گام اول

build کردن ایمیج با استفاده از Dockerfile ساخته شده:

```
nargess-MacBook-Pro:k8s narges$ docker build -t  nargessalehi98/private-note-nginx:1.0.1 .
[+] Building 0.1s (2/2) FINISHED
 => [internal] load build definition from Dockerfile                                                                    0.1s
 => => transferring dockerfile: 2B                                                                                      0.1s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                         0.0s
failed to solve with frontend dockerfile.v0: failed to read dockerfile: open /var/lib/docker/tmp/buildkit-mount933108053/Dockerfile: no such file or directory
```

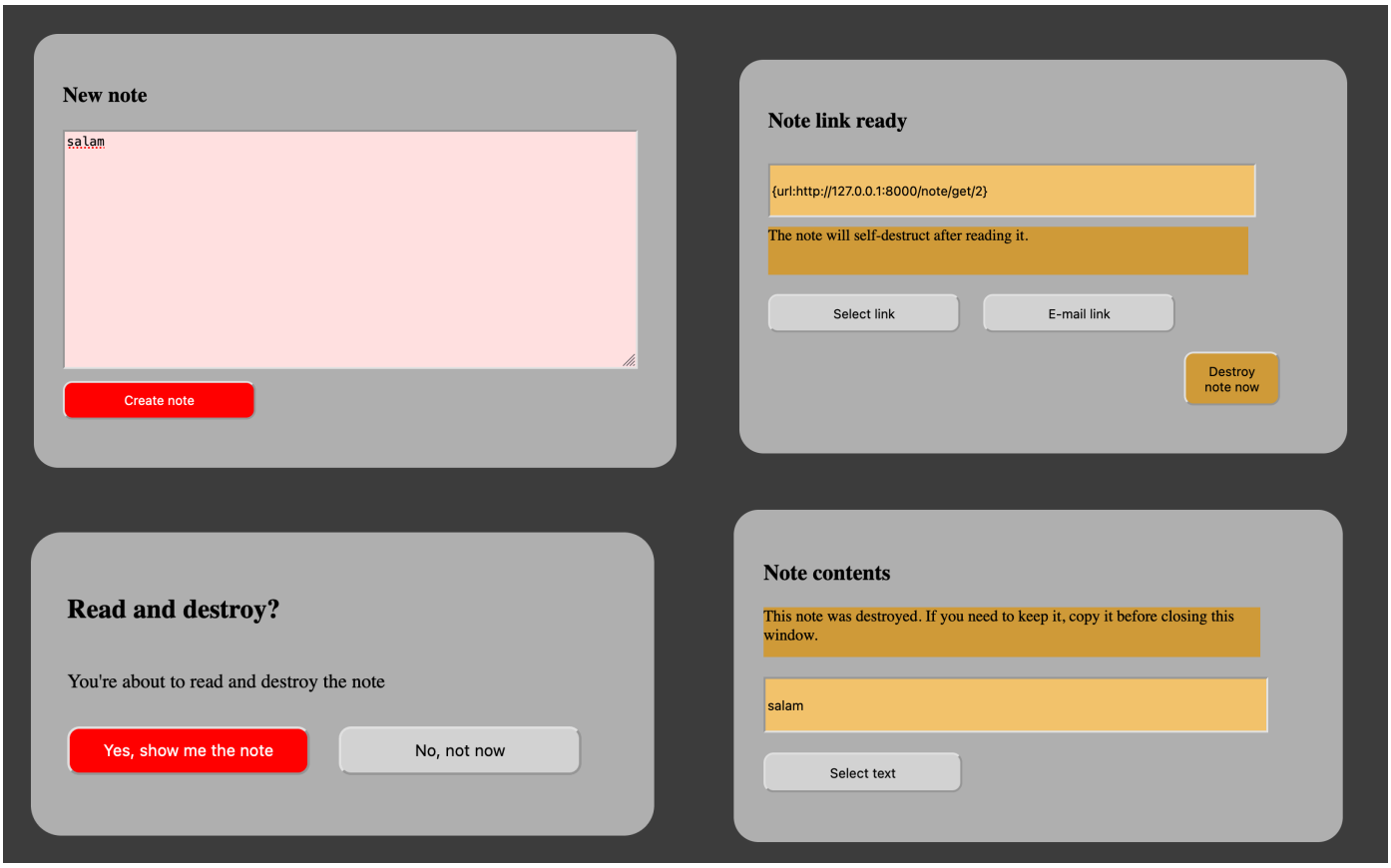
```
nargess-MacBook-Pro:front narges$ docker build -t  nargessalehi98/private-note:0.0.3 .
[+] Building 5.3s (15/15) FINISHED
 => [internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 170B                                                                                    0.0s
 => [internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                         0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                                         3.3s
 => [auth] library/nginx:pull token for registry-1.docker.io                                                            0.0s
 => [1/9] FROM docker.io/library/nginx:latest@sha256:10f14ffa93f8dedf1057897b745e5ac72ac5655c299dade0aa434c71557697ea   0.0s
 => [internal] load build context                                                                                       0.0s
 => => transferring context: 907B                                                                                       0.0s
 => CACHED [2/9] COPY ./index.html /usr/share/nginx/html/index.html                                                     0.0s
 => CACHED [3/9] COPY ./index1.html /usr/share/nginx/html/index1.html                                                   0.0s
 => CACHED [4/9] COPY ./index2.html /usr/share/nginx/html/index2.html                                                   0.0s
 => [5/9] COPY ./index3.html /usr/share/nginx/html/index.html                                                           0.1s
 => [6/9] COPY ./myscripts.js /usr/share/nginx/html/myscripts.js                                                        0.0s
 => [7/9] COPY ./style.css /usr/share/nginx/html/style.css                                                              0.0s
 => [8/9] COPY ./front.conf /etc/nginx/conf.d/                                                                          0.0s
 => [9/9] RUN rm -f /etc/nginx/conf.d/default.conf                                                                      1.2s
 => exporting to image                                                                                                  0.4s
 => => exporting layers                                                                                                 0.3s
 => => writing image sha256:2e77803bb8f5537375ff98c8476a721fb93cc8ff59a88354d38a37c52baf8425                            0.0s
 => => naming to docker.io/nargessalehi98/private-note:0.0.3                                                            0.0s
```

ارسال ایمیج ساخته شده بر روی داکرهاب و نتیجه آن:

```
nargess-MacBook-Pro:front narges$ docker push  nargessalehi98/private-note-nginx:1.0.1
The push refers to repository [docker.io/nargessalehi98/private-note-nginx]
e86357367df8: Layer already exists
ed0220b3f0bf: Layer already exists
78c24f96a324: Layer already exists
bd965242738d: Layer already exists
1c3e88b9a70a: Layer already exists
b4db1fd63586: Layer already exists
967c9cef44c8: Layer already exists
739ce4520e4c: Layer already exists
e7344f8a29a3: Layer already exists
44193d3f4ea2: Layer already exists
41451f050aa8: Layer already exists
b2f82de68e0d: Layer already exists
d5b40e80384b: Layer already exists
08249ce7456a: Layer already exists
1.0.1: digest: sha256:652f2a16c351431a0ff6b88cb5f68e895b1da23a9dd902e3b8fbd34578118948 size: 3229
```

```
nargess-MacBook-Pro:front narges$ docker push  nargessalehi98/private-note:0.0.3
The push refers to repository [docker.io/nargessalehi98/private-note]
2212f7884dcc: Pushed
00d9987abc39: Pushed
9b3edfbdb2f3: Pushed
5f0e9b9f66b5: Pushed
dcba99eeaa57: Pushed
b4db1fd63586: Mounted from nargessalehi98/private-note-nginx
967c9cef44c8: Mounted from nargessalehi98/private-note-nginx
739ce4520e4c: Mounted from nargessalehi98/private-note-nginx
e7344f8a29a3: Mounted from nargessalehi98/private-note-nginx
44193d3f4ea2: Mounted from nargessalehi98/private-note-nginx
41451f050aa8: Mounted from nargessalehi98/private-note-nginx
b2f82de68e0d: Mounted from nargessalehi98/private-note-nginx
d5b40e80384b: Mounted from nargessalehi98/private-note-nginx
08249ce7456a: Mounted from nargessalehi98/private-note-nginx
0.0.3: digest: sha256:23968f544c800ad6ec3227d09c4b7a64f0eec8c255cf50bab5513b7282629704 size: 3229
nargess-MacBook-Pro:front narges$
```

در صورتی که پروژه خود را با استفاده از ایمیج ساخته شده بر روی سیستم شخصی خود تست کردید، تصاویر مربوطه را قرار دهید.



محتویات Dockerfile:

```
FROM nginx:latest
COPY ./index.html /usr/share/nginx/html/index.html
COPY ./index1.html /usr/share/nginx/html/index1.html
COPY ./index2.html /usr/share/nginx/html/index2.html
COPY ./index3.html /usr/share/nginx/html/index.html

COPY ./myscripts.js /usr/share/nginx/html/myscripts.js
COPY ./style.css /usr/share/nginx/html/style.css
COPY ./front.conf /etc/nginx/conf.d/
RUN rm -f /etc/nginx/conf.d/default.conf
```

```
FROM python:3.6-alpine

ENV PYTHONUNBUFFERED=1

ENV  DEFAULT_TIMEDELTA  [0, 0, 1]
ENV  RUNSERVER_PORT  8000
ENV  DATABASE_USERNAME  'admin'
ENV  DATABASE_PASSWORD  'admin'
ENV  DATABASE_URL  'localhost'
ENV  DATABASE_NAME  'privetNote'
ENV  DATABASE_PORT  '27017'

RUN pip install --upgrade pip
ADD ./requirements.txt /requirements.txt
RUN pip install -r /requirements.txt
RUN mkdir -p /home/pn

WORKDIR /home/pn
COPY . .

EXPOSE 8000
RUN chmod +x runserver.sh
ENTRYPOINT ./runserver.sh
```

گام دوم

با استفاده از دستور kubectl get صحت ایجاد منابع بر روی کلاستر را نمایش دهید.

```
[nargess-MacBook-Pro:~ narges$ kubectl get pods
NAME                      READY   STATUS    RESTARTS        AGE
back-657bfd86ff-bfst9     1/1     Running   0               148m
mongodb-standalone-0      1/1     Running   1 (158m ago)    170m
mongodb-standalone-1      1/1     Running   0               106s
mongodb-standalone-2      1/1     Running   0               103s
nginx-6d846f8db8-l29hn    1/1     Running   0               156m
[nargess-MacBook-Pro:~ narges$ kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)           AGE
back         LoadBalancer   10.111.51.16     192.168.99.105   8000:31198/TCP    8h
database     LoadBalancer   10.102.205.176   192.168.99.106   27017:30447/TCP   4h17m
kubernetes   ClusterIP      10.96.0.1        <none>           443/TCP           74d
nginx        LoadBalancer   10.111.126.53    192.168.99.107   80:30176/TCP      45h
[nargess-MacBook-Pro:~ narges$ kubectl get deployments
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
back    1/1     1            1           33h
nginx   1/1     1            1           2d1h
[nargess-MacBook-Pro:~ narges$ kubectl get statefulset
NAME                  READY   AGE
mongodb-standalone    3/3     4h18m
[nargess-MacBook-Pro:~ narges$ kubectl get configmap
NAME                DATA   AGE
kube-root-ca.crt    1      74d
privet-note         7      2d3h
[nargess-MacBook-Pro:~ narges$ kubectl get secret
NAME                  TYPE                                  DATA   AGE
default-token-nk7nt   kubernetes.io/service-account-token   3      74d
mongo-creds           Opaque                                2      39h
nargess-MacBook-Pro:~ narges$
```

آدرس IP پادها و نحوه برقراری ارتباط میان آنها و سرویس ساخته شده

```
NAME↑        TYPE           CLUSTER-IP       EXTERNAL-IP      PORTS            AGE
back         LoadBalancer   10.111.51.16     192.168.99.105   8000►31198       8h
database     LoadBalancer   10.102.205.176   192.168.99.106   27017►30447      4h19m
kubernetes   ClusterIP      10.96.0.1                         https:443►0      74d
nginx        LoadBalancer   10.111.126.53    192.168.99.107   http:80►30176    45h
```

پاد ها به وسیله ای external ip با یکدیگر ارتباط میکنند.

برای دیپلویمنت مربوط به دیتابیس چه تعداد پاد ایجاد کردید؟ دلیل کار خود را توضیح دهید.

سه تا. حالت master-slave برای دیتا بیس مونگو به صورت replicates پیاده سازی میشود. و توصیه میشود حداقل سه instance ساخته شود.

موارد امتیازی

پارامترهای موجود جهت مقیاس کردن خودکار را بیان کنید.

Cpu usage - memory - RPS

شما کدامیک از این پارامترها را برای ایجاد HPA استفاده کردید؟ دلیل خود را شرح دهید.

Cpu usage - برای اینکه response time زیاد نباشد و در دسترس پذیری بالایی داشته باشد.

دستور و یا توصیف مورد استفاده برای ساخت HPA

```
kubectl autoscale deployment back --cpu-percent=50 --min=1 --max=10
```

با استفاده از کامند بالا، هرزمان که درصد استفاده cpu بیشتر از ٪۵۰ شود، یک پاد اضافه میشود تا ماکزیمم تعداد پاد ۱۰ .

دلیل استفاده از stateful set بجای deployment

برای اینکه در حالت deployment پاد ها stateless هستند و با هر بار restart شدن دیتای ان ها

از دست میرود در صورتی که در statefulset پاد ها دیتای خود را روی pvc نگه میدارند.

توصیف مورد استفاده برای ساخت stateful set

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mongodb-standalone
spec:
  serviceName: database
  replicas: 3
  selector:
    matchLabels:
      app: database
  template:
    metadata:
      labels:
        app: database
        selector: mongodb-standalone
    spec:
      containers:
      - name: mongodb-standalone
        image: mongo
        env:
          - name: MONGO_INITDB_ROOT_USERNAME
            value: admin
          - name: MONGO_INITDB_ROOT_PASSWORD
            value: admin
```

نحوه استفاده از سرویس مستر و رپلیکاها

```
rs.initiate({ _id: "MainRepSet", version: 1,
members: [
 { _id: 0, host: "mongod-0.mongodb-service.default.svc.cluster.local:27017" },
 { _id: 1, host: "mongod-1.mongodb-service.default.svc.cluster.local:27017" },
 { _id: 2, host: "mongod-2.mongodb-service.default.svc.cluster.local:27017" } ]});
```

توضیح مختصر ساختار helm chart

Helm مجموعه از فایل هاست که مجموعه ای از منابع کوبرنتیس را توصیف میکند و آنها را در سطح کلاستر deploy میکند. برای ساده سازی مراحل نصب یک اپلیکشن بروی کلاستر استفاده میشود.

محتویات و توضیح مختصر پارامترهای تعریف شده در فایل values مربوط به چارت

مقادیر مشخص شده در فایل values در تمپلیت های چارت استفاده میشوند و برای هرکدام از سرویس های بخش های مربوط به خود را دارند. که حاوی اطلاعات فایل های deployment و service است که helm chart در نهایت آنها را apply میکند.

```
back:
  replicaCount: 2
  image:
    repository: nargessalehi98/private-note
    pullPolicy: IfNotPresent
    tag: ""
  nameOverride: ""
  fullnameOverride: "back"
  podAnnotations: {}
  podSecurityContext: {}
  securityContext: {}
  service:
    type: NodePort
    port: 8000
    nodePort: 31000
  env:
    databasePort: 31001
    databaseName: privetNote
    databaseURL: 192.168.99.107
    databaseAdmin: admin
    databasePassword: admin
  resources:
    limit:
      cpu: 200m
      memory: 256Mi
    requests:
```

```yaml
      cpu: 300m
      memory: 512Mi
  autoscaling:
    enabled: enable
    minReplicas: 2
    maxReplicas: 4
    targetCPUUtilizationPercentage: 80
  nodeSelector: {}
  tolerations: []
  affinity: {}
front:
  replicaCount: 1
  image:
    repository: nargessalehi98/private-note-nginx
    pullPolicy: IfNotPresent
    tag: ""
  nameOverride: ""
  fullnameOverride: "nginx"
  podAnnotations: {}
  podSecurityContext: {}
  securityContext: {}
  service:
    type: NodePort
    port: 80
    nodePort: 31003
  resources:
    limit:
      cpu: 200m
      memory: 256Mi
    requests:
      cpu: 300m
      memory: 512Mi
  nodeSelector: {}
```

```
  tolerations: []
  affinity: {}
mongo:
  username: admin
  password: admin
  service:
    nodePort: 31001
```

محتویات و توضیح مختصر docker compose پیاده سازی شده

در docker-compose سه سرویس برای بک و فرانت و دیتابیس تعریف کردیم ، ایمیج مربوط به هرکدام را مشخص کردیم و پورت های مورد نظر را مشخص کردیم .

```
version: '3.4'

services:

  config:
    image: np:1.0.11
    ports:
      - 8000:8000
    depends_on:
      - mongo
      - front

  front:
    image: front:1.0.2
    ports:
      - 8080:80

  mongo:
    image: mongo:latest
    ports:
      - 27017:27017
```

آزمون پروژه

خروجی برنامه در موارد بالاتر قرار داده شد.

```
nargess-MacBook-Pro:~ narges$ kubectl port-forward svc/nginx 8000:80
Forwarding from 127.0.0.1:8000 -> 80
Forwarding from [::1]:8000 -> 80
```