به نام او

شهنام فیضیان 810100197

شهریار عطار 810100186

تکلیف کامپیوتری نهم

گزارش کار

روند اجرا

در ابتدا فایل run-cluster.bat در ویندوز و یا run-cluster.sh در لینوکس را اجرا می کنیم. این فایل ها روند ساخت کلاستر، secret ها، configmap و pv, pvc را خودکار کرده و در نهایت deployment ها و service ها را می‌سازد. خروجی اجرای run-cluster.bat در عکس زیر آمده است.

```
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment> .\run-cluster.bat
Creating kind cluster...
Creating cluster "dev-cluster" ...
 • Ensuring node image (kindest/node:v1.33.1) 🖼 ...
 ✓ Ensuring node image (kindest/node:v1.33.1) 🖼
 • Preparing nodes 📦 📦 📦 📦   ...
 ✓ Preparing nodes 📦 📦 📦 📦
 • Writing configuration 📜  ...
 ✓ Writing configuration 📜
 • Starting control-plane 🕹️ ...
 ✓ Starting control-plane 🕹️
 • Installing CNI 🔌  ...
 ✓ Installing CNI 🔌
 • Installing StorageClass 💾  ...
 ✓ Installing StorageClass 💾
 • Joining worker nodes 🚜  ...
 ✓ Joining worker nodes 🚜
Set kubectl context to "kind-dev-cluster"
You can now use your cluster with:

kubectl cluster-info --context kind-dev-cluster

Have a nice day! 👋
Waiting for cluster to be ready...
node/dev-cluster-control-plane condition met
node/dev-cluster-worker condition met
node/dev-cluster-worker2 condition met
node/dev-cluster-worker3 condition met
Kind cluster created successfully
Creating ConfigMap and Secrets...
secret/db-secret-env created
secret/server-secret-env created
configmap/webclient-config created
ConfigMap and Secrets created successfully
Creating PersistentVolume and PersistentVolumeClaim...
persistentvolume/db-pv created
persistentvolumeclaim/db-pvc created
PV and PVC created successfully
Deploying MySQL...
deployment.apps/db-deployment created
service/mysql created
MySQL deployed successfully
Deploying backend server...
deployment.apps/server-deployment created
service/miobook-server created
Backend server deployed successfully
Deploying web client...
deployment.apps/webclient-deployment created
service/nginx-service created
Web client deployed successfully
All components deployed successfully.
```

