

گزارش پروژه دوم رایانش ابری - داکر و مقدمات کوبرنتیز

گام اول -

ابتدا یک دیرکتوری برای داکر فایل ساخته سپس یک داکر فایل در آن میسازیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image>mkdir my-docker-image
```

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image>type nul > Dockerfile
```

در

داکر فایل عبارات زیر را مینویسم که برای نصب cURL در ubuntu است:

```

Dockerfile X
rs > Samin > Desktop > University > Term 7 > Cloud Computing > Projects > Project2 > my-doc
1 FROM ubuntu:latest
2 RUN apt-get update && apt-get install -y curl
3

```

حالا داکر فایل را ذخیره کرده و docker image را میسازیم:

```

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image>docker build -t my-ubuntu-image .
[+] Building 232.7s (4/5)
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 31B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> CACHED [1/2] FROM docker.io/library/ubuntu:latest@sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a6d21
=> [2/2] RUN apt-get update && apt-get install -y curl
=> # Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 publicsuffix all 20211207.1025-1 [129 kB]
=> # Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 libbrotli1 amd64 1.0.9-2build6 [315 kB]
=> # Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libssl2 amd64 2.1.2+dfsg-2ubuntu1.2 [20.5 kB]
=> # Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libssl1.1 amd64 2.1.2+dfsg-2ubuntu1.2 [53.8 kB]
=> # Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libldap-2.5-0 amd64 2.5.14+dfsg-0ubuntu0.22.04.1 [183 kB]
=> # Get:10 http://archive.ubuntu.com/ubuntu jammy/main amd64 librtmp1 amd64 2.4+20151223.gitfa8646d.1-2build4 [58.2 kB]

```

مدتی طول میکشد تا docker image ساخته شود:

```

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image>docker build -t my-ubuntu-image .
[+] Building 263.3s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 31B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/ubuntu:latest
=> CACHED [1/2] FROM docker.io/library/ubuntu:latest@sha256:67211c14fa74f070d27cc59d69a7fa9aeff8e28ea118ef3babc295a0428a6d21
=> [2/2] RUN apt-get update && apt-get install -y curl
=> exporting to image
=> => exporting layers
=> => writing image sha256:9bfa632483d99c565ff16f3faddaee84a00eac55516fdf7f2ef4abee1100bea
=> => naming to docker.io/library/my-ubuntu-image

```

حالا آن را چک میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image> docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-ubuntu-image	latest	9bfa632483d9	46 seconds ago	128MB
kiicbase/stable	v0.0.37	01c0ce65fff7	2 months ago	1.15GB
hello-world	latest	feb5d9fea6a5	18 months ago	13.3kB

همانطور که مشاهده میشود در لیست تصاویر داکری که در سیستم وجود دارد تصویر جدید هم وجود دارد پس به درستی ساخته شده است.

میخواهیم این تصویر را روی داکرهاب قرار دهیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image> docker login
Authenticating with existing credentials...
Login Succeeded

