



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر

عنوان

نگارش
بردیا اردکانیان

استاد
دکتر احمد جوادی

۱۴۰۱ آذر

گام اول

در گام نخست می‌بایست عملیات ذیل را انجام دهیم.

- ارسال ایمیج ساخته شده بر روی داکرها و نتیجه آن.
- نمایش لیست ایمیج‌های موجود بر روی سیستم خود.
- دریافت ایمیج ساخته شده از داکرها.
- ساختن کانتینر از ایمیج دریافت شده از داکرها.
- اجرا دستور curl و نتیجه آن.

برای شروع ابتدا پس از نصب و راه اندازی داکر با کمک دستور زیر image لینوکس alpine را دریافت می‌کنیم (اصطلاحا pull می‌کنیم). (عکس ۱)

```

Eververse — bardia@Bardias-MacBook-Pro — ..HW2/Eververse — zsh — 80x24
~/Doc/au/C/CE4/a/HW2/Eververse > docker run alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
261da4162673: Already exists
Digest: sha256:8914eb54f968791faf6a8638949e480fef81e697984fba772b3976835194c6d4
Status: Downloaded newer image for alpine:latest
[~/Doc/au/C/CE4/a/HW2/Eververse > docker image ls
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
eververse        latest       b3b661433fd6   9 hours ago   442MB
ubuntu           latest       4c2c87c6c36e   27 hours ago  69.2MB
redis            latest       9783be5021b5   3 days ago    111MB
alpine           latest       d3156fec8bcb   2 weeks ago   7.46MB

```

عکس ۱-۱

در مرحله بعدی یک Dockerfile می‌نویسیم تا curl را دریافت کند. (عکس ۲)

```

File: Dockerfile
FROM alpine:latest
RUN apk add --no-cache curl
ENTRYPOINT ["/usr/bin/curl"]

```

عکس ۲-۱

حالا باید با دستور docker build یک image بسازیم تا لینوکس مد نظر ساخته شود. (عکس ۳)

در مرحله بعدی می‌بایست یک مخزن در داکرها درست کنیم، مخزن درست شده rossseta نام دارد (عکس ۴-۴). به منظور اپلود کردن image داکرها می‌بایست تگ bardiaardakanian/rossseta image را به docker push تغییر دهیم و بعد با دستور bardiaardakanian/rossseta آن را در داکرها پارگزاری کنیم (عکس ۵-۵).

```

Alpinec — bardia@Bardias-MacBook-Pro — ..ments/Alpinec — -zsh — 80x24
~/Doc/au/C/CE4/a/Alpinec > docker build -t alpine .
[+] Building 15.5s (6/6) FINISHED
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 219B                         0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 2B                           0.0s
=> [internal] load metadata for docker.io/library/alpine:latest 1.1s
=> CACHED [1/2] FROM docker.io/library/alpine:latest@sha256:8914eb54f968 0.0s
=> [2/2] RUN apk add --no-cache curl                      14.3s
=> exporting to image                                     0.0s
=> => exporting layers                                    0.0s
=> => writing image sha256:9ea689d29950e761c36c59efd5845af90972d6469aacc 0.0s
=> => naming to docker.io/library/alpine                  0.0s

```

عکس ۱

bardiaardakanian / rosseta

Description
Docker repository for education. [Edit](#)

⌚ Last pushed: a few seconds ago

Tags and scans
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
latest	🐧	Image	--	a few seconds ago

[See all](#) [Go to Advanced Image Management](#)

VULNERABILITY SCANNING - DISABLED [Enable](#)

Docker commands
To push a new tag to this repository.

```
docker push bardiaardakanian/rosseta:tagname
```

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.

[Upgrade](#) [Learn more](#)

README [Edit](#)
Repository description is empty. Click [here](#) to edit.

عکس ۲

```
~/Doc/au/C/CE4/a/HW2/Eververse > docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
eververse latest b3b661433fd6 10 hours ago 442MB
ubuntu latest 4c2c87c6c36e 28 hours ago 69.2MB
redis latest 9783be5021b5 3 days ago 111MB
alpine latest d3156fec8bcb 2 weeks ago 7.46MB
~/Doc/au/C/CE4/a/HW2/Eververse > docker tag alpine bardiaardakanian/rosseta
~/Doc/au/C/CE4/a/HW2/Eververse > docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
eververse latest b3b661433fd6 10 hours ago 442MB
ubuntu latest 4c2c87c6c36e 28 hours ago 69.2MB
redis latest 9783be5021b5 3 days ago 111MB
bardiaardakanian/rosseta latest d3156fec8bcb 2 weeks ago 7.46MB
alpine latest d3156fec8bcb 2 weeks ago 7.46MB
~/Doc/au/C/CE4/a/HW2/Eververse > docker push bardiaardakanian/rosseta
Using default tag: latest
The push refers to repository [docker.io/bardiaardakanian/rosseta]
1b577a8fb8ce: Layer already exists
latest: digest: sha256:af06af3514c44a964d3b905b498cf6493db8f1cde7c10e078213a89c87308ba0 size: 528
```