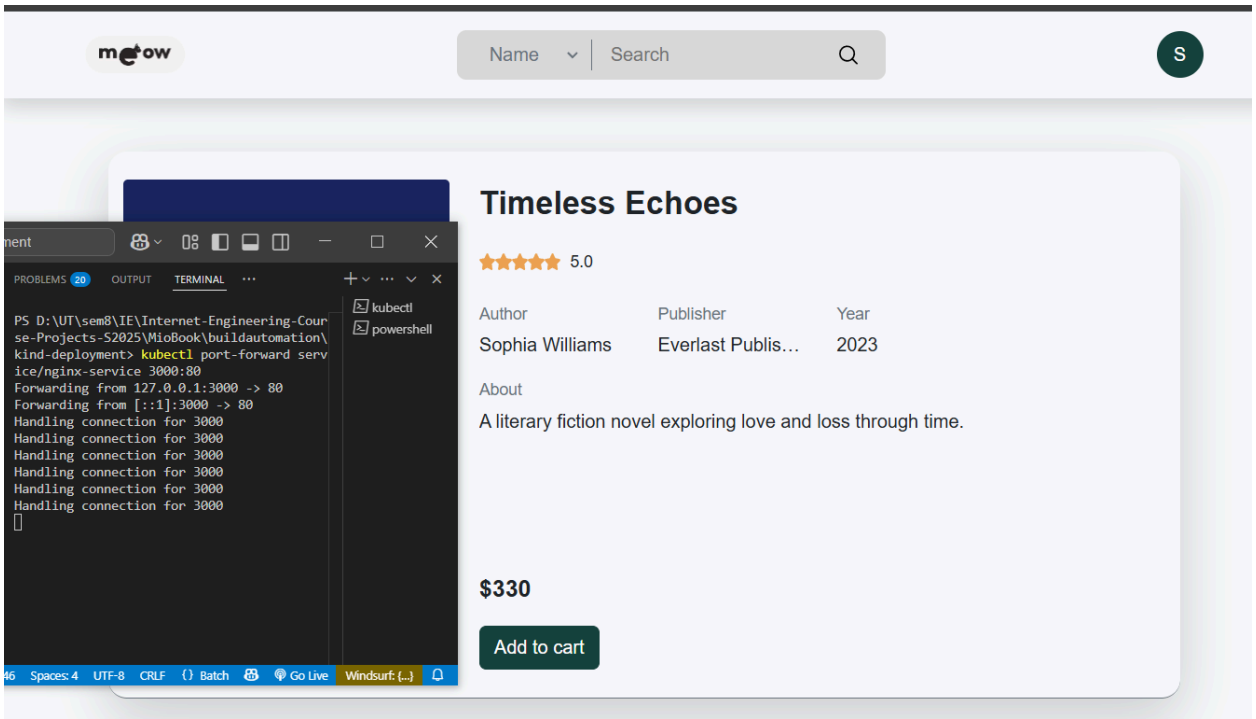
پس از اجرای فایل های مذکور وضعیت موارد ساخته شده را با دستور kubectl get <target> بررسی می‌کنیم که نتیجه آن در عکس زیر آمده است.

```
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment> kubectl get secrets
NAME                TYPE     DATA   AGE
db-secret-env       Opaque   4      26m
server-secret-env   Opaque   4      26m
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment> kubectl get configmap
NAME               DATA   AGE
kube-root-ca.crt   1      26m
webclient-config   1      26m
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment> kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                 STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
db-pv                                      1Gi        RWO            Retain           Available                                        <unset>                          26m
pvc-e720e803-e5be-47aa-85cc-831a40a565a0   1Gi        RWO            Delete           Bound       default/db-pvc    standard       <unset>                          26m
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment> kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
db-pvc    Bound    pvc-e720e803-e5be-47aa-85cc-831a40a565a0   1Gi        RWO            standard       <unset>                 26m
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment> kubectl get service
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP    27m
miobook-server    ClusterIP   10.96.210.29    <none>        8080/TCP   26m
mysql             ClusterIP   10.96.11.170    <none>        3306/TCP   26m
nginx-service     ClusterIP   10.96.165.176   <none>        80/TCP     26m
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment> kubectl get pods
NAME                                  READY   STATUS    RESTARTS      AGE
db-deployment-5b7d6688d7-4scgx        1/1     Running   0             26m
server-deployment-944d968d5-n6svj     1/1     Running   4 (18m ago)   26m
server-deployment-944d968d5-rj5dz     1/1     Running   5 (18m ago)   26m
webclient-deployment-7f5d5765c7-217jp   1/1     Running   0             26m
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment>
```

حال port 80 مربوط به nginx-service را به port 3000 خود forward می‌کنیم تا بتوانیم از روی localhost://3000 به اپلیکیشن وصل شویم.

```
PS D:\UT\sem8\IE\Internet-Engineering-Course-Projects-S2025\MioBook\buildautomation\kind-deployment> kubectl port-forward service/nginx-service 3000:80
Forwarding from 127.0.0.1:3000 -> 80
Forwarding from [::1]:3000 -> 80
```

عکس زیر نیز مربوط به صفحه یک کتاب در کنار port-forward است.



سوال 1)

ClusterIP: این سرویس برای دسترسی به سرویس‌هایی داخل کلاستر استفاده می‌شود. و سرویس مد نظر را روی یک IP مشخص و ثابت در سراسر کلاستر دسترس‌پذیر می‌کند. این IP از بیرون کلاستر قابل مشاهده و دسترس‌پذیر نیست و فقط برای ارتباطات داخلی است.
NodePort: این نوع یک سرویس را روی پورت ثابتی از node هایی که روی آن اجرا می‌شود در دسترس قرار می‌دهد. به عبارتی تضمین می‌شود که سرویس مورد نظر روی پورت مشخص شده بالا بیاید ولی حرفی راجع به node آن نمی‌زند. این سرویس از طریق NodeIP:Port قابل دسترسی هم از داخل کلاستر و هم از خارج کلاستر است. برای دسترسی مستقیم و ساده بدون لود بالانسر مناسب است.
LoadBalancer: یک لود بالانسر تحت کلاود روی یک Public IP ثابت ایجاد می‌کند که سرویس ما را برای عموم روی یک IP ثابت در دسترس قرار می‌دهد. از داخل قابل دسترسی نیست و فقط از بیرون کلاستر و از طریق IP مذکرو می‌توان به سرویس دسترسی داشت.
ExternalName: وقتی می‌خواهیم به یک سرویس خارجی دسترسی پیدا کنیم به گونه‌ای که انگار یکی از سرویس‌های داخلی خودمان است برای آن یک external name تعریف می کنیم. به عبارتی مثل یک DNS alias عمل می‌کند.
Headless: این سرویس لود بالانس داخلی و IP ثابت برای تک‌تک اینستنس‌های یک سرویس را از بین می‌برد و اجازه دسترسی مستقیم به pode IP هایی که سرویس روی آن‌ها اجرا می‌شود می‌دهد. همچنین به هر اینستنس از سرویس مذکور یک hostname ثابت اختصاص می‌دهد تا حتی اگر پایین بیاید و دوباره restart شود باز هم هویتش را از دست ندهد. این سرویس برای HA کردن کامپوننت‌های stateful مناسب است. زیرا در آن‌ها مهم است که هر نود هویت مشخص و مستقلی داشته باشد و همچنین لود بالانس رندوم در هر بخش از ارتباط بعضا باعث بروز مشکل در ارتباط ما با کامپوننت stateful می‌شود.

سوال 2)

namespace ها یک کلاستر مجازی درون کلاستر ما هستند و از آن‌ها عموما برای جدا سازی محیط‌های dev, staging, production و ... استفاده می‌کنند. کوبر به طور پیش‌فرض چهار namespace به نام‌های default, kube-system, kube-public و kube-node-lease دارد. default جایی است که اگر namespace ای را مشخص نکنیم resource های ما به صورت پیش‌فرض درون آن ساخته می‌شوند. kube-system محل قرار گیری کامپوننت‌های سیستمی خود کوبر است. kube-public برای تمامی کاربران قابل مشاهده است و در نهایت kube-node-lease برای heartbeats و health-checking نودها استفاده می‌شود.

برای عوض کردن namespace default ابتدا به کمک دستور

`kubectl create namespace new-namespace` یک فضای نام جدید می‌سازیم. حال می‌توانیم به کمک دستور

```
kubectl config set-context --current --namespace=new-namespace
```

فضای نام کانتکس فعلی را به فضای نام جدیدی که ساختیم تغییر دهیم. همچنین می توانیم فایل kubeconfig را باز کنیم، سپس کانتکس فعلی که روی آن هستیم را پیدا کرده و فضای نام آن را دستی تغییر دهیم.

سوال 3)

فرض کنید اسم deployment ای که با آن سرویس بک‌اند را بالا آورده‌ایم backend باشد. حال با دستور `kubectl scale deployment backend --replicas=5` میتوانیم تعداد رپلیکاهای deployment مان را افزایش دهیم (در این مثال به 5).

حال برای اینکه این کار را به صورت خودکار انجام دهیم به یک Horizontal Pod Autoscaler یا همان HPA نیاز داریم که می‌توانیم برای آن یک yaml فایل درست کنیم که مطابق با IaaC عمل کرده باشیم یا اینکه دستوری مشابه زیر را وارد کنیم.

```
kubectl autoscale deployment backend --cpu-percent=70 --min=3 --max=20
```

در این دستور یک HPA روی deployment ای به اسم backend اعمال کردیم که مشخصات آن هم بیان می‌کند حداقل 3 و حداکثر 20 پاد ساخته شود. همچنین نقطه تریگر کردن اسکیل رو هم روی 70 درصد cpu usage قرار داده‌ایم.