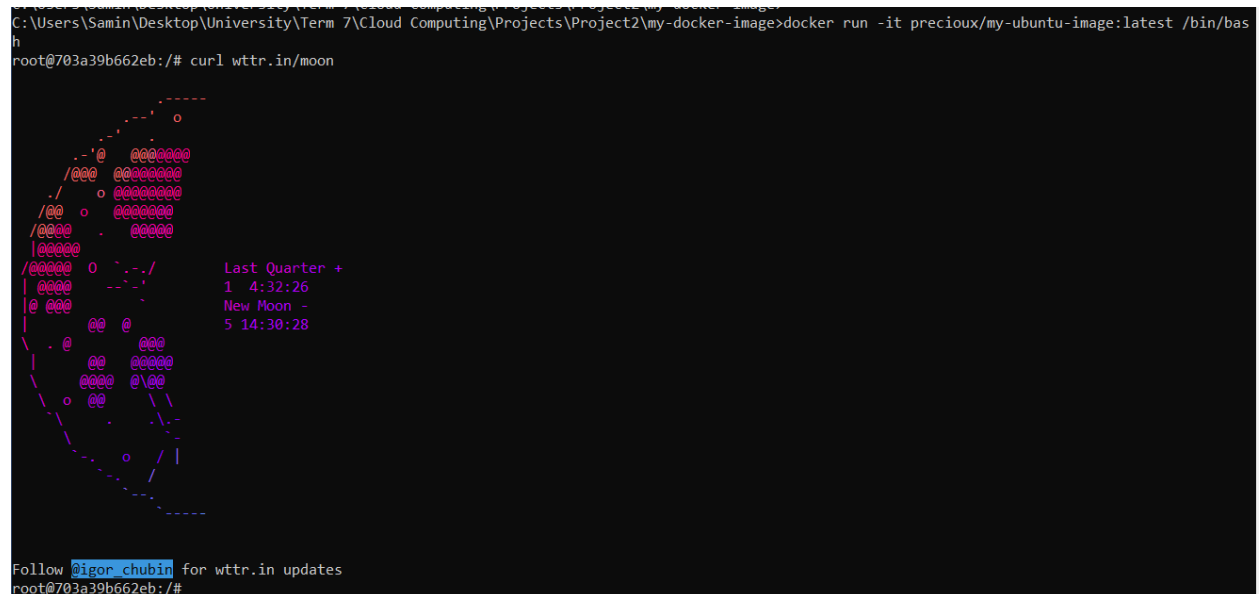
Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/
```

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image> docker tag my-ubuntu-image precieux/my-ubuntu-image:latest
```

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image> docker push precieux/my-ubuntu-image:latest
The push refers to repository [docker.io/precieux/my-ubuntu-image]
cae44c2fe186: Pushed
b93c1bd012ab: Mounted from library/ubuntu
latest: digest: sha256:d1664306c0845e1f77850a2e7469af299f0d35b5687b3243ea4cbe2b27c34015 size: 741
```

حالا برای تست آن را دانلود و ران میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\my-docker-image> docker run -it precieux/my-ubuntu-image:latest /bin/bash
root@703a39b662eb:/# curl wttr.in/moon
```



```
Follow @igor_chubin for wttr.in updates
root@703a39b662eb:/#
```

همانطور که مشاهده میشود یک بش بر پایه لینوکس بالا آمد که با استفاده از آن کرل مورد نظر را زدیم و پاسخ گرفتیم.

گام دوم

برای انجام این بخش از APILayer استفاده میکنیم. برای بالا آوردن یک کانتینر از کش ردیس بصورت زیر

عمل میکنیم:

```
main.py × server\Dockefile × redis\Dockefile ×
1 FROM redis:latest
2
3 CMD ["redis-server"]
4
```

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part II>docker build -t my-redis redis/.
[+] Building 0.1s (5/5) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 56B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/redis:latest 0.0s
=> CACHED [1/1] FROM docker.io/library/redis:latest 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:dd61760911b819214857107dc2d6af55fb9bec24de9f6ac0e1e4c1e928b109a2 0.0s
=> => naming to docker.io/library/my-redis 0.0s
```

حالا داکر فایل سرور را مینویسیم:

```
>> FROM python:3.9

WORKDIR /app

COPY requirements.txt .

RUN pip install -r requirements.txt

COPY . .

CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "8000"]
```

برای این بخش نیاز به requirements.txt داریم:

```
fastapi
redis
requests
uvicorn
PyYAML==5.4.1
```

حالا تصویر سرور را ایجاد میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part II>docker build -t fastapi-server server
/.
[+] Building 3.9s (11/11) FINISHED
=> [internal] load build definition from Dockerfile 0.1s
=> => transferring dockerfile: 32B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load metadata for docker.io/library/python:3.9 3.2s
=> [auth] library/python:pull token for registry-1.docker.io 0.0s
=> [1/5] FROM docker.io/library/python:3.9@sha256:6ea9dafc96d7914c5c1d199f1f0195c4e05cf017b10666ca84cb7ce8e2699d 0.0s
=> [internal] load build context 0.1s
=> => transferring context: 3.44kB 0.0s
=> CACHED [2/5] WORKDIR /app 0.0s
=> CACHED [3/5] COPY requirements.txt . 0.0s
=> CACHED [4/5] RUN pip install -r requirements.txt 0.0s
=> [5/5] COPY . . 0.1s
=> exporting to image 0.1s
=> => exporting layers 0.1s
=> => writing image sha256:d13051ab79c7921e8290a09529d9b28e43eac84ce63b3daa390b5f0f43edb8a3 0.0s
=> => naming to docker.io/library/fastapi-server 0.0s
```

