

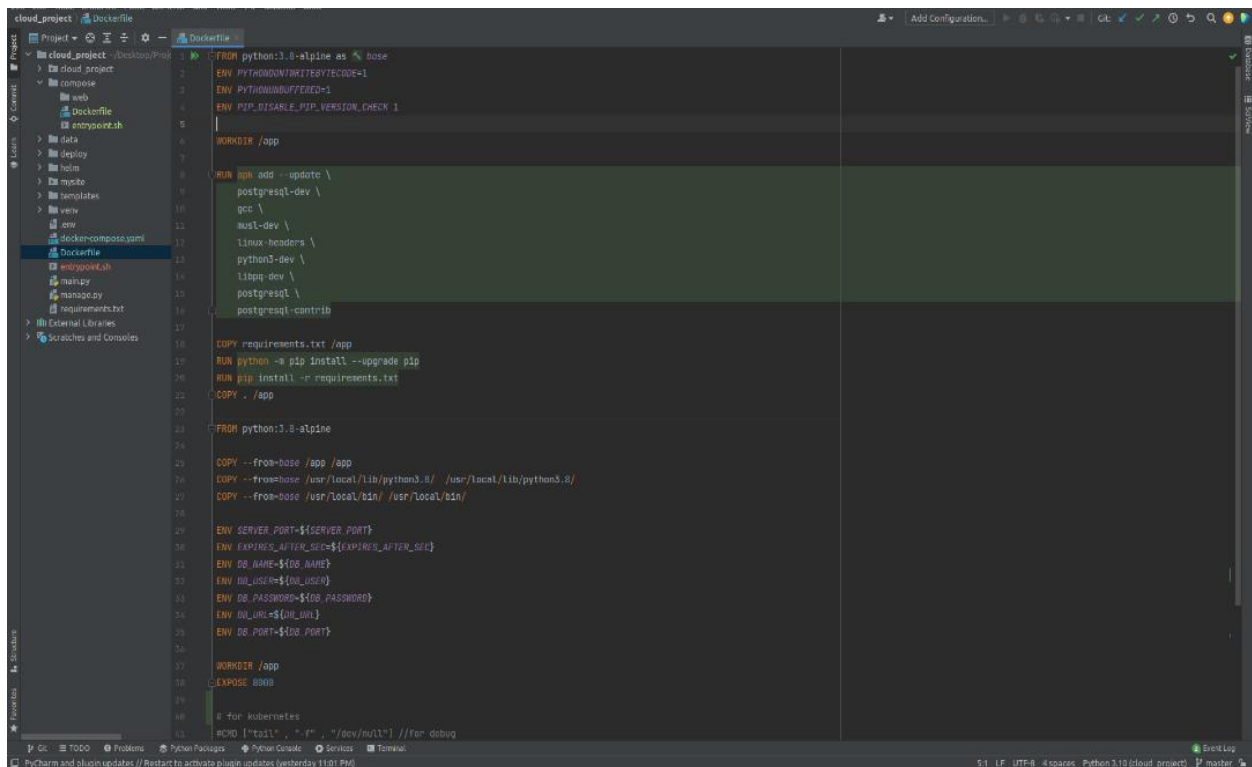
گام اول

اپلیکیشن private note با استفاده از فریمورک جنگو و دیتابیس postgres ساخته شد.

گام دوم

در ابتدا Dockerfile مربوط به برنامه را نوشتیم و سپس با دستور `docker build -t cc-app` ایمج از آن ساختیم.

