Kubernetes Aula 6

Methods and Techniques for Software Development

2019/2020

Kubernetes Lesson 6

Methods and Techniques for Software Development

2019/2020

kubernetes

Hands on lab

Kubernetes

- Namespaces
- Service Discovery
- Fully Qualified Domain Name

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

How do we network containers together?

- How will we do that?

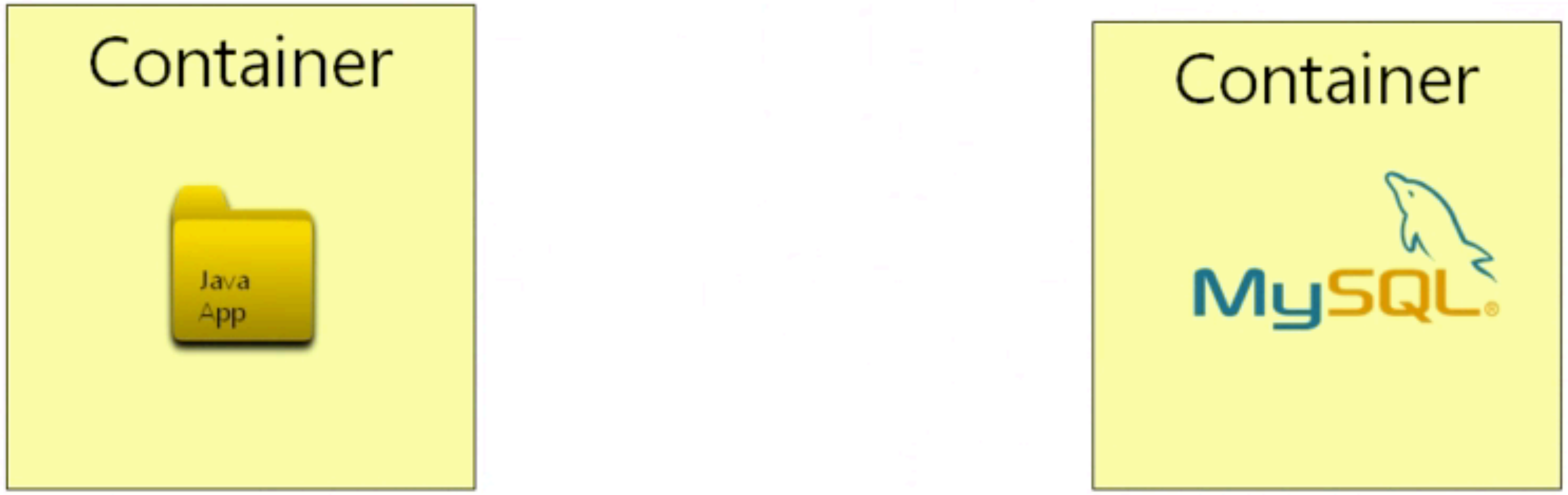
Container
Java App
Container
MySQL

- Lets image I deployed some kind of application container
- I have it here as a java application

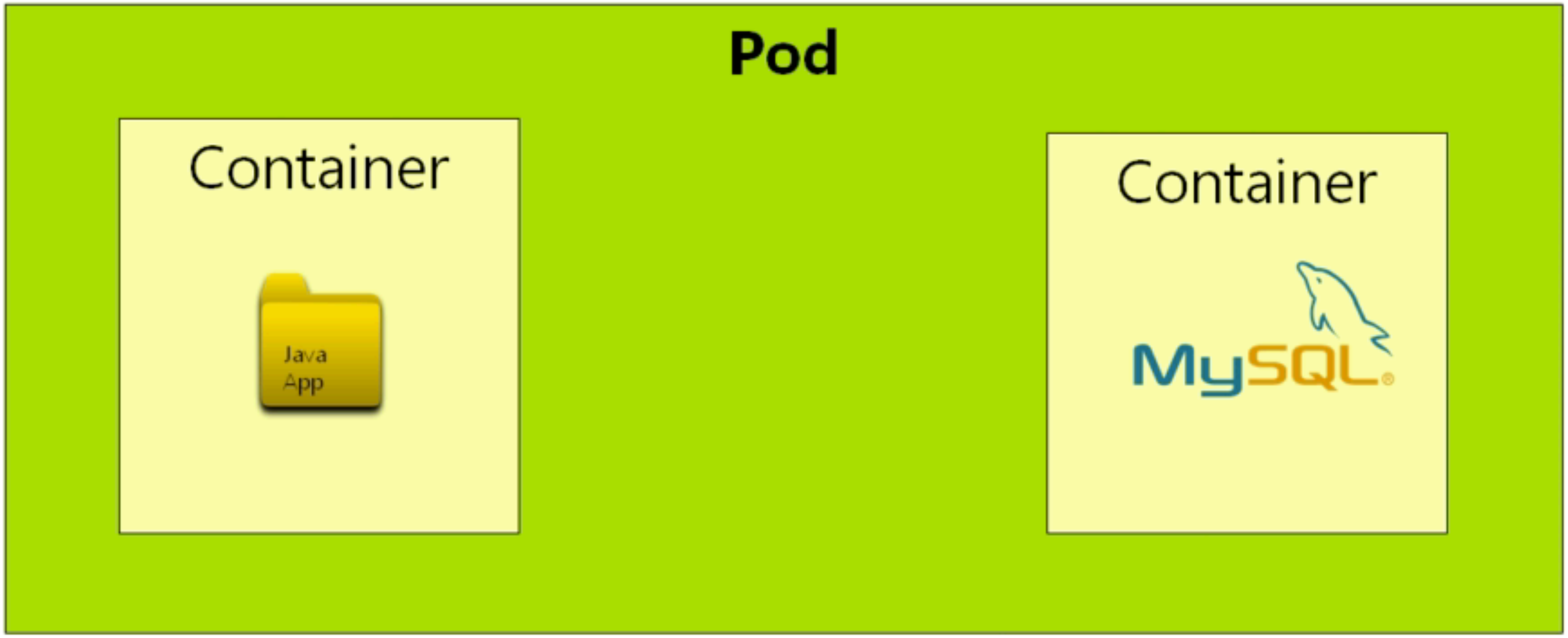
Container
Java App
Container
MySQL

- I want this application to be able to store its data to a database
- I already said that is a bad idea to incorporate a database as well as the application in the same container

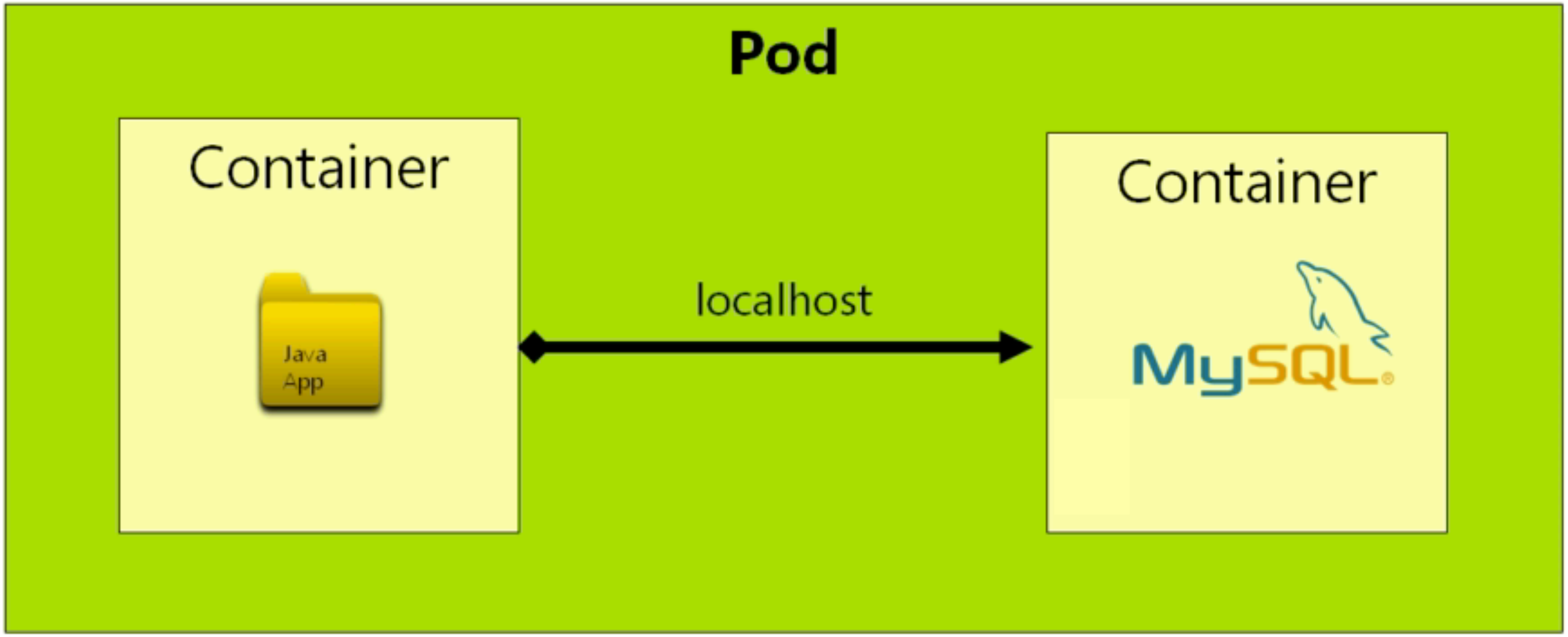
- docker containers are designed to be a single service
- It is possible to expose multiple services in one container but its not something that we want to do
- We want to have a separate container to deploy the image with the database



- We need to have these two containers networking with each other



- In kubernetes if you decide to deploy these two containers into a single Pod its very easy because containers can see each other using localhost



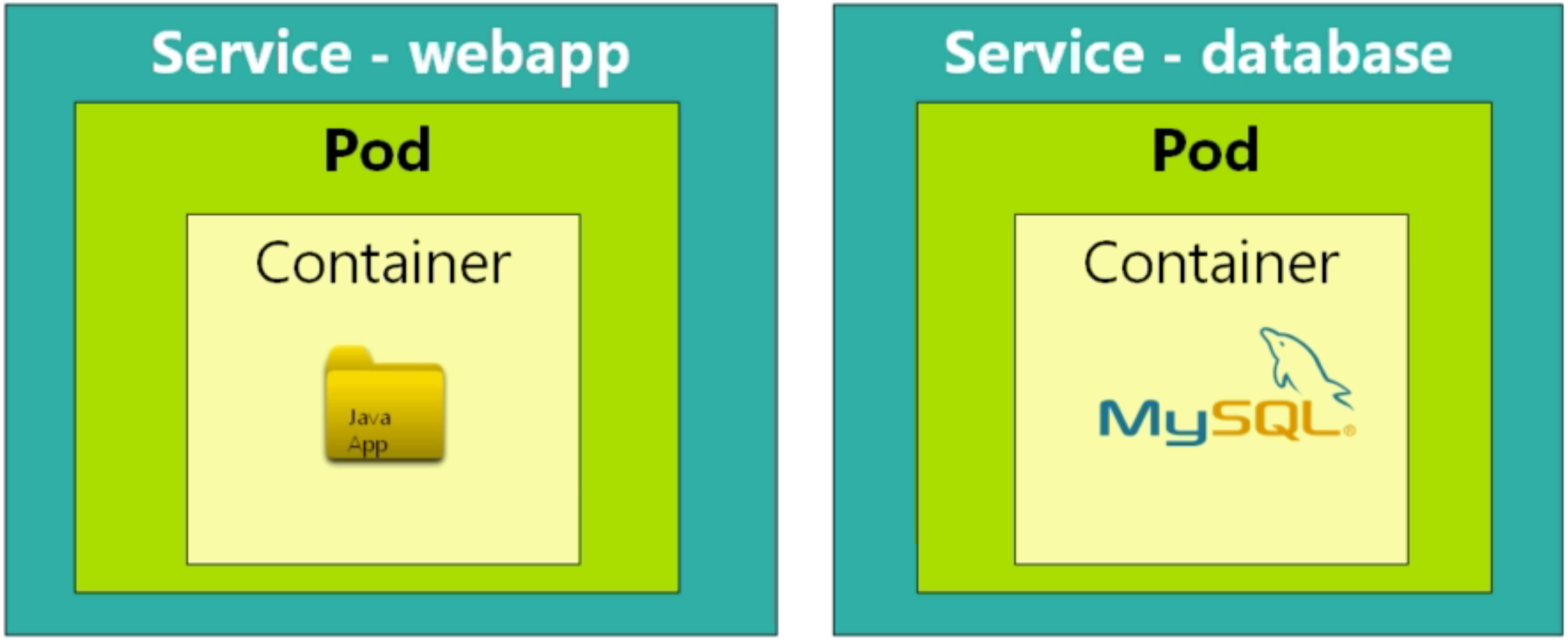
- If I were to write any code in this java application I could just do a lookup of the address `localhost:3306` and I would see the data in mysql container database

- It is acceptable to have multiple containers in a single pod in kubernetes
- But is not a very good idea.... it is not recommended

- Having an application and database container running in the same pod would make the pod more complicated to manage

- If the pods fails we need to find out if it failed because of the application container ou because of the database container - it makes things more complicated

- If we want to deploy the application as well as the database in not only different containers but also in separate pods

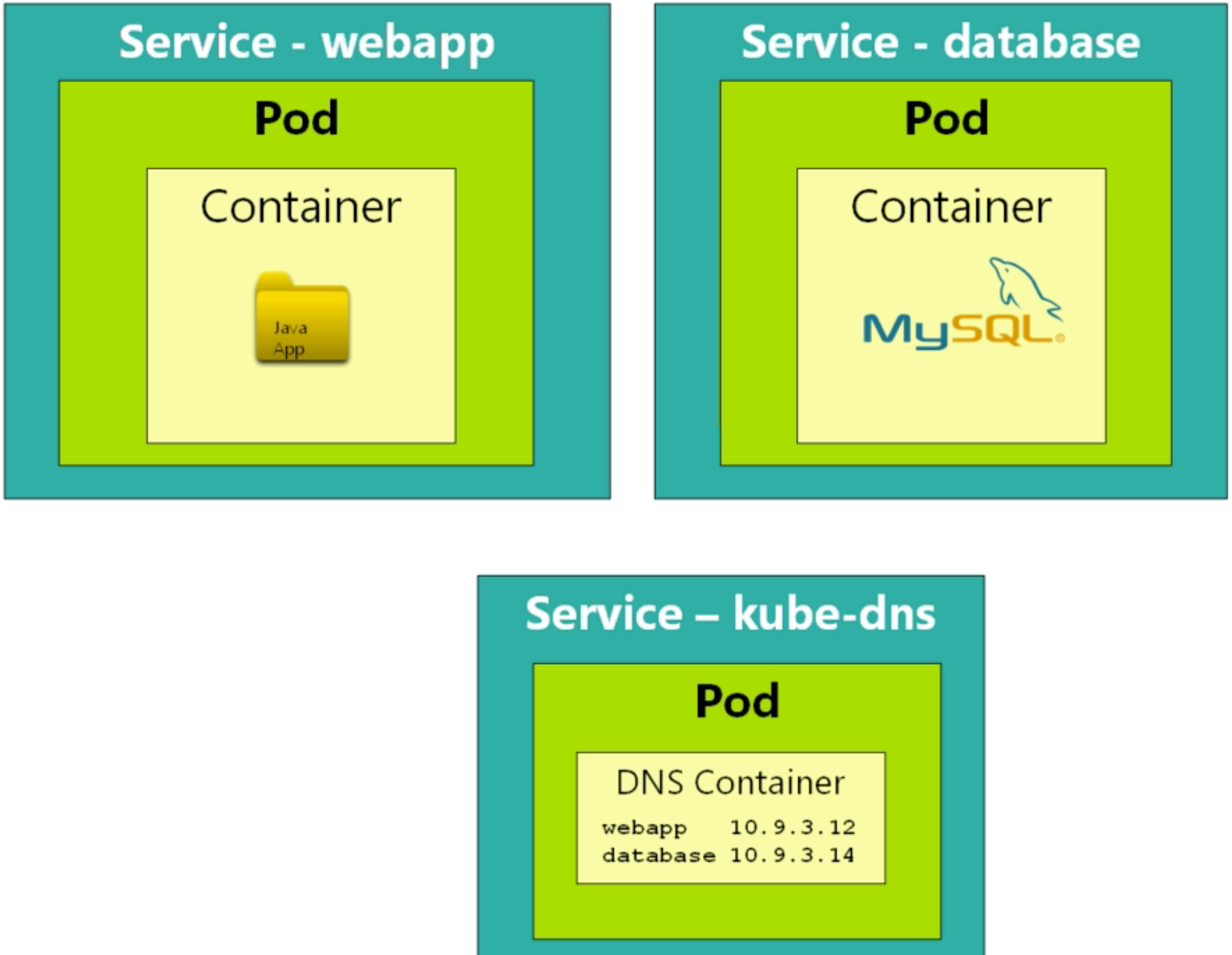


- The problem is, we are not going to know the IP address of that service because it is allocated dynamically by kubernetes

- The next time we run the kubernetes cluster there will be different IP addresses allocated to the services

What is the solution?

