**Kindly Stop And Give Me Your Valuable Feedback..**

Before going to next page kindly read this below paragraph.

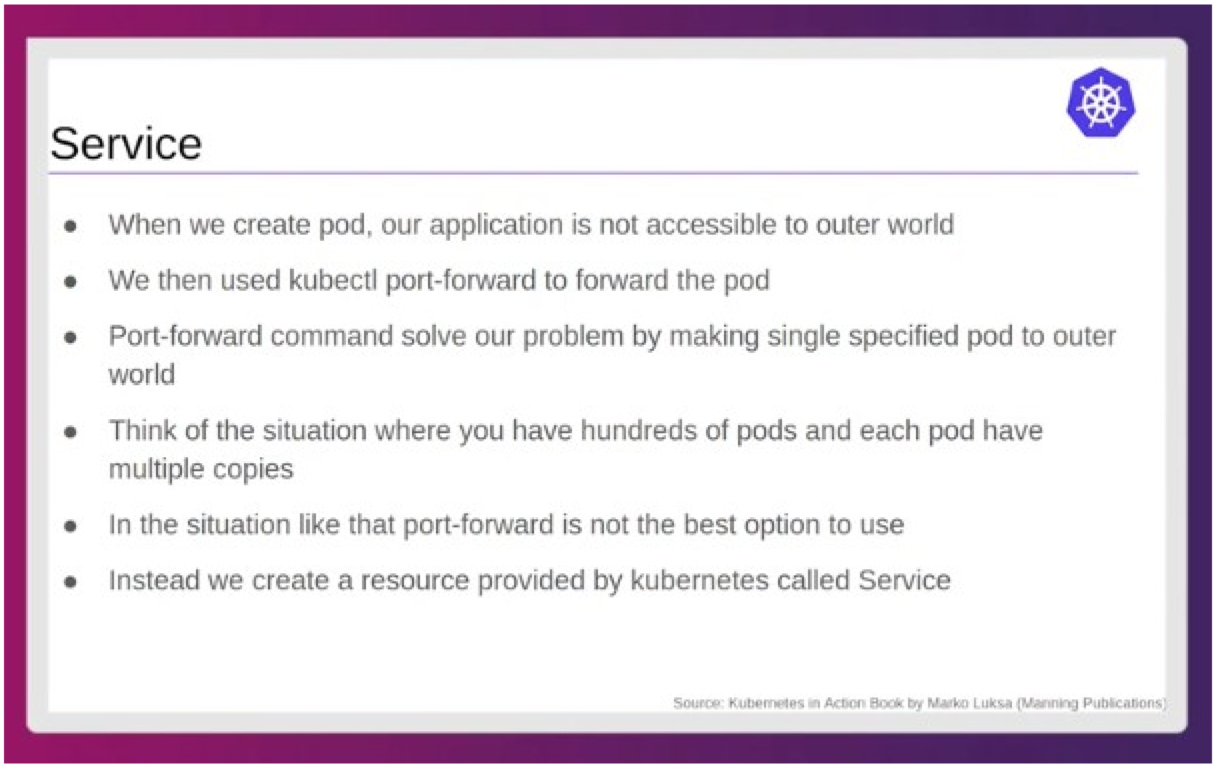
Please give me feedback about this type of helping material. Is this helpful for you ?
Near in future I am energetic for establish the YouTube Channel for education So I want to take your valuable feedback for my about content
So kindly give me your opinion..

In Shaa Allah I Will Establish A YouTube Channel To Make Learning Easy And Simple..For Everyone.

**Thank You**

Kubernetes Lesson 4

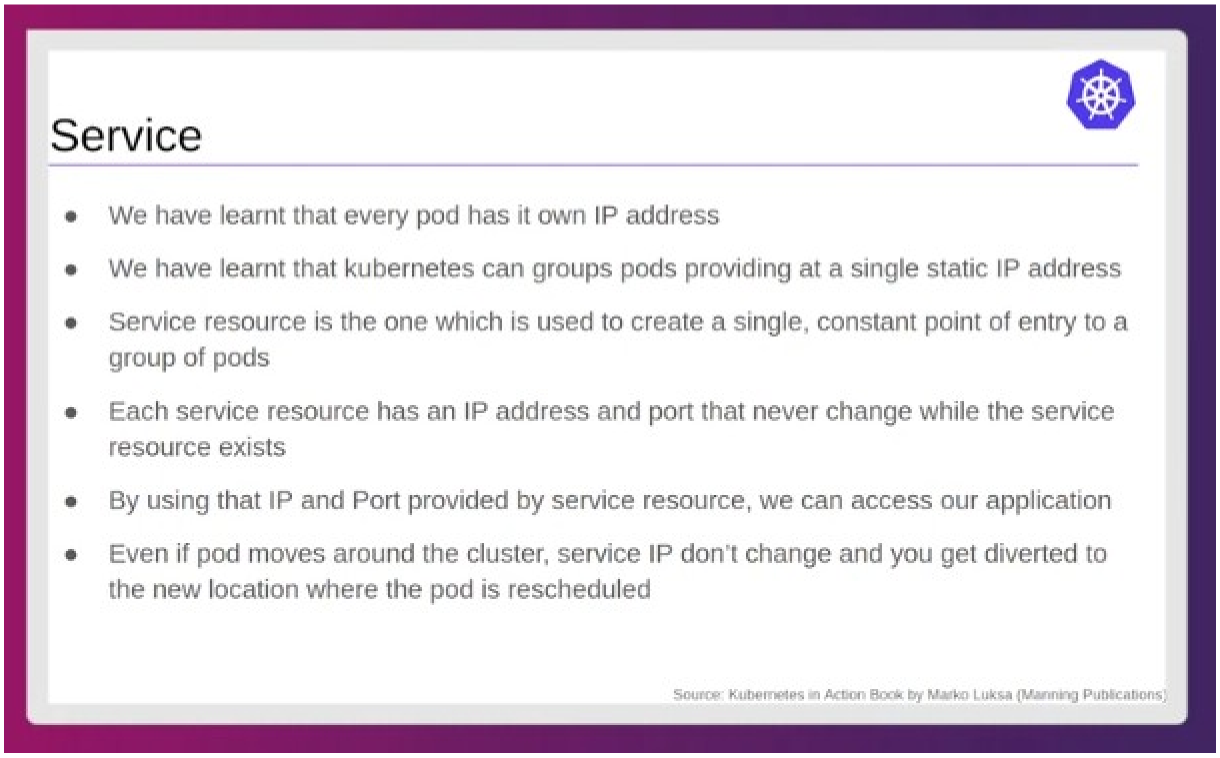
Video 1. “Service”



Service

- When we create pod, our application is not accessible to outer world
- We then used kubectl port-forward to forward the pod
- Port-forward command solve our problem by making single specified pod to outer world
- Think of the situation where you have hundreds of pods and each pod have multiple copies
- In the situation like that port-forward is not the best option to use
- Instead we create a resource provided by kubernetes called Service

