

مرحله اول

```
~ (18.739s)
docker pull alpine:3.17
3.17: Pulling from library/alpine
4060ece20d7a: Pull complete
Digest: sha256:f71a5f071694a785e064f05fed657bf8277f1b2113a8ed70c90ad486d6ee54dc
Status: Downloaded newer image for alpine:3.17
docker.io/library/alpine:3.17
```

ابندا ایمیج **alpine** را از داکر هاب دانلود می کنیم.

```
~
docker run -it --name alpine alpine:3.17
/ # echo nameserver 8.8.8.8 > /etc/resolv.conf
/ # apk update
fetch https://dl-cdn.alpinelinux.org/alpine/v3.17/main/aarch64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.17/community/aarch64/APKINDEX.tar.gz
v3.17.5-144-g9271183f7c5 [https://dl-cdn.alpinelinux.org/alpine/v3.17/main]
v3.17.5-144-g9271183f7c5 [https://dl-cdn.alpinelinux.org/alpine/v3.17/community]
OK: 17692 distinct packages available
/ # apk add curl
(1/5) Installing ca-certificates (20230506-r0)
(2/5) Installing brotli-libs (1.0.9-r9)
(3/5) Installing nghttp2-libs (1.51.0-r2)
(4/5) Installing libcurl (8.4.0-r0)
(5/5) Installing curl (8.4.0-r0)
Executing busybox-1.35.0-r29.trigger
Executing ca-certificates-20230506-r0.trigger
OK: 10 MiB in 20 packages
/ # █
```

در این مرحله از ایمیج **alpine** یک کانتینر می سازیم و وارد آن می شویم تا دستورات مورد نظر را وارد کنیم. **cURL** را روی آن نصب می کنیم.

```

~ (0.441s)
docker commit 013d42001686 alpine-curl-commit:1.0.0
sha256:e7ce1679c744890842d23e68a68339507d3fa1c5506b04fa8ab95326885ba158

~ (0.196s)
docker images | grep alpine
alpine-curl-commit          1.0.0      e7ce1679c744  13 seconds ago  12.4MB
python                      3.11-alpine  143748e8d441  3 weeks ago   57.6MB
alpine                      latest     3cc203321400  4 weeks ago   7.66MB
rabbitmq                    3.12-alpine  b35a2a591395  5 weeks ago   134MB
alpine                      3.17      8650cd65339b  2 months ago  7.39MB

```

در مرحله بعد با استفاده از دستور Commit یک ایمیج از کانتینر خود می‌سازیم.

```

Dockerfile ×
Users > farhad > Desktop > 9931006_HW2 > step-1 > Dockerfile > ...
1 FROM alpine:3.17
2
3 RUN apk update && apk add --no-cache curl
4
5
6

```

همچنین روش دوم ساخت ایمیج مورد نظر استفاده از Dockerfile است که در آن پکیج cURL نصب می‌شود.

```

~/Desktop/9931006_HW2/step-1 (0.602s)
docker build -t alpine-curl-dockerfile:1.0.0 .

[+] Building 0.1s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 107B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/alpine:3.17
=> [1/2] FROM docker.io/library/alpine:3.17
=> CACHED [2/2] RUN apk update && apk add --no-cache curl
=> exporting to image
=> => exporting layers
=> => writing image sha256:8599f7a9138b2361d613742da00989ba25c81b1ad184b1273abc12a717fd454
=> => naming to docker.io/library/alpine-curl-dockerfile:1.0.0

```

و سپس باید با استفاده از دستور Build یک ایمیج جدید برای آن می‌سازیم.

```
~/Desktop/9931006_HW2/step-1 (25.042s)
docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: feri80
Password:
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/
```

در مرحله بعدی باید در داکر هاب لاگین کنیم.

```
~/Desktop/9931006_HW2/step-1 (24.656s)
docker push feri80/alpine-curl-commit:1.0.0
The push refers to repository [docker.io/feri80/alpine-curl-commit]
0b2e35f93a70: Pushed
d2d9d24a8c2a: Mounted from feri80/alpine-curl-dockerfile
1.0.0: digest: sha256:ae009b04311f7cc7d6a679ef5c7c8aa7899725847708765540cf4bd9ce80ca39 size: 739

~/Desktop/9931006_HW2/step-1 (21.372s)
docker push feri80/alpine-curl-dockerfile:1.0.0
The push refers to repository [docker.io/feri80/alpine-curl-dockerfile]
6f5f26cd42c8: Layer already exists
d2d9d24a8c2a: Layer already exists
1.0.0: digest: sha256:8ac7a54df5d5313a0e62130f643ec2f22eebee164e4486d3cc3dd83c6b082785 size: 738
```

سپس ایمیج هایی که در مراحل قبلی ساختیم را روی داکر هاب آپلود می کنیم.

