایمیج الپاین استفاده شده است.

والیوم redis-data برای پرسیست کردن دیتای ردیس استفاده می شود.
کامپوز خودش نتورک مورد نیاز را میسازد.

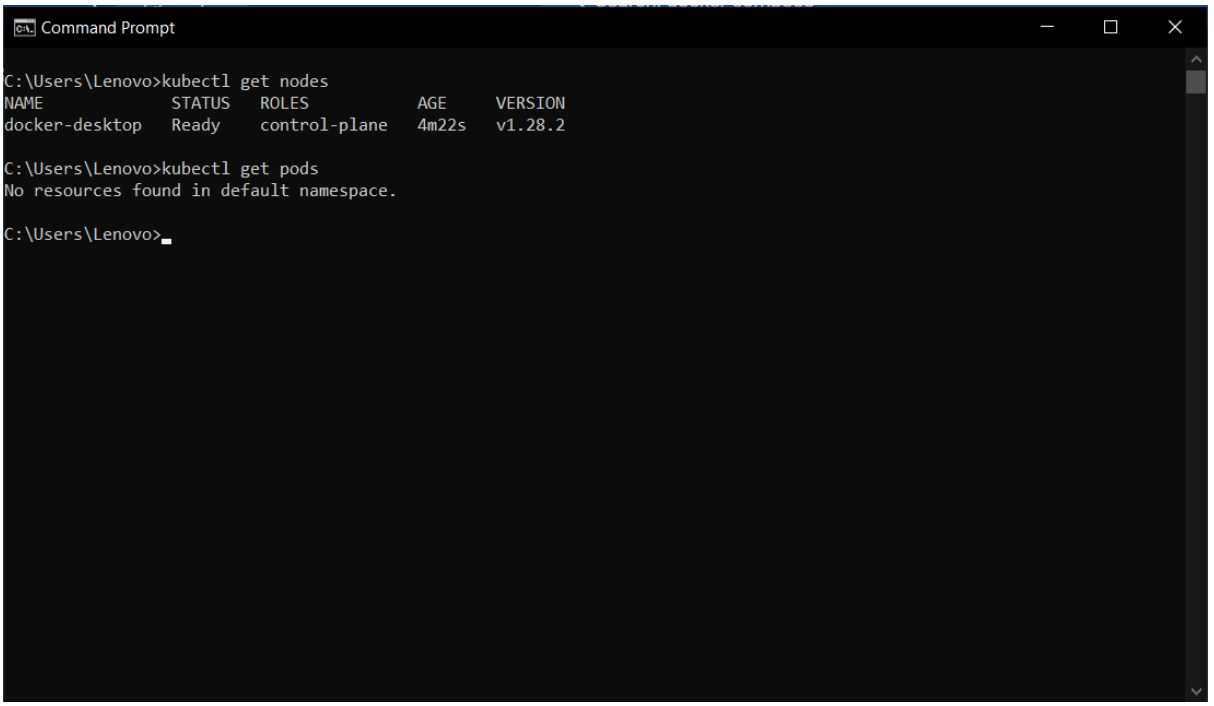
```
weather_server-app-1  |
weather_server-app-1  | > weather@0.0.0 start
weather_server-app-1  | > node index.js
weather_server-app-1  |
weather_server-app-1  | {
weather_server-app-1  |   PORT: '80',
weather_server-app-1  |   REDIS_HOST: 'redis',
weather_server-app-1  |   REDIS_PORT: '6379',
weather_server-app-1  |   REDIS_PAIR_TTL: '300',
weather_server-app-1  |   WEATHER_URL: 'https://api.api-ninjas.com/v1/weather?city=',
weather_server-app-1  |   WEATHER_AUTH: 'YbdgZf59lSo6VSJ6BaklYQ==xghcOBjyCR8UbPPf'
weather_server-app-1  | }
weather_server-app-1  | Server running on 80
weather_server-app-1  | CACHE MISS
weather_server-app-1  | [GET] "/api/v1/weather/tehran" (200) 31 - 3429.227 ms
weather_server-app-1  | CACHE HIT
weather_server-app-1  | [GET] "/api/v1/weather/tehran" (200) 31 - 5.237 ms
weather_server-app-1  | CACHE HIT
weather_server-app-1  | [GET] "/api/v1/weather/tehran" (200) 31 - 135.414 ms
weather_server-app-1  | CACHE HIT
weather_server-app-1  | [GET] "/api/v1/weather/tehraN" (200) 31 - 12.714 ms
weather_server-app-1  | CACHE MISS
weather_server-app-1  | [GET] "/api/v1/weather/rafsanjan" (200) 31 - 2216.120 ms
weather_server-app-1  | CACHE HIT
weather_server-app-1  | [GET] "/api/v1/weather/rafsanjan" (200) 31 - 2.914 ms
weather_server-app-1  | CACHE HIT
weather_server-app-1  | [GET] "/api/v1/weather/rafsanjan" (200) 31 - 8.095 ms
weather_server-app-1  | CACHE HIT
weather_server-app-1  | [GET] "/api/v1/weather/rafsanjan" (200) 31 - 2.307 ms
weather_server-app-1  | CACHE MISS
weather_server-app-1  | [GET] "/api/v1/weather/ardabil" (200) 29 - 1163.215 ms
```

نتایج کش کردن را مشاهده میکنیم.

مقدار اعتبار هر Pair کش 5 دقیقه است.