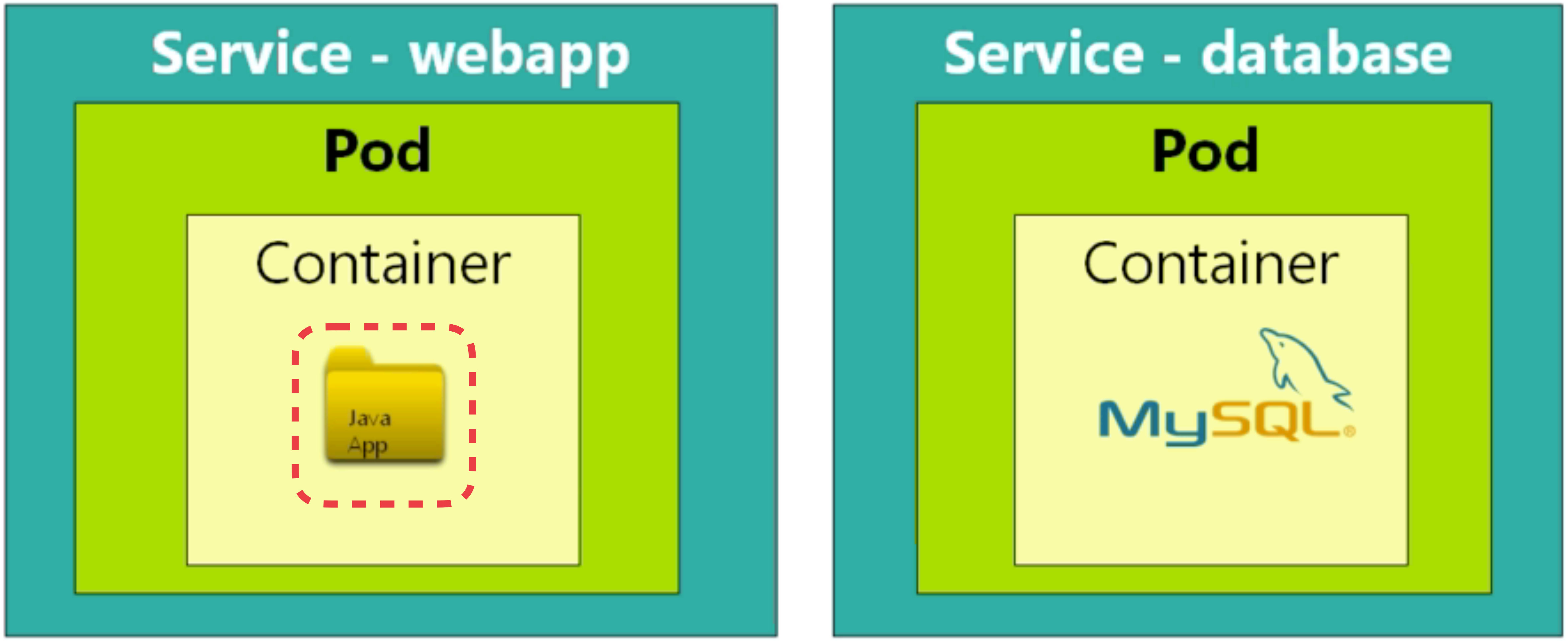
Kubernetes maintains its own private DNS service



```
webapp   10.9.3.12
database 10.9.3.14
```

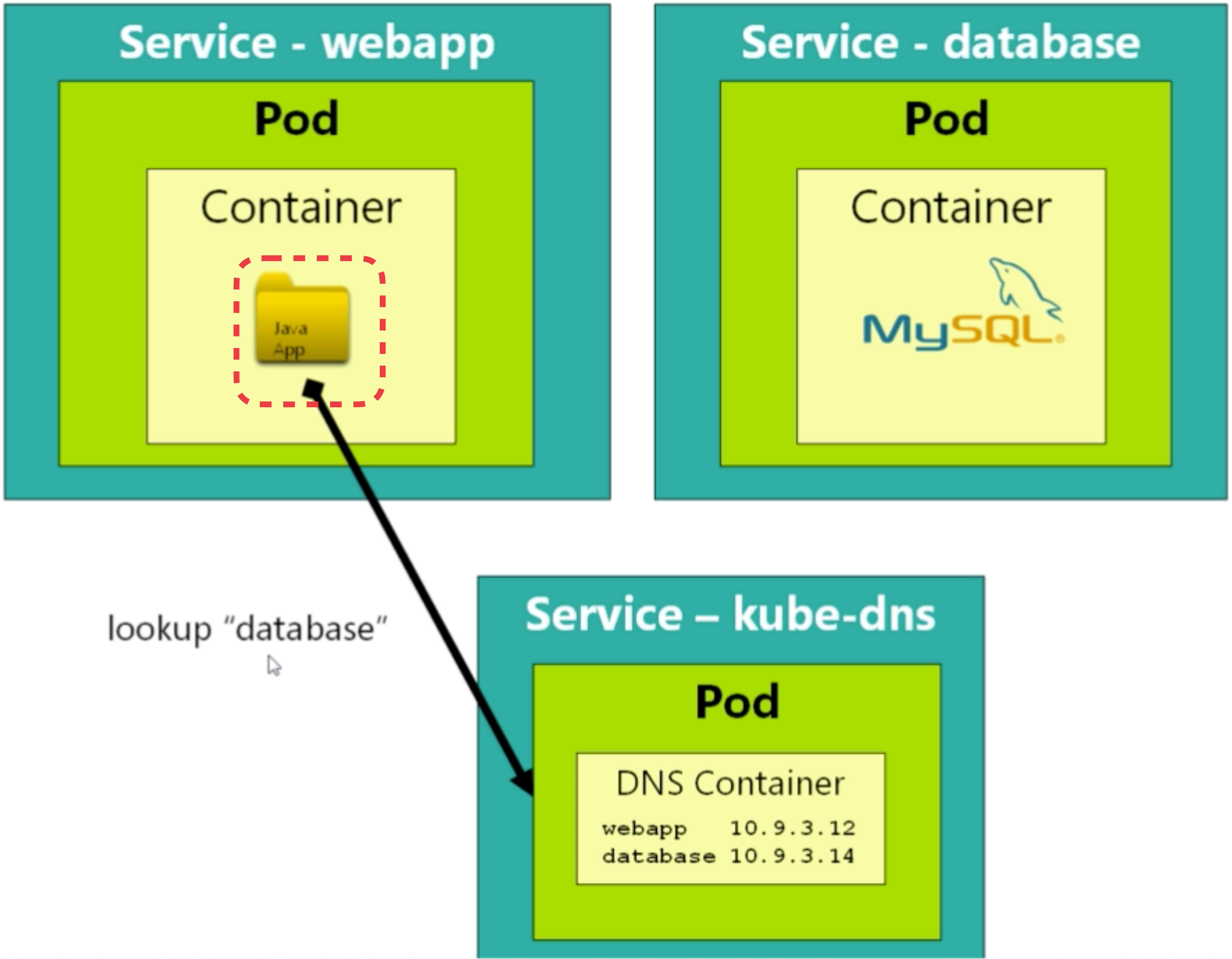
- The service is called kube-dns
- It's not needed to configure this service, it has been running automatically in background
- This service is wrapping a pod named kube-dns

- The DNS services consists in a key-value database
- The keys are labels - the names of the Kubernetes services
- The values are the IP addresses of those services
- Kubernetes has the full responsibility to maintain this service

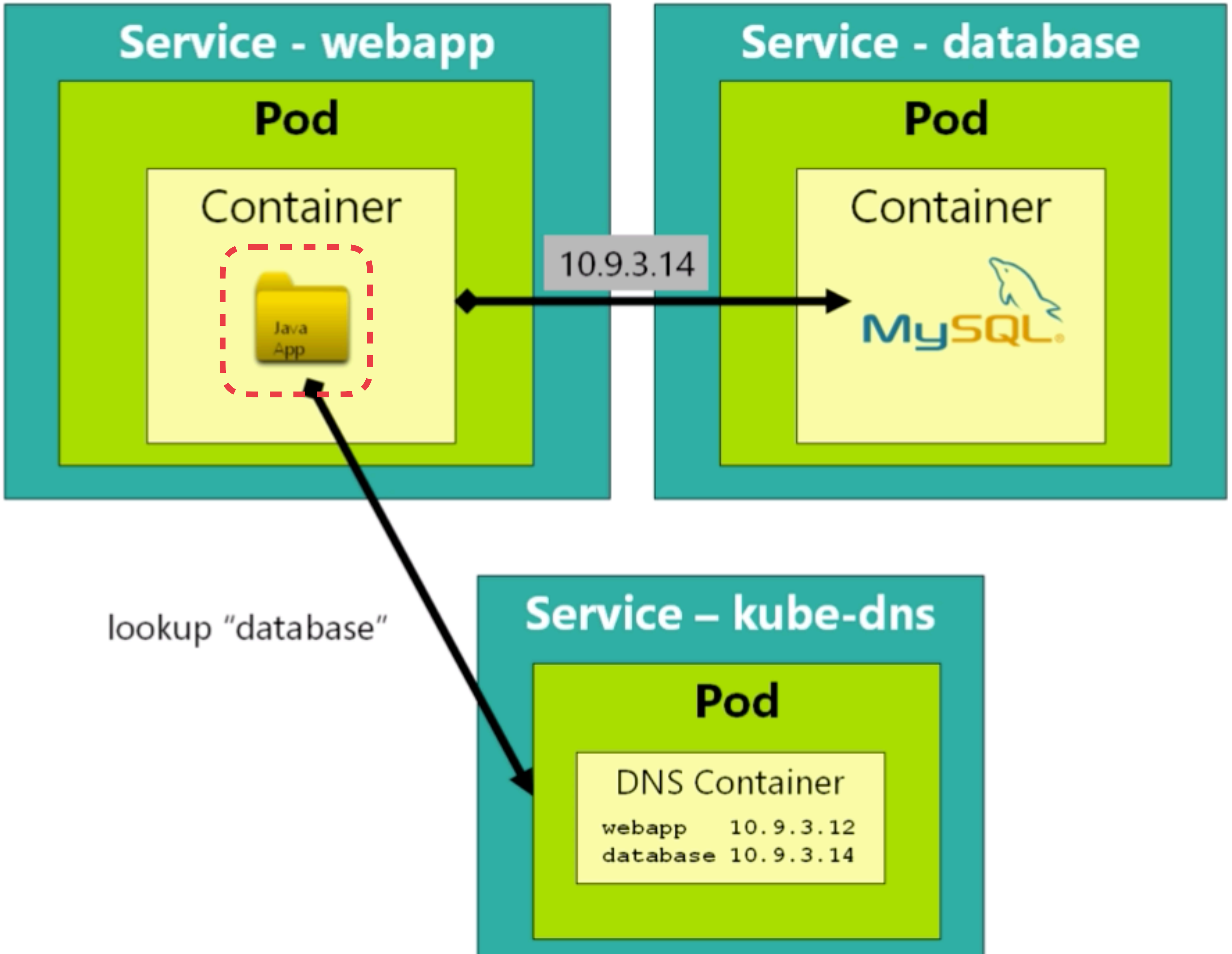


- In our code if we need to refer to the database we could just use the string "database"

P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO



```
webapp   10.9.3.12
database 10.9.3.14
```


Service - webapp
Pod
Container
Java App
10.9.3.14
Service - database
Pod
Container
MySQL
lookup "database"
Service – kube-dns
Pod
DNS Container
webapp 10.9.3.12
database 10.9.3.14

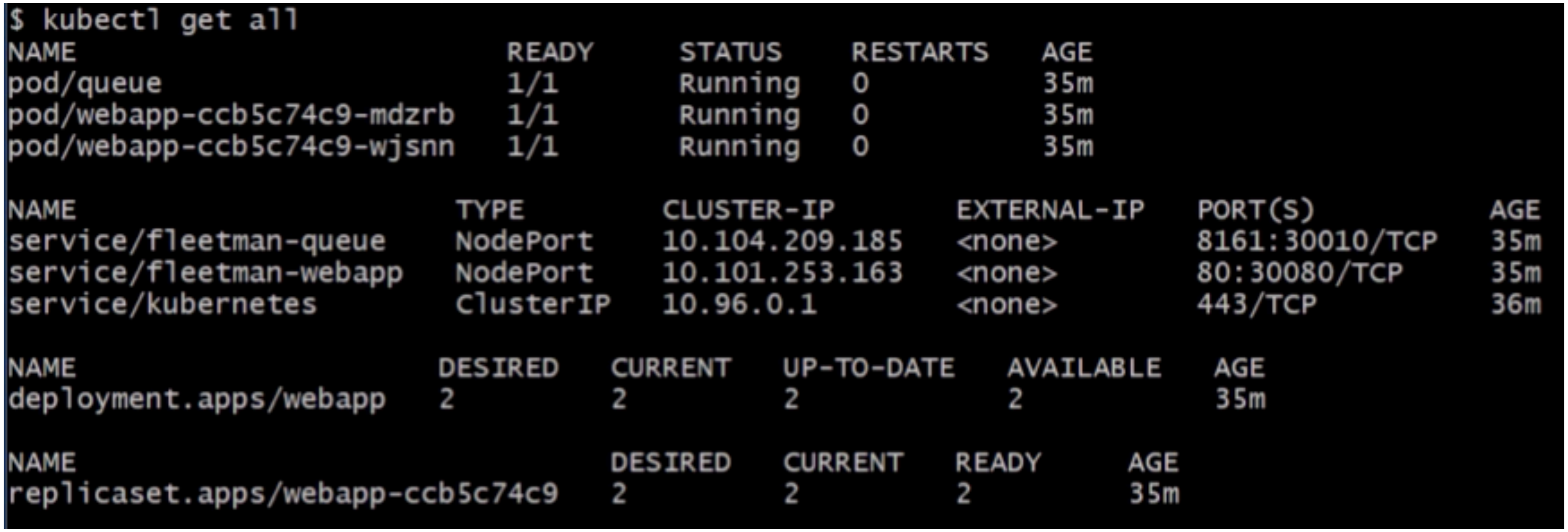
P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Why didn’t we see this kube-dns before?