The screenshot shows the Docker Hub homepage. At the top, there is a search bar with the placeholder "Search Docker Hub" and a "Create repository" button. Below the search bar, there is a dropdown menu set to "feri80" and a "Search by repository name" input field. The main content area displays two repository cards:

- feri80 / alpine-curl-commit**
Contains: Image | Last pushed: a minute ago
Inactive • 0 stars • 0 forks • Public
- feri80 / alpine-curl-dockerfile**
Contains: Image | Last pushed: 2 minutes ago
Inactive • 0 stars • 0 forks • Public

همانطور که مشاهده می شود ایمیج های ما روی داکر هاب موجود هستند.

```
~/Desktop/9931006_HW2/step-1 (15.956s)
docker pull feri80/alpine-curl-commit:1.0.0
1.0.0: Pulling from feri80/alpine-curl-commit
4060ece20d7a: Already exists
2a06d02ef912: Already exists
Digest: sha256:ae009b04311f7cc7d6a679ef5c7c8aa7899725847708765540cf4bd9ce80ca39
Status: Downloaded newer image for feri80/alpine-curl-commit:1.0.0
docker.io/feri80/alpine-curl-commit:1.0.0
```

```
~/Desktop/9931006_HW2/step-1 (15.707s)
docker pull feri80/alpine-curl-dockerfile:1.0.0
1.0.0: Pulling from feri80/alpine-curl-dockerfile
4060ece20d7a: Already exists
98ba3b9b1542: Already exists
Digest: sha256:8ac7a54df5d5313a0e62130f643ec2f22eebee164e4486d3cc3dd83c6b082785
Status: Downloaded newer image for feri80/alpine-curl-dockerfile:1.0.0
docker.io/feri80/alpine-curl-dockerfile:1.0.0
```

حالا می‌توانیم ایمیج‌های خود را از داکر هاب دانلود کنیم.

```
~/Desktop/9931006_HW2/step-1
docker run -it feri80/alpine-curl-commit:1.0.0
/ # 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>
/ # █
```

```
~/Desktop/9931006_HW2/step-1
docker run -it feri80/alpine-curl-dockerfile:1.0.0
/ # 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>
/ # █
```

سپس می‌توانیم ایمیج‌های خود را اجرا کنیم و داخل آن‌ها از دستور **CURL** استفاده کنیم.

مرحله دوم

```
~/Desktop/9931006_HW2 (8.818s)
docker pull redis:7.2-alpine
7.2-alpine: Pulling from library/redis
Digest: sha256:2d44eb73209e5af4118de67e89b01709bfec7cc5a4c1cec29665f6d011c70783
Status: Image is up to date for redis:7.2-alpine
docker.io/library/redis:7.2-alpine
```

ابتدا ایمیج ردیس را از داکر هاب دانلود می کنیم.

```
~/Desktop/9931006_HW2/step-2 (0.685s)
docker run -d --name redis-test redis:7.2-alpine
33ae3b46eff69f24dcfb6d60e02d58cadea4f76ac785d8eef27f4e1bf38c6ac9
```

و سپس یک کانتیر از این ایمیج ایجاد می کنیم.

```
~/Desktop/9931006_HW2/step-2 (0.546s)
docker network create redis-temp
daf406467887f1fcc27804d525d7286a24b36d8d48cff2c655860f31c4989055
```

در مرحله بعد یک شبکه برای ردیس خود می سازیم.

```
~/Desktop/9931006_HW2/step-2 (11.527s)
docker build -t temp-app:1.0.0 .

[+] Building 10.7s (11/11) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> [internal] transferring dockerfile: 37B                                         0.0s
=> [internal] load .dockerignore                                                 0.0s
=> [internal] transferring context: 2B                                         0.0s
=> [internal] load metadata for docker.io/library/python:3.11-slim               7.2s
=> [auth] library/python:pull token for registry-1.docker.io                      0.0s
=> [1/5] FROM docker.io/library/python:3.11-slim@sha256:d36d3fb6c859768ec62ac36ddc7397b5331d8dc05bc8823b3cac24f6ade 0.0s
=> [internal] load build context                                                 1.3s
=> [internal] transferring context: 15.14MB                                     1.2s
=> CACHED [2/5] WORKDIR /app                                                 0.0s
=> CACHED [3/5] COPY requirements.txt /app/requirements.txt                         0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt                0.0s
=> [5/5] COPY . /app                                                       1.4s
=> exporting to image                                                        0.7s
=> => exporting layers                                                       0.7s
=> => writing image sha256:3fedee2a6bc006825fe0abd0fec27600959e164943cebc0ced29e46ebaa71ed2 0.0s
=> => naming to docker.io/library/temp-app:1.0.0                               0.0s
```