حالا این تصویر را روی داکر هاب قرار میدهیم

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part II>docker push precieux/fastapi-server:latest
The push refers to repository [docker.io/precieux/fastapi-server]
7d2e6f454509: Pushed
b489cbbb15ef: Pushed
acb0a7a47e2f: Pushed
7f0a031595db: Pushed
7590f3933b44: Mounted from library/python
6e281c24915d: Mounted from library/python
137c4023273a: Mounted from library/python
0007505dc811: Mounted from library/python
f43725f97b9f: Mounted from library/python
9c42af2c6418: Mounted from library/python
d96e248f10e6: Mounted from library/python
d925e0fae4e6: Mounted from library/python
latest: digest: sha256:7807493f65f07efe3445047ed9bf35fc1f94703b3487bdc1066ba7bc235306b4 size: 2841
```

برای ردیس یک volume تعریف میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part II>docker volume create redis-data
redis-data
```

حالا لازم داریم که شبکه ای تعریف کنیم که این دو کانتینر باهم در ارتباط باشند:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part II>docker network create cc-network
15d8d4e91f84fee10c9749a721c9b1d2ac85ec3ec0a2c87be7d23a08e64f4956
```









```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part II>docker network inspect cc-network
[
  {
    "Name": "cc-network",
    "Id": "15d8d4e91f84fee10c9749a721c9b1d2ac85ec3ec0a2c87be7d23a08e64f4956",
    "Created": "2023-04-15T11:39:00.3295102Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.21.0.0/16",
          "Gateway": "172.21.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "20ff46b50973957091f6208363af215c0e2c3201e383fbc3c2be5d623c2682ba": {
        "Name": "my-server",
        "EndpointID": "72ca275cfb94c2669e0165c392d717e87e8c00436d00c63df9561c15b9c708f3",
        "MacAddress": "02:42:ac:15:00:03",
        "IPv4Address": "172.21.0.3/16",
        "IPv6Address": ""
      },
      "31d4e350c77614b396b16d76ee9f60f756cb00401308ab7339bf6d958df2926b": {
        "Name": "my-redis",
        "EndpointID": "2ee82210620c5096c88cdf33edbbd052f5f29583b1a7a6a3a139d832b8454feb",
        "MacAddress": "02:42:ac:15:00:02",
        "IPv4Address": "172.21.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
```

همانطور که مشاهده میشود شبکه تشکیل شده و دو کانتینر به آن متصل شده اند. کانتینر سرور روی ۱۷۲.۲۱.۰.۳ در حال شنیدن و کانتینر ردیس روی ۱۷۲.۲۱.۰.۲ در حال شنیدن میباشد.

کانتینر هارا بالا می آوریم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part II>docker run -d --name redis-cache --network cc-network -p 6379:6379 -v redis-data:/data redis-cache
2708210d352650530a45e8798cd3b6ad72216a7bc267b83e32b557a414a145b6

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part II>docker run -it --name fastapi-server --network cc-network -p 8000:8000 fastapi-server
Redis container is active!
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
```

<input type="checkbox"/>	 redis-cache 9e9b754135bf	redis:latest	Running	1 minute ago	  
<input type="checkbox"/>	 fastapi-server 7c4a4f7cfa01	fastapi-server:latest	Running	31 seconds ago	  