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

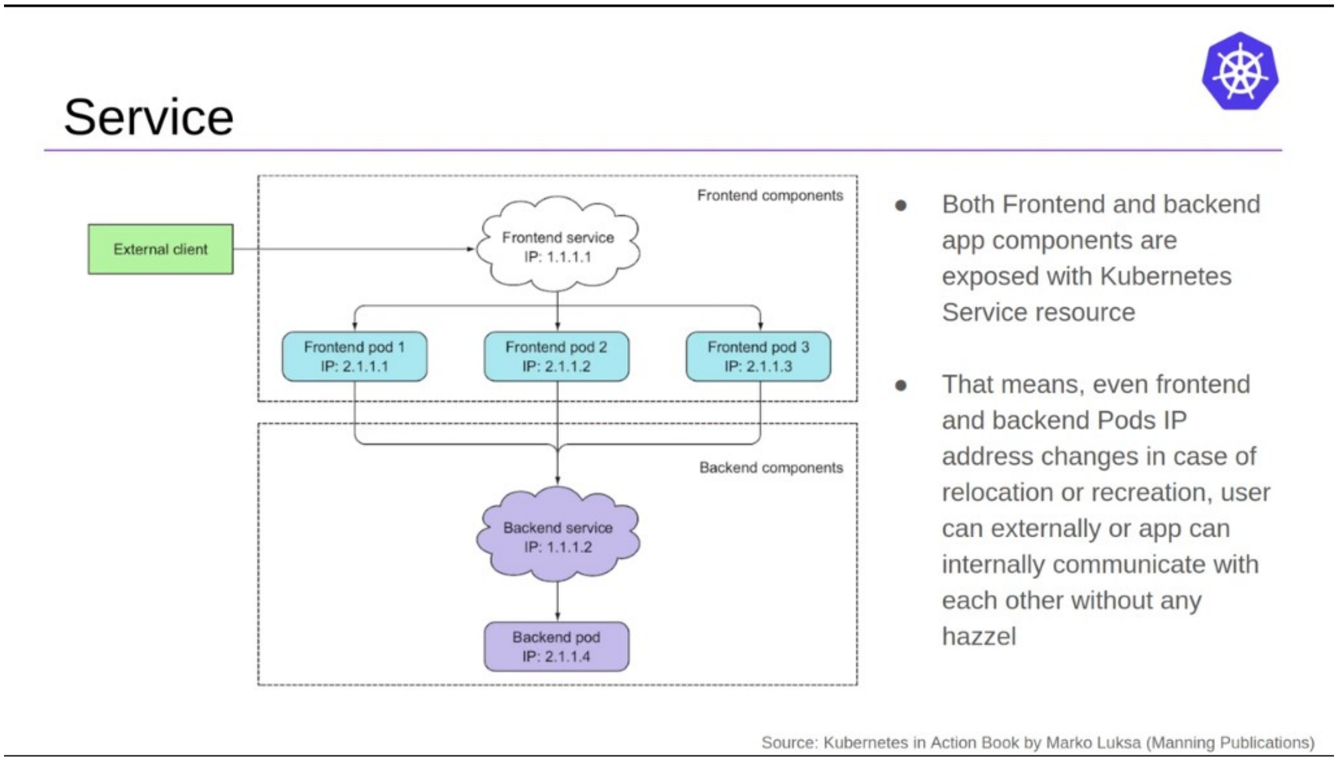


Service

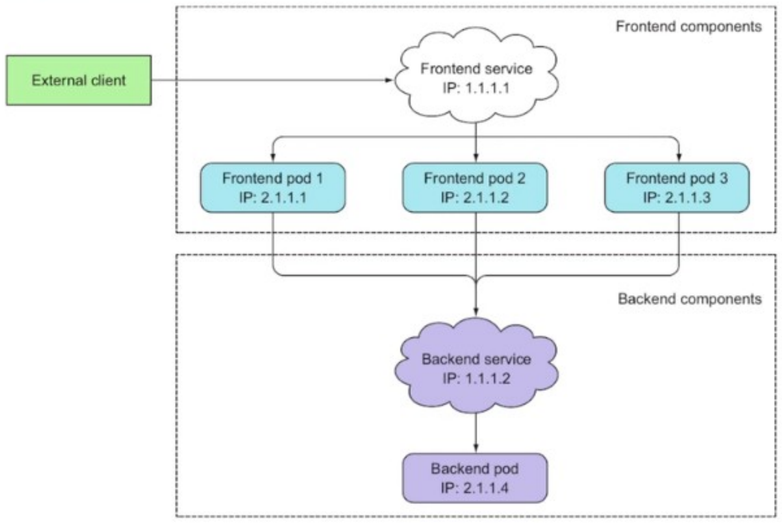
- We have learnt that every pod has it own IP address
- We have learnt that kubernetes can groups pods providing at a single static IP address
- Service resource is the one which is used to create a single, constant point of entry to a group of pods
- Each service resource has an IP address and port that never change while the service resource exists
- By using that IP and Port provided by service resource, we can access our application
- Even if pod moves around the cluster, service IP don’t change and you get diverted to the new location where the pod is rescheduled

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Video 2. “Service Resource Flow”



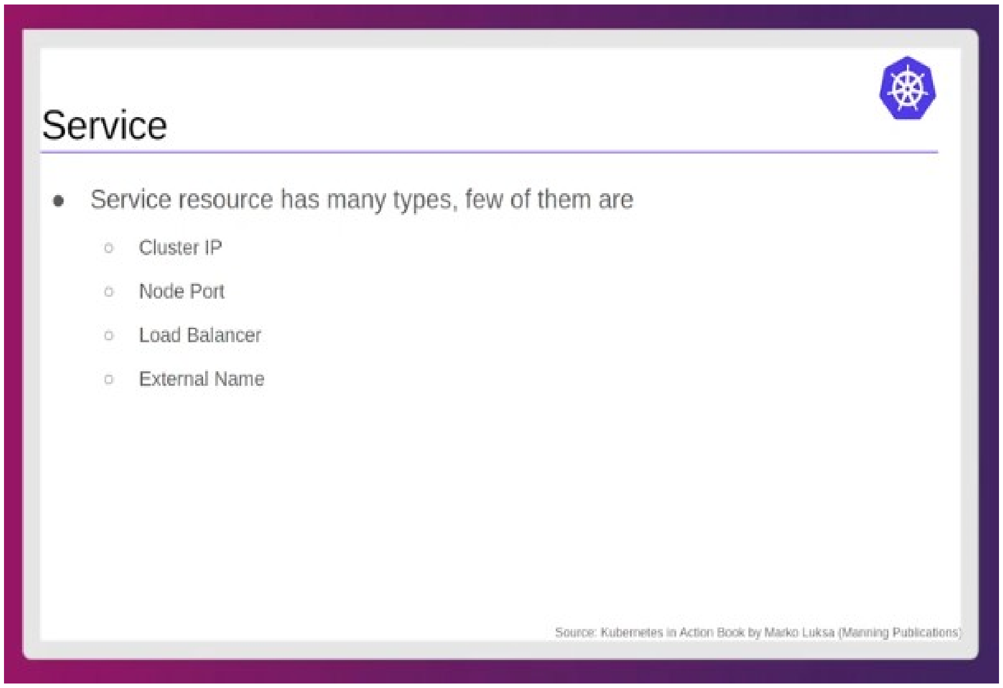
Service



- Both Frontend and backend app components are exposed with Kubernetes Service resource
- That means, even frontend and backend Pods IP address changes in case of relocation or recreation, user can externally or app can internally communicate with each other without any hazzel