حال سرور خود را Build می‌کنیم برای این سرور از قبل یک Dockerfile نوشته‌ایم.

```
1 FROM python:3.11-slim
2
3 WORKDIR /app
4
5 COPY requirements.txt /app/requirements.txt
6
7 RUN pip install --no-cache-dir -r requirements.txt
8
9 COPY . /app
10
11 EXPOSE 5001
12
13 CMD ["python", "temperature.py"]
14
15
```

حال ایمیج ساخته شده خود را اجرا می‌کنیم.

```
~/Desktop/9931006_HW2/step-2 (2.879s)
docker run -d --name temp-app temp-app:1.0.0
db30816601311aff2e4e9ac49038f1a287e8cc9d6ee1e2945281a13b772ca4c5
```

```
~/Desktop/9931006_HW2/step-2 (0.415s)
docker ps
CONTAINER ID   IMAGE      COMMAND           CREATED        STATUS          PORTS     NAMES
db3081660131   temp-app:1.0.0   "python temperature..."   39 seconds ago   Up 36 seconds   5001/tcp   temp-app
33ae3b46eff6   redis:7.2-alpine "docker-entrypoint.s..."   9 minutes ago    Up 9 minutes   6379/tcp   redis-test

~/Desktop/9931006_HW2/step-2 (0.718s)
docker network connect redis-temp temp-app

~/Desktop/9931006_HW2/step-2 (0.454s)
docker network connect redis-temp redis-test
```

در مرحله بعد کانتینرهای Redis و سرور خود را به شبکه ساخته شده متصل می‌کنیم.

```
~/Desktop/9931006_HW2/step-2 (0.184s)
docker volume create redis-test-vol
redis-test-vol
```

برای نگهداری از داده‌های ردیس یک Volume برای آن می‌سازیم.

```
~/Desktop/9931006_HW2/step-2 (0.703s)
docker run -d --name redis-test -v redis-test-vol:/data redis:7.2-alpine
27de17c3c9430a5178367bb82a921b575838bb3fa9778edb34e28578e7024175
```

سپس می‌توانیم ردیس را با Volume ساخته شده اجرا کنیم.

	File: Dockerfile
1	<code>FROM python:3.11-slim</code>
2	<code>WORKDIR /app</code>
3	<code>COPY requirements.txt /app/requirements.txt</code>
4	
5	<code>RUN pip install --no-cache-dir -r requirements.txt</code>
6	
7	<code>COPY . /app</code>
8	
9	<code>EXPOSE 5001</code>
10	
11	<code>CMD ["python", "temperature.py"]</code>
12	
13	

```

~/Desktop/9931006_HW2/step-2 (3.809s)
docker build -t temp-app:1.0.0 .

[+] Building 2.9s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 37B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.11-slim
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load build context
=> => transferring context: 115.31kB
=> [1/5] FROM docker.io/library/python:3.11-slim@sha256:d36d3fb6c859768ec62ac36ddc7397b5331d8dc05bc8823b3cac24f6ade
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY requirements.txt /app/requirements.txt
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [5/5] COPY . /app
=> exporting to image
=> => exporting layers
=> => writing image sha256:3fedee2a6bc006825fe0abd0fec27600959e164943cebc0ced29e46ebaa71ed2
=> => naming to docker.io/library/temp-app:1.0.0

```

```

~/Desktop/9931006_HW2/step-2 (0.318s)
docker tag temp-app:1.0.0 feri80/temp-app:1.0.0

~/Desktop/9931006_HW2/step-2 (0.172s)
docker images

REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
temp-app            1.0.0    3fedee2a6bc0  17 minutes ago  191MB
feri80/temp-app     1.0.0    3fedee2a6bc0  17 minutes ago  191MB
step-2-app          latest   6d97f95fc1fe  9 hours ago   191MB
feri80/alpine-curl-dockerfile 1.0.0    8599ff7a9138b  42 hours ago  12.4MB
feri80/alpine-curl-commit 1.0.0    e7ce1679c744  42 hours ago  12.4MB
image-authenticator-cloud-computing-listener  latest   e2f4b5efb219  9 days ago   280MB
image-authenticator-cloud-computing-publisher  latest   bd0d8a922d0a  9 days ago   26.5MB
redis               7.2-alpine  9d4c89bd4fa6  10 days ago  39.2MB
python              3.11-alpine 143748e8d441  3 weeks ago  57.6MB
rabbitmq            3.12-alpine b35a2a591395  5 weeks ago  134MB
postgres            16        c2a319b219d0  6 weeks ago  438MB
moby/buildkit        buildx-stable-1 6b5b3e76f5a6  2 months ago  168MB
alpine              3.17     8650cd65339b  2 months ago  7.39MB
gradle              jdk17    ae8ad657951b  4 months ago  745MB