با دستور `docker inspect` تصویر `fastapi-server` را بررسی میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects>docker inspect fastapi-server
[
  {
    "Id": "551cd1e15f16162b750c38a8af112a014f019333a2a4218e8eb2a8f4f15c52",
    "Created": "2023-05-10T04:11:57.146893Z",
    "Path": "uvicorn",
    "Args": [
      "main:app",
      "--host",
      "0.0.0.0",
      "--port",
      "8000"
    ],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 2563,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2023-05-10T04:11:58.6284699Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:d13051ab79c7921e8290a09529d9b28e43eac84ce63b3daa390b5f0f43edb8a3",
    "ResolvConfPath": "/var/lib/docker/containers/551cd1e15f16162b750c38a8af112a014f019333a2a4218e8eb2a8f4f15c52/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/551cd1e15f16162b750c38a8af112a014f019333a2a4218e8eb2a8f4f15c52/hostname",
    "HostsPath": "/var/lib/docker/containers/551cd1e15f16162b750c38a8af112a014f019333a2a4218e8eb2a8f4f15c52/hosts",
    "LogPath": "/var/lib/docker/containers/551cd1e15f16162b750c38a8af112a014f019333a2a4218e8eb2a8f4f15c52/json.log",
    "Name": "/fastapi-server",
    "RestartCount": 0,
    "Driver": "overlay2",
    "Platform": "linux",
    "MountLabel": "",
    "ProcessLabel": "",
    "AppArmorProfile": "",
    "ExecIDs": null,
    "HostConfig": {
      "Binds": null,
      "ContainerIDFile": "",
      "LogConfig": {
        "Type": "json-file",
        "Config": {}
      },
      "NetworkMode": "cc-network",
      "PortBindings": {
        "8000/tcp": [

```

کانتینرهای موجود در سیستم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
551cd1e15f16   fastapi-server "uvicorn main:app --..." 3 minutes ago  Up 3 minutes  0.0.0.0:8000->8000/tcp              fastapi-server
2708210d3526   redis-cache    "docker-entrypoint.s..." 4 minutes ago  Up 4 minutes  0.0.0.0:6379->6379/tcp              redis-cache
```

منابع مورد استفاده این کانتینرها بصورت زیر است:

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
551cd1e15f16	fastapi-server	0.26%	26.71MiB / 6.124GiB	0.43%	1.27kB / 370B	0B / 0B	1
2708210d3526	redis-cache	0.19%	7.586MiB / 6.124GiB	0.12%	1.49kB / 297B	0B / 0B	5

حالا از برنامه تست میگیریم و آدرس <https://ce.aut.ac.ir> را برای آن ارسال میکنیم:

```
import json
import requests

url = "http://localhost:8000/shorten_url/"
long_url = "https://ce.aut.ac.ir/"
payload = {
    "long_url": long_url
}
headers = {
    "Accept": "application/json"
}

response = requests.get(url, params=payload, headers=headers)

if response.status_code == 200:
    data = response.json()
    res = json.dumps(data)
    print(f"Response: {res}")
else:
    print("Failed to shorten URL")
```

نتیجه بصورت زیر خواهد بود:

Response:

```
{\"longUrl\": \"https://ce.aut.ac.ir/\",
 \"shortUrl\": \"https://p.rs/Ohh°v\",
 \"isCached\": false,
 \"hostname\": \"Nyx\"}
```

گام سوم

ابتدا مینیکوب را روی داکر بالا می آوریم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III>minikube start
* minikube v1.29.0 on Microsoft Windows 10 Pro 10.0.19044.2846 Build 19044.2846
* Using the docker driver based on existing profile
* Starting control plane node minikube in cluster minikube
* Pulling base image ...
* Restarting existing docker container for "minikube" ...
* minikube 1.30.1 is available! Download it: https://github.com/kubernetes/minikube/releases/tag/v1.30.1
* To disable this notice, run: 'minikube config set WantUpdateNotification false'

! This container is having trouble accessing https://registry.k8s.io
* To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
* Preparing Kubernetes v1.26.1 on Docker 20.10.23 ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

- برای ساخت configmap از فایل کانفیگ زیر استفاده میکنیم:

```
config.yaml X
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes> ! config.yaml
1 # config.yaml
2 port: 6379
3 cache_expiry: 5
4 api_endpoint: f"https://api.apilayer.com/short_url/hash"
5 api_key: "8P71Zm1JorY60oeVYV7LXm9ID3VD2ICO"
6 hostname: "Nyx"
7 |
```