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

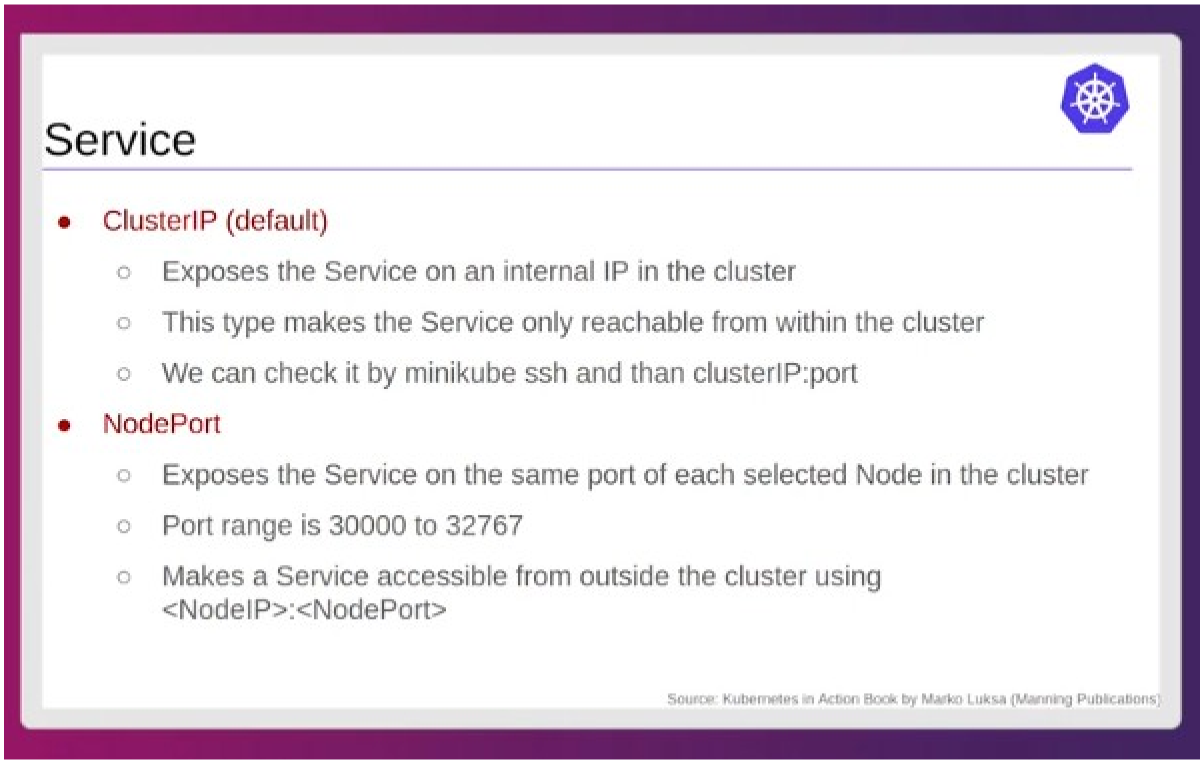
Video 3. “Service Resourse Types”



Service

- Service resource has many types, few of them are
  - Cluster IP
  - Node Port
  - Load Balancer
  - External Name

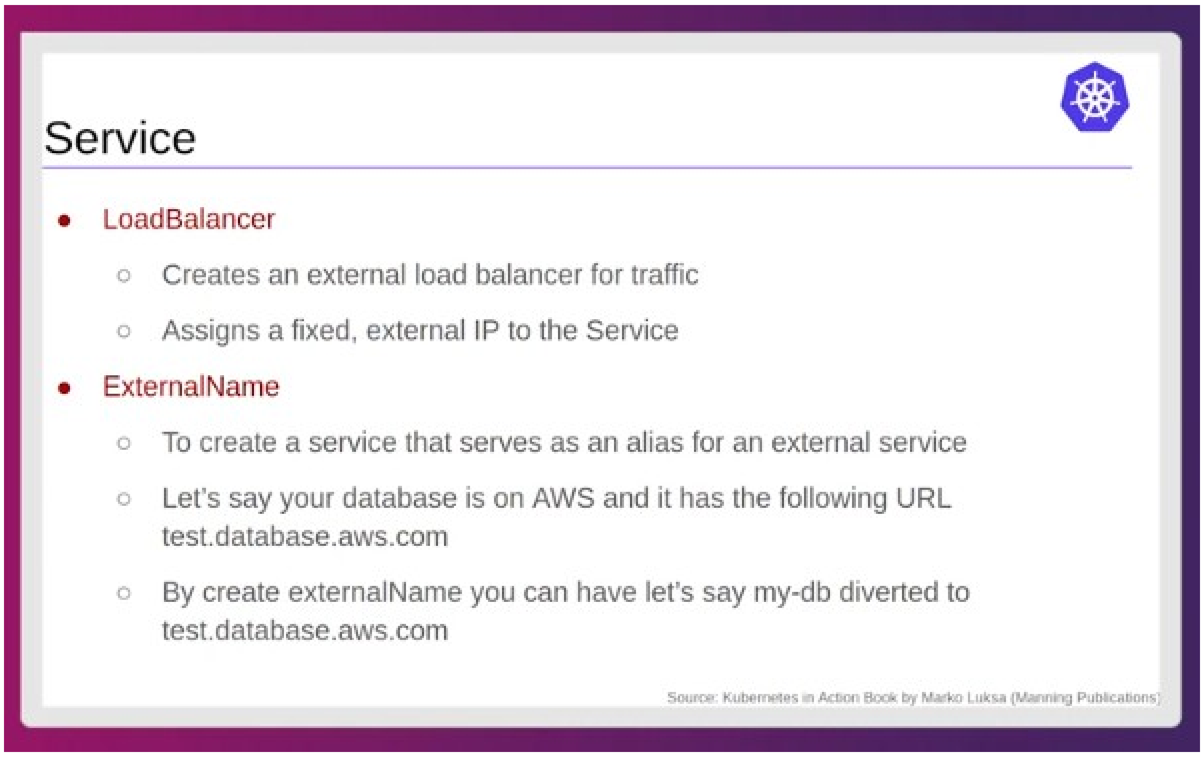
Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)



Service

- ClusterIP (default)
  - Exposes the Service on an internal IP in the cluster
  - This type makes the Service only reachable from within the cluster
  - We can check it by minikube ssh and than clusterIP:port
- NodePort
  - Exposes the Service on the same port of each selected Node in the cluster
  - Port range is 30000 to 32767
  - Makes a Service accessible from outside the cluster using <NodeIP>:<NodePort>

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)