سپس آن را تگ زدیم و با دستور `docker push` بر روی docker hub قرار دادیم.



```
1 FROM python:3.8-alpine as base
2 ENV PYTHONUNBUFFERED=1
3 ENV PYTHONBUFCECQ=1
4 ENV PIP_DISABLE_PIP_VERSION_CHECK 1
5
6 WORKDIR /app
7
8 RUN apk add --update \
9     postgresql-dev \
10    gcc \
11    musl-dev \
12    linux-headers \
13    python3-dev \
14    libpq-dev \
15    postgresql \
16    postgresql-contrib
17
18 COPY requirements.txt /app
19 RUN python -m pip install --upgrade pip
20 RUN pip install -r requirements.txt
21 COPY . /app
22
23 FROM python:3.8-alpine
24
25 COPY --from=base /app /app
26 COPY --from=base /usr/local/lib/python3.8/ /usr/local/lib/python3.8/
27 COPY --from=base /usr/local/bin/ /usr/local/bin/
28
29 ENV SERVER_PORT=${SERVER_PORT}
30 ENV EXPIRES_AFTER_SEC=${EXPIRES_AFTER_SEC}
31 ENV DB_NAME=${DB_NAME}
32 ENV DB_USER=${DB_USER}
33 ENV DB_PASSWORD=${DB_PASSWORD}
34 ENV DB_URL=${DB_URL}
35 ENV DB_PORT=${DB_PORT}
36
37 WORKDIR /app
38 EXPOSE 8000
39
40 # for Kubernetes
41 CMD ["tail", "-f", "/dev/null"] //for debug
```

این تصویر محتویات docker file را نشان میدهد.

که مشخص است از multistage build استفاده شده است.

پروژه پایانی

The screenshot shows a Docker Hub repository page for 'amirhasance79/cc_app'. The page is viewed in a Google Chrome browser window. The repository name is 'amirhasance79/cc_app'. The description states 'This repository does not have a description'. The last pushed time is '7 hours ago'. The 'Tags and Scans' section shows two tags: '2' and '1', both pushed '7 hours ago' and 'a day ago' respectively. The 'Automated Builds' section is disabled. The page also includes a 'Docker commands' section with the command 'docker push amirhasance79/cc_app:tagname'. At the bottom, there is a cookie consent banner.

Activities Google Chrome Jul 2 05:33

Docker Hub

hub.docker.com/repository/docker/amirhasance79/cc_app

General Tags Builds Collaborators Webhooks Settings

1 Add a short description for this repository
The short description is used to index your content on Docker Hub and in search engines. It's visible to users in search results. Update

2 Advanced Image Management
View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions. View preview

amirhasance79/cc_app

Description
This repository does not have a description

Last pushed: 7 hours ago

Docker commands
To push a new tag to this repository.
docker push amirhasance79/cc_app:tagname Public View

Tags and Scans
This repository contains 2 tags.

TAG	OS	PULLED	PUSHED
2		—	7 hours ago
1		—	a day ago

See all

VULNERABILITY SCANNING - DISABLED

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 to Pro Learn more

We and third parties use cookies or similar technologies ("Cookies") as described below to collect and process personal data, such as your IP address or browser information. You can learn more about how this site uses Cookies by reading our privacy policy linked below. By clicking "I consent to cookies", you accept the placement and use of these Cookies for these purposes. You can change your mind and revisit your preferences at any time by accessing the "Cookie Preferences" link in the footer of this site.

I consent to cookies Essential cookies only Customize Privacy Policy

شکل بالا نیز نشان میدهد ایمج ساخته شده با موفقیت بر روی داکرهاب قرار گرفته است.

```

Terminal: Local x Local (2) x Local (3) x + v
(venv) amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─ kubectl get deployment
NAME                READY    UP-TO-DATE    AVAILABLE    AGE
cc-app-deployment   2/2      2              2             35m
postgres            1/1      1              1             35m
(venv) amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─ kubectl get pods
NAME                                READY    STATUS    RESTARTS    AGE
cc-app-deployment-ff4c98767-4qpw7   1/1      Running   2 (35m ago)  35m
cc-app-deployment-ff4c98767-xb4jk   1/1      Running   2 (35m ago)  35m
postgres-5764d6b968-tnw59          1/1      Running   0            35m
(venv) amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─ kubectl get service
NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
cc-app-service      LoadBalancer  10.107.130.188  10.107.130.188  8000:30000/TCP    35m
kubernetes           ClusterIP      10.96.0.1       <none>          443/TCP           100d
postgres            NodePort       10.111.51.21    <none>          5432:31804/TCP    35m
(venv) amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─ kubectl get secret
NAME                                TYPE          DATA    AGE
cc-app-secret                       Opaque        2        35m
default-token-nmvqk                 kubernetes.io/service-account-token  3        100d
sh.helm.release.v1.helm-1656721252.v1  helm.sh/release.v1  1        35m
(venv) amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─ kubectl get configmap
NAME                DATA    AGE
cc-app-configmap    5        35m
kube-root-ca.crt    1        100d
postgres-config     2        35m
(venv) amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─

```

این تصویر صحت ایجاد منابع را نشان میدهد که میبینیم همه ی منابع به درستی بالا آمده و در وضعیت Ready قرار گرفته اند و هیچ نودی نیست که Ready نشده باشد.