حالا از این فایل کانفیگ مپ میسازیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes>kubectl create configmap
my-config --from-file=config.yaml
configmap/my-config created
```

آن را بررسی میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes>kubectl describe configmap
p my-config
Name:         my-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
config.yaml:
----
# config.yaml
port: 6379
cache_expiry: 5
api_endpoint: f"https://api.apilayer.com/short_url/hash"
api_key: "8P71Zm1JorY60oeVYV7LXm9ID3VD2ICO"
hostname: "Nyx"

BinaryData
====

Events:  <none>
```


با انجام این مراحل کانفیگ مپ اپلای میشود، حالا فایل های خواسته شده دیگر را هم تعریف کرده سپس به ترتیب اپلای میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl apply -f deployment.yaml
deployment.apps/fastapi-server created

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl apply -f fastapi-service.yaml
service/fastapi-server created

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl get deployment fastapi-server
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
fastapi-server	0/2	2	0	45s

با بررسی مشاهده میشود که فایل ها به درستی ساخته شده اند.

همچنین کانفیگ مپ را مجددا چک میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl get configmap
```

NAME	DATA	AGE
kube-root-ca.crt	1	41d
my-config	1	5h23m

سه فایل قبلی را برای ردیس کش میسازیم:

Configmap

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl create -f redis-config.yaml
configmap/redis-config created

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl get configmap
```

NAME	DATA	AGE
kube-root-ca.crt	1	41d
my-config	1	5h34m
redis-config	1	64s

Deployment and service

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl apply -f deployment.yaml
deployment.apps/redis-cache created

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl apply -f redis-cache-service.yaml
service/redis-cache created

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl get deployment redis-cache
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
redis-cache	0/1	1	0	32s

دو فایل دیگر را میسازیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl apply
-f redis-pv.yaml
persistentvolume/redis-pv created
```

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl apply
-f pvc.yaml
persistentvolumeclaim/redis-pvc created
```

حالا فایل های قرار داده شده را بررسی میکنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl get d
eployment redis-cache
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
redis-cache	1/1	1	1	10m

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl get p
vc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
redis-pvc	Bound	redis-pv	1Gi	RWO	standard	4m20s

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\redis-cache>kubectl get p
v
```

NAME	CAPACITY	ACCESS MODES	RECLAIM POLICY	STATUS	CLAIM	STORAGECLASS	REASON	AGE
redis-pv	1Gi	RWO	Retain	Bound	default/redis-pvc	standard		8m12s

برای بررسی آدرس ها بصورت زیر عمل میکنیم:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/fastapi-server	LoadBalancer	10.101.212.202	<pending>	80:31132/TCP	26h
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	42d
service/redis-cache	ClusterIP	10.104.144.233	<none>	6379/TCP	26h

نهایتا مجددا منابع را با دستور `kubectl get` بررسی میکنیم:

کانفیگ مپ ها:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl get configmap
```

NAME	DATA	AGE
kube-root-ca.crt	1	155m
redis-config	1	104m
server-config	1	64m

همانطور که مشاهده میشود کانفیگ مپ برای سرور و ردیس به درستی ایجاد شده اند.

دیپلویمنت ها:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
fastapi-server-deployment  2/2     2             2           65m
redis-cache          1/1     1             1           102m
```

همانطور که مشاهده میشود فایل های دیپلویمنت به درستی ساخته شده اند

سرویس ها:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl get svc
NAME                TYPE        CLUSTER-IP      EXTERNAL-IP  PORT(S)    AGE
fastapi-server-deployment  ClusterIP   10.103.130.247  <none>       8000/TCP   27m
kubernetes           ClusterIP   10.96.0.1       <none>       443/TCP    157m
redis-cache-service     ClusterIP   10.104.98.129  <none>       6379/TCP   89m
```

همانطور که مشاهده میشود سرویس ها به درستی ساخته شده اند.