Service

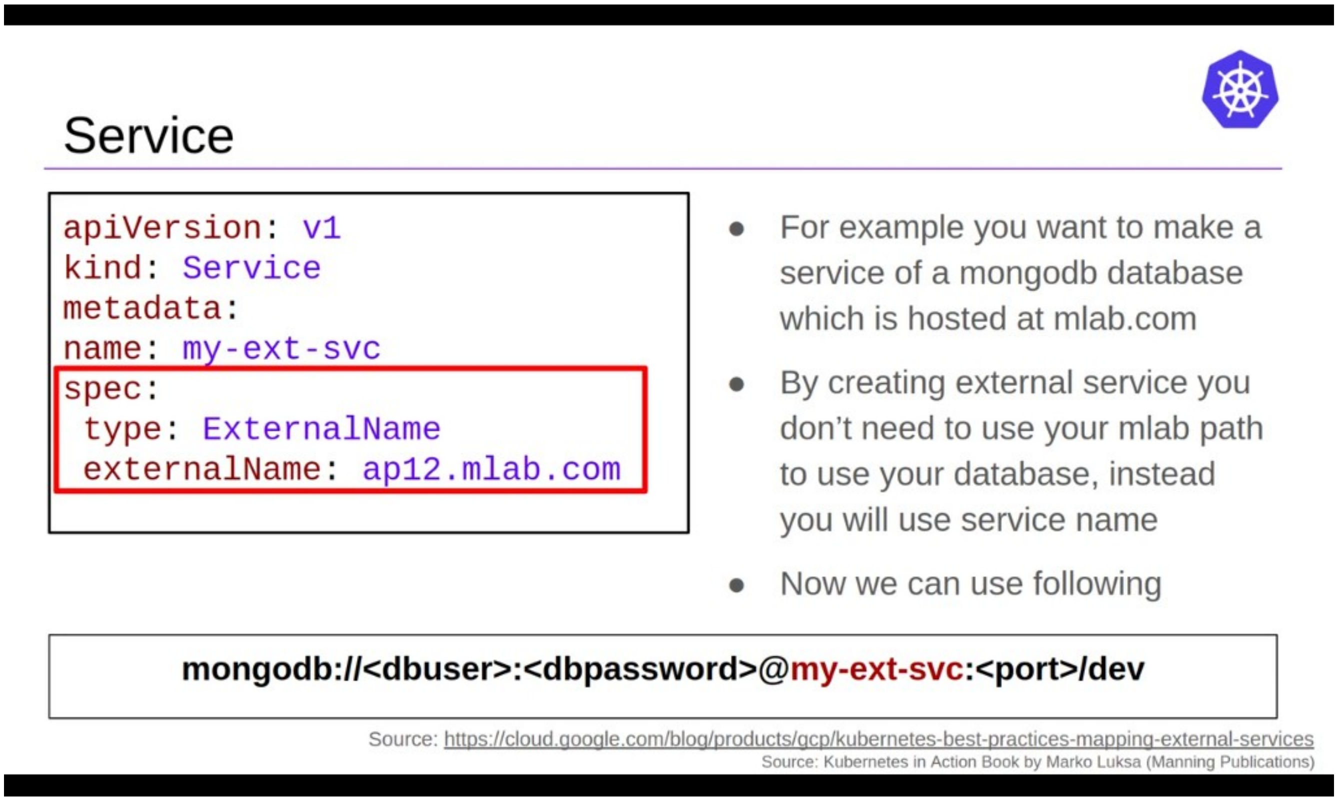
- LoadBalancer
  - Creates an external load balancer for traffic
  - Assigns a fixed, external IP to the Service
- ExternalName
  - To create a service that serves as an alias for an external service
  - Let's say your database is on AWS and it has the following URL test.database.aws.com
  - By create externalName you can have let's say my-db diverted to test.database.aws.com

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

What is mongoDB's database?
Answer is still pending.

Video 4. "External Name Example"

This Video is little theory and practical based.



Service

```
apiVersion: v1
kind: Service
metadata:
name: my-ext-svc
spec:
  type: ExternalName
  externalName: ap12.mlab.com
```

- For example you want to make a service of a mongodb database which is hosted at mlab.com
- By creating external service you don't need to use your mlab path to use your database, instead you will use service name
- Now we can use following

```
mongodb://<dbuser>:<dbpassword>@my-ext-svc:<port>/dev
```

Source: https://cloud.google.com/blog/products/gcp/kubernetes-best-practices-mapping-external-services
Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Video 5. "LoadBalancer Service Example YAML File"