وضعیت pod ها ... service, configmap سالم است.

پروژه پایانی

```
(venv) └─amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─ kubectl describe service cc-app-service
Name: cc-app-service
Namespace: default
Labels: app.kubernetes.io/managed-by=Helm
Annotations: meta.helm.sh/release-name: helm-1656721252
             meta.helm.sh/release-namespace: default
Selector: app=cc-app-deployment
Type: LoadBalancer
IP: 10.107.130.188
LoadBalancer Ingress: 10.107.130.188
Port: <unset> 8000/TCP
TargetPort: 8000/TCP
NodePort: <unset> 30000/TCP
Endpoints: 172.17.0.3:8000,172.17.0.4:8000
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
(venv) └─amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─ kubectl describe service postgres
Name: postgres
Namespace: default
Labels: app=postgres
       app.kubernetes.io/managed-by=Helm
Annotations: meta.helm.sh/release-name: helm-1656721252
             meta.helm.sh/release-namespace: default
Selector: app=postgres
Type: NodePort
IP: 10.111.51.21
Port: <unset> 5432/TCP
TargetPort: 5432/TCP
NodePort: <unset> 31804/TCP
Endpoints: 172.17.0.5:5432
Session Affinity: None
External Traffic Policy: Cluster
Events: <none>
(venv) └─amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─
```

شکل بالا آدرس IP پادها و نحوه ی برقراری ارتباط آنها و سرویس ساخته شده را نشان میدهد.

همچنین تعداد replica های مربوط به دیتابیس را ۱ قرار دادیم که درواقع از مدل standalone استفاده کردیم. یعنی تنها یک گره از دیتابیس را ایجاد میکنیم. دلیل اینکار نیز این است که اپ جنگو ساخته شده api های بسیار کمی دارد و تعداد یوزر های کمی از آن استفاده میکنند و بار زیادی برروی دیتابیس نیست. در نتیجه تعداد transaction های دیتابیس و تعداد های read/write

پروژه پایانی

برروی دیتابیس بسیار کم است که باعث میشود نیازی برای لودبالانس کردن دیتابیس نداشته باشیم
بنابراین تعداد replica هارا برابر ۱ در نظر میگیریم.

ساختن یک کامپوننت HPA

۱) پارامترهای autoscaling میتوانند معیارهای خارجی مانند وضعیت منابع کلاستر object و یا pod ها و resource ها باشند.

پارامترهای resource وابسته به منابع هستند و میشود بر اساس بهره وری و میزان مصرف CPU, Disk, Network Resources, GPU و یا memory باشند.

برای مثال پارامترهای استفاده شده برای مصرف CPU تعداد هسته های CPU که توسط پاد ها مصرف شده اند و همچنین میزان بهره وری CPU میتوانند باشند.

پارامترهای دیگر دیسک شبکه حافظه GPU و ... هستند.

۲) پارامتر CPU را در این قسمت در نظر گرفته ایم چون میزان سرعت و دسترسی به CPU نسبت به پارامترهای دیگر برای ما میتواند باعث ایجاد bottleneck شود.

```

1  apiVersion: autoscaling/v2
2  kind: HorizontalPodAutoscaler
3  metadata:
4    name: private-note-app-hpa
5  spec:
6    scaleTargetRef:
7      apiVersion: apps/v1
8      kind: Deployment
9      name: cc-app-deployment
10   minReplicas: 1
11   maxReplicas: 3
12   metrics:
13     - type: Resource
14       resource:
15         name: cpu
16       target:
17         type: Utilization
18         averageUtilization: 50

```

این تصویر نیز کد مربوط به HPA را نشان میدهد که پارامتر CPU انتخاب شده است و averageUtilization ۵۰ برای آن در نظر گرفته شده است.