```

در این مرحله بر روی ایمیج ساخته شده خود تگ جدید می‌زنیم.

```

~/Desktop/9931006_HW2/step-2 (30.523s)
docker push feri80/temp-app:1.0.0

The push refers to repository [docker.io/feri80/temp-app]
27b1d9335dbc: Pushed
c8bc042c7f1b: Pushed
a08616619529: Pushed
1b3a540a0db5: Pushed
1d3c26097f6f: Mounted from library/python
e02fcddad509e: Mounted from library/python
4e7e2e312a26: Mounted from library/python
fd887e1d7390: Mounted from library/python
32f2ee38f285: Mounted from library/python
1.0.0: digest: sha256:4ec39a0c851cd59823dcdb58ab64427e422736cc2a5193333b84491ebf8b150a size: 2205

```

و سپس آن را در داکرهاپ آپلود می‌کنیم.

feri80 / temp-app
Contains: Image | Last pushed: a few seconds ago

Inactive ⭐ 0 0 Public

همانطور که مشاهده می‌شود اینجع در داکر هاب قرار دارد.

```
~/Desktop/9931006_HW2/step-2 (0.397s)
docker ps
CONTAINER ID   IMAGE      COMMAND           CREATED          STATUS          PORTS     NAMES
f12efb8f38a0   temp-app:1.0.0    "python temperature..."  3 seconds ago   Up 2 seconds   0.0.0.0:5001->5001/tcp   temp-a
pp
d1aa786f46ed   redis:7.2-alpine  "docker-entrypoint.s..."  13 seconds ago  Up 12 seconds  6379/tcp        redis-test
```

در اینجا هم می‌تواند کانتینرهای در حال اجرا بر روی سیستم را مشاهده کرد.

```
~/Desktop/9931006_HW2/step-2 (0.42s)
docker inspect temp-app:1.0.0
[{"Id": "sha256:3fedee2a6bc006825fe0abd0fec27600959e164943cebc0ced29e46ebaa71ed2", "RepoTags": ["feri80/temp-app:1.0.0", "temp-app:1.0.0"], "RepoDigests": ["feri80/temp-app@sha256:4ec39a0c851cd59823dcdb58ab64427e422736cc2a5193333b84491ebf8b150a"], "Parent": "", "Comment": "buildkit.dockerfile.v0", "Created": "2023-10-29T07:22:24.372367512Z", "Container": "", "ContainerConfig": {"Hostname": "", "Domainname": "", "User": "", "AttachStdin": false, "AttachStdout": false, "AttachStderr": false, "Tty": false, "OpenStdin": false, "StdinOnce": false, "Env": null, "Cmd": null, "Image": "", "Volumes": null, "WorkingDir": ""}}
```

```

~/Desktop/9931006_HW2/step-2 (0.42s)
docker inspect temp-app:1.0.0
  "Path": "/",
    "LowerDir": "/var/lib/docker/overlay2/vw8s4oyaiaiki32q2k8elocv1n/diff:/var/lib/docker/overlay2/tgz93wv0pkyutnbjmhpz9tz06/diff:/var/lib/docker/overlay2/uq7k4rnd3rjahlk8ld18mb9dw/diff:/var/lib/docker/overlay2/73f1203fefd213310f706819466dc97bbf7666d43157a7fb1dbdff2ffffc4b45/diff:/var/lib/docker/overlay2/2f5a4742add795680b98cfb575e7455daba129b39324aedba5aecc8dbc452b2/diff:/var/lib/docker/overlay2/892e76b1d785b99ea931c597c4e0a7b528b21a2d2bf72d553bf4e827f7635843/diff:/var/lib/docker/overlay2/19ef67d2af5df16a13901c9ccfc9cbb982a90bc0834393e8580845596896942e7/diff:/var/lib/docker/overlay2/85ec843583978754a3aff08510d0e44158e039e08ab540a8b56/diff",
      "MergedDir": "/var/lib/docker/overlay2/9vt1l91dyimxl87q2sp8t9yf/merged",
        "UpperDir": "/var/lib/docker/overlay2/9vt1l91dyimxl87q2sp8t9yf/diff",
          "WorkDir": "/var/lib/docker/overlay2/9vt1l91dyimxl87q2sp8t9yf/work"
        },
        "Name": "overlay2"
      },
      "RootFS": {
        "Type": "layers",
        "Layers": [
          "sha256:32f2ee38f285d7a349380070ea0dbfa284c9181e5e8b1a736f7a9eca726b1027",
          "sha256:fdb887e1d7390a04fb751b930aa20ce410b0c37444b2834e7e75d92aadc62a00c",
          "sha256:4e7e2e312a2658ef8c72417ce15986c0f7e6078f1fde4532f5b9099e97808704",
          "sha256:e02fcadad509e493eb6adfbe388b7e042275cad00a392375141791a066f4c4e",
          "sha256:1d3c26097f6f4ef46ff63de2b3e50f6e4d769e08a9c653b0c63729bfab1db457",
          "sha256:1b3a540a0db5a4fab37262be7d8577018bb370089ac46e6f0fb91278313422d",
          "sha256:a086166195295a6b5e8a3c079dfb80e3a45986c7b94eb7a3811096f56e38cce",
          "sha256:c8bc042c7f1b7b2af72fa49e4076b6fa5924ad324f54d068ebc698845fe9ca3c",
          "sha256:27b1d9335dbc7a488c34cc6967c996a224a19b6b9772b9c6efe41fc3eaa214a1"
        ]
      },
      "Metadata": {
        "LastTagTime": "2023-10-29T07:39:44.386218423Z"
      }
    }
  }
}