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
  - port: 8080
    targetPort: 80
  type: LoadBalancer
  selector:
    app: rsexample
```

This is the yaml file for creating the service,type of ***LoadBalancer.*** *In this type type of service we have to define the main things like ports,then type of the service and selector..* ***the first one port is 8080,*** *the purpose of this type of the port is to define the port of the pod and then the* ***targetPort: 80*** *is the port of the container port for nginx port.*
Then the Type *is define the type of the service means which one the service is going to be made like* ***(cluster IP, Node Port,Load Balancer,External Name).***
Then the selector *is used to define that which label of the group that should be run under this service.*
Then 8080:31137/TCP *this results will be come so the* ***LoadBalancer*** *will expose an* ***port*** *that will be use to hit from out side of the world. In this case the external* ***port*** *is* ***31137*** *and the* ***8080*** *is the port of the* ***pod***

Video 6. "LoadBalancer Service Example Command Line"

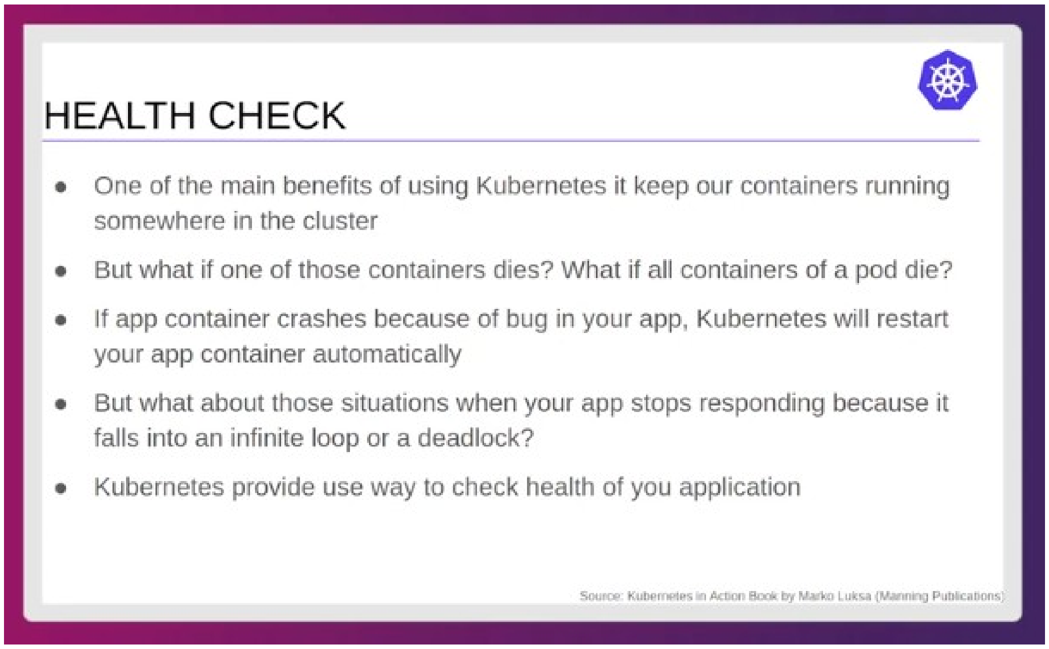
For creating *the service of the* ***LoadBalancer*** *from command line so the kubernetes gives us the expose command to get an ip.*

- ***kubectl expose pod***

This command means that if we want to get and ip for single pod so we can also this. So we should have to choose the group of the pods so the kubernetes can handle the load. If not then the problem we have to face.

- ***kubectl expose rs my1strs --name=myclsvc --selector=app=rsexample --port=8000 --target-port=80 --type=LoadBalancer***

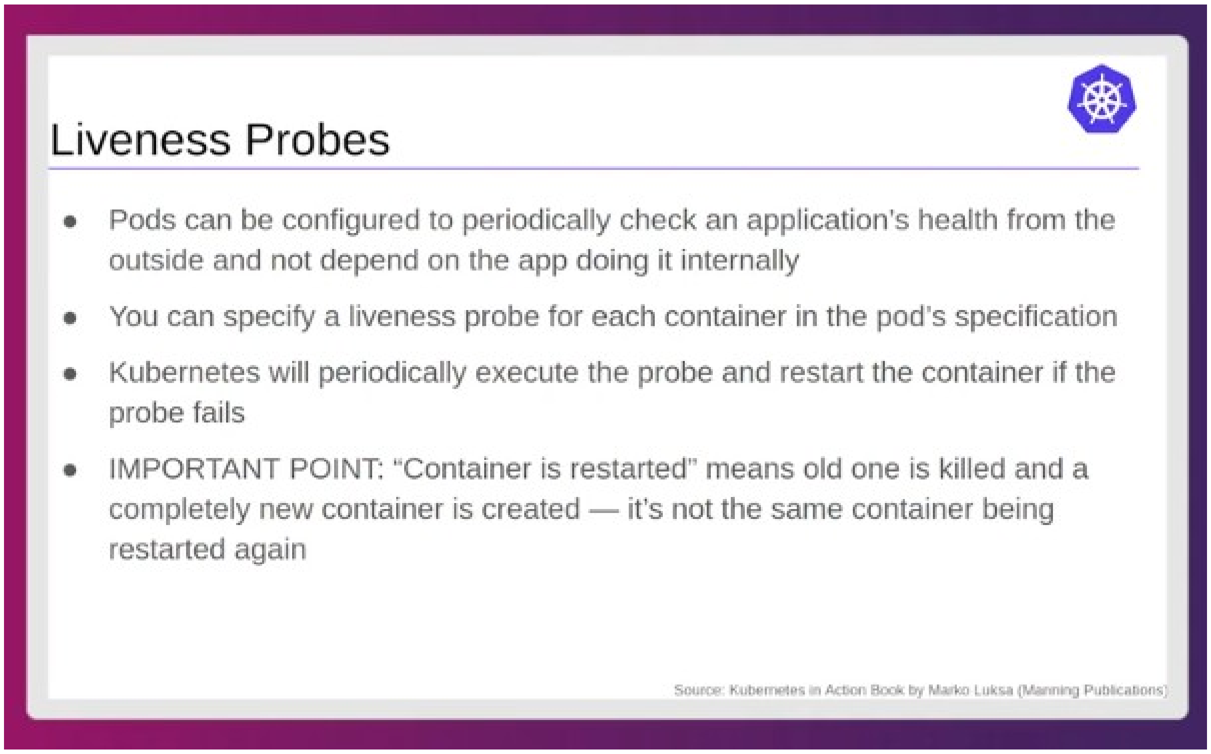
Video 7. "Liveness Probe"



HEALTH CHECK

- One of the main benefits of using Kubernetes it keep our containers running somewhere in the cluster
- But what if one of those containers dies? What if all containers of a pod die?
- If app container crashes because of bug in your app, Kubernetes will restart your app container automatically
- But what about those situations when your app stops responding because it falls into an infinite loop or a deadlock?
- Kubernetes provide use way to check health of you application

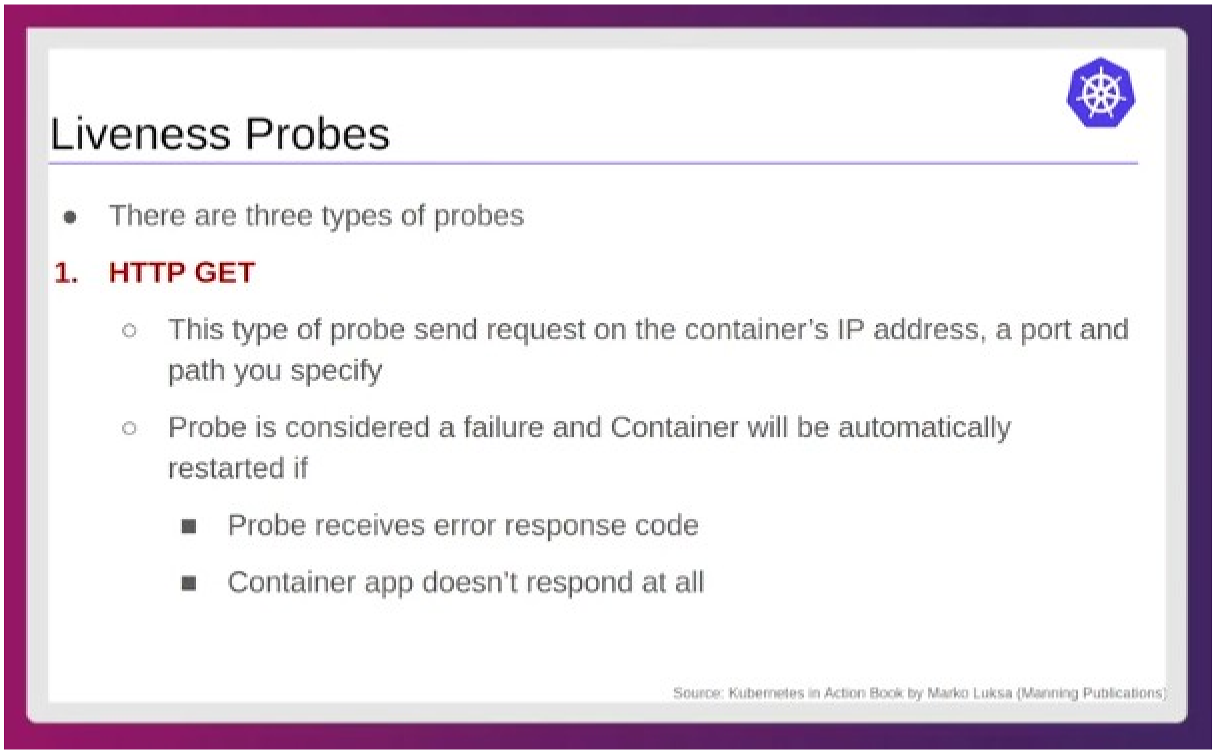
Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)



Liveness Probes

- Pods can be configured to periodically check an application's health from the outside and not depend on the app doing it internally
- You can specify a liveness probe for each container in the pod's specification
- Kubernetes will periodically execute the probe and restart the container if the probe fails
- IMPORTANT POINT: "Container is restarted" means old one is killed and a completely new container is created — it's not the same container being restarted again

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

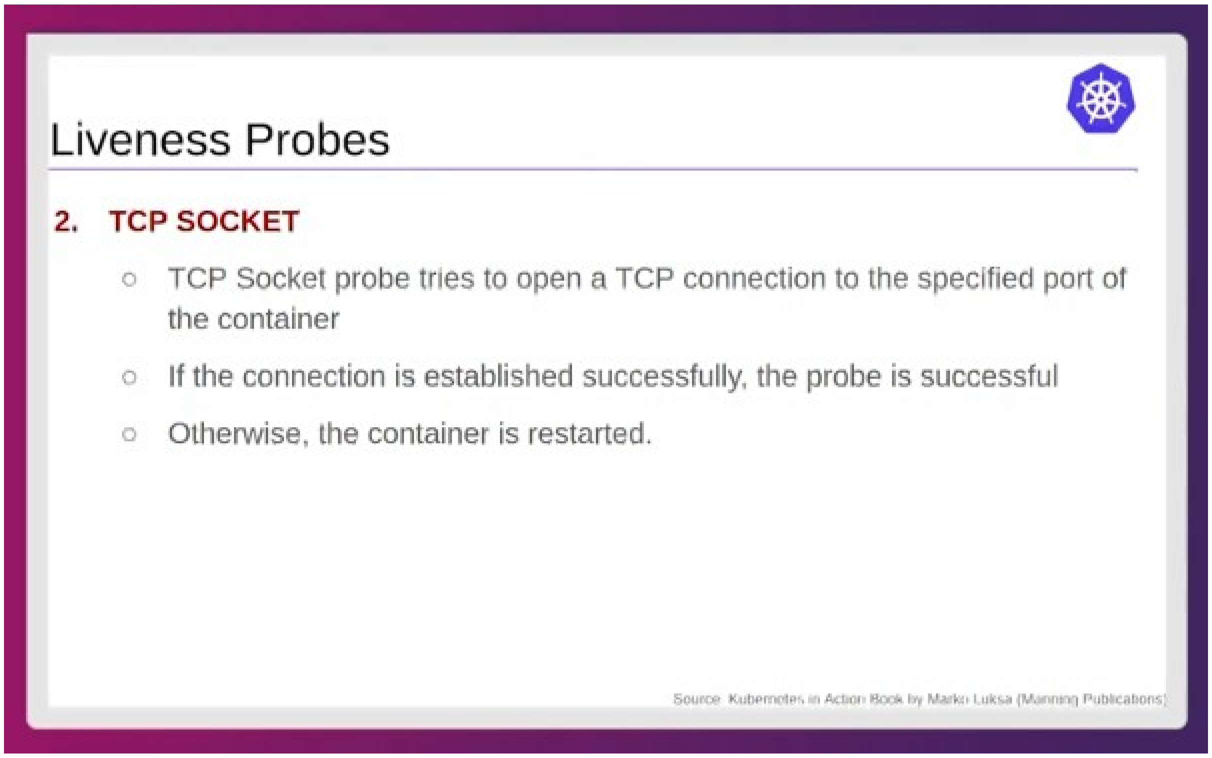


Liveness Probes

- There are three types of probes

1. **HTTP GET**
    - This type of probe send request on the container's IP address, a port and path you specify
    - Probe is considered a failure and Container will be automatically restarted if
        - Probe receives error response code
        - Container app doesn't respond at all

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

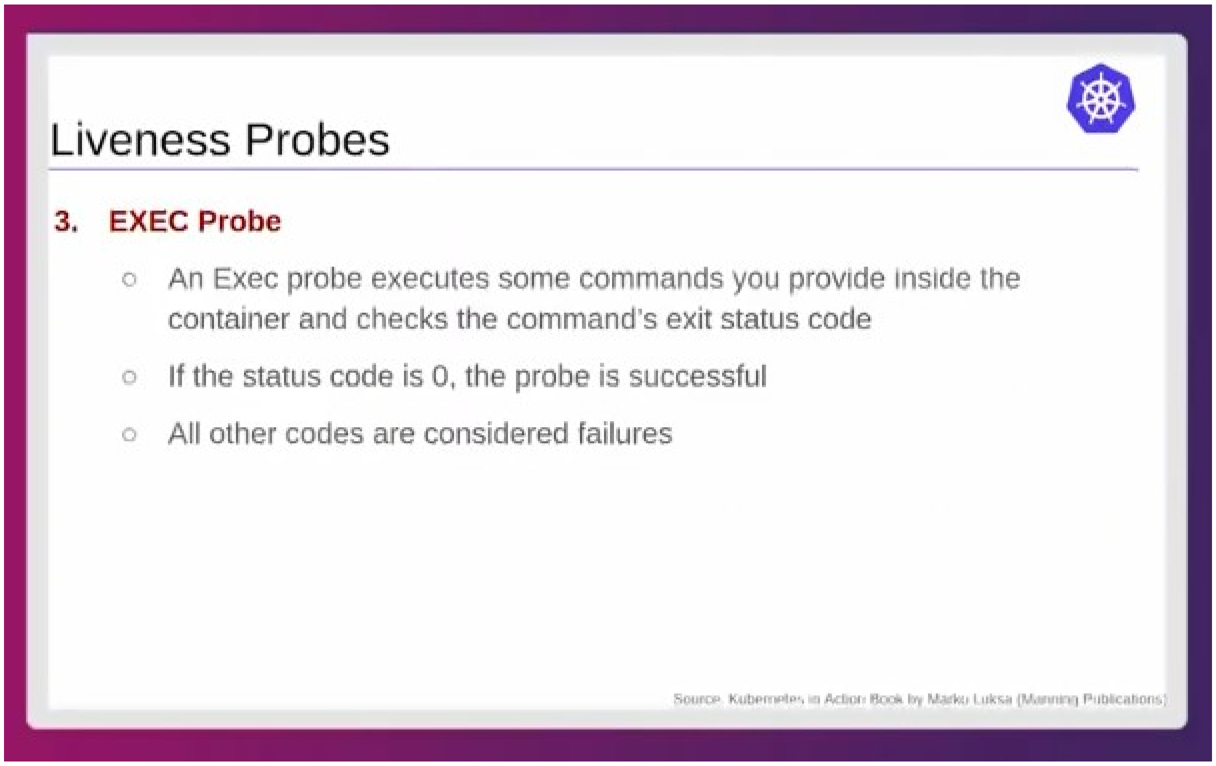


Liveness Probes

2. TCP SOCKET

- TCP Socket probe tries to open a TCP connection to the specified port of the container
- If the connection is established successfully, the probe is successful
- Otherwise, the container is restarted.

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)



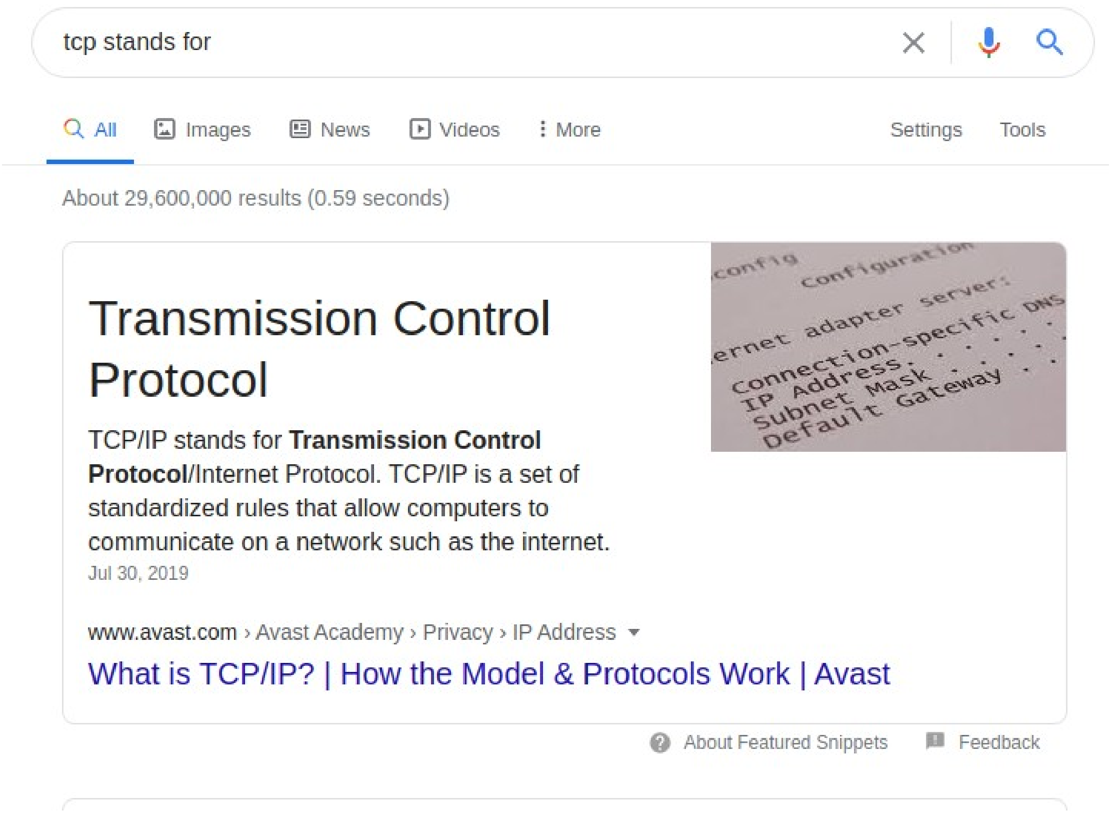
Liveness Probes

3. EXEC Probe

- An Exec probe executes some commands you provide inside the container and checks the command's exit status code
- If the status code is 0, the probe is successful
- All other codes are considered failures

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Some Extra Concepts



tcp stands for

All | Images | News | Videos | More | Settings | Tools

About 29,600,000 results (0.59 seconds)

Transmission Control Protocol

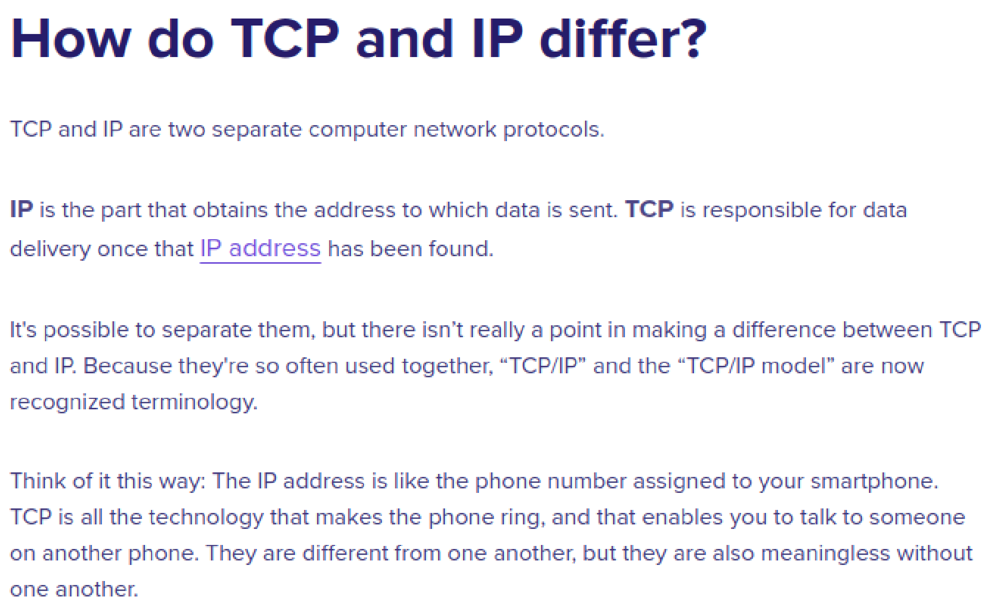
TCP/IP stands for **Transmission Control Protocol**/Internet Protocol. TCP/IP is a set of standardized rules that allow computers to communicate on a network such as the internet.

Jul 30, 2019

www.avast.com › Avast Academy › Privacy › IP Address

What is TCP/IP? | How the Model & Protocols Work | Avast

About Featured Snippets | Feedback



How do TCP and IP differ?

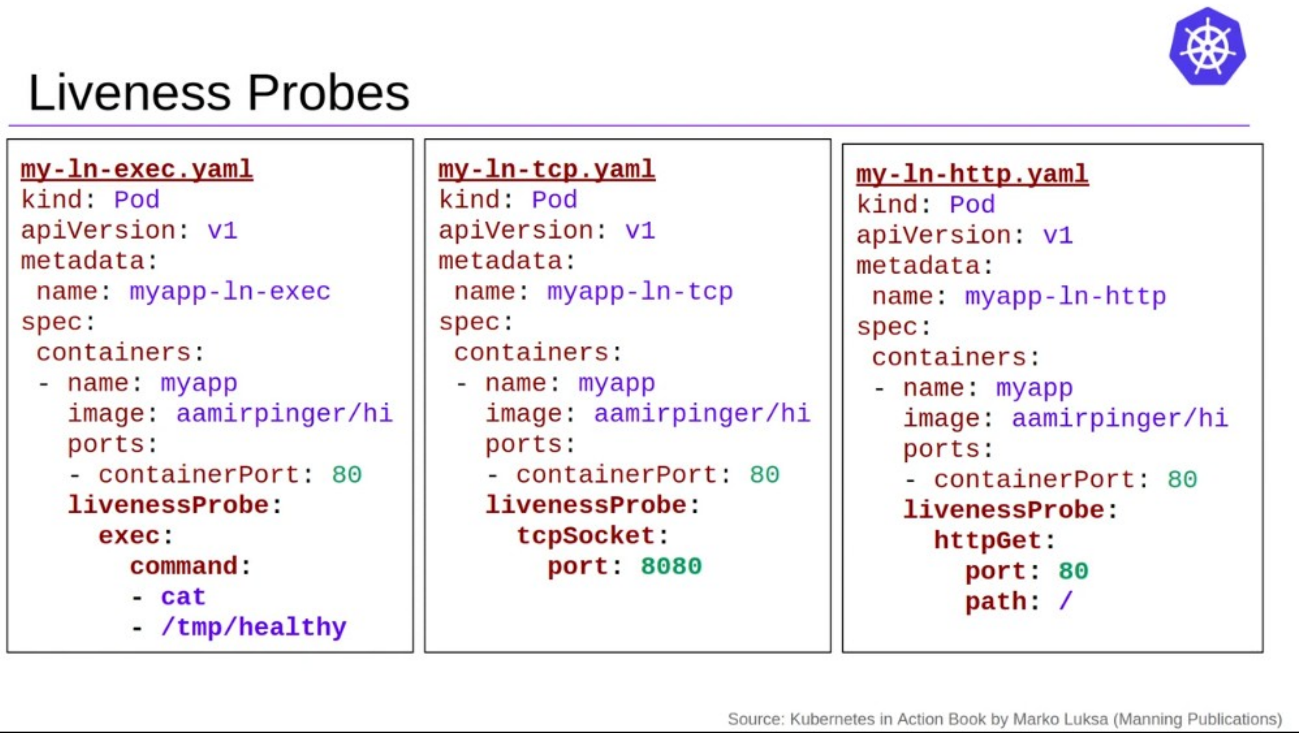
TCP and IP are two separate computer network protocols.

IP is the part that obtains the address to which data is sent. **TCP** is responsible for data delivery once that IP address has been found.

It's possible to separate them, but there isn't really a point in making a difference between TCP and IP. Because they're so often used together, "TCP/IP" and the "TCP/IP model" are now recognized terminology.

Think of it this way: The IP address is like the phone number assigned to your smartphone. TCP is all the technology that makes the phone ring, and that enables you to talk to someone on another phone. They are different from one another, but they are also meaningless without one another.

Video 8. “Liveness Probe Example Part 1”



Liveness Probes