سوال 4)

مهمترین واحد درگیر در این فرایند Controller Manager است. این واحد یک چرخه ابدی به اسم control loop دارد که در آن وضعیت فعلی سیستم (actual state) را با وضعیت مطلوب (desired state) مقایسه می‌کند و در صورت تفاوت اقدام‌های لازم را برای یکی کردن این دو انجام می‌دهد. البته واحدهای Api Server و etcd نیز قطعا درگیر می‌شوند. به این صورت که در مرحله اول درخواست تغییر وضعیت مطلوب از طرف ما به Api Server می‌رود و او وضعیت مطلوب جدید را در etcd ذخیره می‌کند. حال control loop می‌تواند وضعیت مطلوب را از روی etcd بخواند و مابقی کارها را انجام دهد.

نکته: control loop یک ماژول یا یک پروسه زنده درون Controller Manager نیست. بلکه یک design pattern است و controller های مختلفی نظیر ،Deployment Controller ReplicaSet Controller و ... که درون Controller Manager قرار دارند آن را پیاده سازی کرده‌اند و شبیه به life cycle آن که از سه مرحله استخراج وضعیت فعلی، مقایسه با وضعیت مطلوب و اقدام تشکیل شده است رفتار می‌کنند.

سوال 5)

Operator ها را می‌توان custom controller هایی دانست که از control loop design pattern پیروی می‌کنند و ما آنها را دستی به اکوسیستم کوبر بر اساس نیاز خود اضافه می‌کنیم.

می‌توانیم اپلیکیشن‌های پیچیده و stateful را با کمک آنها به همان گونه که یک اپراتور انسانی باید آن را مدیریت کند به صورت خودکار مدیریت کنیم.

سوال 6)

وظیفه liveness probe تشخیص این است که آیا کانتینر مورد بررسی زنده است یا خیر. اگر این probe فیلد شود کوبر کانتینر را می‌کشد و آن را restart می‌کند. وظیفه readiness probe این است که تشخیص دهد آیا کانتینر مورد بررسی برای دریافت ترافیک آماده است یا خیر. اگر جواب مثبت بود load balancer مقداری از ترافیک را به سمت آن می‌فرستد تا کار انجام دهد. در غیر این صورت load balancer صبر می کند تا جواب readiness probe مثبت شود و تا قبل از آن ترافیکی را به آن اختصاص نمی‌دهد. بر خلاف liveness probe که در طول اجرای کانتینر و بعد از وضعیت startup فعال می‌شود، startup probe در لحظات آغازین شروع به کار کردن کانتینر فعال است و چک می‌کند که کانتینر به سلامت این مرحله startup را پشت سر می‌گذارد یا خیر. در صورت اینکه جواب منفی باشد آن را restart می‌کند. هنگامی که startup probe فعال است liveness و readiness probe غیر فعال هستند و منتظرند تا مرحله startup به پایان برسد.

سوال 7)

ReplicaSet مطمئن میشود که در هر لحظه تعداد مشخصی پاد با لیبل مشخص بالا خواهد بود و تعداد پادهای مذکور از این تعداد مشخص کمتر یا بیشتر نخواهد شد. Deployment ها می‌توانند روی ReplicaSet ها سوار شوند و قابلیت‌هایی را به آنها اضافه کنند. از جمله این قابلیت‌ها می‌توان به مدیریت چرخه حیات پادها، مدیریت بروزرسانی و ورژنینگ اپلیکیشن با انواع استراتژی، rollback کردن خودکار در صورت بروز مشکل و ... اشاره کرد.

DaemonSet یک پاد مشخص را روی همه نودها اجرا می‌کند و مطمئن می‌شود همه نودها پاد مذکور را اجرا کنند. عموما برای مدیریت cluster-level agent ها استفاده می‌شود. برای مثال در زمینه‌های monitoring, logging و networking کاربرد دارد.

سوال 8)

فیلد requests میزان حداقل cpu و memory که تضمین می‌شود در اختیار کانتینر قرار می‌گیرد را مشخص می‌کند و فیلد limits حداکثر میزان cpu و memory که به کانتینر اختصاص داده خواهد شد را مشخص می‌کند.