```

در مرحله بعد میتوانیم با استفاده از دستور **Inspect** اطلاعات ایمیج سرور خود را مشاهده کنیم.

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK I/O	PIDS
f12efb8f38a0	temp-app	0.74%	81.62MiB / 3.841GiB	2.08%	1.05kB / 604B	0B / 3.08MB	3
d1aa786f46ed	redis-test	0.47%	2.684MiB / 3.841GiB	0.07%	1.06kB / 0B	0B / 0B	5

در مرحله بعد با استفاده از دستور **Stats** میزان منابع استفاده شده توسط کانتینر های خود را میبینیم.

```

~/Desktop/9931006_HW2/step-2
docker-compose up --build

[+] Building 6.2s (11/11) FINISHED
  => [internal] load build definition from Dockerfile
  => => transferring dockerfile: 32B
  => [internal] load .dockerrcignore
  => => transferring context: 2B
  => [internal] load metadata for docker.io/library/python:3.11-slim
  => [auth] library/python:pull token for registry-1.docker.io
  => [1/5] FROM docker.io/library/python:3.11-slim@sha256:d36d3fb6c859768ec62ac36ddc7397b5331d8dc05bc8823b3cac24f6ade
  => [internal] load build context
  => => transferring context: 118.03kB
  => CACHED [2/5] WORKDIR /app
  => CACHED [3/5] COPY requirements.txt /app/requirements.txt
  => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
  => [5/5] COPY . /app
  => exporting to image
  => => exporting layers
  => => writing image sha256:a28f649d9f97031b2ee93d6f6a55c71005ce734ba52215d3cb9f463a58189bf3
  => => naming to docker.io/library/step-2-app
[+] Running 3/3
  ✓ Network step-2_redis-temp-network  Created
  ✓ Container temp-redis           Created
  ✓ Container temp-app            Created
Attaching to temp-app, temp-redis

```

```
1  version: '3'
2
3  services:
4
5    redis:
6      container_name: "temp-redis"
7      image: "redis:7.2-alpine"
8      ports:
9        - "6379:6379"
10     volumes:
11       - redis_data:/data
12     networks:
13       - redis-temp-network
14
15   app:
16     container_name: "temp-app"
17     build: .
18     ports:
19       - "5001:5001"
20     environment:
21       - API_KEY=waSUBz0YclQkXRErgqLM3g==CfH9igxWNSa3aoaY
22       - SERVER_PORT=5001
23       - REDIS_TTL=300
24       - REDIS_HOST=redis
25     networks:
26       - redis-temp-network
27     depends_on:
28       - redis
29
30   volumes:
31     redis_data:
32
33   networks:
34     redis-temp-network:
35       driver: bridge
:
```

در مرحله بعد با استفاده از داکر کامپوز برنامه خود را اجرا می‌کنیم.

```
~/Desktop/9931006_HW2/step-2 (0.151s)
curl localhost:5001/temp

{
  "error": "City Is Missing"
}
```

```
~/Desktop/9931006_HW2/step-2 (2.094s)
curl "localhost:5001/temp?city=tehran"

{
  "city": "tehran",
  "max_temperature": "24",
  "min_temperature": "22"
}
```

```
~/Desktop/9931006_HW2/step-2 (1.305s)
curl "localhost:5001/temp?city=paris"

{
  "city": "paris",
  "max_temperature": "13",
  "min_temperature": "12"
}
```

در انتها برنامه خود را تست می‌کنیم.