P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

- `kubectl get all`



```
$ kubectl get all
NAME                            READY     STATUS     RESTARTS    AGE
pod/queue                       1/1       Running    0           35m
pod/webapp-ccb5c74c9-mdzrb      1/1       Running    0           35m
pod/webapp-ccb5c74c9-wjsnn      1/1       Running    0           35m

NAME                         TYPE        CLUSTER-IP        EXTERNAL-IP    PORT(S)            AGE
service/fleetman-queue       NodePort    10.104.209.185    <none>         8161:30010/TCP     35m
service/fleetman-webapp      NodePort    10.101.253.163    <none>         80:30080/TCP       35m
service/kubernetes           ClusterIP   10.96.0.1         <none>         443/TCP            36m

NAME                       DESIRED    CURRENT    UP-TO-DATE    AVAILABLE    AGE
deployment.apps/webapp     2          2          2             2            35m

NAME                               DESIRED    CURRENT    READY      AGE
replicaset.apps/webapp-ccb5c74c9   2          2          2          35m
```

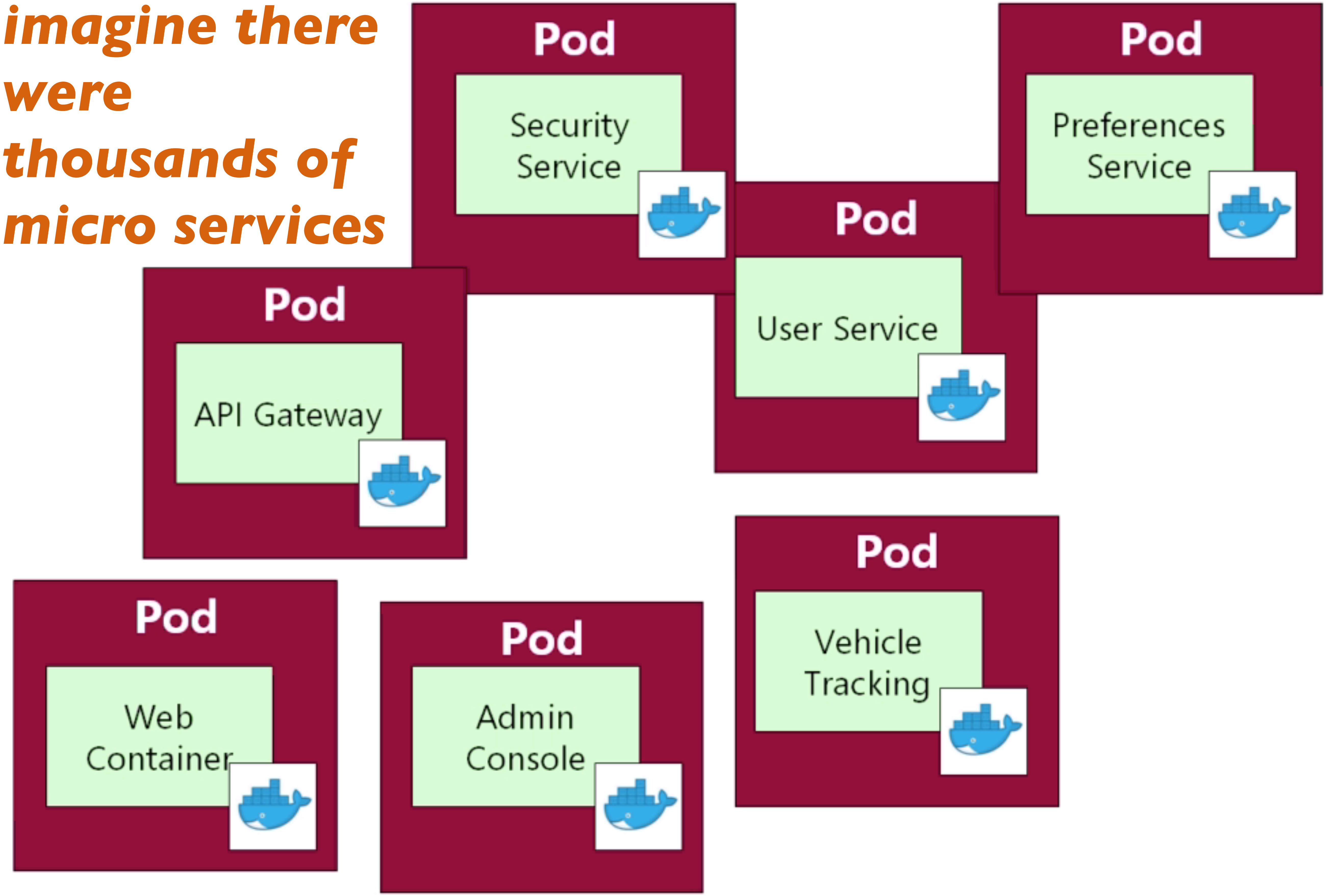
- There’s not any reference to DNS…

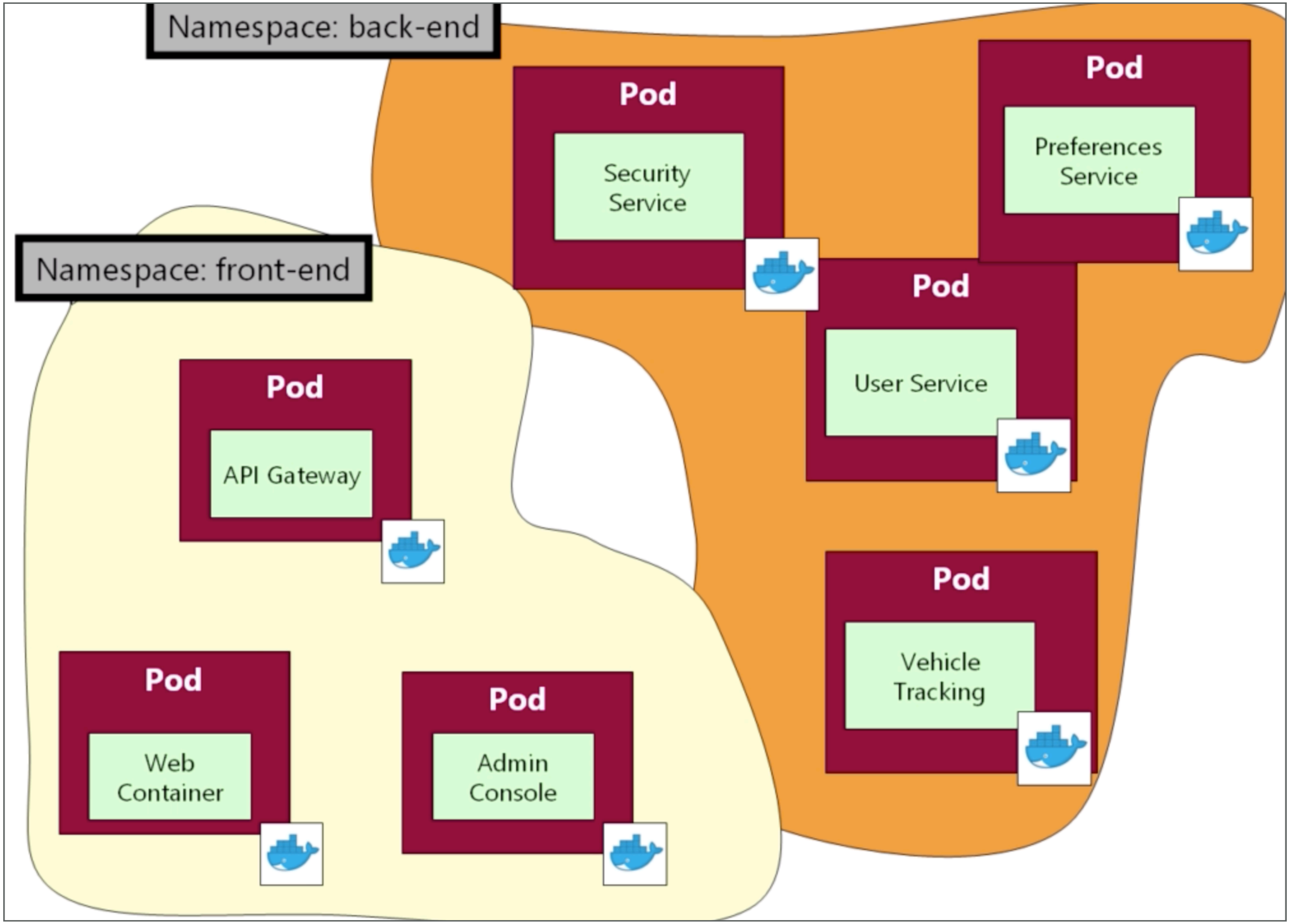
There is a new concept we didn’t talk about…

namespace

- Namespaces are a way of partitioning resources in kubernetes into separate areas

imagine there were thousands of micro services

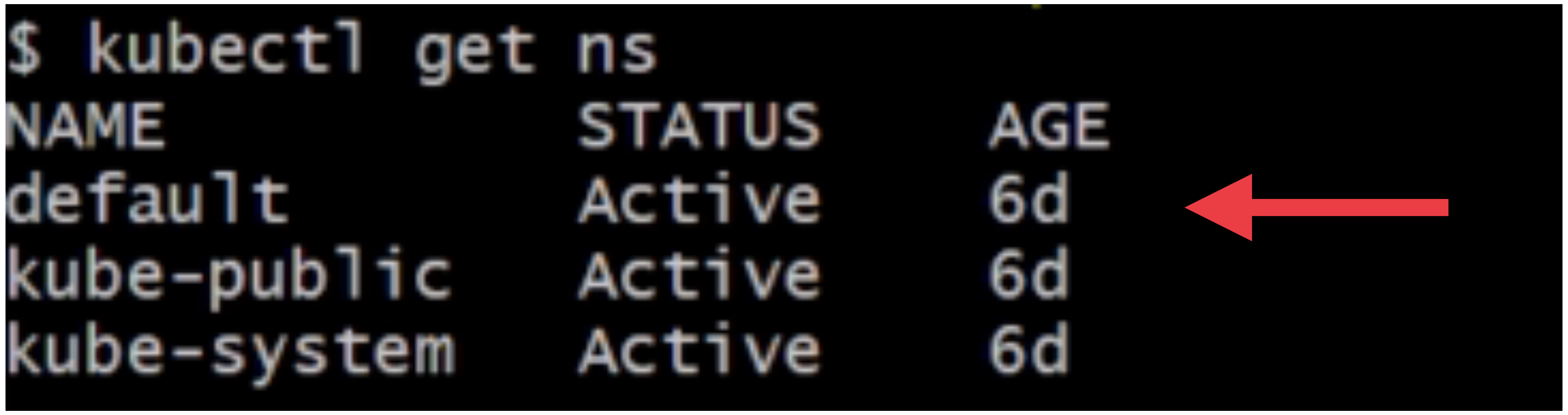



Namespace: back-end
Pod
Security Service
Pod
Preferences Service
Pod
User Service
Pod
Vehicle Tracking
Namespace: front-end
Pod
API Gateway
Pod
Web Container
Pod
Admin Console

- We just created pods and services
- What happens when you don't specify a namespace?
  - That resource is put in the default namespace
- In the same way, when we don't specify the namespace we only see the resources on the default namespace
- Kubernetes comes with additional namespaces that we never seen

Lets verify them!

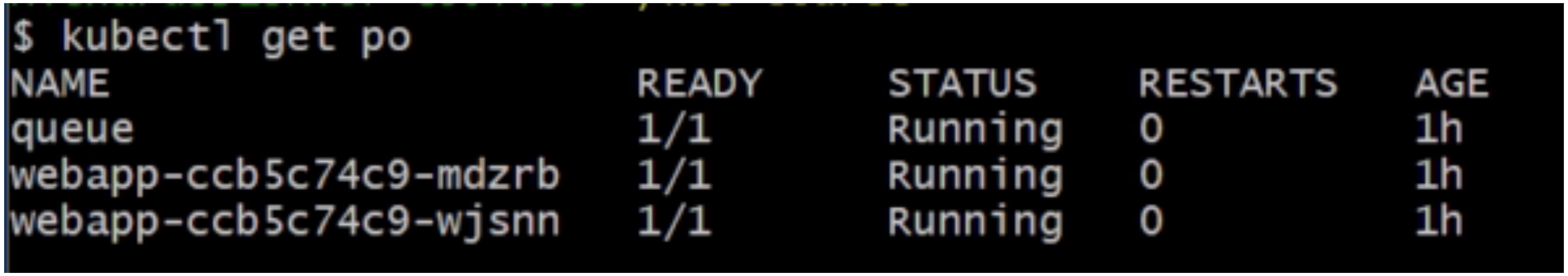
- `kubectl get namespaces`
- `kubectl get ns`



```
$ kubectl get ns
NAME          STATUS    AGE
default       Active    6d
kube-public   Active    6d
kube-system   Active    6d
```

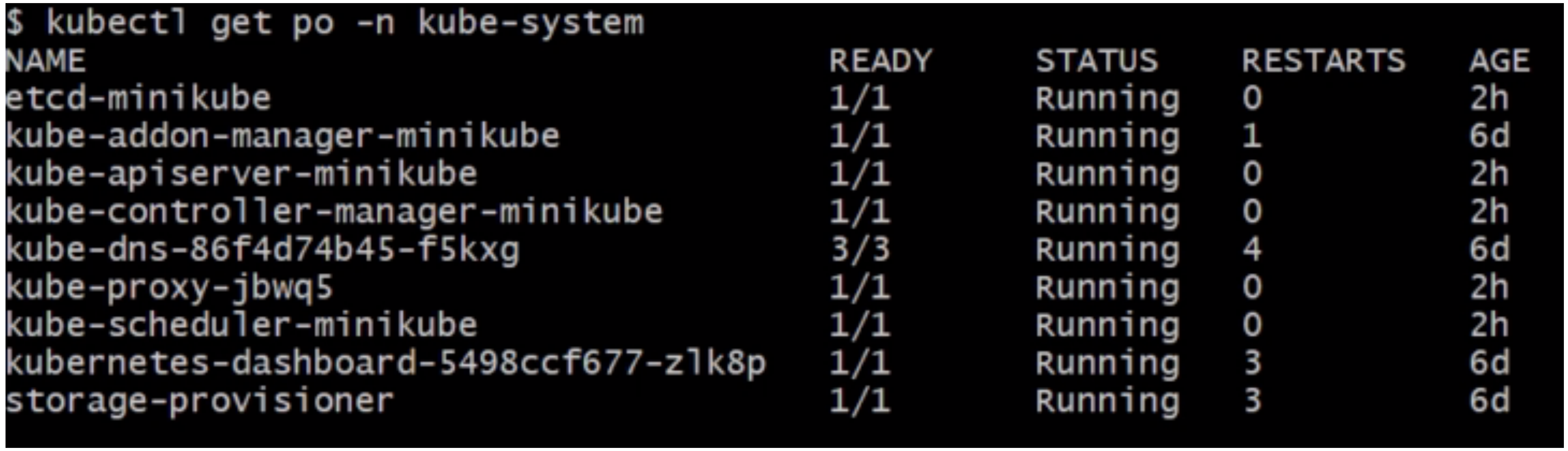
- These area all the namespaces defined in the system
- Let’s see the other namespaces

- kubectl get po ***default ns***



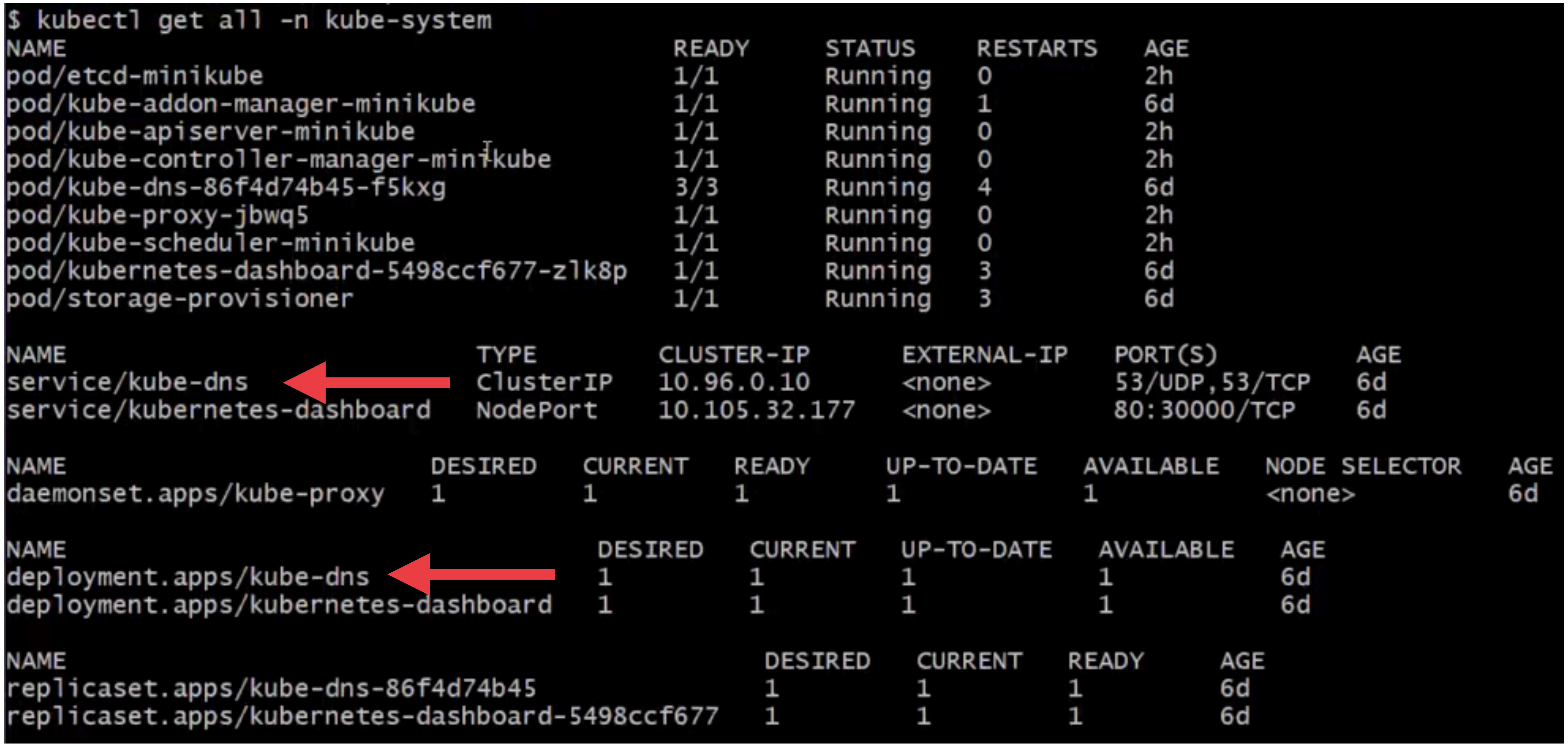
```
$ kubectl get po
NAME                      READY     STATUS    RESTARTS   AGE
queue                     1/1       Running   0          1h
webapp-ccb5c74c9-mdzrb    1/1       Running   0          1h
webapp-ccb5c74c9-wjsnn    1/1       Running   0          1h
```

- kubectl get po -n kube-system



```
$ kubectl get po -n kube-system
NAME                                    READY     STATUS    RESTARTS   AGE
etcd-minikube                           1/1       Running   0          2h
kube-addon-manager-minikube             1/1       Running   1          6d
kube-apiserver-minikube                 1/1       Running   0          2h
kube-controller-manager-minikube        1/1       Running   0          2h
kube-dns-86f4d74b45-f5kxg               3/3       Running   4          6d
kube-proxy-jbwq5                        1/1       Running   0          2h
kube-scheduler-minikube                 1/1       Running   0          2h
kubernetes-dashboard-5498ccf677-zlk8p   1/1       Running   3          6d
storage-provisioner                     1/1       Running   3          6d
```

kubectl get all -n kube-system



```
$ kubectl get all -n kube-system
NAME                                         READY     STATUS    RESTARTS   AGE
pod/etcd-minikube                            1/1       Running   0          2h
pod/kube-addon-manager-minikube              1/1       Running   1          6d
pod/kube-apiserver-minikube                  1/1       Running   0          2h
pod/kube-controller-manager-minikube         1/1       Running   0          2h
pod/kube-dns-86f4d74b45-f5kxg                3/3       Running   4          6d
pod/kube-proxy-jbwq5                         1/1       Running   0          2h
pod/kube-scheduler-minikube                  1/1       Running   0          2h
pod/kubernetes-dashboard-5498ccf677-zlk8p    1/1       Running   3          6d
pod/storage-provisioner                      1/1       Running   3          6d

NAME                            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/kube-dns                ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP    6d
service/kubernetes-dashboard    NodePort    10.105.32.177    <none>        80:30000/TCP     6d

NAME                         DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/kube-proxy    1         1         1         1            1           <none>          6d

NAME                                    DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/kube-dns                1         1         1            1           6d
deployment.apps/kubernetes-dashboard    1         1         1            1           6d

NAME                                              DESIRED   CURRENT   READY     AGE
replicaset.apps/kube-dns-86f4d74b45               1         1         1         6d
replicaset.apps/kubernetes-dashboard-5498ccf677   1         1         1         6d
```

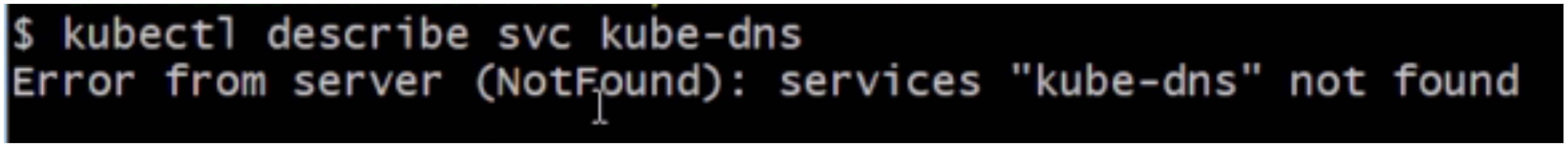
P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Exercise

Describe the service kube-dns

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

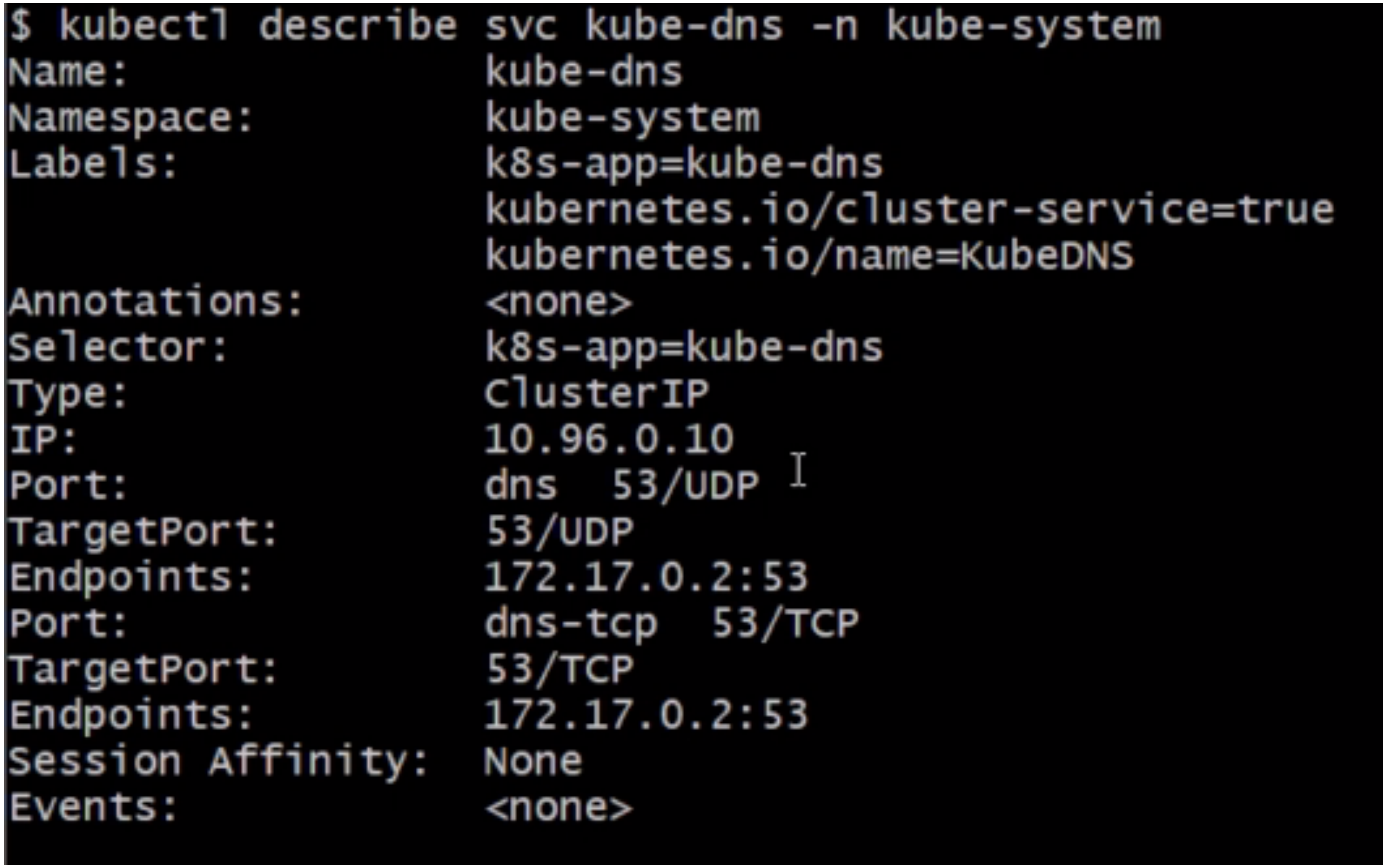
- `kubectl describe svc kube-dns`



```
$ kubectl describe svc kube-dns
Error from server (NotFound): services "kube-dns" not found
```

??

kubectl describe svc kube-dns -n kube-system

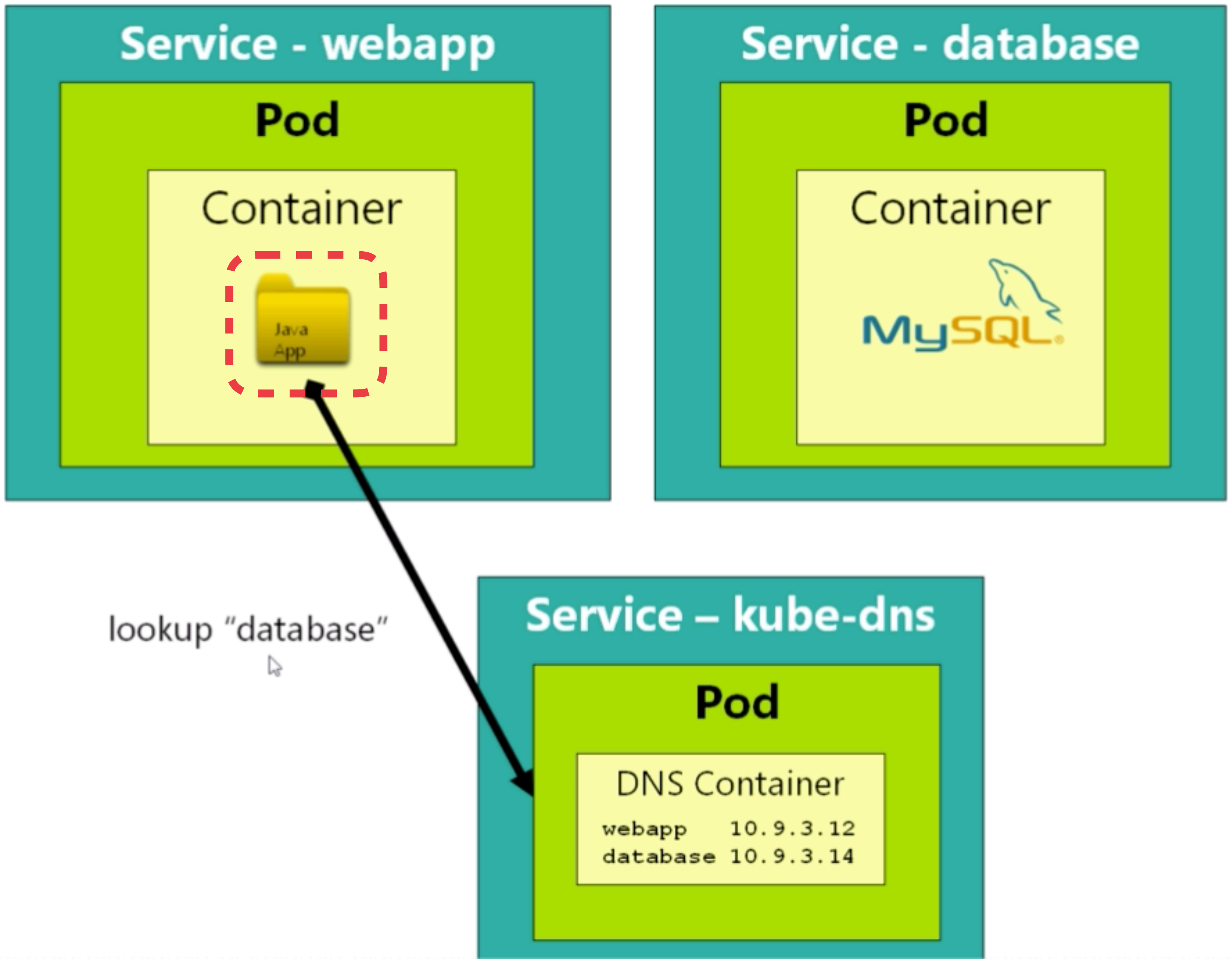


```
$ kubectl describe svc kube-dns -n kube-system
Name:                  kube-dns
Namespace:             kube-system
Labels:                k8s-app=kube-dns
                       kubernetes.io/cluster-service=true
                       kubernetes.io/name=KubeDNS
Annotations:           <none>
Selector:              k8s-app=kube-dns
Type:                  ClusterIP
IP:                    10.96.0.10
Port:                  dns  53/UDP
TargetPort:            53/UDP
Endpoints:             172.17.0.2:53
Port:                  dns-tcp  53/TCP
TargetPort:            53/TCP
Endpoints:             172.17.0.2:53
Session Affinity:      None
Events:                <none>
```

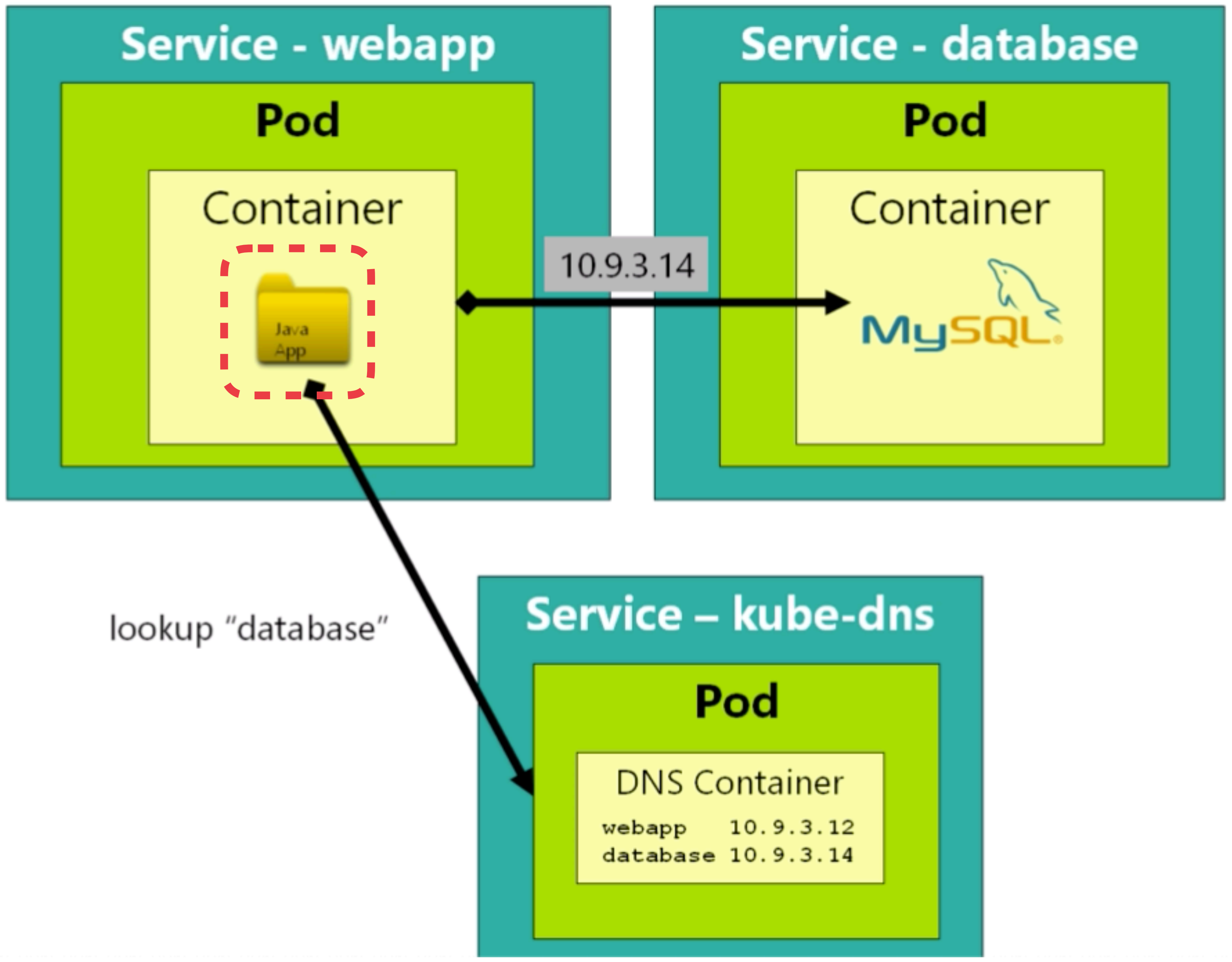
- We already understand now why by default we can't see the kube-dns service

Back to the work we were doing

- We want from our webapp application to reference a database container in a different pod



- Now we know that its as easy as doing a lookup for "database" in the kube-dns service

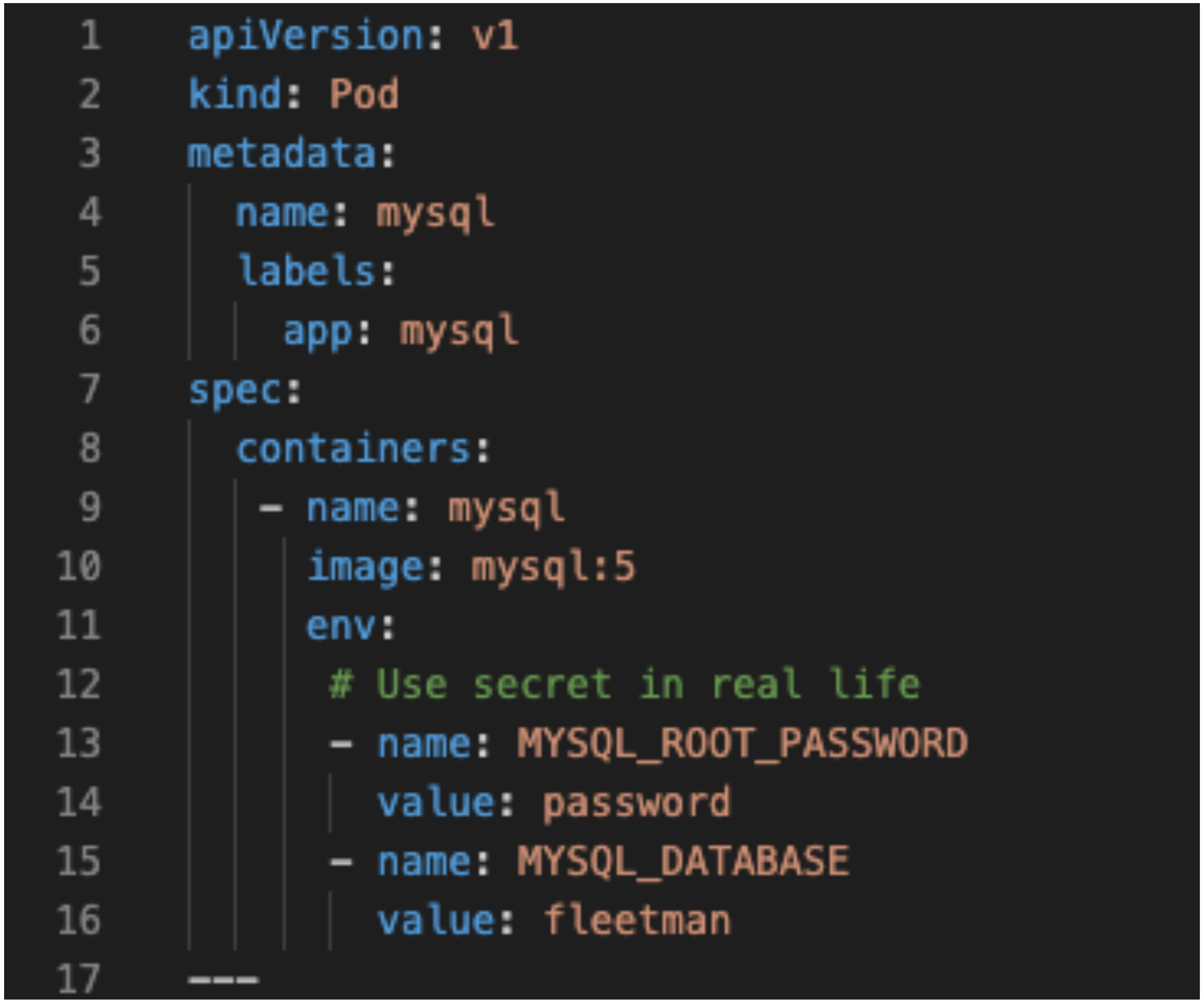


```
webapp   10.9.3.12
database 10.9.3.14
```

- That will give us the IP address of the target service

- We already have the webapp container, but we don't have the database container in our
- Let's create a new MySQL container in tis own Pod and a Service

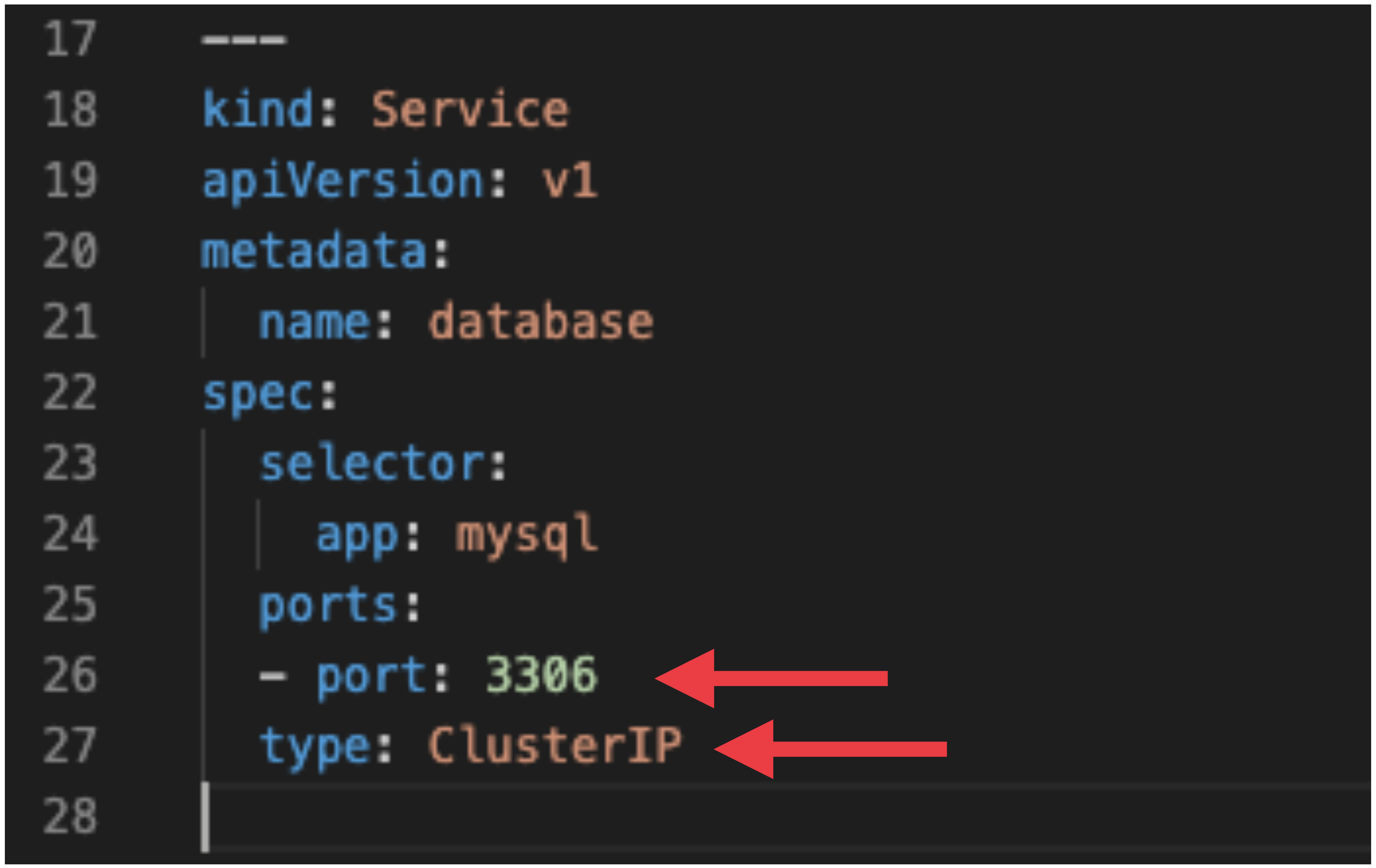
networking-tests.yaml



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  containers:
  - name: mysql
    image: mysql:5
    env:
     # Use secret in real life
     - name: MYSQL_ROOT_PASSWORD
       value: password
     - name: MYSQL_DATABASE
       value: fleetman
---
```

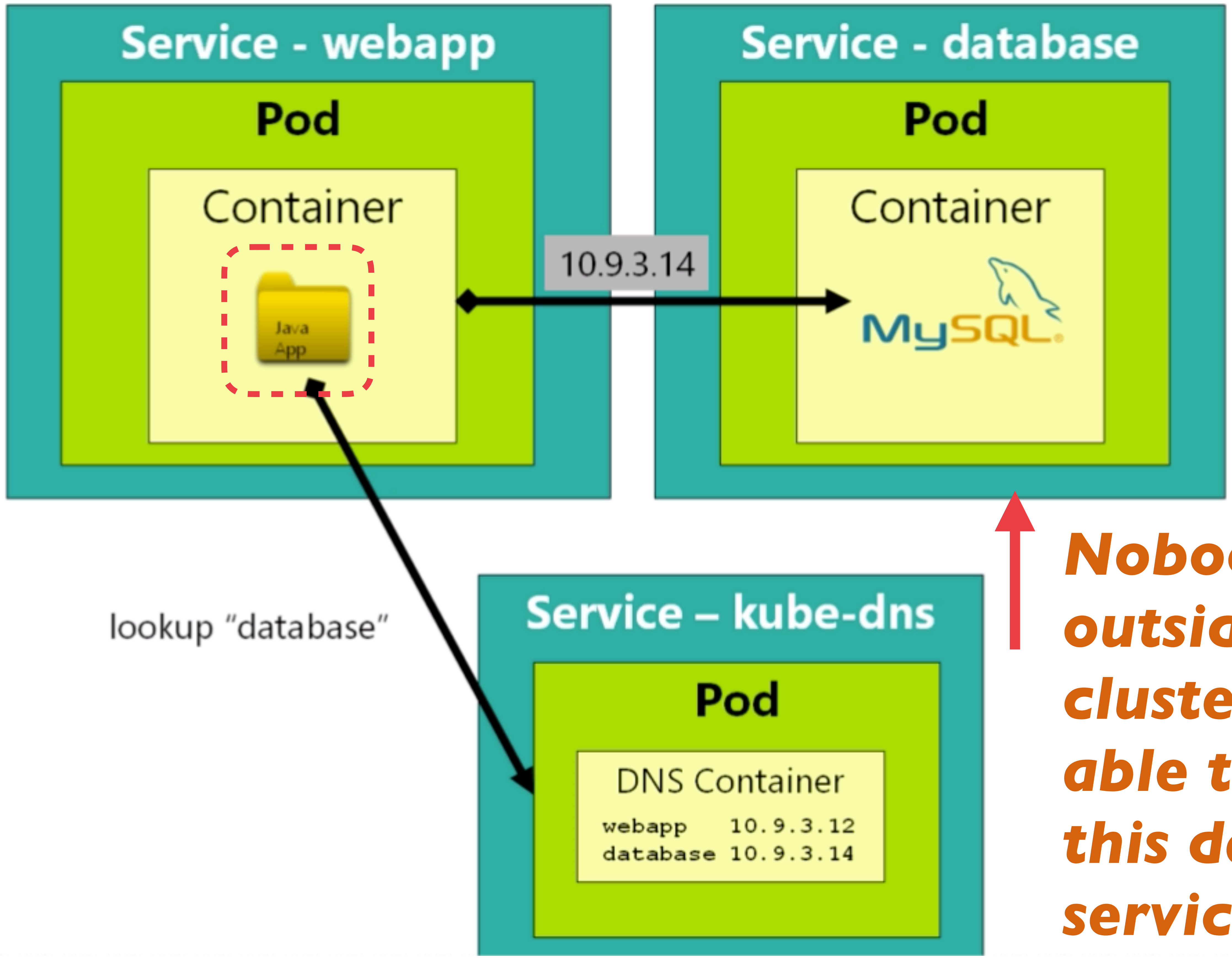
Pod

P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO



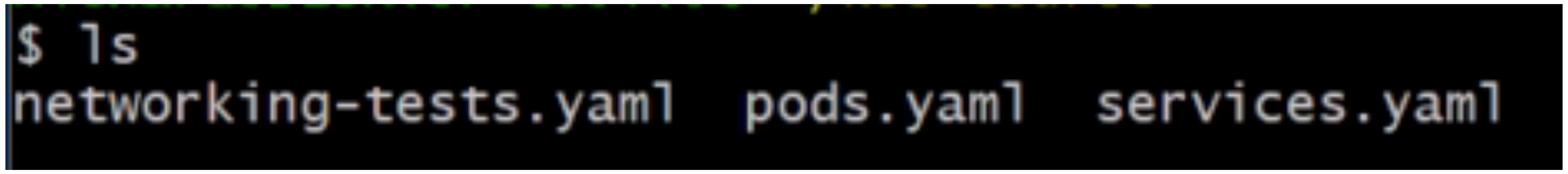
```yaml
---
kind: Service
apiVersion: v1
metadata:
  name: database
spec:
  selector:
    app: mysql
  ports:
  - port: 3306
  type: ClusterIP

```

Service



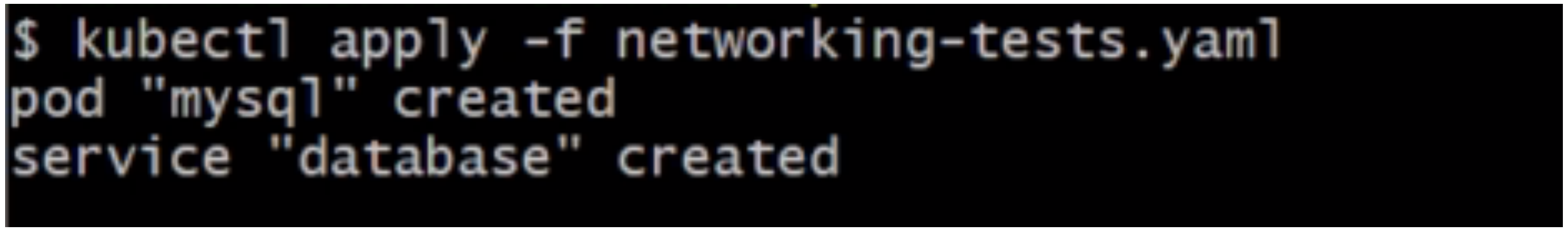
Nobody outside this cluster will be able to access this database service

- ls



```
$ ls
networking-tests.yaml  pods.yaml  services.yaml
```

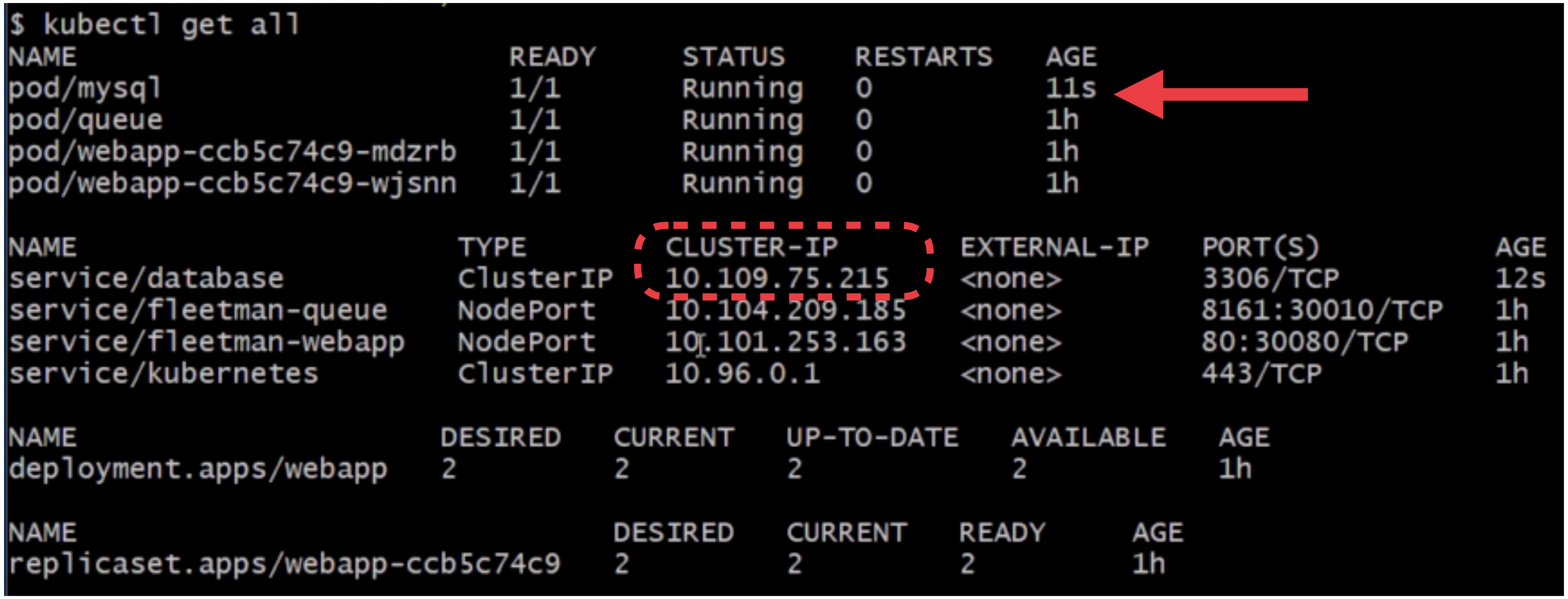
- kubectl apply -f networking-tests.yaml



```
$ kubectl apply -f networking-tests.yaml
pod "mysql" created
service "database" created
```

P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

kubectl get all



```
$ kubectl get all
NAME                            READY     STATUS    RESTARTS   AGE
pod/mysql                       1/1       Running   0          11s
pod/queue                       1/1       Running   0          1h
pod/webapp-ccb5c74c9-mdzrb      1/1       Running   0          1h
pod/webapp-ccb5c74c9-wjsnn      1/1       Running   0          1h

NAME                         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/database             ClusterIP   10.109.75.215    <none>        3306/TCP         12s
service/fleetman-queue       NodePort    10.104.209.185   <none>        8161:30010/TCP   1h
service/fleetman-webapp      NodePort    10.101.253.163   <none>        80:30080/TCP     1h
service/kubernetes           ClusterIP   10.96.0.1        <none>        443/TCP          1h

NAME                      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/webapp    2         2         2            2           1h

NAME                                DESIRED   CURRENT   READY     AGE
replicaset.apps/webapp-ccb5c74c9    2         2         2         1h
```

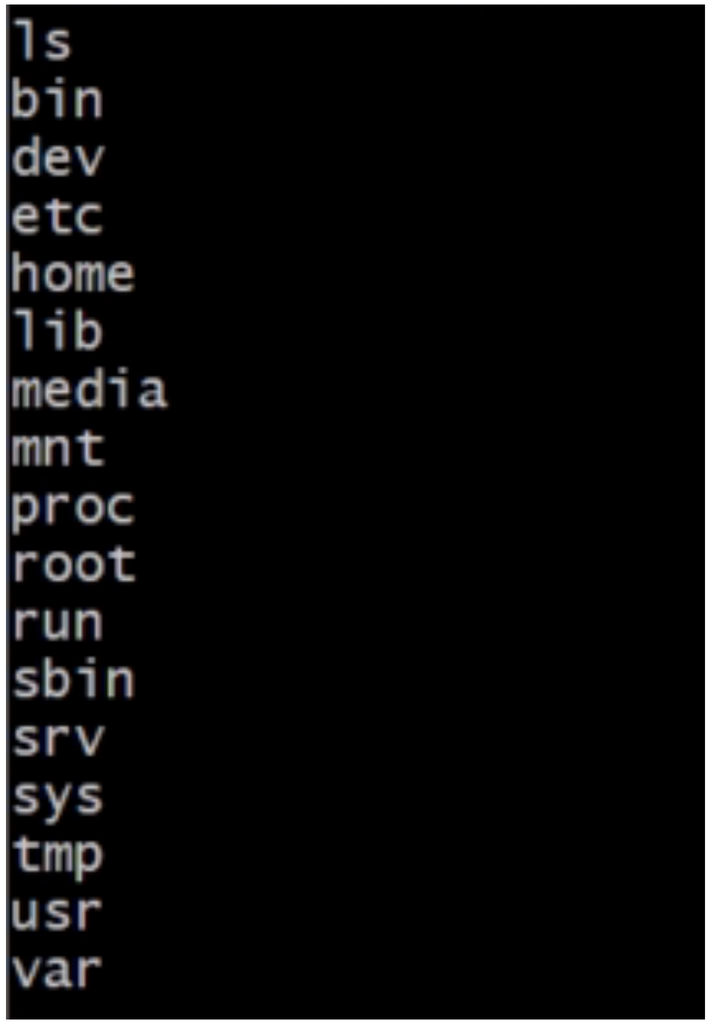
P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

- We are not going to write any code inside this webapp container that calls the database

- we are just going to do a very basic demonstration - we are going to simulate that happening

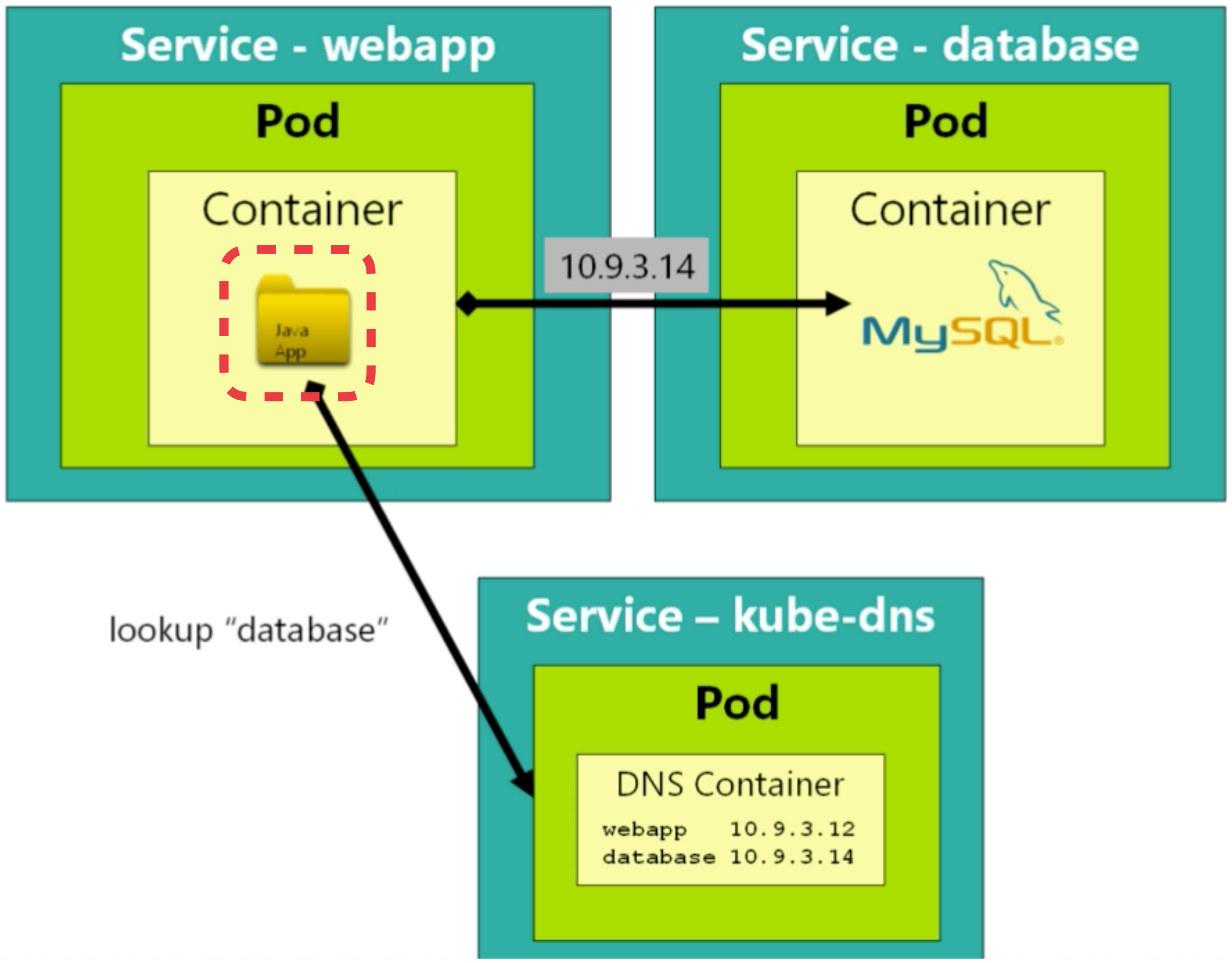
P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

- `kubectl exec -it webapp-ccb5c74c9-mdzrb sh`
- `ls`



```
ls
bin
dev
etc
home
lib
media
mnt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

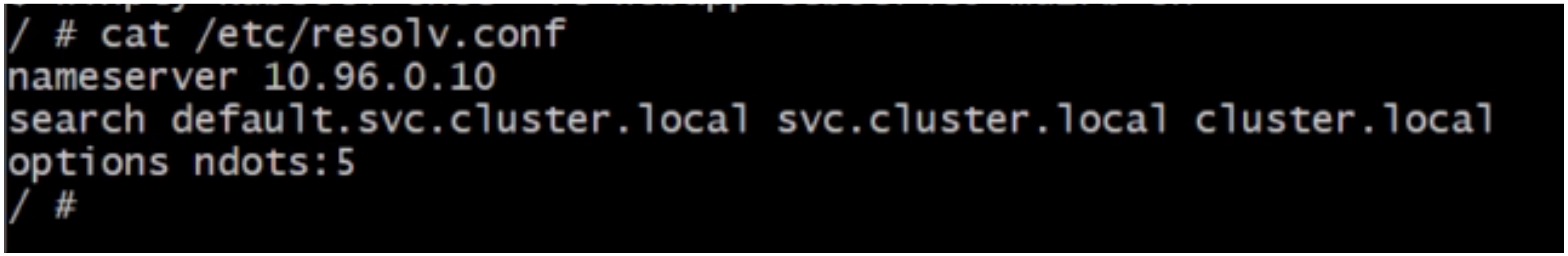


- How this networking mechanism works?
- I want to be able to access the remote database container
- How does our webapp container know how to find the dns service?

- The answer is in a piece of automatic configuration that Kubernetes is doing behind the scenes for us in all of our containers
- Kubernetes will automatically do somme management of the container and automatically configures the dns system

We can verify that

- Remember you are in the webapp container
- `cat /etc/resolv.conf`

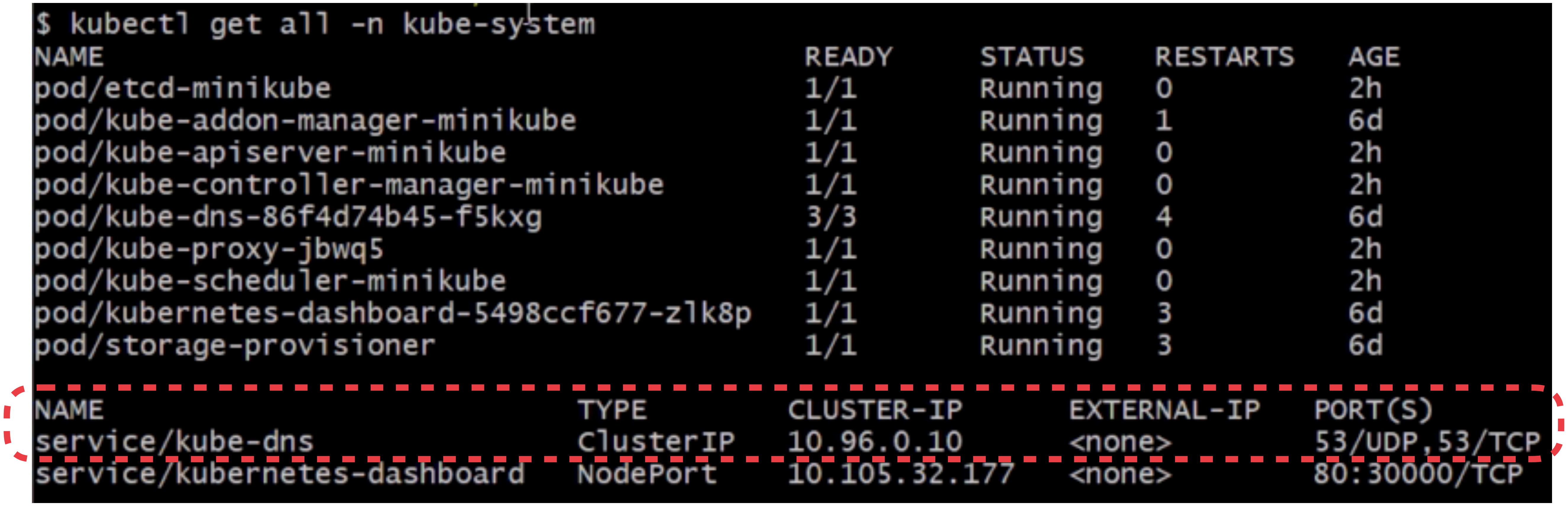


```
/ # cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
/ #
```

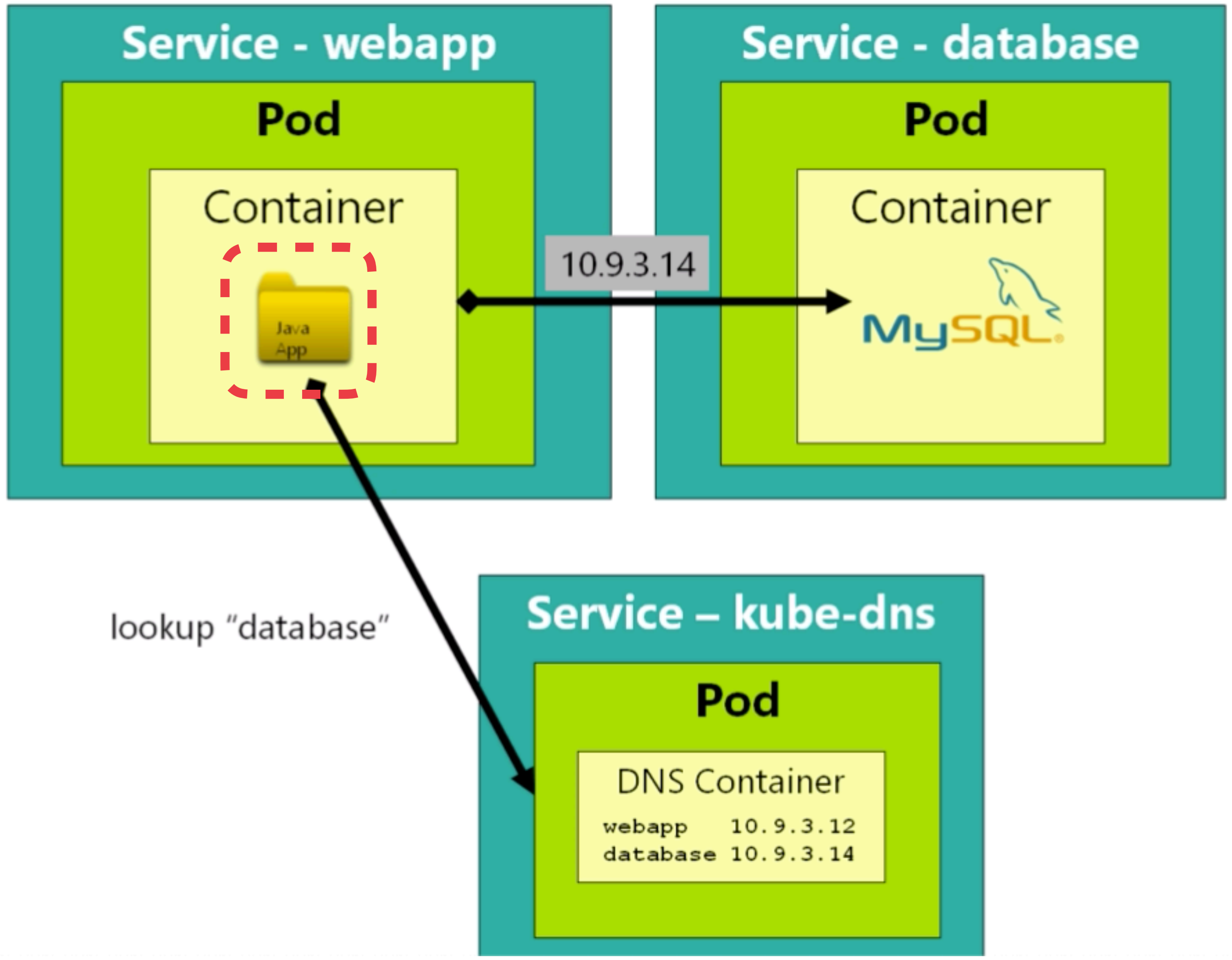
- This is the file that configures how the dns name resolution is going to work

P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

- The first line tells the linux system, the container, when we make a reference to the domain name we need to find the dns server located in "`10.96.0.10`" (in my case)
- `kubectl get all -n kube-system`



```
$ kubectl get all -n kube-system
NAME                                      READY    STATUS    RESTARTS    AGE
pod/etcd-minikube                         1/1      Running   0           2h
pod/kube-addon-manager-minikube           1/1      Running   1           6d
pod/kube-apiserver-minikube               1/1      Running   0           2h
pod/kube-controller-manager-minikube      1/1      Running   0           2h
pod/kube-dns-86f4d74b45-f5kxg             3/3      Running   4           6d
pod/kube-proxy-jbwq5                      1/1      Running   0           2h
pod/kube-scheduler-minikube               1/1      Running   0           2h
pod/kubernetes-dashboard-5498ccf677-zlk8p 1/1      Running   3           6d
pod/storage-provisioner                   1/1      Running   3           6d

NAME                          TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)
service/kube-dns              ClusterIP   10.96.0.10      <none>         53/UDP,53/TCP
service/kubernetes-dashboard  NodePort    10.105.32.177   <none>         80:30000/TCP
```



- Now we understand how it works (next slide)

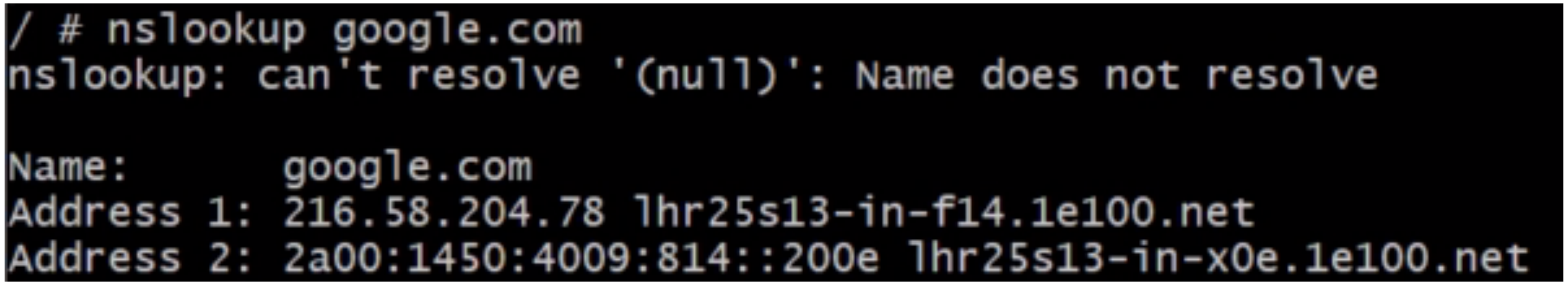
P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

1. We try to make a request to a URL with the domain database inside it - this container doesn't know what that means
2. Uses that resolv.conf configuration to tell where the DNS service is
3. Where it will lookup that name and it will respond with the correct IP address

P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

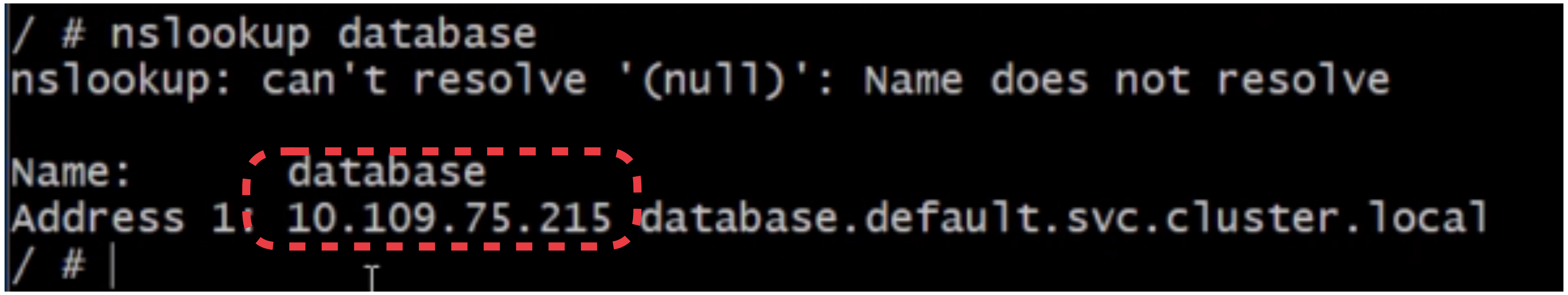
- We can see this working by using the nslookup command

- `nslookup google.com`



```
/ # nslookup google.com
nslookup: can't resolve '(null)': Name does not resolve

Name:      google.com
Address 1: 216.58.204.78 lhr25s13-in-f14.1e100.net
Address 2: 2a00:1450:4009:814::200e lhr25s13-in-x0e.1e100.net
```

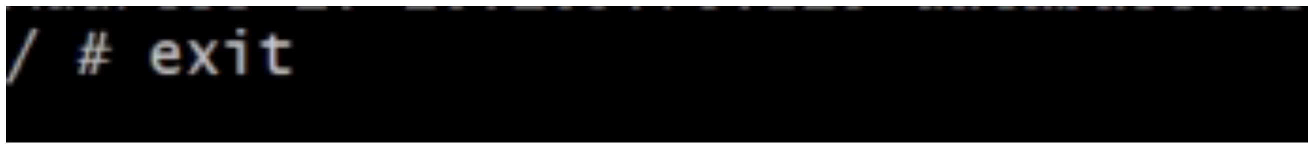
- `nslookup database`



```
/ # nslookup database
nslookup: can't resolve '(null)': Name does not resolve

Name:      database
Address 1: 10.109.75.215 database.default.svc.cluster.local
/ # |
```

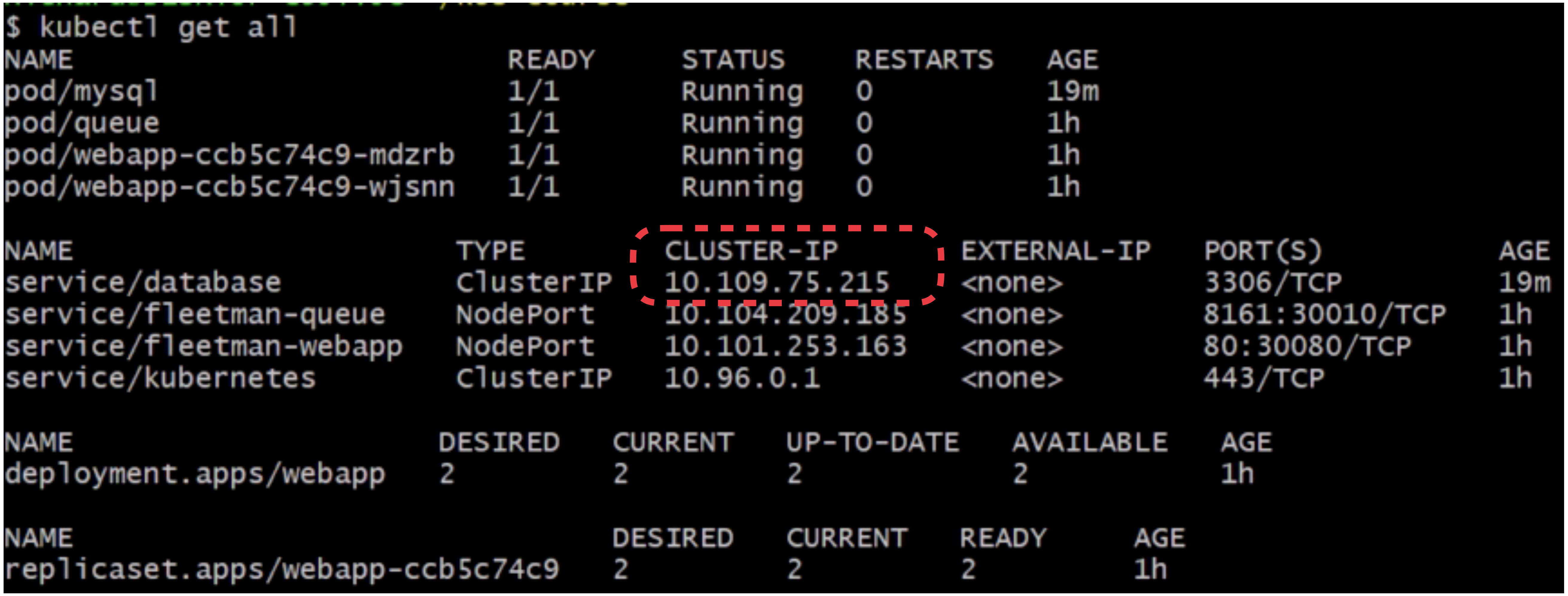
P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

- exit of our shell - **`exit`**



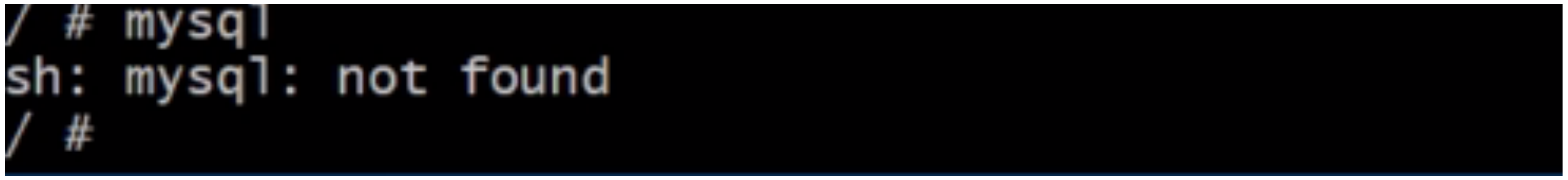
```
/ # exit
```

- **`kubectl get all`**



```
$ kubectl get all
NAME                          READY     STATUS    RESTARTS   AGE
pod/mysql                     1/1       Running   0          19m
pod/queue                     1/1       Running   0          1h
pod/webapp-ccb5c74c9-mdzrb    1/1       Running   0          1h
pod/webapp-ccb5c74c9-wjsnn    1/1       Running   0          1h

NAME                         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
service/database             ClusterIP   10.109.75.215    <none>        3306/TCP         19m
service/fleetman-queue       NodePort    10.104.209.185   <none>        8161:30010/TCP   1h
service/fleetman-webapp      NodePort    10.101.253.163   <none>        80:30080/TCP     1h
service/kubernetes           ClusterIP   10.96.0.1        <none>        443/TCP          1h

NAME                        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/webapp      2         2         2            2           1h

NAME                                DESIRED   CURRENT   READY     AGE
replicaset.apps/webapp-ccb5c74c9    2         2         2         1h
```

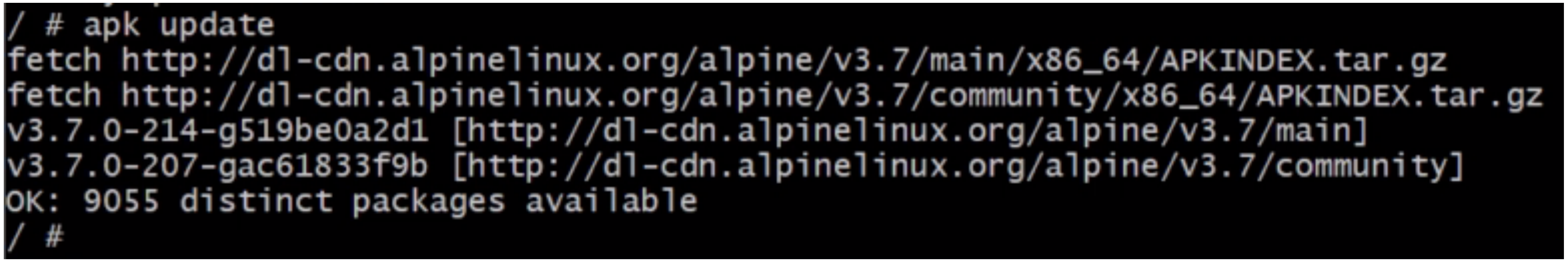
- go back to the shell
- `kubectl exec -it webapp-ccb5c74c9-mdzrb sh`
- We can't use mysql because is not installed
- `mysql`



```
/ # mysql
sh: mysql: not found
/ #
```

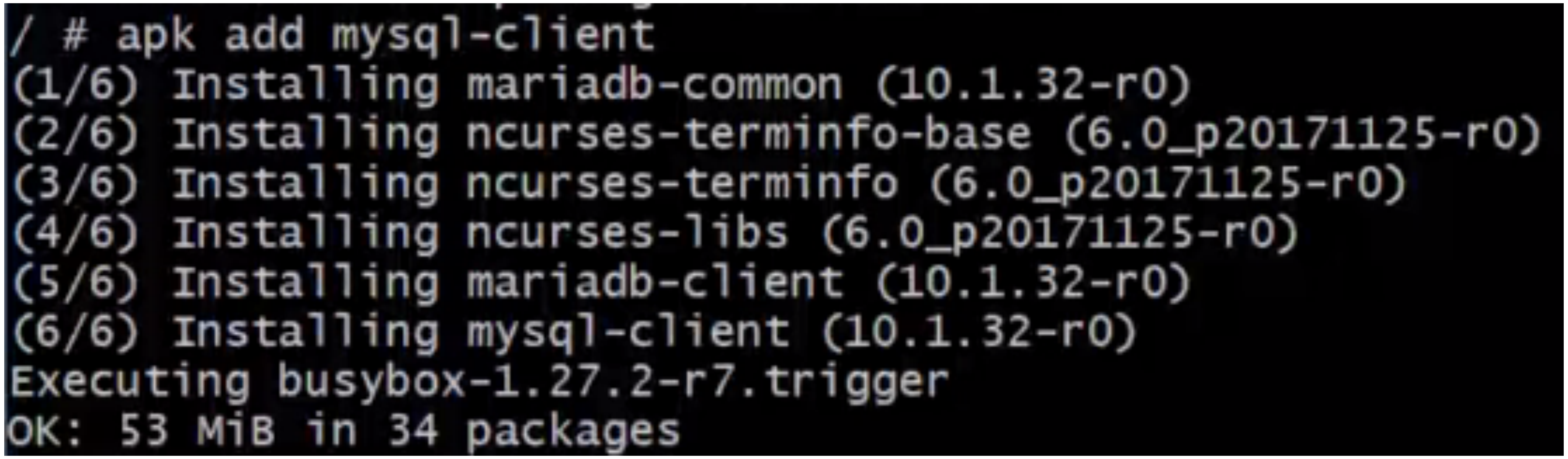
- Let's install the mysql client

- `apk update` (this a is linux alpine)



```
/ # apk update
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.7/community/x86_64/APKINDEX.tar.gz
v3.7.0-214-g519be0a2d1 [http://dl-cdn.alpinelinux.org/alpine/v3.7/main]
v3.7.0-207-gac61833f9b [http://dl-cdn.alpinelinux.org/alpine/v3.7/community]
OK: 9055 distinct packages available
/ #
```

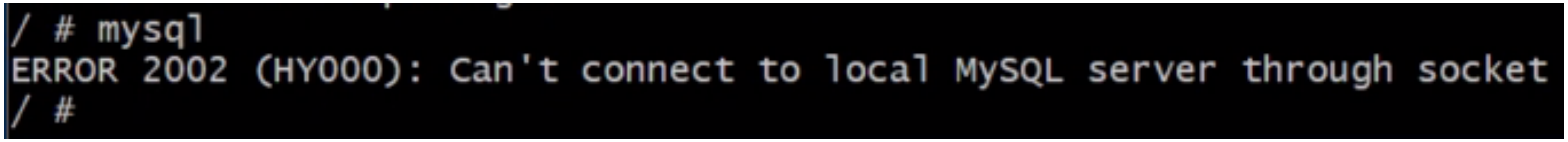
- `apk add mysql-client`



```
/ # apk add mysql-client
(1/6) Installing mariadb-common (10.1.32-r0)
(2/6) Installing ncurses-terminfo-base (6.0_p20171125-r0)
(3/6) Installing ncurses-terminfo (6.0_p20171125-r0)
(4/6) Installing ncurses-libs (6.0_p20171125-r0)
(5/6) Installing mariadb-client (10.1.32-r0)
(6/6) Installing mysql-client (10.1.32-r0)
Executing busybox-1.27.2-r7.trigger
OK: 53 MiB in 34 packages
```

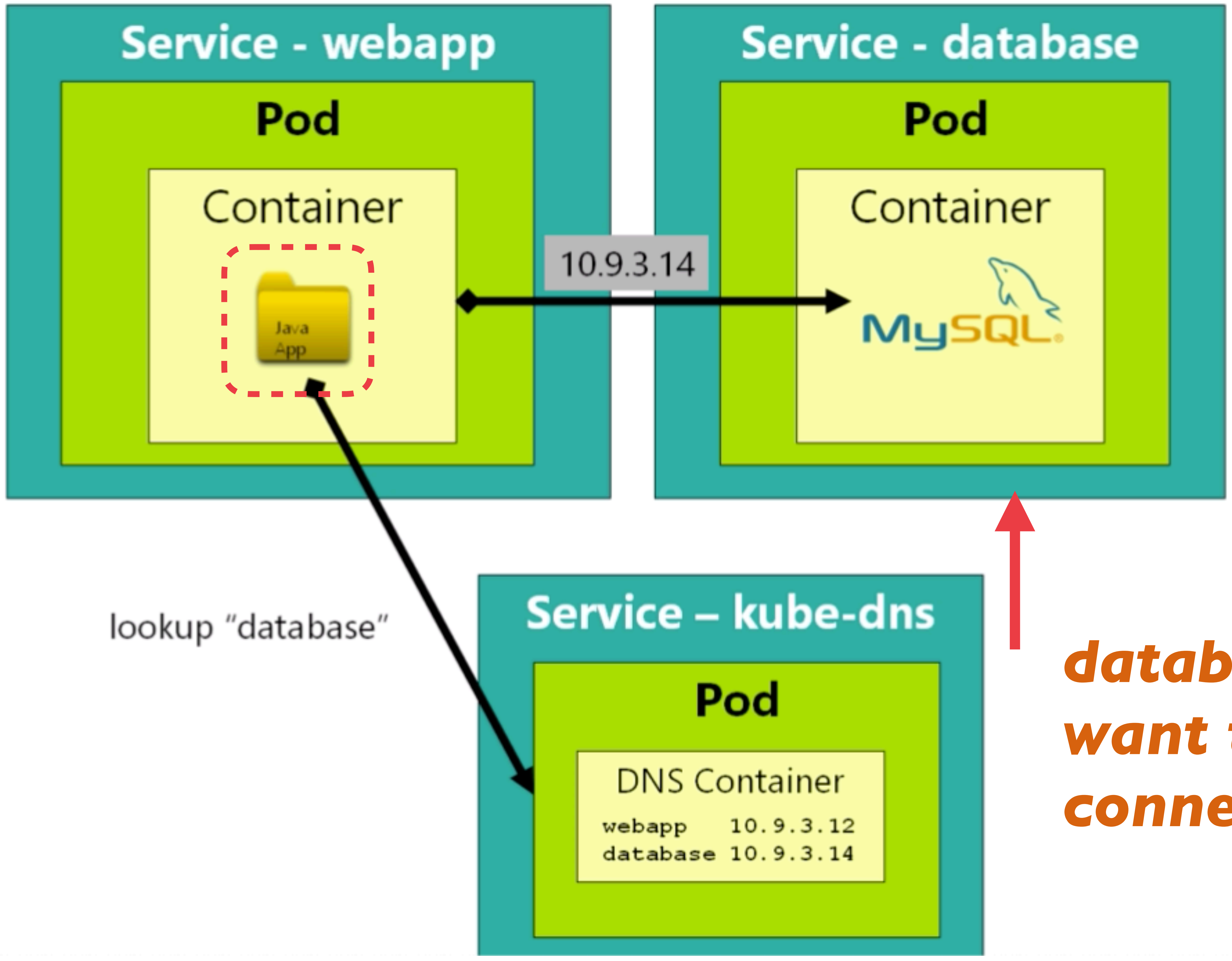
- We are not installing a database, this is just a command line for my mysql

- If we try to use mysql - **`mysql`**



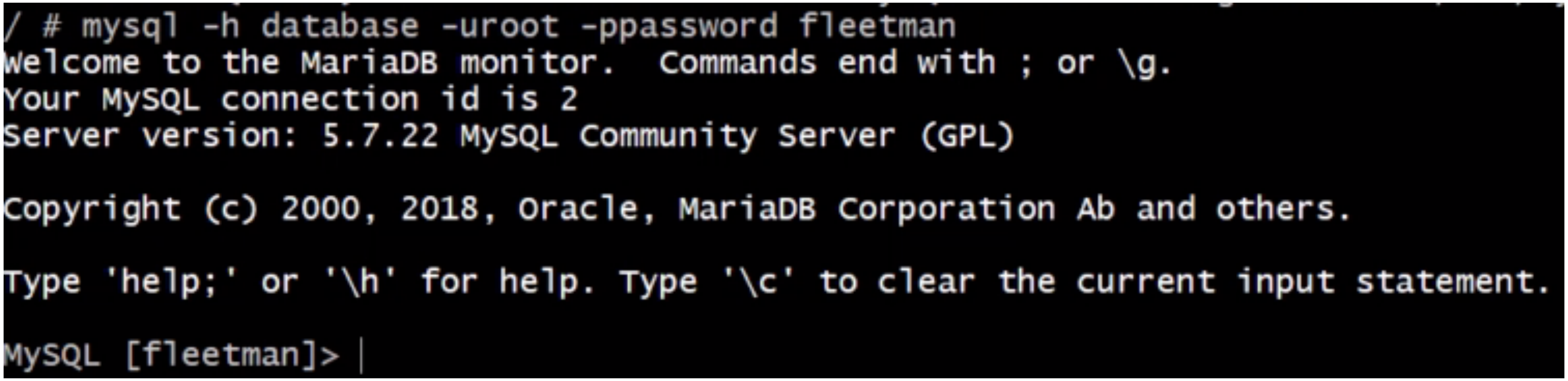
```
/ # mysql
ERROR 2002 (HY000): Can't connect to local MySQL server through socket
/ #
```

- We got this error because we don’t have any database server running on this container
- We want to connect to a database running on a different container on a different pod - represented by the service database



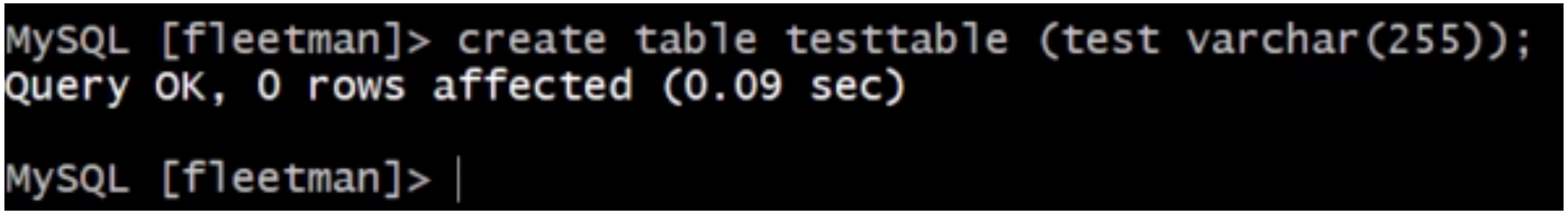
database we want to connect

- `mysql -h database -uroot -ppassword fleetman`



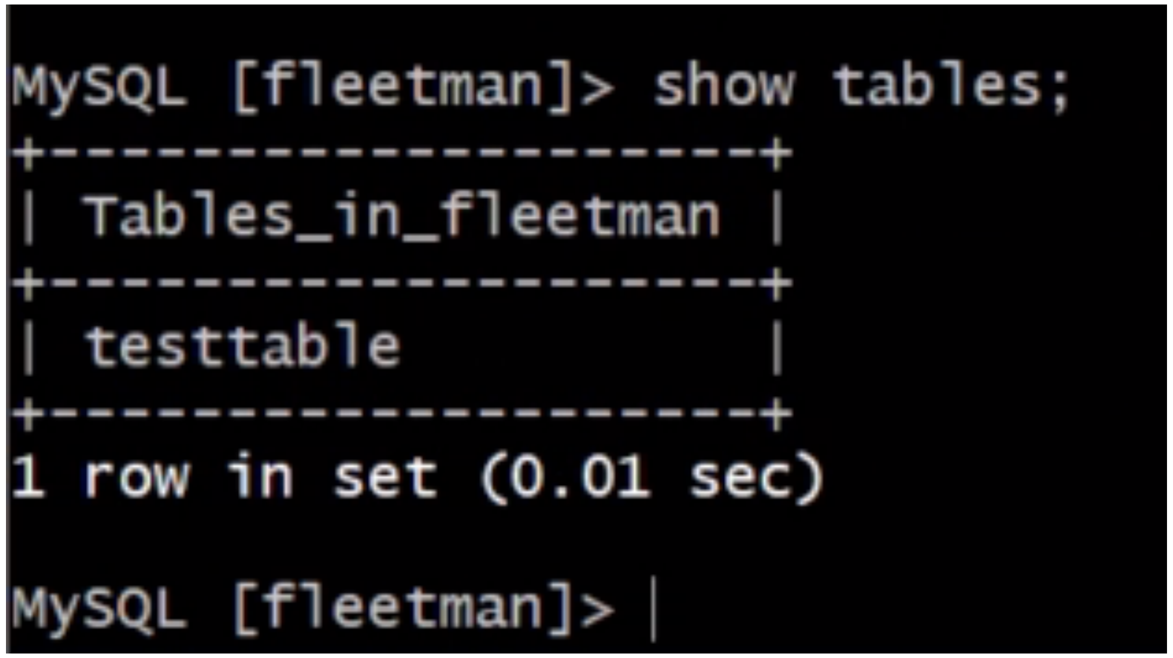
```
/ # mysql -h database -uroot -ppassword fleetman
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.22 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [fleetman]> |
```

- `create table testtable (test varchar(255));`



```
MySQL [fleetman]> create table testtable (test varchar(255));
Query OK, 0 rows affected (0.09 sec)

MySQL [fleetman]> |
```

show tables



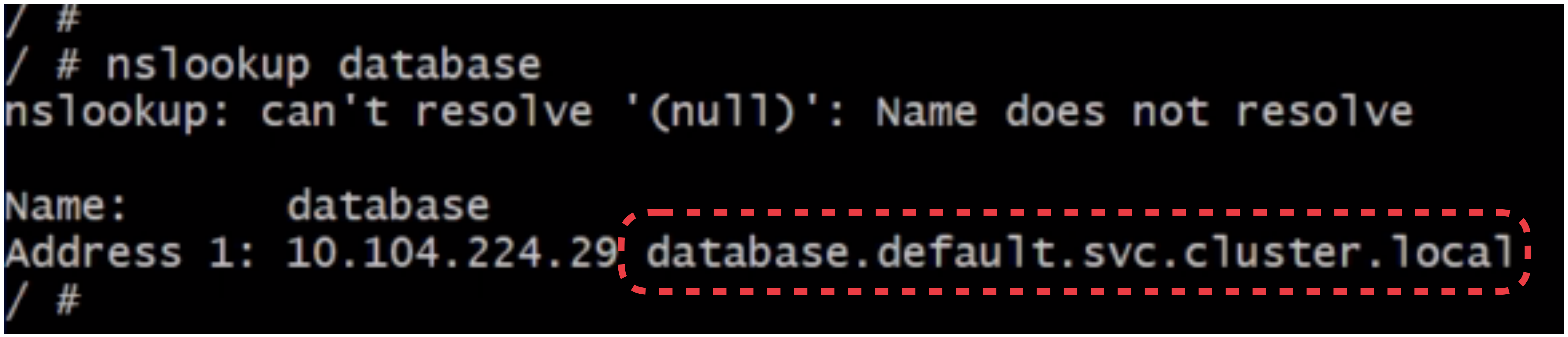
```
MySQL [fleetman]> show tables;
+---------------------+
| Tables_in_fleetman |
+---------------------+
| testtable           |
+---------------------+
1 row in set (0.01 sec)

MySQL [fleetman]> |
```

Networking in kubernetes is very easy when we understand there is a DNS service

this technique - service discovery - is very important to connect all of our micro services together

P.PORTO ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

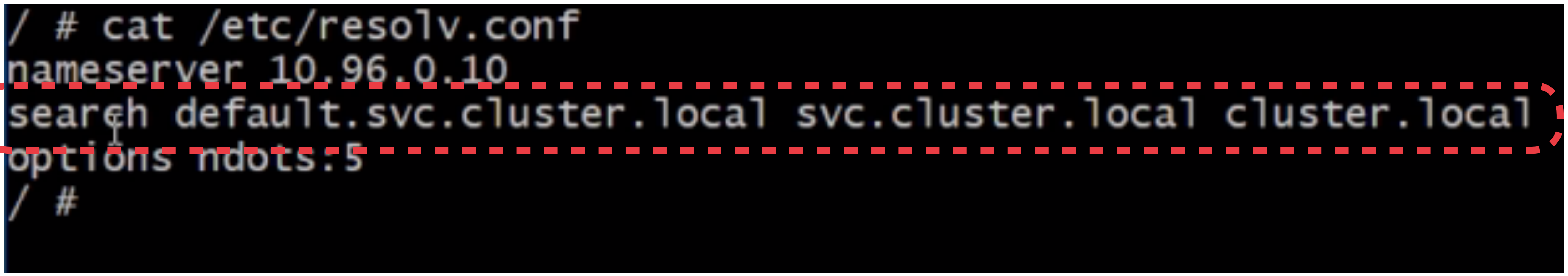
- We discovered how to find any service by its name
- Using nslookup we could locate the database service through its name (nslookup database)
- But the service is not registered with the name database



```
/ #
/ # nslookup database
nslookup: can't resolve '(null)': Name does not resolve

Name:      database
Address 1: 10.104.224.29 database.default.svc.cluster.local
/ #
```

- Its registered under its Fully Qualified Domain Name - FQDN: **`database.default.svc.cluster.local`**
- What is happenning when we do the lookup, is that it can't find "database" because it didn't matches the FQDN string

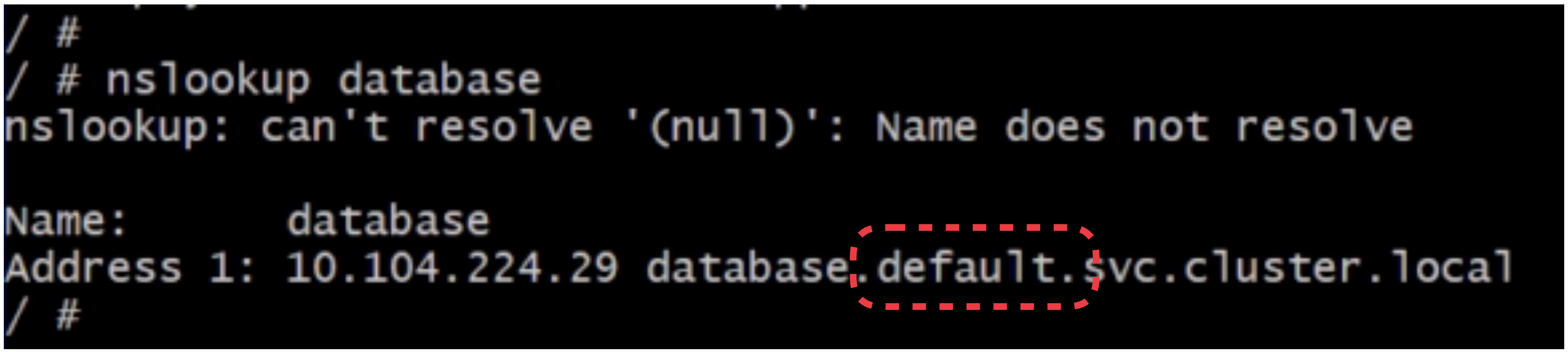
- `cat /etc/resolv.conf`



```
/ # cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local
options ndots:5
/ #
```

- It says that in the event of the name not being found then it should try to append the string **`default.svc.cluster.local`** then if it's still not found append **`svc.cluster.local`** and then **`cluster.local`**

- The most important point here is the name after **`database`**., in this case "**`default`**", because it relates to the namespace that that service is in



```
/ #
/ # nslookup database
nslookup: can't resolve '(null)': Name does not resolve

Name:      database
Address 1: 10.104.224.29 database.default.svc.cluster.local
/ #
```

P.PORTO

Polytechnic of
Porto

ESTG - School of
Management and Technology