اجرای دیتابیس با استفاده از stateful set

(۱) دلیل استفاده این است که منابعی که در deployment استفاده میشوند دارای state نیستند یعنی پادهای مختلف از deployment ها ساخته میشوند که state آنها دائمی نیست و بعد از از بین رفتن pod ها وضعیت آنها نیز از بین میرود. اما در stateful به اینصورت نیست و داده ها در مقابل restart و یا از بین رفتن پادها مقاوم هستند به این صورت که VolumClaimTemplate استفاده میکنند.

(۲)

```
# PostgreSQL StatefulSet Service
apiVersion: v1
kind: Service
metadata:
  name: postgres-db-lb
spec:
  selector:
    app: postgresql-db
    type: LoadBalancer
  ports:
  - port: 5432
    targetPort: 5432
```


پروژه پایانی

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: postgres-sts
spec:
  serviceName: postgres-headless-svc
  replicas: 3
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      securityContext:
        fsGroup: 1001
      containers:
        - name: postgresql
          lifecycle:
            preStop:
              exec:
                command:
                  - /pre-stop.sh
          image: docker.io/bitnami/postgresql-repmgr:11.12.0-debian-10-r44
          imagePullPolicy: "IfNotPresent"
          securityContext:
            runAsUser: 1001
          # Auxiliary vars to populate environment variables
          env:
            - name: BITNAMI_DEBUG
              value: "false"
            # PostgreSQL configuration
            - name: POSTGRESQL_VOLUME_DIR
              value: "/bitnami/postgresql"
            - name: PGDATA
```

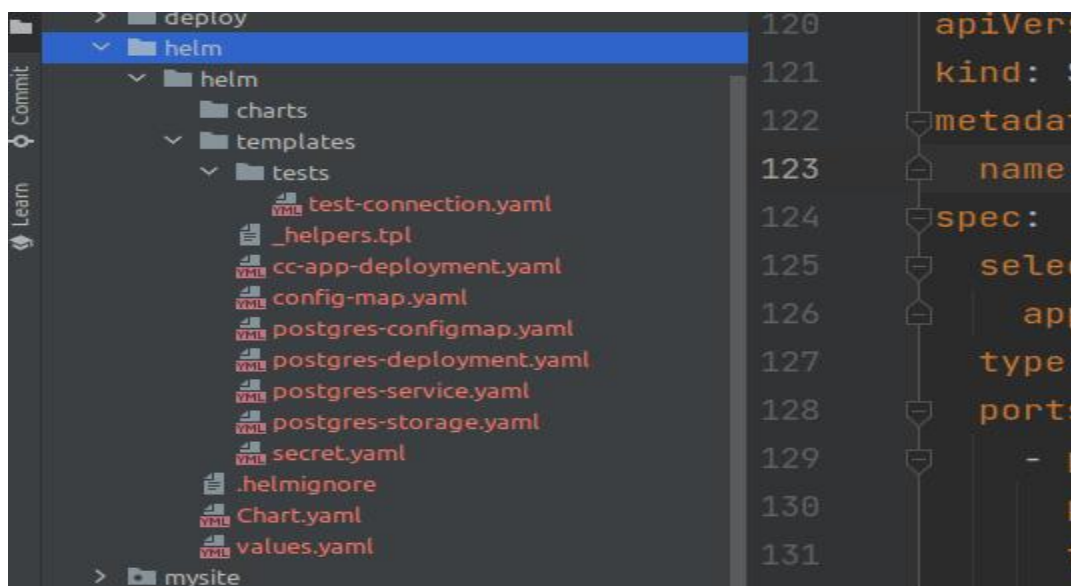
۳) در اینجا سه نود stateful postgres ایجاد میکنیم که یکی از آنها نقش مستر و دو تا دیگر نقش slave را بازی میکنند. باید توجه کنیم باید دستورات write را به مستر بفرستیم اما دستورات read را میتوانیم به هر سه نود بفرستیم. که علت آن جلوگیری از data inconsistency میباشد.

پیاده سازی helm chart

۱) میدانییم helm برای مدیریت فایل های yml دیپلویمنت های ساخته برای کلاستر های کوبر مورد استفاده قرار میگیرد و ورژن ریلیز های مختلف را مدیریت میکند و باعث میشود که بتوانیم فایل های yml را گروه بندی کنیم. بدینصورت که در صورت استفاده زیاد از یک سری مقادیر دیگر آن هارا در همه ی فایل های yml تکرار نمیکنیم و یکبار آن را در helm تعریف میکنیم و در فایل های yml دیگر از آن استفاده میکنیم.

ساختار helm به صورت زیر است:

```
chart/  
└─ microservices/  
    ├── Chart.yaml  
    ├── values.yaml  
    └─ templates/  
        ├── deployment.yaml  
        ├── service.yaml  
        └─ ingress.yaml
```



مطابق شکل مشاهده میکنیم ابتدا اسم chart ساخته شده قرار میگیرد.

پروژه پایانی

سپس یک فایل Chart.yaml داریم که اسم chart ورژن و دیگر تنظیمات مربوط به چارت در این فایل مشخص و ذخیره میشوند.

سپس یک فایل به اسم values.yaml داریم که فایل مهمی است زیرا تمام مقادیر در این فایل تعریف میشوند. درواقع مقادیر مشترکی که فایل های yml از آنها استفاده میکنند را در این فایل قرار میدهیم.

سپس دایرکتوری templates را داریم که تمامی فایل های deployment مربوط به کلاستر های کوبر را در این فایل قرار میدهیم و سپس باید مقادیر فیلد های آن فایل های yml ای که مقدار آنها در فایل values.yaml در heml قرار گرفته اند را با فرمتی مثل {{.Values.}} جایگزین میکنیم.

```

replicaCount: 1

appName: cc-app
dbName: postgres
dbReplicaCount: 1
appReplicaCount: 2
expiresAfterSec: 60

image:
  repository: nginx
  pullPolicy: IfNotPresent
  # Overrides the image tag whose default is the chart appVersion.
  tag: ""

imagePullSecrets: []
nameOverride: ""
fullnameOverride: ""

serviceAccount:
  # Specifies whether a service account should be created
  create: true
  # Annotations to add to the service account
  annotations: {}
  # The name of the service account to use.
  # If not set and create is true, a name is generated using the fullname template
  name: ""

podAnnotations: {}

podSecurityContext: {}

```

پروژه پایانی

```
podSecurityContext: {}
  # fsGroup: 2000

securityContext: {}
  # capabilities:
  #   drop:
  #     - ALL
  # readOnlyRootFilesystem: true
  # runAsNonRoot: true
  # runAsUser: 1000

service:
  type: ClusterIP
  port: 5432

ingress:
  enabled: false
  annotations: {}
    # kubernetes.io/ingress.class: nginx
    # kubernetes.io/tls-acme: "true"
  hosts:
    - host: chart-example.local
      paths: []
  tls: []
    # - secretName: chart-example-tls
    #   hosts:
    #     - chart-example.local

resources: {}
  # We usually recommend not to specify default resources and to leave this as a conscious
  # choice for the user. This also increases chances charts run on environments with little
  # resources, such as Minikube. If you do want to specify resources, uncomment the following
  # lines, adjust them as necessary, and remove the curly braces after 'resources:'.
  # limits:
  #   cpu: 100m
```

پروژه پایانی

```
autoscaling:
  enabled: false
  minReplicas: 1
  maxReplicas: 100
  targetCPUUtilizationPercentage: 80
  # targetMemoryUtilizationPercentage: 80

nodeSelector: {}

tolerations: []

affinity: {}
```

همچنین عکس های زیر مربوط به helm پروژه است که نشان میدهد helm به درستی عمل کرده است:

```
(venv) └─amir@Amir ~/Desktop/Projects/cloud_project/helm <master*>
└─ helm install helm --generate-name
NAME: helm-1656721252
LAST DEPLOYED: Sat Jul  2 04:50:52 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
```