بخش دوم

ابتدا Minikube را نصب می‌کنیم و با استفاده از درایور Docker آن را اجرا می‌کنیم.

```
~ (32.781s)
minikube start docker
享誉 minikube v1.32.0 on Darwin 14.0 (arm64)
NEW Kubernetes 1.28.3 is now available. If you would like to upgrade, specify: --kubernetes-version=v1.28.3
+ Using the docker driver based on existing profile
👍 Starting control plane node minikube in cluster minikube
👉 Pulling base image ...
🏃 Updating the running docker "minikube" container ...
❗ Image was not built for the current minikube version. To resolve this you can delete and recreate your minikube cluster
using the latest images. Expected minikube version: v1.31.0 -> Actual minikube version: v1.32.0
🌐 Preparing Kubernetes v1.27.4 on Docker 24.0.4 ...
🌐 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
```

```
~ (0.16s)
k describe namespace default

Name:           default
Labels:         kubernetes.io/metadata.name=default
Annotations:   <none>
Status:        Active

No resource quota.

No LimitRange resource.
```

همانطور که مشاهده می‌شود Minikube اجرا شده و می‌توانیم Namespace Default را در آن مشاهده کنیم.

```
~/Desktop/9931006_HW2/part-2 (0.251s)
k get pods

NAME                      READY   STATUS    RESTARTS   AGE
redis-deployment-7f758db768-q4g5g   1/1     Running   0          8h
temp-deployment-7b85f845cc-n5dpx   1/1     Running   0          7h58m
temp-deployment-7b85f845cc-pdj7z   1/1     Running   0          7h58m
temp-deployment-7b85f845cc-spbxm  1/1     Running   0          7h46m
```

در اینجا می‌توانیم پادهای ساخته شده را ببینیم که ۳ تای آن متعلق به سرور برنامه و یکی از آن‌ها مربوط به ردیس است.

نام‌های پادها بر اساس نام دیپلولیمنت در **Metadata** به اضافه یک پسوند انتخاب می‌شوند.

```
~/Desktop/9931006_HW2/part-2 (0.136s)
k describe pod temp-deployment-7b85f845cc-n5dpx

Name:          temp-deployment-7b85f845cc-n5dp
Namespace:     default
Priority:      0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Thu, 16 Nov 2023 13:28:47 +0330
Labels:         app=temp
                pod-template-hash=7b85f845cc
Annotations:   <none>
Status:        Running
IP:            10.244.0.7
IPs:
  IP:          10.244.0.7
Controlled By: ReplicaSet/temp-deployment-7b85f845cc
```

```
~/Desktop/9931006_HW2/part-2 (0.152s)
k describe pod temp-deployment-7b85f845cc-pdj7z

Name:          temp-deployment-7b85f845cc-pdj7z
Namespace:     default
Priority:      0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Thu, 16 Nov 2023 13:28:47 +0330
Labels:         app=temp
                pod-template-hash=7b85f845cc
Annotations:   <none>
Status:        Running
IP:            10.244.0.8
IPs:
  IP:          10.244.0.8
Controlled By: ReplicaSet/temp-deployment-7b85f845cc
```

همانطور که می‌بینیم IP پادهای مختلف یک دیپلویمنت متفاوت است و با حذف شدن یک پاد و جایگزینی آن هم این IP ها ثابت نیستند. در نتیجه باید از منبع Service استفاده کنیم.

با اینکه در کوبرنتیز چند نوع سرویس وجود دارد استفاده از نوع ClusterIP برای ما کافی است. با تعریف این سرویس به ما یک IP ثابت داده می‌شود که می‌توانیم از آن در برنامه‌های خود استفاده کنیم. همچنین ClusterIP نوع پیشفرض سرویس است پس اگر هنگام تعریف سرویس نوع آن را مشخص نکنیم نوع به صورت خودکار ClusterIP خواهد بود.

```
~/Desktop/9931006_HW2/part-2 (0.15s)
k describe svc temp-service

Name:           temp-service
Namespace:      default
Labels:          <none>
Annotations:    <none>
Selector:        app=temp
Type:            ClusterIP
IP Family Policy: SingleStack
IP Families:    IPv4
IP:              10.109.95.239
IPs:             10.109.95.239
Port:            <unset>  5001/TCP
TargetPort:      5001/TCP
Endpoints:       10.244.0.7:5001,10.244.0.8:5001,10.244.0.9:5001
Session Affinity: None
Events:          <none>
```

در اینجا می‌تواند Port و TargetPort سرویس را هم مشاهده کرد که 5001 است.

حال ردیس خود را بر روی کلاستر بالا می‌آوریم. در حالت کلی اطلاعات یک پاد روی فضای ذخیره می‌شوند که Ephemeral Storage نام دارد همانطور که از نام آن پیداست این فضا میرا بوده با و از بین رفقن پاد اطلاعات نیز پاک می‌شوند برای اینکه اطلاعات پاک نشوند باید از Persistant PVC این قابلیت را به ما می‌دهد تا اطلاعات خود را بر روی دیسک ذخیره Volume کنیم. تا از پاک شدن آن جلوگیری شود.

```
~/Desktop/9931006_HW2/part-2 (0.168s)
k describe pvredis-pvcvc

Name:          redis-pvc
Namespace:     default
StorageClass:  standard
Status:        Bound
Volume:        pvc-127c7518-de0e-4f59-b4f1-a629c7f9c521
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
                pv.kubernetes.io/bound-by-controller: yes
                volume.beta.kubernetes.io/storage-provisioner: k8s.io/minikube-hostpath
                volume.kubernetes.io/storage-provisioner: k8s.io/minikube-hostpath
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      100Mi
Access Modes:  RWO
VolumeMode:    Filesystem
Used By:       redis-deployment-7f758db768-q4g5g
Events:        <none>
```