قسمت دوم تمرین (کوبرنتیز)

از داکر برای نصب مینی کیوب استفاده شده است و نتایج نصب آن:



ر

یسورس های مورد استفاده برای اجرای سرویس شامل Serivce Deployment ConfigMap هستند:

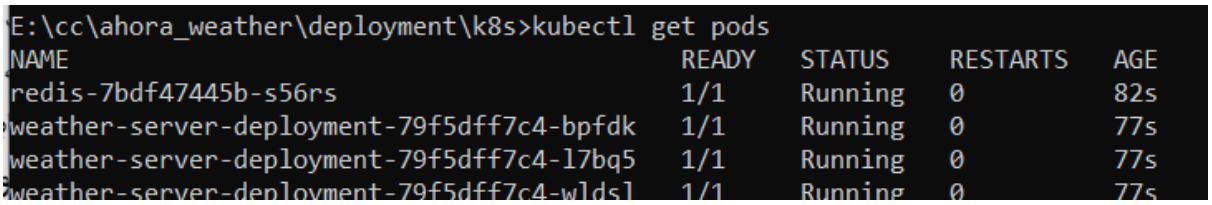
```yaml
apiVersion: v1
kind: Service
metadata:
  name: weather-server-service
spec:
  selector:
    app: weather-server
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80

```


apiVersion: v1
kind: ConfigMap
metadata:
name: my-configmap
data:
config.yaml: -
PORT: 80
REDIS_HOST: redis
REDIS_PORT: 6379
WEATHER_URL: https://api.api-ninjas.com/v1/weather?city=
WEATHER_AUTH: YbdgZf59lSo6VSJ6BaklYQ==xghcOBjyCR8UbPPf
REDIS_PAIR_TTL: 300

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: weather-server-deployment
  labels:
    app: weather-server
spec:
  replicas: 3
  selector:
    matchLabels:
      app: weather-server
  template:
    metadata:
      labels:
        app: weather-server
    spec:
      containers:
      - name: wather-server
        image: azare242/weather-server:latest
        ports:
        - containerPort: 80
        volumeMounts:
          - mountPath: /configs
            name: config-volume
      volumes:
        - name: config-volume
          configMap:
            name: my-configmap
```

با دستور kubectl apply -f میتوان آن هارا به گره خود اضافه کرد.

در اینجا چون تعداد رپلیکاهای سرور هواشناسی را 3 در نظر گرفتیم 3 پاد از ان ساخته می شود.

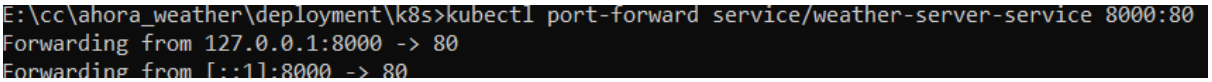


چون پاد در صورت از بین رفتن ایپی خودرا از دست میدهد نیاز داریم که از ایپی سرویس استفاده کنیم. که سرویس و کلاسترایپی درخواست هارا به پاد ها ارسال میکند. که فقط از داخل کلاستر قابل دستیابی است. برای دسترسی از خارج کلاستر باید از نود پورت یا لود بالانسر استفاده کرد. (در فایل های yml ما نوع سرویس را مشخص نکردیم چون مقدار پیشفرض از طرف کوبرنتیز کلاستر ایپی است)

حال نوبت به دیپلوی کردن ردیس است.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis
spec:
  replicas: 1
  selector:
    matchLabels:
      name: redis
  template:
    metadata:
      labels:
        name: redis
    spec:
      volumes:
        - name: redis-data-store
          persistentVolumeClaim:
            claimName: redis-pv-claim
      containers:
        - name: redis
          image: redis:7.2-alpine
          ports:
            - containerPort: 6379
          volumeMounts:
            - mountPath: "/data"
              name: redis-data-store

```

با پورت فورواردکردن با استفاده از دستور kubectl port-forward می توان به آن دسترسی داشت.



حال اگر پاد ردیس از بین برود و دوباره آن را اجرا کنیم دیتای آن از دست می رود پس باید pv و pvc برای ردیس نوشت:

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: redis-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/home/dev/redis_data"
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: redis-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 500Mi
```

مینی کیوب به ما dynamic provisioning را ارائه میکند. این قابلیت خودکار حاقظه تخصیص میدهد. که باعث مدیریت بهتر در کلاستر ها می شود.

بصورت کلی هر pv را فقط میتوان به pvc های همان نود اختصاص داد و در صورتی که یک pvc نود بخواهد به آن دسترسی یابد باید pvc دیگری دسترسی نخواهد داشت مگر اینکه بقیه pvc ها خودرا به pv بایند کنند.. این قابلیت به isolation فضای ذخیره سازی کمک میکند.

```
E:\cc\ahora_weather\deployment\k8s>curl localhost:8000/api/v1/weather/tehran
{"weather":{"min":8,"max":11}}
```

```
E:\cc\ahora_weather\deployment\k8s>kubectl logs weather-server-deployment-79f5dff7c4-wdxw6

> weather@0.0.0 start
> node index.js

{
  PORT: '80',
  REDIS_HOST: 'redis',
  REDIS_PORT: '6379',
  REDIS_PAIR_TTL: '300',
  WEATHER_URL: 'https://api.api-ninjas.com/v1/weather?city=',
  WEATHER_AUTH: 'YbdgZf59lSo6VSJ6BaklYQ==xghcOBjyCR8UbPPf'
}
Server running on 80
```

با port-forward کردن میتوان به سرویس دسترسی داشت که سرویس خود با انتخاب پاد به آن درخواست میزند.

حال چون pv برای ردیس داریم هربار که پاد ردیس نابود شود و جایگزین شود دیتا همیشه پابرجاست و میتوان به آن دسترسی داشت.

برای اینکه ببینیم همه پادها رکوست میخورد از ایمیج بخش اول تمرین به عنوان یک پاد درخواست دهنده استفاده میکنیم.