عکس ۱-۶

به منظور تست کردن ابتدا image فعلی را پاک می‌کنیم و بعد را از داکرهاب بارگیری می‌کنیم (عکس ۱-۶). در نهایت image را اجرا می‌کنیم و با کمک curl که دانلود شده است به گوگل درخواستی را ارسال می‌کنیم (عکس ۱-۷).

```
~/Doc/au/C/CE4/a/HW2/Eververse > docker rm alpine:bardiaardakanian/rosseta
Error: No such container: alpine:bardiaardakanian/rosseta
~/Doc/au/C/CE4/a/HW2/Eververse > docker pull bardiaardakanian/rosseta
Using default tag: latest
latest: Pulling from bardiaardakanian/rosseta
Digest: sha256:af06af3514c44a964d3b905b498cf6493db8f1cde7c10e078213a89c87308ba0
Status: Image is up to date for bardiaardakanian/rosseta:latest
docker.io/bardiaardakanian/rosseta:latest
~/Doc/au/C/CE4/a/HW2/Eververse > docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
eververse latest b3b661433fd6 10 hours ago 442MB
ubuntu latest 4c2c87c6c36e 28 hours ago 69.2MB
redis latest 9783be5021b5 3 days ago 111MB
bardiaardakanian/rosseta latest d3156fec8bcb 2 weeks ago 7.46MB
alpine latest d3156fec8bcb 2 weeks ago 7.46MB
```

عکس ۱-۷

```
Alpinec — bardia@Bardias-MacBook-Pro ..ments/Alpinec — zsh — 124x30
~/Doc/au/C/CE4/a/Alpinec > docker run --rm bardiaardakanian/rosseta https://www.google.com
% Total    % Received % Xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload   Total   Spent    Left  Speed
 0     0    0     0     0     0      0 --::---- --::---- --::--- 0<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en-IR"><head><meta content="Search the world's information, including webpages, images, video and more. Google has many special features to help you find exactly what you're looking for." name="description"><meta content="noodp" name="robots"><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleleg/1x/googleleg_standard_color_128dp.png" itemprop="image"><title>Google</title><script nonce="JnMqzmHVeSp9LpmqGwahZA">(function(){window.google={kEI:'bzWUY5SUJYy5AsE9Yi4DQ',kEXPI:'0,1359409,6059,206,4804,2316,383,246,5,1129120,1197710,691,380089,16115,28684,22431,1361,284,12032,2819,1930,12834,4998,13228,1838,2009,10622,22741,5081,1593,1279,2742,149,1103,842,2195,4100,3514,606,2023,2297,14670,3227,2845,7,29075,4143,552,1851,2614,13142,3,346,230,1014,1,5445,148,11323,2652,4,1528,2304,27348,7422,7357,13658,21223,5809,2548,4094,4052,3,3541,1,42154,2,16737,3533,19491,5679,1021,2380,28741,4568,6256,17113,6308,1252,5835,14967,4333,7484,445,2,2,1,6960,17666,2006,8155,6680,701,2,15967,874,19633,7,1922,5784,25774,9543,4832,23189,3314,14572,1722,3843,14,82,3890,751,14144,739,3,679,1341,281,1779,668,1186,1997,1125,1581,6549,779,89,243,458,1505,1284,2412,1742,813,3860,936,280,285,2,563,991,2095,170,270,398,97,427,937,138,291,3392,144,12,769,1431,3,260,524,3,239,766,420,916,289,135,963,235,106,246,930,1,863,599,777,947,266,591,320,506,159,782,31,1638,228,79,3,48,3,391,96,593,2,342,213,157,327,773,95,65,307,576,394,497,427,1415,5,1519,130,108,139,324,218,84,160,68,124,10,162,271,14,3,144,27,339,769,238,471,57,282,258,591,60,1,506,59,838,105,1121,263,5,3,744,33,1352,259,213,37,136,326,77,331,382,699,290,2,1129,39,357,1431,939,260,281,973,725,233,3,4,1166,202,327,2,369,5277053,4219,5995616,2803414,3311,141,795,19735,1,1,346,1755,1004,41,403,1,3,2,123,27,6,3,3,4,1,3,2,1,2,2,3,1,2,5,2,1,2,2,3,2,23947076,511,19,4041612,1964,14297,2376,3101,304,5595,11,3835,1922,2668,541,1766,1520751',kBL:'oABN'};google.sn='webhp';google.kHL='en-IR'}))();(function(){
var f=this||self;var h,k=[];function l(a){for(var b;a&&(a.getAttribute||!(b=a.getAttribute("eid"))));a=a.parentNode;return b||h}function m(a){for(var b=null;a&&(a.getAttribute||!(b=a.getAttribute("leid"))));a=a.parentNode;return b}
function n(a,b,c,d,g){var e="";c||-1!=b.search("=&ei=")||("e=&ei="+l(d),-1==b.search("=&lei=")||!(d=m(d))&&(e+="&lei="+d));d="";!c&&f._cshid&&-1==b.search("=&cshid=")&&"slh"!=a&&(d="=&cshid="+f._cshid);c=c||"/"+(g||"gen_204")?"#":?apt=i&ct="+"+a+"&cad=" +b+e+"&zxx=" +Date.now() +d;/^http://i.test(c)&&"https://" == window.location.protocol&&(google.ml&&google.ml(Error("a")),!1,{src:c,g1mm:1}),c="";return c};h=google.kEI;google.getFI=l;google.getLEI=m;google.ml=function(){return null};google.log=function(a,b,c,d,g){if(c=n(a,b,c,d,g)){a=new Image;var e=k.length;k[e]=a;a.onerror=a.onload=a.onabort=function(){delete k[e]};a.src=c}};google.logUrl=n});call(this);(function(){google.y={};google.sy=[];google.x=function(a,b){if(a)var c=a.id;else do c=Math.random();while(google.y[c])google.y[c]=[a,b];return!1};google.sx=function(a){google.sy.push(a)};google.lm=[];google.plm=func
```

عکس ۱-۷

همانطور که مشاهده می‌کنید بعد از ران شدن curl دستور alpine image به گوگل فرستاده شده و پاسخ فرستاده شده است.

گام دوم

برای پیاده‌سازی بک‌اند پروژه از زبان گولنگ استفاده شده است. همچنین در Dockerfile چندین کامند جدید نوشته شده است تا تمامی فایل‌های go در مسیر مربوطه کپی و وابستگی‌های پروژه دانلود شود (عکس ۱-۲). برای استفاده از ردیس به عنوان کش در برنامه از docker-compose.yml و سرویس ردیس در کانتینر ردیس قرار داده شود (عکس ۲-۲).

```

1  # syntax=docker/dockerfile:1
2
3  ## Build
4 >> FROM golang:1.19-alpine
5
6  WORKDIR /app
7
8  COPY main/go.mod .
9  COPY main/go.sum .
10 RUN go mod download && go mod verify
11
12 COPY main/*.go .
13 COPY .env .
14
15 RUN go build -o /eververse
16
17 CMD [ "/eververse"]

```

عکس ۱-۲

```

1   version: '3.8'
2
3 >> services:
4 >>   cache:
5     container_name: redis
6     image: redis:alpine
7     restart: always
8     ports:
9       - ${REDIS_MAP_PORT:-6379} :${REDIS_EXPOSE_PORT:-6379}
10    volumes:
11      - cache:/data
12    networks:
13      - eververse-local
14
15 >>   web:
16     container_name: web
17     build:
18       context: .
19       dockerfile: Dockerfile
20     ports:
21       - ${MAP_PORT:-80} :${EXPOSE_PORT:-1323}
22     networks:
23       - eververse-local
24
25   networks:
26     eververse-local:
27       external: true
28
29   volumes:
30     cache:
31       driver: local

```

عکس ۲-۲

همچنین از فایل env برای کانفیگ کردن پورت مب سرور، پورت مب ردیس، پورت EXPOSE سرور، پورت مب ردیس، کلید CoinApi.io و مدت زمان کش استفاده شده است. تمامی این موارد به راحتی کانفیگ پذیر هستند (عکس ۳-۲).

```

1 TAG=v1.0
2 MAP_PORT=80
3 EXPOSE_PORT=1323
4 REDIS_MAP_PORT=6379
5 REDIS_EXPOSE_PORT=6379
6 COIN_API_KEY=9900DC68-EB5F-4FF5-8820-F8345AC607F1
7 CACHE_TIME=5

```

عکس ۳-۲

در سرور نوشته شده سرویسی پیاده سازی شده است که با کمک پارامترهای ورودی درخواست، قیمت رمزارز درخواست داده شده را برمی‌گرداند. این سرویس بعد بالا اوردن داکر با کمک دستور زیر، می‌توان به localhost/get?name=<coin_name> درخواست داد و قیمت رمزارز خواسته شده را دریافت کرد. این سرویس با کمک ردیس به مدت پنج دقیقه به صورت پیش‌فرز قیمت رمز ارز را در ردیس کش می‌کند و بعد آن درخواستی به سایت CoinApi.io می‌فرستد.

```
docker-compose up -d -build -force-recreate
```

در فایل docker-compose.yml یک volume برای ردیس تعریف شده است که با پایین آمدن داکر اطلاعاتش حفظ شود و از بین نرود. همچنین بین کانتینر وب و ردیس از طریق شبکه تعريف شده eververse-local شبکه‌ای ساخته شده تا با یکدیگر صحبت کنند (عکس ۴-۲).

```

networks:
  eververse-local:
    external: true

volumes:
  cache:
    driver: local

```

عکس ۴-۲

موارد ذکر شده در فایل پروژه:

دربافت image و ساختن کانتینر:

با دستور docker run به دنبال image ردیس می‌گردد و در صورتی که به صورت لوکال وجود نداشته باشد آن را دانلود می‌کند (عکس ۵-۵). ساخت شبکه برای برقرار ارتباط بین دو کانتینر:

با دستور docker network create eververse-local یک شبکه ایجاد می‌کنیم (عکس ۶-۶).

```

CE422-CC-Eververse — docker run -it redis — docker — com.docker.cli < docker run -it redis — 102x31
~/Doc/au/C/CE4/assignments/CE422-CC-Eververse ➜ 🐄 main ?1 ➜ base ✘

> docker run -it redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
6064e7e5b6af: Already exists
f6bd55a0e6ff: Pull complete
7525aa3c12b6: Pull complete
e5e9839bbc03: Pull complete
e0f57d95d120: Pull complete
79307332ce5c: Pull complete
Digest: sha256:4970a4bb34f9072b56389e85185204dd07dc86ba74a1be441931d551f74b472
Status: Downloaded newer image for redis:latest
1:C 13 Dec 2022 11:34:24.534 # o000o000o000o Redis is starting o000o000o000o
1:C 13 Dec 2022 11:34:24.534 # Redis version=7.0.6, bits=64, commit=00000000, modified=0, pid=1, just
started
1:C 13 Dec 2022 11:34:24.534 # Warning: no config file specified, using the default config. In order t
o specify a config file use redis-server /path/to/redis.conf
1:M 13 Dec 2022 11:34:24.534 * monotonic clock: POSIX clock_gettime

                                     Redis 7.0.6 (00000000/0) 64 bit
                                     Running in standalone mode
                                     Port: 6379
                                     PID: 1
                                     https://redis.io

```

عکس ۲

```

Bardia — bardia@Bardias-MacBook-Pro — ~ — zsh — 80x24
> docker network create eververse-local
Error response from daemon: network with name eververse-local already exists
> docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
2fc8b92f5205    bridge    bridge      local
8d01eb081aed    eververse-local    bridge      local
198660ef49d0    host      host       local
ef7703a481e9    local-dev    bridge      local
7524069e6275    none      null       local

```

عکس ۶

ساخت Persist جهت Volume کردن کش دیتابیس:

در عکس ۴-۲ نمایش داده شده است.

ساختن اپمیج سرور نوشته شده با استفاده از داکرفایل:

با کمک دستور docker run و docker build سرور نوشته شده را می‌سازیم و اجرا می‌کنیم (عکس ۲-۷).

```
CE422-CC-Eververse — docker run -it eververse — docker — com.docker.cli • docker run -it eververse — 127x37
base •
~/Documents/aut/Courses/CE422-CC/assignments/CE422-CC-Eververse ➜ p main ?1
❯ docker build -t eververse .
[+] Building 2.0s (17/17) FINISHED
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 74B 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> resolve image config for docker.io/docker/dockerfile:1 0.0s
=> CACHED docker-image://docker.io/docker/dockerfile:1@sha256:9ba7531bd80fb0a858632727cf7a112fbfd19b17e94c4e84ced81e24e 0.0s
=> [internal] load build definition from Dockerfile 0.0s
=> [internal] load .dockerignore 0.0s
=> [internal] load metadata for docker.io/library/golang:1.19-alpine 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 400B 0.0s
=> [1/8] FROM docker.io/library/golang:1.19-alpine@sha256:a9b24b67dc83b3383d22a14941c2b2b2ca6a103d805cac6820fd1355943be 0.0s
=> CACHED [2/8] WORKDIR /app 0.0s
=> CACHED [3/8] COPY main/go.mod ./ 0.0s
=> CACHED [4/8] COPY main/go.sum ./ 0.0s
=> CACHED [5/8] RUN go mod download && go mod verify 0.0s
=> CACHED [6/8] COPY main/*.go ./ 0.0s
=> CACHED [7/8] COPY .env ./ 0.0s
=> CACHED [8/8] RUN go build -o /eververse 0.0s
=> exporting to image 0.0s
=> => exporting layers 0.0s
=> => writing image sha256:1702cf9c280f87b5bab7fae516f703ee5ba97e1c9bfa20bb62d11ee86a7c9646 0.0s
=> => naming to docker.io/library/eververse 0.0s
base •
~/Documents/aut/Courses/CE422-CC/assignments/CE422-CC-Eververse ➜ p main ?1
❯ docker run -it eververse
2022/12/13 11:51:11 - [ENV] EXPOSE_PORT:1323
2022/12/13 11:51:11 - [ENV] REDIS_MAP_PORT:6379

/_/ _/ _/
/_/ _/ _\ _\ _\ v4.9.0
High performance, minimalist Go web framework
https://echo.labstack.com
```

٧-٢ عکس

ارسال اینمیج ساخته شده بر روی داکرهای و نمایش نتیجه آن:

با کمک دستور `tg`, تگ `image` ساخته شده را عوض می‌کنیم و بعد در داکرهاب `push` می‌کنیم (عکس ۲-۸). نتیجه را در داکرهاب می‌توانیم مشاهده کنیم (عکس ۲-۹).

`docker inspect` نمایش اطلاعات اینجی سرور خود را استفاده از دسته.

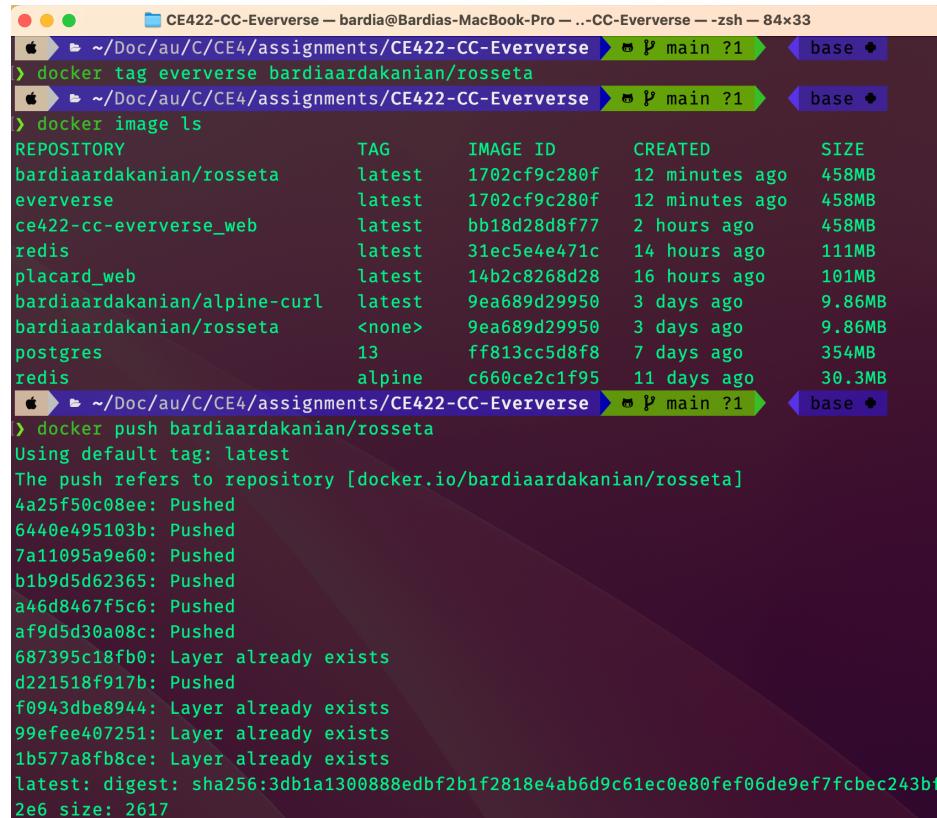
اب: دستور metadata می بیند که image داده شده ای نمایش می دهد (عکس :-).

`docker ps` کارتینگ های موجود در سیستم خود را استفاده از دستور نمایش.

این دسته پوستهای EXPOSED شده را به نمایش می‌گذارد (عکس ۲-۱۱).

`docker stats` 命令可以实时查看容器的 CPU 和内存使用情况。

اب: دسته، منابع استفاده شده توسط هر کانتینر، ابه نمایش م- گذا.د (عکس ۱۲-۲).

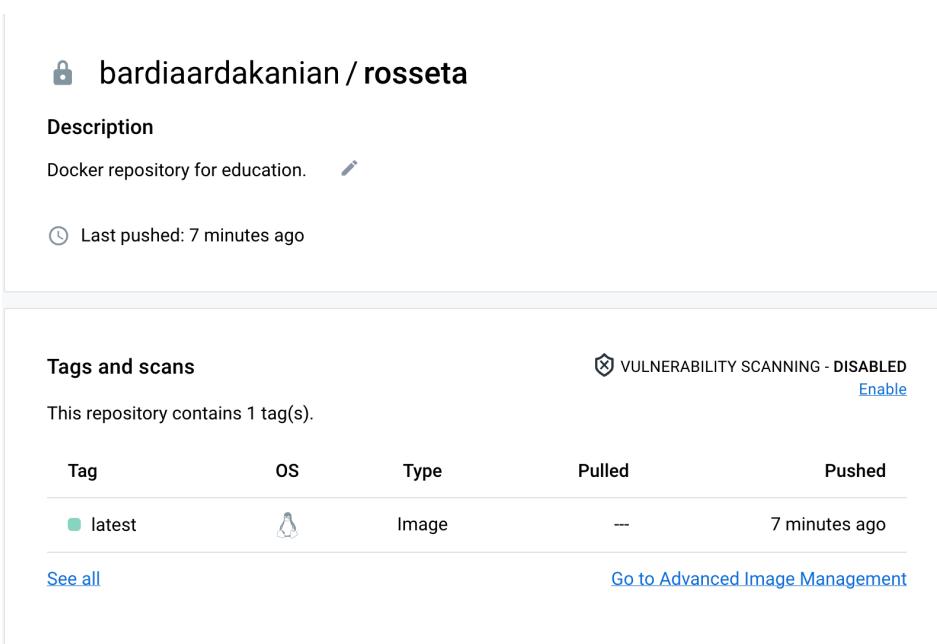


```

CE422-CC-Eververse — bardia@Bardias-MacBook-Pro — ..-CC-Eververse — zsh — 84x33
> docker tag eververse bardiaardakanian/rosseta
> docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
bardiaardakanian/rosseta    latest   1702cf9c280f  12 minutes ago  458MB
eververse           latest   1702cf9c280f  12 minutes ago  458MB
ce422-cc-eververse_web     latest   bb18d28d8f77  2 hours ago   458MB
redis               latest   31ec5e4e471c  14 hours ago   111MB
placard_web         latest   14b2c8268d28  16 hours ago   101MB
bardiaardakanian/alpine-curl latest   9ea689d29950  3 days ago    9.86MB
bardiaardakanian/rosseta    <none>  9ea689d29950  3 days ago    9.86MB
postgres            13      ff813cc5d8f8  7 days ago    354MB
redis               alpine   c660ce2c1f95  11 days ago   30.3MB
> docker push bardiaardakanian/rosseta
Using default tag: latest
The push refers to repository [docker.io/bardiaardakanian/rosseta]
4a25f50c08ee: Pushed
6440e495103b: Pushed
7a11095a9e60: Pushed
b1b9d5d62365: Pushed
a46d8467f5c6: Pushed
af9d5d30a08c: Pushed
687395c18fb0: Layer already exists
d221518f917b: Pushed
f0943dbe8944: Layer already exists
99efee407251: Layer already exists
1b577a8fb8cce: Layer already exists
latest: digest: sha256:3db1a1300888edb2b1f2818e4ab6d9c61ec0e80fef06de9ef7fcbec243bf
2e6 size: 2617

```

عکس ۲



bardiaardakanian / rosseta

Description

Docker repository for education. 

Last pushed: 7 minutes ago 

Tags and scans		VULNERABILITY SCANNING - DISABLED		
This repository contains 1 tag(s).		Enable		
Tag	OS	Type	Pulled	Pushed
latest		Image	---	7 minutes ago
See all		Go to Advanced Image Management		

عکس ۳

```

{
  "Id": "sha256:1702cf9c280f87b5bab7fae516f703ee5ba97e1c9bfa20bb62d11ee86a7c9646",
  "RepoTags": [
    "bardiaardakanian/rosseta:latest",
    "eververse:latest"
  ],
  "RepoDigests": [
    "bardiaardakanian/rosseta@sha256:3db1a1300888edbf2b1f2818e4ab6d9c61ec0e80fef06de9ef7fcbec243bf2e6"
  ],
  "Parent": "",
  "Comment": "buildkit.dockerfile.v0",
  "Created": "2022-12-13T11:50:44.486744251Z",
  "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": ""
  }
}

```

عکس ۱۰-۲

```

$ docker-compose up --build -d --force-recreate
Building web
[+] Building 2.4s (17/17) FINISHED
=> [internal] load build definition from Dockerfile          0.0s
=> => transferring dockerfile: 74B                          0.0s
=> [internal] load .dockerignore                            0.0s
=> => transferring context: 28                           0.0s
=> resolve image config for docker.io/docker/dockerfile:1   1.1s
=> CACHED docker-image://docker.io/docker/dockerfile:10@sha256:9ba7531bd80fb0 0.0s
=> [internal] load build definition from Dockerfile          0.0s
=> [internal] load .dockerignore                            0.0s
=> [internal] load metadata for docker.io/library/golang:1.19-alpine 1.0s
=> [1/8] FROM docker.io/library/golang:1.19-alpine@sha256:a9b24b67dc83b3383d 0.0s
=> [internal] load build context                           0.0s
=> => transferring context: 400B                         0.0s
=> CACHED [2/8] WORKDIR /app                            0.0s
=> CACHED [3/8] COPY main/go.mod ./                      0.0s
=> CACHED [4/8] COPY main/go.sum ./                      0.0s
=> CACHED [5/8] RUN go mod download && go mod verify 0.0s
=> CACHED [6/8] COPY main/*.go ./                      0.0s
=> CACHED [7/8] COPY .env ./                           0.0s
=> CACHED [8/8] RUN go build -o /eververse             0.0s
=> exporting to image                                    0.0s
=> => exporting layers                                 0.0s
=> => writing image sha256:1702cf9c280f87b5bab7fae516f703ee5ba97e1c9bfa20bb6 0.0s
=> => naming to docker.io/library/ce422-cc-eververse_web 0.0s
Creating redis ... done
Creating web ... done
$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
S PORTS NAMES
ec7028e60d30 redis:alpine "docker-entrypoint.s..." 3 seconds ago Up 2
seconds 0.0.0.0:6379->6379/tcp redis
029ddceff194 ce422-cc-eververse_web "/eververse" 3 seconds ago Up 2
seconds 0.0.0.0:80->1323/tcp web

```

عکس ۱۱-۲

CONTAINER ID	NAME	CPU %	MEM USAGE / LIMIT	MEM %	NET I/O	BLOCK
I/O	PIDS					
ec7028e60d30	redis	0.33%	2.547MiB / 7.667GiB	0.03%	1.16kB / 0B	0B /
0B	5					
029ddceff194	web	0.00%	1.441MiB / 7.667GiB	0.02%	1.16kB / 0B	0B /
0B	6					

۱۲-۲ عکس

گام سوم

در این مرحله ابتدا فایل config-map را ایجاد می‌کنیم تا کانفیگ سرور و ردیس را ذخیره کنیم (عکس ۱-۳). در این فایل تمامی اطلاعات مورد نیاز برای راه اندازی و اجرای سرور موجود می‌باشد. همچنین در فایل deployment سرور یک فایل .env ماوونت می‌کنیم. در ادامه فایل .env سرور را کامل می‌کنیم و تعداد پادها، پورت کانتینرها و ایمیج مورد نیاز پادها را مشخص می‌کنیم (عکس ۲-۳).

```

1  apiVersion: v1
2  data:
3    .env:
4      HOST=0.0.0.0
5      PORT=8080
6      SERVER_PORT=1323
7      API_KEY=9900DC68-EB5F-4FF5-8820-F8345AC607F1
8
9      REDIS_HOST=redis-service
10     REDIS_PORT=6379
11     REDIS_DB=0
12     KEY_EX=5
13   kind: ConfigMap
14   metadata:
15     creationTimestamp: null
16   name: eververse-config

```

عکس ۱-۳

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    creationTimestamp: null
5    labels:
6      app: eververse
7      name: eververse
8  spec:
9    replicas: 2
10   selector:
11     matchLabels:
12       app: eververse
13     strategy: {}
14   template:
15     metadata:
16       creationTimestamp: null
17       labels:
18         app: eververse
19     spec:
20       containers:
21         - image: docker.io/bardiaardakanian/eververse
22           imagePullPolicy: Always
23           name: eververse
24           resources:
25             requests:
26               cpu: "40m"
27           ports:
28             - containerPort: 8080
29           volumeMounts:
30             - mountPath: /env/.env
31               subPath: .env
32               name: config-map
33               readOnly: true
34             volumes:
35               - name: config-map
36               configMap:
37                 name: eververse-config
38   status: {}

```

عکس ۲-۳

در نهایت برای دسترسی به سرور یک service برای پادها از نوع ClusterIP تعریف می‌کنیم (عکس ۳-۳).

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: eververse-service
5    namespace: default
6  spec:
7    selector:
8      app: eververse
9    ports:
10   - protocol: TCP
11     port: 1323
12     targetPort: 8080

```

عکس ۳-۳

برای ردیس نیز دو فایل pv (عکس ۴-۳) و cpv (عکس ۴-۵) علاوه بر فایل‌های deployment (عکس ۴-۶) و service (عکس ۴-۷) داریم. در فایل کانفیگ (عکس ۱-۱) برای درخواست فرستادن به ردیس می‌بایس به ردیس سرویس رخدوست بفرستند.

```

1  apiVersion: v1
2  kind: PersistentVolume
3  metadata:
4    name: redis
5  spec:
6    storageClassName: manual
7    accessModes:
8      - ReadWriteOnce
9    capacity:
10   storage: 100Mi
11   hostPath:
12     path: /data/redis/

```

عکس ۴-۳

```

1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: redis-pvc
5  spec:
6    storageClassName: manual
7    accessModes:
8      - ReadWriteOnce
9    resources:
10   requests:
11     storage: 10Mi

```

عکس ۴-۵

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    creationTimestamp: null
5    labels:
6      app: redis
7    name: redis
8  spec:
9    replicas: 1
10   selector:
11     matchLabels:
12       app: redis
13   strategy: {}
14   template:
15     metadata:
16       creationTimestamp: null
17       labels:
18         app: redis
19     spec:
20       containers:
21         - image: redis:alpine
22           name: redis
23           command: [ "redis-server" ]
24           ports:
25             - containerPort: 6379
26           resources: {}
27           volumeMounts:
28             - mountPath: /data
29               name: data
30           volumes:
31             - name: data
32           persistentVolumeClaim:
33             claimName: redis-pvc
34   status: {}

```

عکس ۶-۳

```

1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: redis-service
5  spec:
6    selector:
7      app: redis
8    ports:
9      - protocol: TCP
10        port: 6379
11        targetPort: 6379

```

عکس ۷-۳

صحت ایجاد کانفیگ مب، پادها، service و deployment را با دستور get چک می‌کنیم (عکس ۸-۳)

```

❯ kubectl get configmaps
NAME      DATA   AGE
kube-root-ca.crt   1      3d18h

❯ kubectl get deployments
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
eververse   2/2     2           2           3h16m
redis       1/1     1           1           3h16m

❯ kubectl get services
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)      AGE
eververse-service ClusterIP  10.110.216.141 <none>        1323/TCP    3h16m
kubernetes     ClusterIP  10.96.0.1     <none>        443/TCP     3h17m
redis-service  ClusterIP  10.103.148.191 <none>        6379/TCP    3h16m

❯ kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
eververse-54b69c76c-27vpz  1/1     Running   0          3h16m
eververse-54b69c76c-h6h49  1/1     Running   0          3h16m
redis-79d9fcfd6cb-zshnv  1/1     Running   0          3h16m

❯ kubectl get pv
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM   STORAGECLASS
S   REASON   AGE
redis    100Mi      RWO           Retain           Bound   default/redis-pvc   manual
                           2d18h

❯ kubectl get pvc
NAME      STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
redis-pvc Bound   redis    100Mi     RWO           manual        2d18h

```

عکس ۸-۳

همچنین برای دیدن آدرس ای پی get endpoints ها از دستور استفاده می‌کنیم (عکس ۹-۳).

```

❯ kubectl get endpoints
NAME      ENDPOINTS   AGE
eververse-service  172.17.0.5:8080,172.17.0.6:8080  3h18m
kubernetes     192.168.49.2:8443   3h19m
redis-service   172.17.0.7:6379   3h18m

```

عکس ۹-۳

گام چهارم

ابتدا با دستورات get از وجود منابع روی کلاستر کوبرنیزیز اطمینان خاطر پیدا می‌کنیم (عکس ۱-۴). در برای تست کردن سرویس‌های خود یک پاد با کمک دستور ذیل ایجاد می‌کنیم (عکس ۲-۴). در ادامه با کمک دستورات curl به سرویس‌های ایجاد شده درخواست می‌زنیم و از صحت برنامه آگاه می‌شویم (عکس ۳-۴ و ۴-۴). همچنین به منظور تست کردن پخش شدن لود روی replicas از پاد اول (عکس ۵) و پاد دوم (عکس ۶-۴) لاغ می‌گیریم.

```

Bardia — bardia@Bardias-MacBook-Pro ~ ~ -zsh — 85x41
(base) bardia:~ bardia$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
eververse  2/2     2           2           6m22s
redis      1/1     1           1           6m21s
(base) bardia:~ bardia$ kubectl get services
NAME        TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
eververse-service  ClusterIP  10.105.39.94  <none>        8080/TCP    6m27s
kubernetes    ClusterIP  10.96.0.1    <none>        443/TCP     3h21m
redis-service  ClusterIP  10.107.86.22  <none>        6379/TCP    6m27s
(base) bardia:~ bardia$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
eververse-7846f64657-pnbm6  1/1     Running   0          6m30s
eververse-7846f64657-w8h2k  1/1     Running   0          6m30s
redis-79d9fc6cb-vz5rf     1/1     Running   0          6m30s
(base) bardia:~ bardia$ kubectl get ep
NAME        ENDPOINTS      AGE
eververse-service  172.17.0.5:8080,172.17.0.6:8080  6m36s
kubernetes    192.168.49.2:8443  3h22m
redis-service  172.17.0.7:6379  6m36s

```

عکس ۱-۴

```

Bardia — minikube kubectl -- run webkit --image=bardiaardakanian/webkit -i --tty -- sh — kubectl — kubectl <minikube...
(base) bardia:~ bardia$ kubectl run webkit --image=bardiaardakanian/webkit -i --tty -- sh
If you don't see a command prompt, try pressing enter.

(# curl --help
Usage: curl [options...] <url>
--abstract-socket <path> Connect via abstract Unix domain socket
--anyauth      Pick any authentication method
-a, --append    Append to target file when uploading
--basic        Use HTTP Basic Authentication
--cacert <file> CA certificate to verify peer against
--capath <dir> CA directory to verify peer against
-E, --cert <certificate[:password]> Client certificate file and password
--cert-status  Verify the status of the server certificate
--cert-type <type> Certificate file type (DER/PEM/ENG)
--ciphers <list of ciphers> SSL ciphers to use
--compressed   Request compressed response
--compressed-ssh Enable SSH compression
-K, --config <file> Read config from a file
--connect-timeout <seconds> Maximum time allowed for connection
--connect-to <HOST1:PORT1:HOST2:PORT2> Connect to host
-C, --continue-at <offset> Resumed transfer offset
-b, --cookie <data> Send cookies from string/file
-c, --cookie-jar <filename> Write cookies to <filename> after operation
--create-dirs  Create necessary local directory hierarchy
--crlf        Convert LF to CRLF in upload

```

عکس ۲-۴

```
# curl -X POST -v -Ss http://eververse-service.default.svc.cluster.local:8080/get?name=ETH
* Trying 10.105.39.94...
* TCP_NODELAY set
* Connected to eververse-service.default.svc.cluster.local (10.105.39.94) port 8080 (#0)
> POST /get?name=ETH HTTP/1.1
> Host: eververse-service.default.svc.cluster.local:8080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Date: Mon, 19 Dec 2022 19:19:45 GMT
< Content-Length: 109
<
{"Date":"2022-12-19 19:19:45.061344501 +0000 UTC m+=555.217472213","Name":"ETH","Rate":"1170.100
2457606498"}
* Connection #0 to host eververse-service.default.svc.cluster.local left intact
# curl -X POST -v -Ss http://eververse-service.default.svc.cluster.local:8080/get?name=ETH
* Trying 10.105.39.94...
* TCP_NODELAY set
* Connected to eververse-service.default.svc.cluster.local (10.105.39.94) port 8080 (#0)
> POST /get?name=ETH HTTP/1.1
> Host: eververse-service.default.svc.cluster.local:8080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Date: Mon, 19 Dec 2022 19:20:04 GMT
< Content-Length: 109
<
{"Date":"2022-12-19 19:20:04.224710885 +0000 UTC m+=574.380838597","Name":"ETH","Rate":"1170.100
2457606498"}
* Connection #0 to host eververse-service.default.svc.cluster.local left intact
```

عکس ۳-۴

```
# curl -X POST -v -Ss http://eververse-service.default.svc.cluster.local:8080/get?name=DOG
* Trying 10.105.39.94...
* TCP_NODELAY set
* Connected to eververse-service.default.svc.cluster.local (10.105.39.94) port 8080 (#0)
> POST /get?name=DOG HTTP/1.1
> Host: eververse-service.default.svc.cluster.local:8080
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Date: Mon, 19 Dec 2022 19:20:15 GMT
< Content-Length: 112
<
{"Date":"2022-12-19 19:20:15.310135543 +0000 UTC m+=585.496489185","Name":"DOG","Rate":"0.000670
9751874307417"}
* Connection #0 to host eververse-service.default.svc.cluster.local left intact
#
```

عکس ۴-۴

همانطور که مشاهده می‌کنید دو درخواست اول به پاد دوم و درخواست اخر به پاد اول فرستاده شده است (عکس ۴-۴ و ۵-۴). در نتیجه Load Balancing به خوبی انجام شده و تمامی فشار روی یک پاد قرار نگرفته است. تمامی کدها و فایل‌های کوبرنیز در فایل زیپ قرار داده شده‌اند.

```

$ kubectl logs eververse-7846f64657-wbh2k
2022/12/19 19:10:29 - [ENV] HOST:0.0.0.0
2022/12/19 19:10:29 - [ENV] SERVER_PORT:8080
2022/12/19 19:10:29 - [ENV] REDIS_HOST:redis-service
2022/12/19 19:10:29 - [ENV] REDIS_PORT:6379
2022/12/19 19:10:29 - [ENV] REDIS_DB:0
2022/12/19 19:10:29 - [ENV] KEY_EX:5
2022/12/19 19:10:29 - [ENV] API_KEY:9900DC68-EB5F-4FF5-8820-F8345AC607F1

/ _/_/_/ \
/ / / _/ _ \ \_ \
/ _/ \/_/_/_/ v4.9.0
High performance, minimalist Go web framework
https://echo.labstack.com
0/
0\

⇒ http server started on [::]:8080
2022/12/19 19:19:43 - [ERROR] Fail get key ETH, Error redigo: nil returned
2022/12/19 19:19:45 - [TRY] Set (key ETH, val 1170.1002457606498)
2022/12/19 19:19:45 - [INFO] SET ETH:1170.1002457606498
2022/12/19 19:19:45
- [INFO] Exchange rate by https://www.coinapi.io/
{"time":"2022-12-19T19:19:45.061430293Z","id":"","remote_ip":"172.17.0.1","host":"eververse-service.default.svc.cluster.local:8080","method":"POST","uri":"/get?name=ETH","user_agent":"curl/7.58.0","status":200,"error":"","latency":1222562834,"latency_human":"1.222562834s","bytes_in":0,"bytes_out":109}
{"time":"2022-12-19T19:20:04.224752677Z","id":"","remote_ip":"172.17.0.1","host":"eververse-service.default.svc.cluster.local:8080","method":"POST","uri":"/get?name=ETH","user_agent":"curl/7.58.0","status":200,"error":"","latency":404750,"latency_human":"404.75µs","bytes_in":0,"bytes_out":109}
2022/12/19 19:20:04 - [INFO] GET 1170.1002457606498
2022/12/19 19:20:04
- [INFO] Exchange rate by Cache

```

عکس ۴

```

$ kubectl logs eververse-7846f64657-pnbm6
2022/12/19 19:10:29 - [ENV] HOST:0.0.0.0
2022/12/19 19:10:29 - [ENV] SERVER_PORT:8080
2022/12/19 19:10:29 - [ENV] REDIS_HOST:redis-service
2022/12/19 19:10:29 - [ENV] REDIS_PORT:6379
2022/12/19 19:10:29 - [ENV] REDIS_DB:0
2022/12/19 19:10:29 - [ENV] KEY_EX:5
2022/12/19 19:10:29 - [ENV] API_KEY:9900DC68-EB5F-4FF5-8820-F8345AC607F1

/ _/_/_/ \
/ / / _/ _ \ \_ \
/ _/ \/_/_/_/ v4.9.0
High performance, minimalist Go web framework
https://echo.labstack.com
0/
0\

⇒ http server started on [::]:8080
2022/12/19 19:20:14 - [ERROR] Fail get key DOG, Error redigo: nil returned
2022/12/19 19:20:15 - [TRY] Set (key DOG, val 0.0006709751874307417)
{"time":"2022-12-19T19:20:15.310264543Z","id":"","remote_ip":"172.17.0.1","host":"eververse-service.default.svc.cluster.local:8080","method":"POST","uri":"/get?name=DOG","user_agent":"curl/7.58.0","status":200,"error":"","latency":1248137668,"latency_human":"1.248137668s","bytes_in":0,"bytes_out":112}
2022/12/19 19:20:15 - [INFO] SET DOG:0.0006709751874307417
2022/12/19 19:20:15
- [INFO] Exchange rate by https://www.coinapi.io/

```

عکس ۵