```
(venv) └─amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─ kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/cc-app-deployment-ff4c98767-4qpw7	1/1	Running	2 (44m ago)	44m
pod/cc-app-deployment-ff4c98767-xb4jk	1/1	Running	2 (44m ago)	44m
pod/postgres-5764d6b968-tnw59	1/1	Running	0	44m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/cc-app-service	LoadBalancer	10.107.130.188	10.107.130.188	8000:30000/TCP	44m
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	108d
service/postgres	NodePort	10.111.51.21	<none>	5432:31804/TCP	44m

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/cc-app-deployment	2/2	2	2	44m
deployment.apps/postgres	1/1	1	1	44m

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/cc-app-deployment-ff4c98767	2	2	2	44m
replicaset.apps/postgres-5764d6b968	1	1	1	44m

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
horizontalpodautoscaler.autoscaling/private-note-app-hpa	Deployment/cc-app-deployment	<unknown>/50%	1	3	2	29m

```
(venv) └─amir@Amir ~/Desktop/Projects/cloud_project <master*>
└─
```

پروژه پایانی

پایه سازی docker compose

The image shows a PyCharm IDE with two panels. The top panel displays the `docker-compose.yml` file, and the bottom panel shows the Kubernetes deployment status.

docker-compose.yml

```
version: "3.9"

services:
  db:
    image: postgres:14.1-alpine
    volumes:
      - ./data/db:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
  web:
    build: .
    command: sh -c "sleep 5 && python manage.py migrate && python manage.py runserver 0.0.0.0:8000 --noreload"
    ports:
      - "8000:8000"
    environment:
      - DB_NAME=postgres
      - DB_USER=postgres
      - DB_PASSWORD=postgres
      - DB_URL=db
      - DB_PORT=5432
      - EXPIRES_AFTER_SEC=30
    depends_on:
      - db
```

Kubernetes Deployment Status

Annotations:

- deployment.kubernetes.io/revision: 1
- meta.helm.sh/release-name: helm-1656721252
- meta.helm.sh/release-namespace: default

Selector:

- app=cc-app-deployment

Replicas:

- 2 desired | 2 updated | 2 total | 2 available | 0 unavailable

StrategyType:

- RollingUpdate

MinReadySeconds:

- 0

RollingUpdateStrategy:

- 25% max unavailable, 25% max surge

Pod Template:

- Labels: app=cc-app-deployment
- Containers:
- cc-app-deployment:
- Image: amirhasance79/cc_app:2
- Port: 8000/TCP
- Host Port: 0/TCP
- Environment:
- EXPIRES_AFTER_SEC: <set to the key 'EXPIRES_AFTER_SEC' of config map 'cc-app-configmap'> Optional: false
- DB_URL: <set to the key 'DB_URL' of config map 'postgres-config'> Optional: false
- DB_NAME: <set to the key 'POSTGRES_DB' of config map 'postgres-config'> Optional: false
- DB_PORT: <set to the key 'DB_PORT' of config map 'cc-app-configmap'> Optional: false
- DB_PASSWORD: <set to the key 'db-password' in secret 'cc-app-secret'> Optional: false
- DB_USER: <set to the key 'db-user' in secret 'cc-app-secret'> Optional: false
- Mounts: <none>
- Volumes: <none>

Conditions:

Type	Status	Reason
Progressing	True	NewReplicaSetAvailable
Available	True	MinimumReplicasAvailable
OldReplicaSets	<none>	
NewReplicaSet	cc-app-deployment-ff4c98767	(2/2 replicas created)

Events:

Type	Reason	Age	From	Message
Normal	ScalingReplicaSet	36m	deployment-controller	Scaled up replica set cc-app-deployment-ff4c98767 to 2

پروژه پایانی

در فایل docker compose بالا دو سرویس تعریف کرده ایم. یکی سرویس web و دیگری سرویس db

سرویس web همان اپ ما است که به سرویس db که همان دیتابیس postgres است وابسته است.

تصاویر زیر نیز نشان میدهد پروژه به درستی بالا آمده است:

