به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

رایانش ابری

پروژه پایانی

داکر و کوبرنتیز

آرمین ذوالفقاری داریانی 9731082

محمد حسین لوک‌زاده 9731056

گام اول (پروژه private-note)

این گام از پروژه با استفاده از Nodejs و MongoDB پیاده‌سازی شده است که برای جزییات پیاده‌سازی می‌توانید به لینک زیر مراجعه کنید:

https://github.com/arminZolfaghari/FInal_CloudComputing_Project/tree/main/PrivateNote

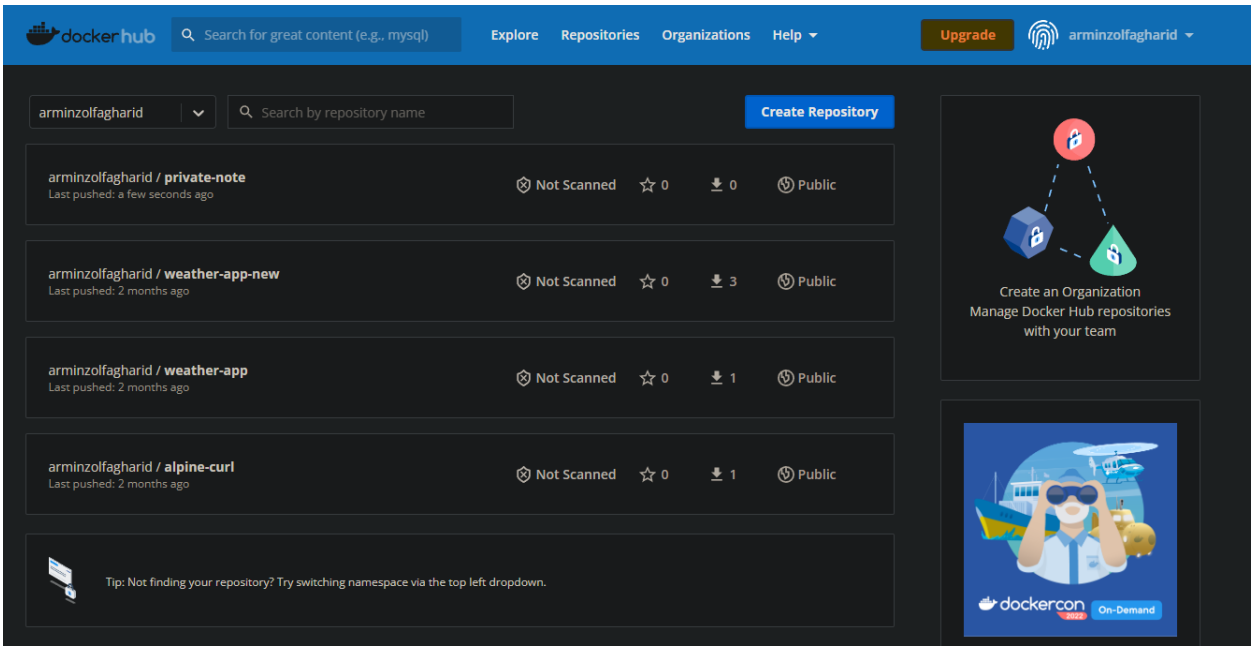
گام دوم

1) بیلد کردن ایمیج با استفاده از Dockerfile ساخته شده

```
PS E:\Armin\AUT\CEIT\Sem8\Cloud Computing (Javadi)\HWs\FInal_CloudComputing_Project> docker build -t private-note:latest .
 => [internal] load build context
 => => transferring context: 146.99kB
 => CACHED [stage-1 2/5] RUN apk add curl
 => CACHED [stage-1 3/5] RUN apk add nodejs npm
 => CACHED [stage-1 4/5] WORKDIR /PrivateNote
 => CACHED [build 2/5] WORKDIR /PrivateNote
 => CACHED [build 3/5] COPY /PrivateNote/package*.json .
 => CACHED [build 4/5] RUN npm install
 => CACHED [build 5/5] COPY /PrivateNote .
 => CACHED [stage-1 5/5] COPY --from=build /PrivateNote .
 => exporting to image
 => => exporting layers
 => => writing image sha256:a848f46701abd624acb2c235519c3271eabfd5773235e150fcef0ad937fe47fd
 => => naming to docker.io/library/private-note:latest
```

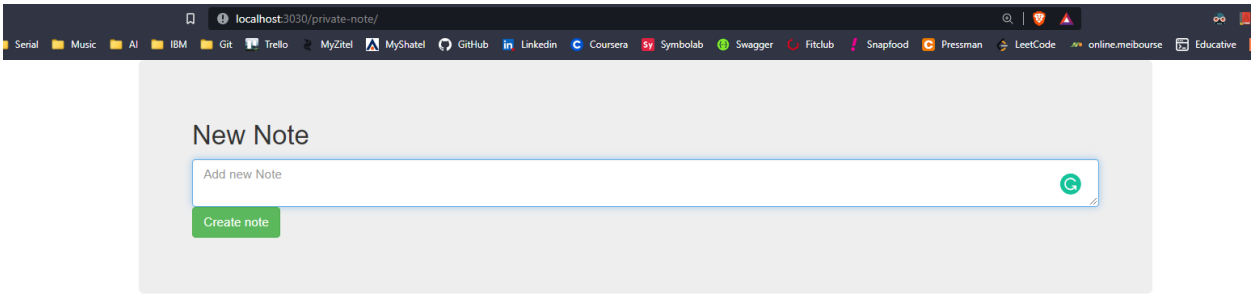
2) ارسال ایمیج ساخته‌شده بر روی داکرهاب و نتیجه آن

```
PS E:\Armin\AUT\CEIT\Sem8\Cloud Computing (Javadi)\HWs\FInal_CloudComputing_Project> docker push arminzolfagharid/private-note:latest
The push refers to repository [docker.io/arminzolfagharid/private-note]
ae5277a0f99d: Layer already exists
14b4b0f4aa03: Layer already exists
7177b9baa2da: Layer already exists
34b937672ad6: Layer already exists
24302eb7d908: Layer already exists
latest: digest: sha256:2169489d7394bb142779940378d5dcfcdd4f2e06b1bb5c5d3e4f77883d986323 size: 1369
```

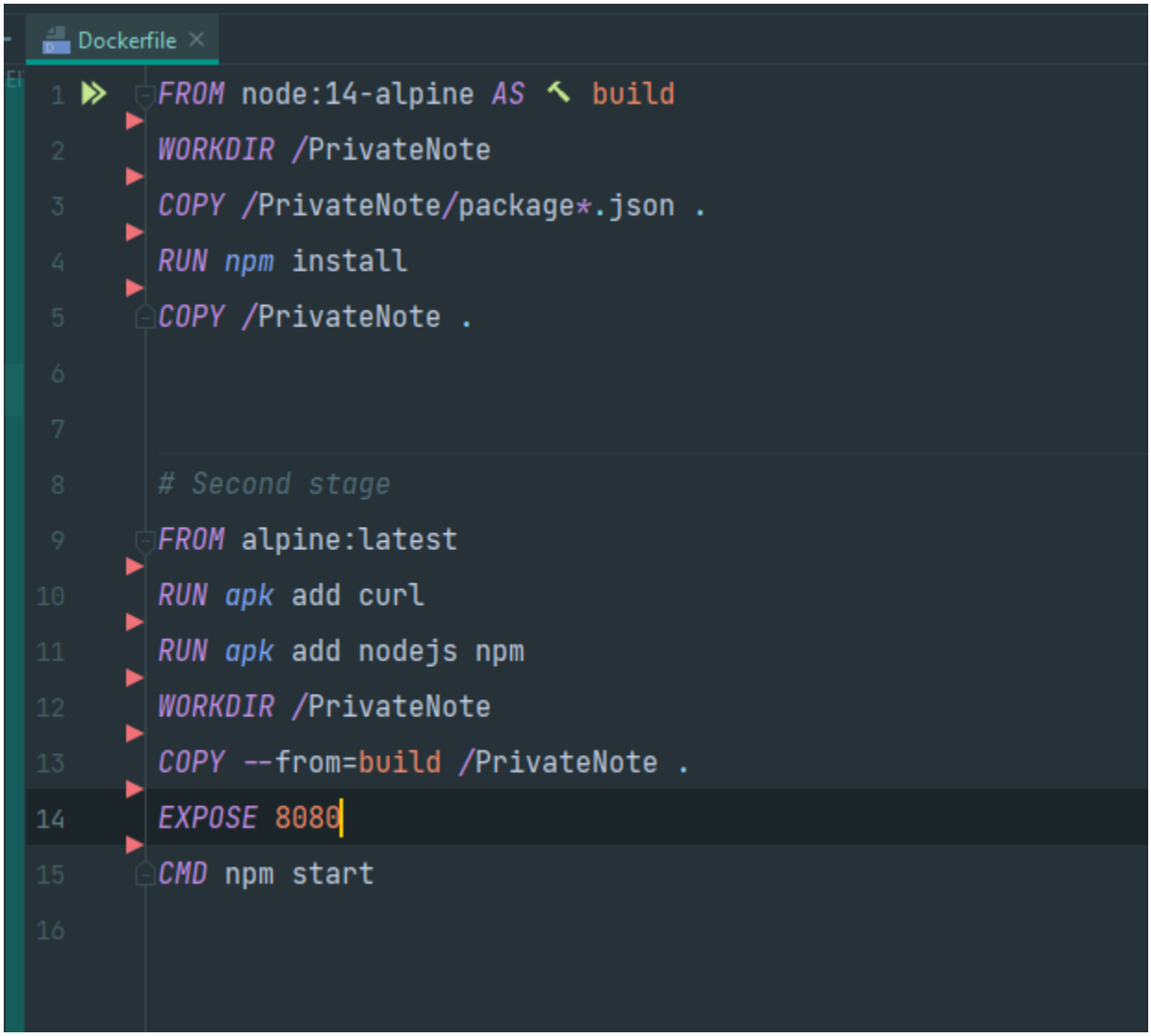


3) تست

```
PS E:\Armin\AUT\CEIT\Sem8\Cloud Computing (Javadi)\HWs\FInal_CloudComputing_Project> docker run -p 3000:3000 -i -t private-note

> private-note@1.0.0 start
> node ./src/index.js
```



4) محتویات داکرفایل (که به صورت multistage پیاده‌سازی شده‌است)



گام سوم

1) صحت ایجاد منابع بر روی کلاستر

دیتابیس مونگو:

برای persistent-volume

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % ls
Dockerfile              PrivateNote             README.md               docker-compose.yaml     k8s
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/mongodb/pv.yaml
persistentvolume/mongodb-volume created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get pv
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   REASON   AGE
mongodb-volume   500Mi      RWO            Retain           Available           manual                  48s
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project %
```

برای persistent-volume-claim

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/mongodb/pvc.yaml
persistentvolumeclaim/mongodb-pvc created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get pvc
NAME          STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
mongodb-pvc   Bound    pvc-1dddb040-4a51-46c8-b3f1-f36df929b84b   500Mi      RWO            standard       4s
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project %
```

برای configmaps

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/mongodb/config-map.yaml
configmap/mongodb-config created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get configmaps
NAME               DATA   AGE
kube-root-ca.crt   1      71d
mongodb-config     3      3s
myapp-configs      2      63d
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project %
```

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl describe configmaps mongodb-config
Name:         mongodb-config
Namespace:    default
Labels:       <none>
Annotations:  <none>

Data
====
MONGO_HOST:
----
mongodb
MONGO_INITDB_DATABASE:
----
private_notes
MONGO_PORT:
----
27017

BinaryData
====

Events:  <none>
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project %
```

برای secret

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/mongodb/secret.yaml
secret/mongodb-secret created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-22gjx   kubernetes.io/service-account-token   3      71d
mongodb-secret        Opaque                                2      10s
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project %
```

برای دیپلوینمنت

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/mongodb/deployment.yaml
deployment.apps/mongodb-deployment created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get pod
NAME                                READY   STATUS    RESTARTS        AGE
mongodb-deployment-56d787649b-lmfct   1/1     Running   0               6s
myapp-deployment-f4d7d588-2kbbd       1/1     Running   1 (3h26m ago)   63d
myapp-deployment-f4d7d588-wp5dt       1/1     Running   1 (3h26m ago)   63d
```

برای سرویس

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/mongodb/service.yaml
service/mongodb-service created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get service
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)     AGE
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP     71d
mongodb-service   ClusterIP   10.103.167.6    <none>        27017/TCP   8s
myapp-service     ClusterIP   10.101.57.114   <none>        5010/TCP    63d
```

برنامه private-note

برای configmaps

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/private-note-app/config-map.yaml
configmap/private-note-app-config created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get configmaps
NAME                     DATA   AGE
kube-root-ca.crt         1      71d
mongodb-config           3      25m
myapp-configs            2      63d
private-note-app-config  7      8s
```

برای secret

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/private-note-app/secret.yaml
error: no objects passed to apply
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/private-note-app/secret.yaml
secret/private-note-secret created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get secret private-note-secret
NAME                TYPE     DATA   AGE
private-note-secret   Opaque   2      17s
```

برای دیپلوینمنت

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/private-note-app/deployment.yaml
deployment.apps/private-note-app-deployment created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get deployment
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
mongodb-deployment           1/1     1            1           28m
myapp-deployment             2/2     2            2           63d
private-note-app-deployment  2/2     2            2           8s
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get pod
NAME                                        READY   STATUS    RESTARTS        AGE
mongodb-deployment-56d787649b-lmfct         1/1     Running   0               28m
myapp-deployment-f4d7d588-2kbbd             1/1     Running   1 (3h55m ago)   63d
myapp-deployment-f4d7d588-wp5dt             1/1     Running   1 (3h55m ago)   63d
private-note-app-deployment-74957f4997-kjqmz   1/1     Running   0               14s
private-note-app-deployment-74957f4997-kr9r5   1/1     Running   0               14s
```

برای سرویس

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl apply -f k8s/private-note-app/service.yaml
service/private-note-app-service created
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get service
NAME                     TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)     AGE
kubernetes               ClusterIP   10.96.0.1       <none>        443/TCP     71d
mongodb-service          ClusterIP   10.103.167.6    <none>        27017/TCP   29m
myapp-service            ClusterIP   10.101.57.114   <none>        5010/TCP    63d
private-note-app-service   ClusterIP   10.97.57.50     <none>        3000/TCP    6s
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project %
```

2) آدرس IP پادها و نحوه برقراری ارتباط میان آنها و سرویس ساخته‌شده

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl get pod -o wide
NAME                                        READY   STATUS    RESTARTS      AGE    IP           NODE       NOMINATED NODE   READINESS GATES
mongodb-deployment-56d787649b-z5qmz         1/1     Running   1 (72m ago)   114m   172.17.0.6   minikube   <none>           <none>
myapp-deployment-f4d7d588-2kbbd             1/1     Running   2 (72m ago)   63d    172.17.0.5   minikube   <none>           <none>
myapp-deployment-f4d7d588-wp5dt             1/1     Running   2 (72m ago)   63d    172.17.0.2   minikube   <none>           <none>
private-note-app-deployment-74957f4997-cx4kg   1/1     Running   1 (72m ago)   108m   172.17.0.8   minikube   <none>           <none>
private-note-app-deployment-74957f4997-zbc7r   1/1     Running   1 (72m ago)   108m   172.17.0.3   minikube   <none>           <none>
```

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl describe service mongodb-service
Name:              mongodb-service
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=mongodb-deployment
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.104.123.221
IPs:               10.104.123.221
Port:              <unset>  27017/TCP
TargetPort:        27017/TCP
Endpoints:         172.17.0.6:27017
Session Affinity:  None
Events:            <none>
```

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl describe service private-note-app-service
Name:              private-note-app-service
Namespace:         default
Labels:            <none>
Annotations:       <none>
Selector:          app=private-note-app
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.103.42.101
IPs:               10.103.42.101
Port:              <unset>  3000/TCP
TargetPort:        3000/TCP
Endpoints:         172.17.0.3:3000,172.17.0.8:3000
Session Affinity:  None
Events:            <none>
```

3) تعداد پادها برای دیپلویمنت مربوط به دیتابیس: در دیپلویمنت از مونگو دیبی به صورت standalone استفاده کردیم که به صورت یک پاد قرار دارد. دلیل استفاده از standalone این است که عملیا replication در دیتابیس‌ها از الگوریتم خاصی استفاده می‌کنند و stateful set هستند.

موارد امتیازی

ساختن یک کامپوننت HPA در کلاستر کوبرنتیز

1. پارامترهای موجود: پارامترهای Autoscaling می‌تواند معیاری خارجی (مستقل از وضعیت منابع کلاستر)، Object (معیاری وابسته به یک شئ خاص در کوبرنتیز)، و یا Pods و Resources باشد.

   پارامترهای Resource یا همان وابسته به منبع Autoscaling پادها می‌تواند برحسب میزان بهره‌وری و مصرف CPU، یا Disk یا Network Resources یا GPU و Memory باشند.

   برای مثال پارامترهای بر اساس مصرف CPU را در زیر توضیح داده‌ایم.

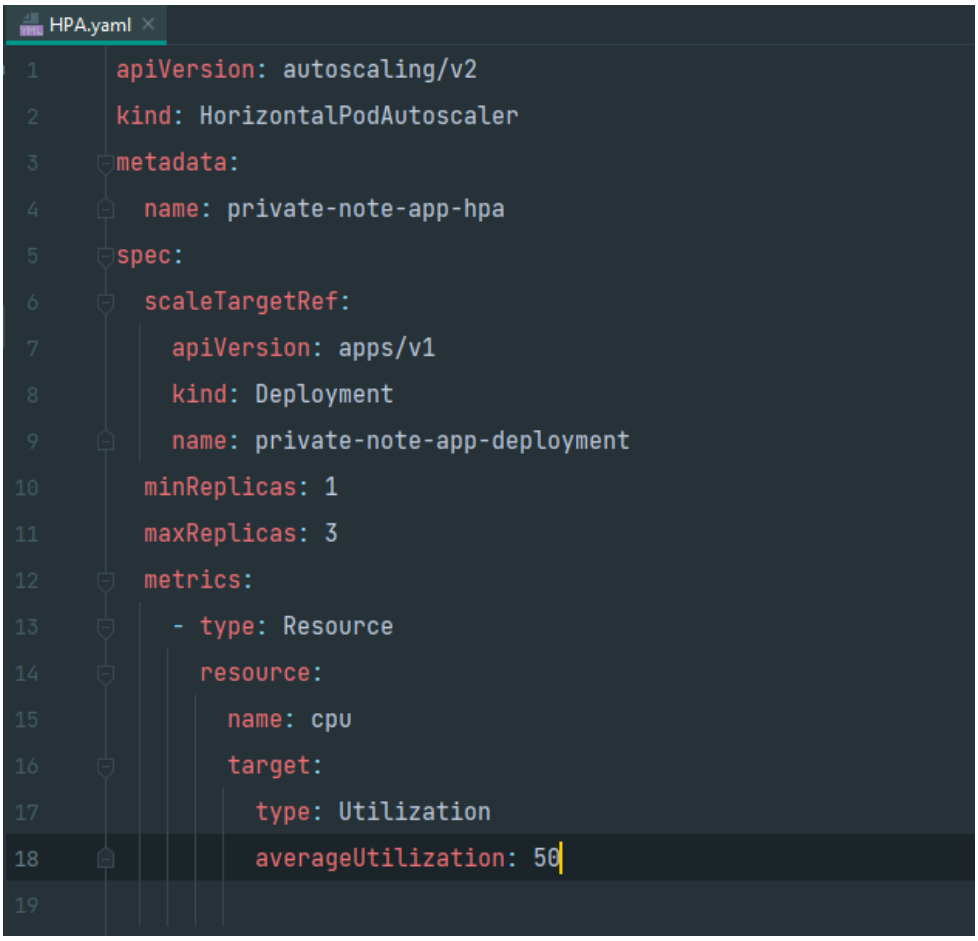
   - تعداد هسته‌ی CPU مصرف شده توسط پاد
   - بهره‌وری CPU: درصدی از CPU که Node که توسط پاد مصرف شده است.

   پارامترهای دیگر می‌تواند دیسک، شبکه، حافظه ، GPU و ... باشد. (ترکیبی از این‌ها هم می‌تواند باشد)

2. پارامتری که در نظر گرفتیم، مقدار CPU است چون bottleneck پروژه ما روی CPU می‌تواند باشد نسبت به بقیه پارامترها. (البته پارامتر شبکه نیز می‌تواند bottleneck نیز باشد)

3. توصیف مورد استفاده برای ساخت HPA

در عکس زیر تعداد کمترین و بیشترین رپلیکا تعیین شده و پارامتر مورد بررسی نیز همانطور که گفته شد CPU است.



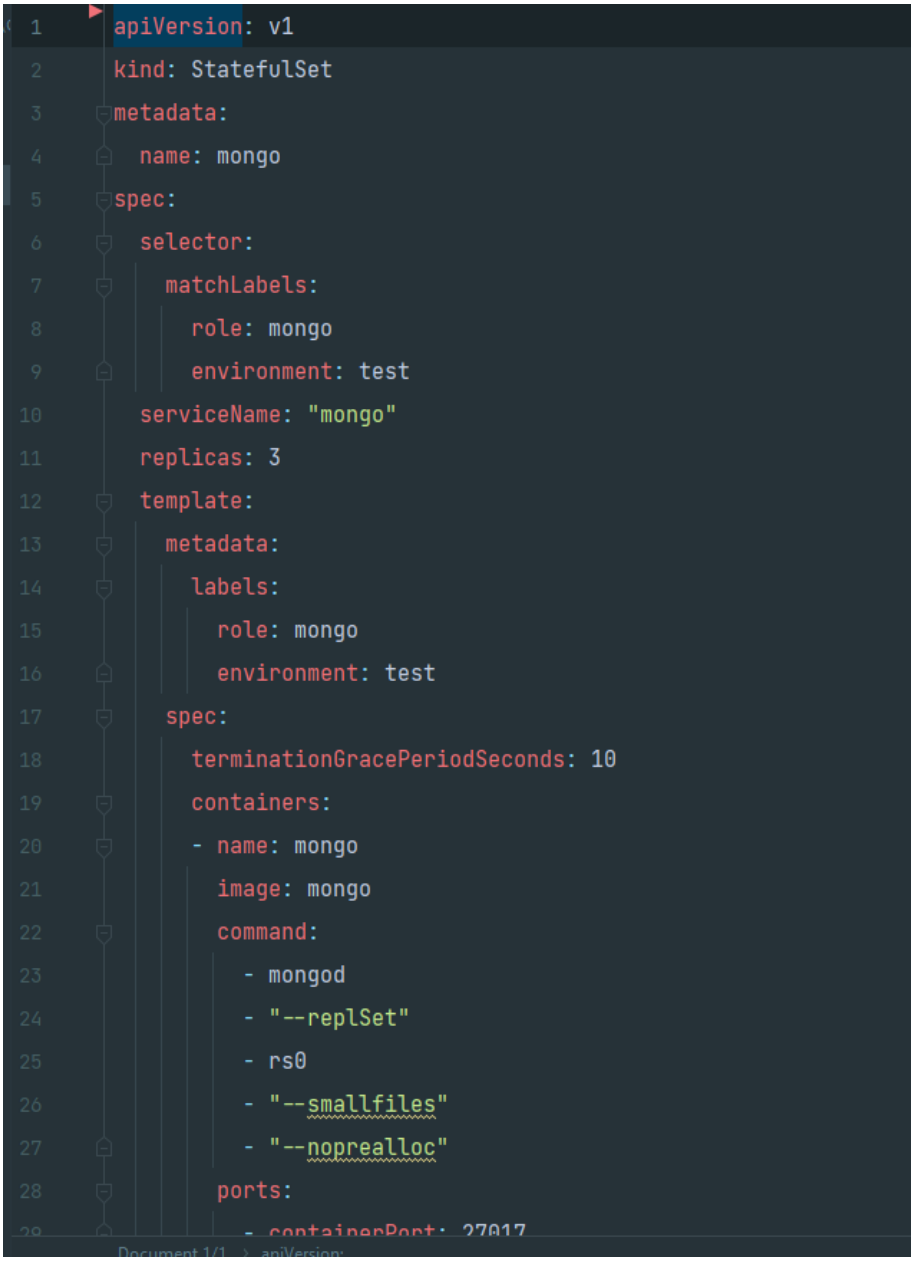
اجرای دیتابیس خود با استفاده از توصیف stateful set و جایگزین کردن آن با deployment

1. دلیل استفاده: همانطور که می‌دانیم منابعی که در deployment استفاده می‌شوند، دارای state نیستند (stateless هستند) یعنی پادهای مختلف از deployment ساخته می‌شوند و state آنها ماندگار و ابدی نیست و بعد از اینکه پاد از بین رفت، state آن نیز حذف می‌شود.

   ولی در stateful set اینگونه نیست و داده‌ها در مقابل restart و یا از بین رفتن پادها مقاوم هستند به این صورت که از VolumeClaimTemplate استفاده می‌کنند. VolumeClaimTemplate برای هر رپلیکای stateful set یکتاست.

2. توصیف مورد استفاده برای ساخت stateful set:

فایل statefulset.yaml



```yaml
apiVersion: v1
kind: StatefulSet
metadata:
  name: mongo
spec:
  selector:
    matchLabels:
      role: mongo
      environment: test
  serviceName: "mongo"
  replicas: 3
  template:
    metadata:
      labels:
        role: mongo
        environment: test
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: mongo
        image: mongo
        command:
          - mongod
          - "--replSet"
          - rs0
          - "--smallfiles"
          - "--noprealloc"
        ports:
          - containerPort: 27017
```

```
        - containerPort: 27017
        volumeMounts:
          - name: mongo-persistent-storage
            mountPath: /data/db
      - name: mongo-sidecar
        image: cvallance/mongo-k8s-sidecar
        env:
          - name: MONGO_SIDECAR_POD_LABELS
            value: "role=mongo,environment=test"
  volumeClaimTemplates:
    - metadata:
        name: mongo-persistent-storage
      spec:
        storageClassName: "fast"
        accessModes: ["ReadWriteOnce"]
        resources:
          requests:
            storage: 100Gi
```

3. نحوه استفاده از سرویس مستر و رپلیکاها: در روش stateful mongoDB، سه نود stateful mongoDB ساخته می‌شود که یکی از آنها نقش مستر و دوتای دیگر نقش رپلیکا یا slave را دارند. همواره باید دستورات write به مستر فرستاده شود ولی برای دستورات read از هر سه نود می‌توان استفاده کرد. دلیل این اتفاق، جلوگیری از ناسازگاری دیتا در سه نود است.

پیاده‌سازی helm chart جهت خودکار سازی ایجاد منابع و توصیف‌های تعریف شده، بر روی کلاستر کوبرنتیز

1. توضیح ساختار helm chart:

در این ساختار ما فایل‌های templates, values.yaml ، chart.yaml را داریم.

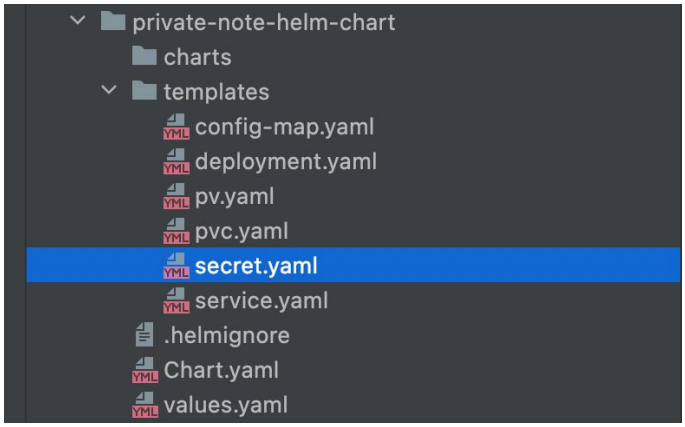
فایل chart.yaml اطلاعات کلی درمورد چارت را می‌دهد.(مثلا ورژن چارت موجود و ...)

پوشه templates که شامل توصیفات منابع کوبرنتیز است.(منابعی مثل deployment و ...) مثالی دیگر تعداد رپلیکاهای دیپلویمنت است که آن را به صورت {{replica.count}} توصیف می‌کنیم.

فایل values.yaml مقدارهای متغیرهایی که در templates توصیف کردیم را مشخص می‌کنیم مثلا تعداد رپلیکاهای دیپلویمنت. (در بعضی مواقع فایل values.yaml به پوشه values تبدیل می‌شود چون از چند فایل values.yaml بسته به نیاز استفاده می‌شود)

2. توضیح مختصر پارامترهای تعریف شده در فایل values:

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % helm install private-note-helm-chart --generate-name
NAME: private-note-helm-chart-1656722759
LAST DEPLOYED: Sat Jul  2 05:16:00 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

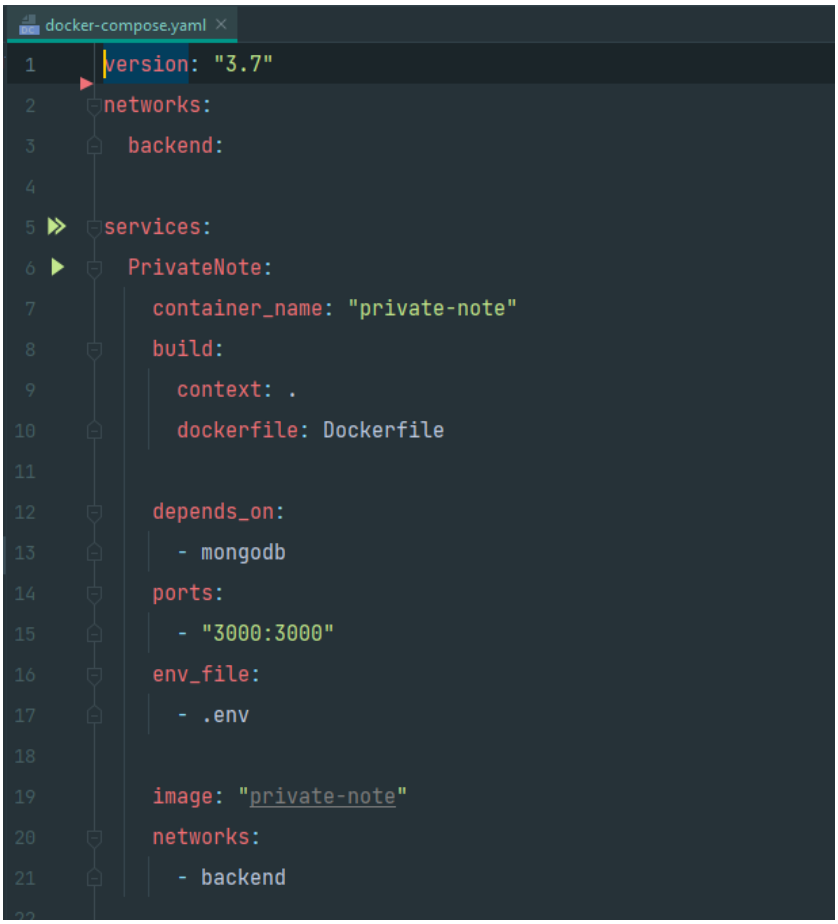


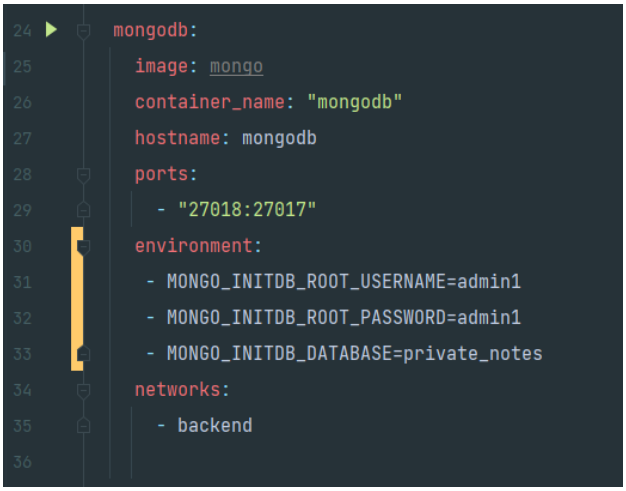
```yaml
deployment:
  name: mongodb-deployment

service:
  name: mongodb-service
  port: 27017
  targetPort: 27017

config:
  name: mongodb-config

secret:
  name: mongodb-secret
  MONGO_INITDB_ROOT_PASSWORD: "admin1"
  MONGO_INITDB_ROOT_USERNAME: "admin1"

pvc:
  name: mongodb-pvc
```

پارامترهای تعریف شده، اسم‌های فایل‌های manifest مونگو است. همچنین برای username, password مونگو نیز از این پارامترها استفاده شده است.

پیاده‌سازی docker compose جهت خودکار سازی ایجاد منابع و وابستگی‌های مورد نیاز پروژه و نهایتا build و اجرای آن

1. محتویات docker compose





در docker compose بالا، دو سرویس تعریف کردیم. اولی برای private-note-app است که به مونگو دیبی وابسته است، یعنی حتما مونگو دیبی باید بالا باشد تا private-note-app اجرا شود. دومی برای مونگو دیبی است که پورت آن، متغیرهای محلی (ENVIRONMENT VARIABLE)ها تعریف شده اند.

برای private-note-app پورت 3000 کانتینر متناظرش را به پورت 3000 میزبان نگاشت می‌کنیم. برای مونگودیبی نیز پورت 27017 کانتینر را به 27017 میزبان نگاشت می‌کنیم.

نتورک مجازی backend را نیز برای هر دو سرویس در نظر می‌گیریم.

گام چهارم (آزمون پروژه)

با استفاده از port forwarding سرویس ایجاد شده برای پروژه را تست می‌کنیم.

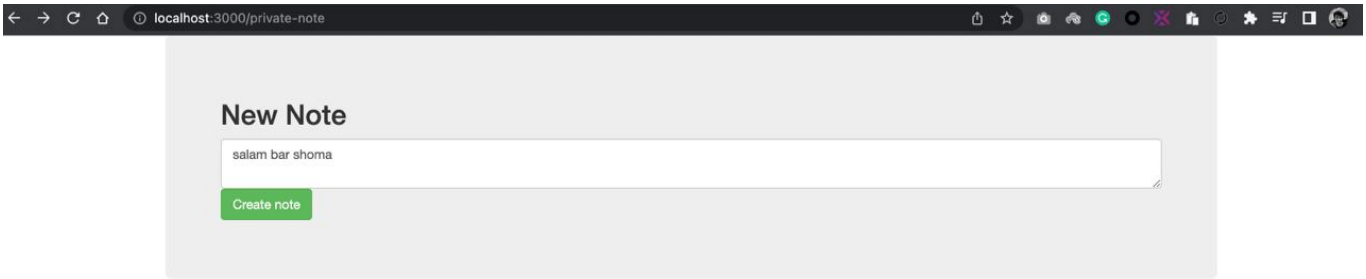
عکس زیر مربوط به port forwarding برای سرویس mongodb است.

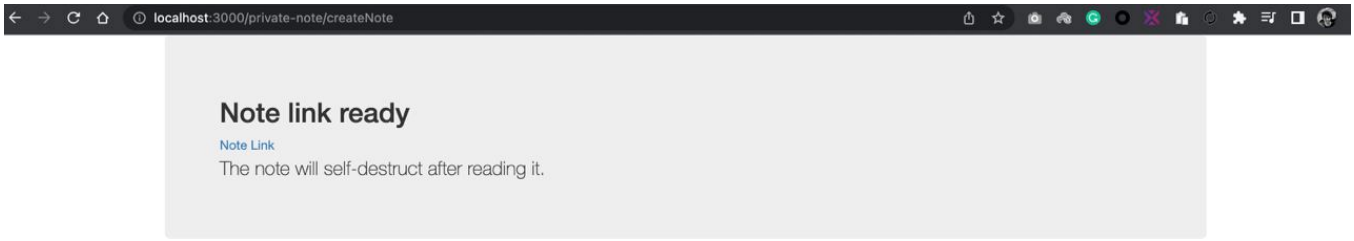
```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl port-forward service/mongodb-service 27017:27017
Forwarding from 127.0.0.1:27017 -> 27017
Forwarding from [::1]:27017 -> 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
Handling connection for 27017
```

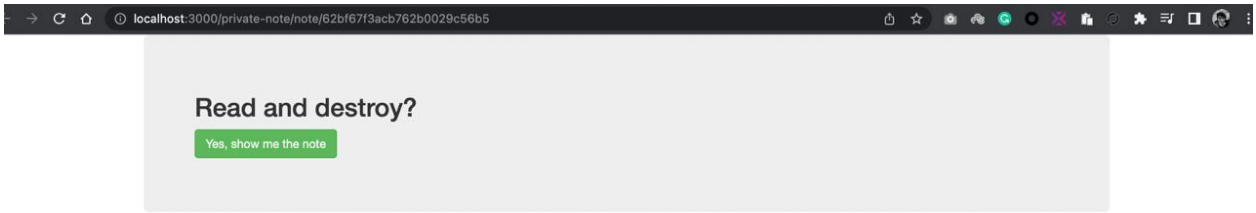
عکس زیر مربوط به port forwarding برای سرویس private-note-app است.

```
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project %
hoseinlook@hoseins-MacBook-Pro FInal_CloudComputing_Project % kubectl port-forward service/private-note-app-service 3000:3000
Forwarding from 127.0.0.1:3000 -> 3000
Forwarding from [::1]:3000 -> 3000
```

عکس‌های زیر مربوط به تست هستند.







عکس زیر مربوط به دیتابیس مونگو است که به صورت سرویس در دسترس است.