Scenario

Your team manages an online storefront. They want to have a simple service in their Kubernetes cluster that is able to provide a list of productions. Other pieces of the application running as other pods in the cluster will use this service in the future, but for now ally need to is deploy the service's pods to the cluster and create a Kubernetes services to provide access to those pods. The team estimates that you will need four replicas of the service pod for the time being. There is already a testing pod in the cluster that you can use to test your new service once it is created.

There is a public Docker image for the store-productions app called linuxcademycontent/store-products:1.0.0

You will need to do the following:
- Create a deployment for the store-products service with four replicas
- Create a store-products service and verify that you can access it from the busybox testing pods

Steps

- **Create a deployment for the store-products service with four replicas**

1. Log in to the Kubernetes master node

```
$ ssh cloud_user@<Master-node-public-IP-address>
```

2. Create a deployment for the store-products service

```
cloud_user@ip-10-0-1-101:~$ cat << EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: store-products
  labels:
    app: store-products
spec:
  replicas: 4
  selector:
    matchLabels:
      app: store-products
  template:
    metadata:
      labels:
        app: store-products
    spec:
      containers:
      - name: store-products
        image: linuxcademycontent/store-products:1.0.0
        ports:
        - containerPort: 80
EOF
deployment.apps/store-products created
cloud_user@ip-10-0-1-101:~$
```

3. Run the command to check the deployment

```
$ kubectl get deployments
```

```
cloud_user@ip-10-0-1-101:~$ kubectl get deployments
NAME             DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
store-products   4         4         4            4           5m23s
```

4. To see the four replica pods

```
$ kubectl get pods
```

```
cloud_user@ip-10-0-1-101:~$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
busybox                           1/1     Running   1          132m
store-products-576bb96d6d-2ns7x   1/1     Running   0          5m28s
store-products-576bb96d6d-fbscs   1/1     Running   0          5m28s
store-products-576bb96d6d-mp2r8   1/1     Running   0          5m28s
store-products-576bb96d6d-wbd8x   1/1     Running   0          5m28s
```

- **Create a store-products service and verify that you can access it from the busybox testing pods**

1. So now we have store-products deployment and four replicas spun up
2. So now it is time to create a service on top of that store-products deployment

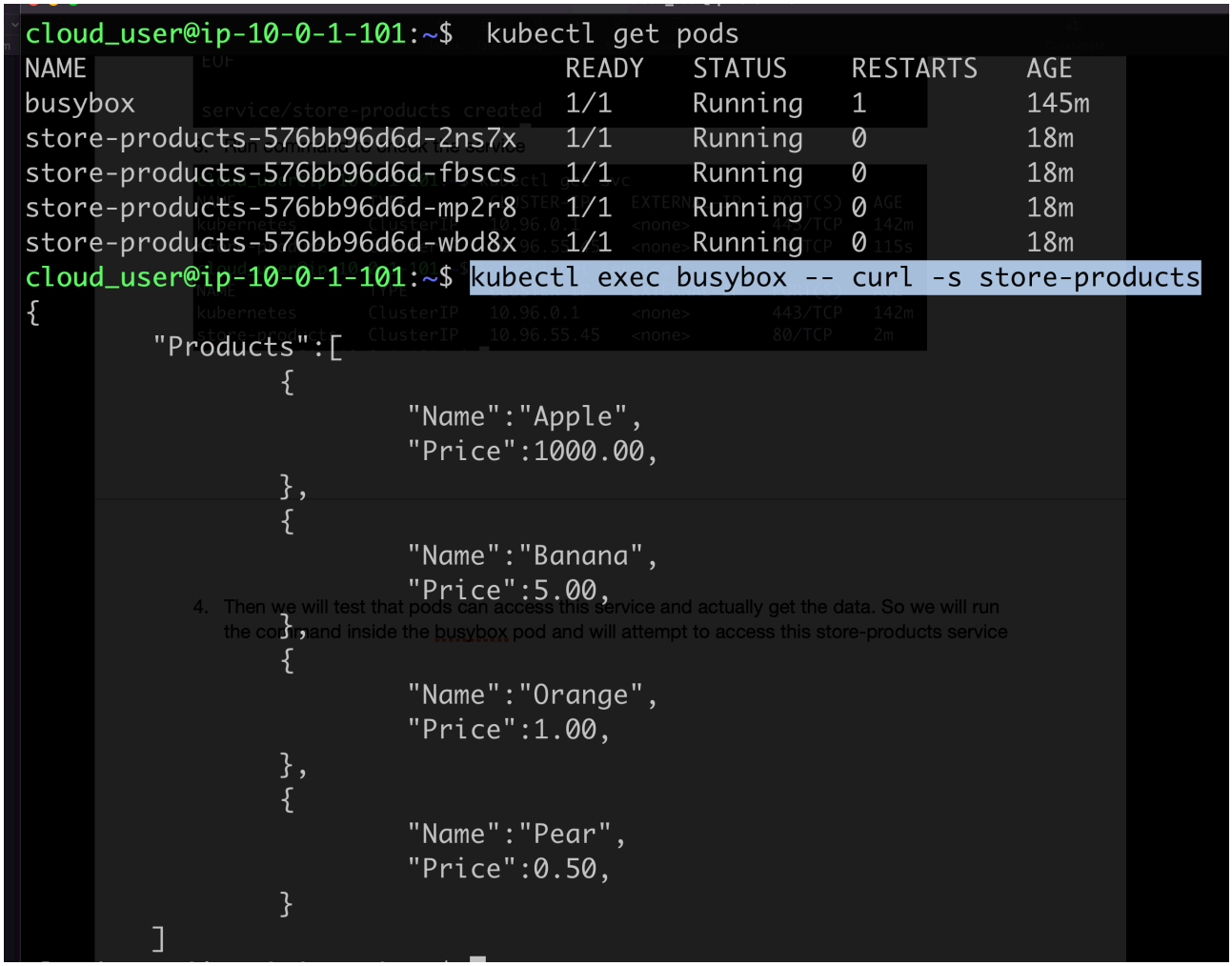
```
cloud_user@ip-10-0-1-101:~$ cat << EOF | kubectl apply -f -
kind: Service
apiVersion: v1
metadata:
  name: store-products
spec:
  selector:
    app: store-products
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
EOF

service/store-products created
```