همانطور که مشاهده می‌شود کافیست که PVC مورد نظر را بنویسیم و آن را اپلای کنیم.

```
16   spec:
17     containers:
18       - name: redis
19         image: redis:7.2-alpine
20       ports:
21         - containerPort: 6379
22       volumeMounts:
23         - mountPath: /data
24           name: redis-data
25       resources:
26         limits:
27           cpu: 200m
28           memory: 128Mi
29         requests:
30           cpu: 200m
31           memory: 128Mi
32       volumes:
33         - name: redis-data
34           persistentVolumeClaim:
35             claimName: redis-pvc
36
```

سپس از آن داخل دیپلولیمنت خود استفاده می‌کنیم.

```

~/Desktop/9931006_HW2/part-2 (0.131s)
k describe pv pvc-127c7518-de0e-4f59-b4f1-a629c7f9c521

Name:           pvc-127c7518-de0e-4f59-b4f1-a629c7f9c521
Labels:         <none>
Annotations:   hostPathProvisionerIdentity: 6f3d4605-4fc8-455e-a9bf-99b38f866aa9
                pv.kubernetes.io/provisioned-by: k8s.io/minikube-hostpath
Finalizers:    [kubernetes.io/pv-protection]
StorageClass:  standard
Status:        Bound
Claim:         default/redis-pvc
Reclaim Policy: Delete
Access Modes:  RWO
VolumeMode:   Filesystem
Capacity:     100Mi
Node Affinity: <none>
Message:
Source:
  Type:       HostPath (bare host directory volume)
  Path:      /tmp/hostpath-provisioner/default/redis-pvc
  HostPathType:
Events:        <none>

```

همانطور که می‌بینیم PV مد نظر نیز وجود دارد و اطلاعات با از بین رفتن پاد حفظ خواهد شد.
در این مثال همانطور که مشاهده شد هنگامی که ما یک PVC را می‌سازیم Mimikube به صورت خودکار PV متناظر با آن را ایجاد می‌کند و به PVC متصل می‌کند در نتیجه نیازی نیست که به صورت دستی PV تعریف این نشان می‌دهد که Minikube به صورت پیشفرض از قابلیت Dynamic Provisioning پشتیبانی می‌کند و فعال می‌کند.

در کوبرنتیز، یک PV به طور معمول به تنها یک PVC متصل می‌شود. این بدان معناست که یک PV نمی‌تواند به طور همزمان به چندین PVC اختصاص یابد. این رفتار بر اساس مدل طراحی کوبرنتیز است که در آن هر PVC به طور مستقل به یک PV اختصاص می‌یابد.

جلوگیری از تداخل داده‌ها و اطمینان از ایمنی: با این روش، اطمینان حاصل می‌شود که هر برنامه یا سرویس فقط به فضای ذخیره‌سازی که به آن اختصاص داده شده دسترسی دارد. این امر از تداخل داده‌ها جلوگیری می‌کند و امنیت داده‌ها را در محیط‌های چندمنظوره تقویت می‌کند.

```
~/Desktop/9931006_HW2/part-2
k run curl-temp --image=feri80/alpine-curl-dockerfile:1.0.0 -it -- sh
If you don't see a command prompt, try pressing enter.
/ # curl "10.109.95.239:5001/temp?city=mashhad"
{
  "city": "mashhad",
  "max_temperature": "20",
  "min_temperature": "20"
}
/ # curl "10.109.95.239:5001/temp?city=tehran"
{
  "city": "tehran",
  "max_temperature": "16",
  "min_temperature": "15"
}
/ # curl "10.109.95.239:5001/temp?city=tehran"
{
  "city": "tehran",
  "max_temperature": "16",
  "min_temperature": "15"
}
/ # █
```

در مرحله بعدی با از استفاده از دستور Run یک پاد از ایمیج cURL را که ساخته بودیم اجرا می‌کنیم و در آن به IP سرویس خود درخواست می‌زنیم و جواب را دریافت می‌کنیم.

```
~/Desktop/9931006_HW2/part-2
k port-forward svc/temp-service 5001:5001
```

```
Forwarding from 127.0.0.1:5001 -> 5001
```

```
Forwarding from [::1]:5001 -> 5001
```

```
Handling connection for 5001
```

```
Handling connection for 5001
```

```
Handling connection for 5001
```

```
█
```

روش دیگر این است که با استفاده از دستور port-forward این سرویس یا یک پاد را Port Forward کنیم و به این شکل می‌توانیم از داخل سیستم خود درخواست بزنیم.

```
~/Desktop/9931006_HW2/part-2 (0.159s)
curl "127.0.0.1:5001/temp?city=mashhad"
```

```
{
  "city": "mashhad",
  "max_temperature": "20",
  "min_temperature": "20"
}
```

```
~/Desktop/9931006_HW2/part-2 (0.098s)
curl "127.0.0.1:5001/temp?city=tehran"
```

```
{
  "city": "tehran",
  "max_temperature": "16",
  "min_temperature": "15"
}
```

```
~/Desktop/9931006_HW2/part-2 (2.902s)
curl "127.0.0.1:5001/temp?city=paris"
```

```
{
  "city": "paris",
  "max_temperature": "9",
  "min_temperature": "8"
}
```

برای تست صحت سیستم و اینکه از درخواست ما به همه پادها داده می‌شود تعداد زیادی درخواست به سرویس می‌زنیم و لاگ هر سه پاد سرور را بررسی می‌کنیم.

```
~/Desktop/9931006_HW2/part-2 (0.189s)
k logs temp-deployment-b57dcb495-2fft6
/usr/local/lib/python3.11/site-packages/redis/connection.py:77: UserWarning: redis-py works best
der installing
    warnings.warn(msg)
    * Serving Flask app 'temperature'
    * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment.
    instead.
    * Running on all addresses (0.0.0.0)
    * Running on http://127.0.0.1:5001
    * Running on http://10.244.0.13:5001
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
/usr/local/lib/python3.11/site-packages/redis/connection.py:77: UserWarning: redis-py works best
der installing
    warnings.warn(msg)
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 503-015-277
INFO:__main__:Temperature Data Retrieved from Ninjas City: paris
INFO:__main__:('7', '9')
INFO:werkzeug:10.244.0.16 - - [16/Nov/2023 19:31:50] "GET /temp?city=paris HTTP/1.1" 200 -
INFO:__main__:Temperature Data Retrieved from Redis City: paris
INFO:__main__:('7', '9')
INFO:werkzeug:10.244.0.16 - - [16/Nov/2023 19:31:52] "GET /temp?city=paris HTTP/1.1" 200 -
INFO:__main__:Temperature Data Retrieved from Redis City: paris
INFO:__main__:('7', '9')
```

```
~/Desktop/9931006_HW2/part-2 (0.17s)
k logs temp-deployment-b57dcb495-h7mfs
/usr/local/lib/python3.11/site-packages/redis/connection.py:77: UserWarning: redis-py works best
der installing
    warnings.warn(msg)
    * Serving Flask app 'temperature'
    * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. U
    instead.
    * Running on all addresses (0.0.0.0)
    * Running on http://127.0.0.1:5001
    * Running on http://10.244.0.14:5001
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
/usr/local/lib/python3.11/site-packages/redis/connection.py:77: UserWarning: redis-py works best
der installing
    warnings.warn(msg)
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 140-048-931
INFO:__main__:Temperature Data Retrieved from Redis City: paris
INFO:__main__:('7', '9')
INFO:werkzeug:10.244.0.16 - - [16/Nov/2023 19:31:51] "GET /temp?city=paris HTTP/1.1" 200 -
INFO:__main__:Temperature Data Retrieved from Redis City: paris
INFO:__main__:('7', '9')
```

```
~/Desktop/9931006_HW2/part-2 (0.15s)
k logs temp-deployment-b57dcb495-rd5x6
/usr/local/lib/python3.11/site-packages/redis/connection.py:77: UserWarning: redis-py works best
der installing
    warnings.warn(msg)
    * Serving Flask app 'temperature'
    * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. U
instead.
    * Running on all addresses (0.0.0.0)
    * Running on http://127.0.0.1:5001
    * Running on http://10.244.0.15:5001
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with stat
/usr/local/lib/python3.11/site-packages/redis/connection.py:77: UserWarning: redis-py works best
der installing
    warnings.warn(msg)
WARNING:werkzeug: * Debugger is active!
INFO:werkzeug: * Debugger PIN: 878-692-954
INFO:__main__:Temperature Data Retrieved from Redis City: paris
INFO:__main__:('7', '9')
INFO:werkzeug:10.244.0.16 - - [16/Nov/2023 19:32:27] "GET /temp?city=paris HTTP/1.1" 200 -
```

همانطور که مشاهده می‌شود درخواست به همه پادهای سرور ما خورده است.
هنگامی که برای یک شهر را برای اولین بار درخواست می‌دهیم به وضوح مقدار بیشتری نسبت به بار دوم طول می‌کشد تا سرور پاسخ دهد این نشان دهنده عملکرد درست کش ردیس در سیستم ما است.