: Pv,pvc

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                STORAGECLASS   REASON   AGE
redis-pv   1Gi        RWO            Retain           Bound    default/redis-pvc    standard              2d7h

C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl get pvc
NAME      STATUS   VOLUME    CAPACITY   ACCESS MODES   STORAGECLASS   AGE
redis-pvc Bound    redis-pv   1Gi        RWO            standard              2d7h
```

همانطور که مشاهده میشود فایل ها به درستی تعریف و ایجاد شده اند.

برای نمایش توزیع درخواست ها بین دو پاد سرور چندین درخواست به سرور میزنیم:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl exec -it ubuntu-deployment-546bf655c4-p58c9 -- /bin/bash
root@ubuntu-deployment-546bf655c4-p58c9:/# curl fastapi-server-service:8000/test
{"longUrl": "https://www.aut.ac.ir", "shortUrl": "https://p1.rs/BLVQ1", "isCached": false, "root@ubuntu-deployment-546bf655c4-p58c9:/# curl fastapi-server-service:8000/shorten_url?long_url=https://www.lms.aut.ac.ir
{"longUrl": "https://www.lms.aut.ac.ir", "shortUrl": "https://p1.rs/3vLGC", "isCached": false, "root@ubuntu-deployment-546bf655c4-p58c9:/# curl fastapi-server-service:8000/shorten_url?long_url=https://www.samad.aut.ac.ir
{"longUrl": "https://www.samad.aut.ac.ir", "shortUrl": "https://p1.rs/wjHY4", "isCached": false, "root@ubuntu-deployment-546bf655c4-p58c9:/# curl fastapi-server-service:8000/shorten_url?long_url=https://www.portal.aut.ac.ir
{"longUrl": "https://www.portal.aut.ac.ir", "shortUrl": "https://p1.rs/2geXy", "isCached": false, "hostname": "root@ubuntu-deplroot@uburoot@ubroot@u
root@ubuntu-deployment-546bf655c4-p58c9:/#
```

حالا با لاگ گرفتن از پاد ها بررسی میکنیم این درخواست ها را چه کسی پاسخ گفته:

پاد اول سه درخواست را جواب داده:

```
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
Redis container is active!
INFO: 10.244.0.41:40054 - "GET /test HTTP/1.1" 200 OK
long URL : https://www.aut.ac.ir
Status : 200
Result : {"hash": "BLVQ1", "short_url": "https://p1.rs/BLVQ1", "long_url": "https://www.aut.ac.ir"}

Shorten url : https://p1.rs/BLVQ1
INFO: 10.244.0.41:41212 - "GET /shorten_url?long_url=https://www.aut.ac.ir HTTP/1.1" 200 OK
long URL : https://www.lms.aut.ac.ir
Status : 200
Result : {"hash": "3vLGC", "short_url": "https://p1.rs/3vLGC", "long_url": "https://www.lms.aut.ac.ir"}

Shorten url : https://p1.rs/3vLGC
INFO: 10.244.0.41:42090 - "GET /shorten_url?long_url=https://www.lms.aut.ac.ir HTTP/1.1" 200 OK
long URL : https://www.samad.aut.ac.ir
Status : 200
Result : {"hash": "wjHY4", "short_url": "https://p1.rs/wjHY4", "long_url": "https://www.samad.aut.ac.ir"}

Shorten url : https://p1.rs/wjHY4
INFO: 10.244.0.41:42430 - "GET /shorten_url?long_url=https://www.samad.aut.ac.ir HTTP/1.1" 200 OK
```

و پاد دوم یکی از درخواست ها را:

```
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>kubectl logs fastapi-server-deployment-768857fcd-b8zh
INFO: Started server process [1]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
Redis container is active!
long URL : https://www.portal.aut.ac.ir
Status : 200
Result : {"hash": "2geXy", "short_url": "https://p1.rs/2geXy", "long_url": "https://www.portal.aut.ac.ir"}

Shorten url : https://p1.rs/2geXy
INFO: 10.244.0.41:42558 - "GET /shorten_url?long_url=https://www.portal.aut.ac.ir HTTP/1.1" 200 OK
C:\Users\Samin\Desktop\University\Term 7\Cloud Computing\Projects\Project2\Part III\kubernetes\fastapi-server>
```

بنابراین درخواست ها بین این دو پاد سرور توزیع شده بوده است.