3. Run command to check the service

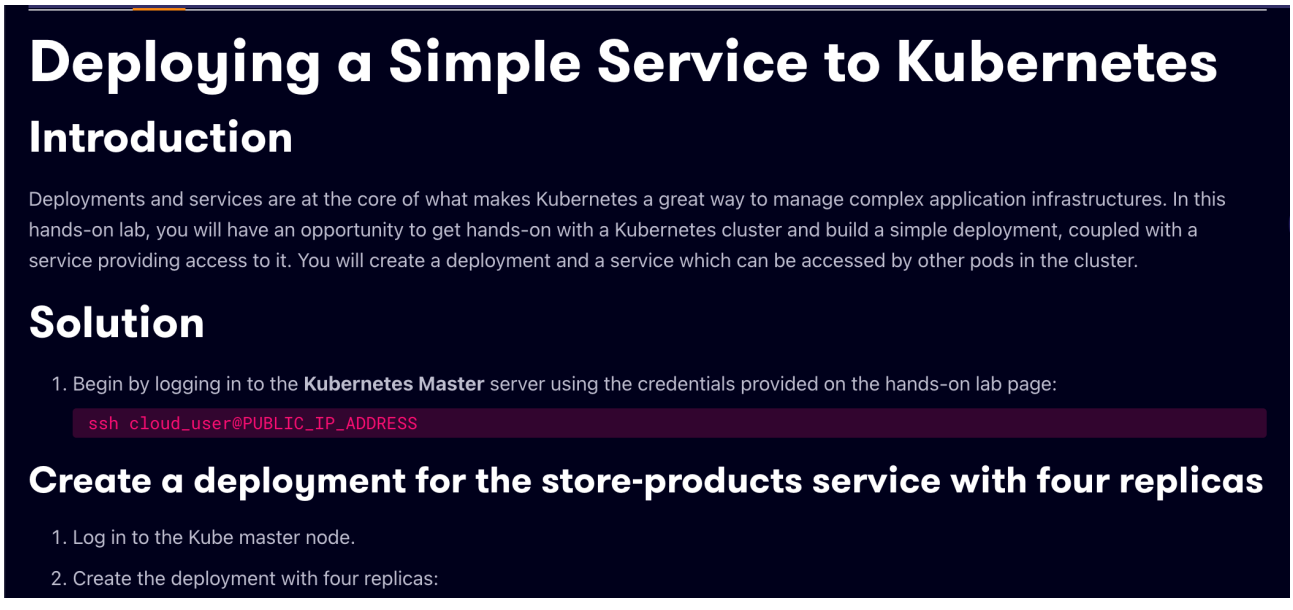
```
cloud_user@ip-10-0-1-101:~$ kubectl get svc
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes       ClusterIP   10.96.0.1      <none>        443/TCP   142m
store-products   ClusterIP   10.96.55.45    <none>        80/TCP    115s
cloud_user@ip-10-0-1-101:~$ kubectl get service
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes       ClusterIP   10.96.0.1      <none>        443/TCP   142m
store-products   ClusterIP   10.96.55.45    <none>        80/TCP    2m
```

4. Then we will test that pods can access this service and actually get the data. So we will run the command inside the busybox pod and will attempt to access this store-products service



And we can see the data ("Products") from one of those four replicas of the store-products service

5. So we were able to successfully get the data from our store-products service from inside of another pod in the cluster using that load balance service that we created.



```
cat << EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: store-products
  labels:
    app: store-products
spec:
  replicas: 4
  selector:
    matchLabels:
      app: store-products
  template:
    metadata:
      labels:
        app: store-products
    spec:
      containers:
      - name: store-products
        image: linuxacademycontent/store-products:1.0.0
        ports:
        - containerPort: 80
EOF
```

Create a store-products service and verify that you can access it from the busybox testing pod

1. Create a service for the store-products pods:

```
cat << EOF | kubectl apply -f -
kind: Service
apiVersion: v1
metadata:
  name: store-products
spec:
  selector:
    app: store-products
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
EOF
```

2. Make sure the service is up in the cluster:

```
kubectl get svc store-products
```

The output will look something like this:

```
NAME             TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
store-products   ClusterIP   10.104.11.230   <none>        80/TCP    59s
```

3. Use `kubectl exec` to query the store-products service from the busybox testing pod.

```
kubectl exec busybox -- curl -s store-products
```