```
my-ln-exec.yaml
kind: Pod
apiVersion: v1
metadata:
 name: myapp-ln-exec
spec:
 containers:
 - name: myapp
   image: aamirpinger/hi
   ports:
   - containerPort: 80
   livenessProbe:
     exec:
       command:
       - cat
       - /tmp/healthy
```

```
my-ln-tcp.yaml
kind: Pod
apiVersion: v1
metadata:
 name: myapp-ln-tcp
spec:
 containers:
 - name: myapp
   image: aamirpinger/hi
   ports:
   - containerPort: 80
   livenessProbe:
     tcpSocket:
       port: 8080
```

```
my-ln-http.yaml
kind: Pod
apiVersion: v1
metadata:
 name: myapp-ln-http
spec:
 containers:
 - name: myapp
   image: aamirpinger/hi
   ports:
   - containerPort: 80
   livenessProbe:
     httpGet:
       port: 80
       path: /
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

```
apiVersion: v1
kind: Pod
metadata:
  name: lp-podnew
spec:
  containers:
  - name: container1
    image: aamirpinger/hi:latest
    ports:
    - containerPort: 80
    livenessProbe:
      exec:
        command:
        - ls
      initialDelaySeconds: 5
      periodSeconds: 5
```

In this the ls is the simple linux command and if the command will be successful run the and not restart again again the container after specified time period so the health of the pod is ok

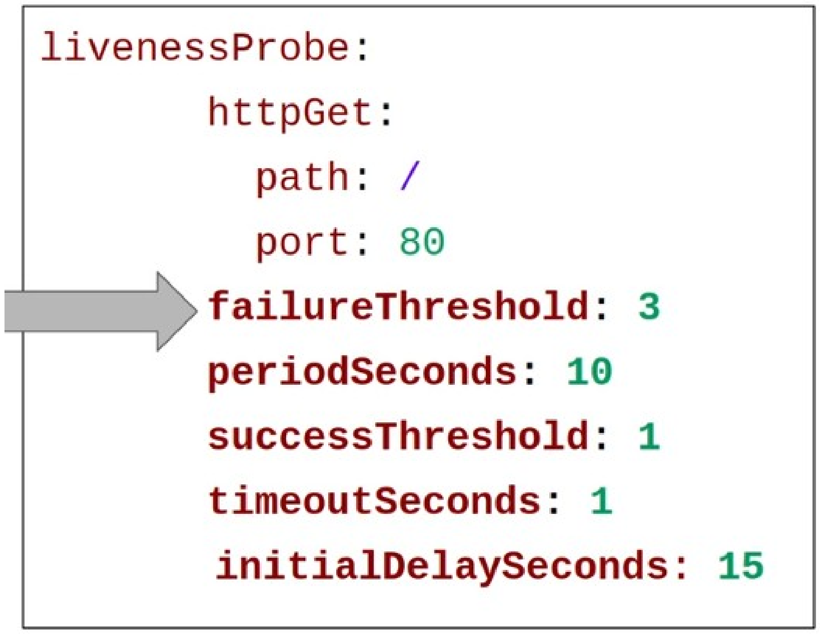
Video 9. "Liveness Probe Example Part 2"

```
apiVersion: v1
kind: Pod
metadata:
  name: lp-podnew
spec:
  containers:
  - name: container1
    image: aamirpinger/hi:latest
    ports:
    - containerPort: 80
    livenessProbe:
      exec:
        command:
        - cat
        - abc.txt
      initialDelaySeconds: 5
      periodSeconds: 5
```

Here I have entered the command of cat and the file of that which is not exists on the filesystem of the container so the container will be restart again again in the specified period of time.

Video 10. "Liveness Probe Additional Properties"

There are some extra properties of the liveness probe that which can be defined with any number in any liveness probe



```
livenessProbe:
        httpGet:
          path: /
          port: 80
        failureThreshold: 3
        periodSeconds: 10
        successThreshold: 1
        timeoutSeconds: 1
        initialDelaySeconds: 15
```

- ***failureThreshold: 3***

_By default the failureThreshold count is 3.
The purpose of the this is to check if the probe/command fail for consecutive 3 times so the container restart. This count for this threshold we can do more or less too.

- ***PeriodSeconds: 10***

Runs the liveness probe every 10 seconds. The liveness probe will be run this by every 10 seconds for the checking of the health of the pod.

- ***SuccessThreshold: 1***

Reset the faiureThreshold counter on 1 successful response. And if the consecutive 3 times the result is failure so the probe will consider failure and the container will be restarted.

- ***<u>timeoutSeconds: 1</u>***

The response must come within one second. Mark failure even the successful response come after specified time in seconds.

- ***InitialDelaySeconds: 15***

The kubernetes will wait for 15 seconds for starting the first liveness probe.
Note: *For example our application is big will be consume 60 seconds for up and running so the first probe will be start after 15 seconds so the application will be considered as failure because the app is not in the running state so this will be an infinite loop.* ***So if*** *our application will take the 60 seconds so we should give 65-70 seconds 5-10 seconds as grace seconds for the starting the app.*

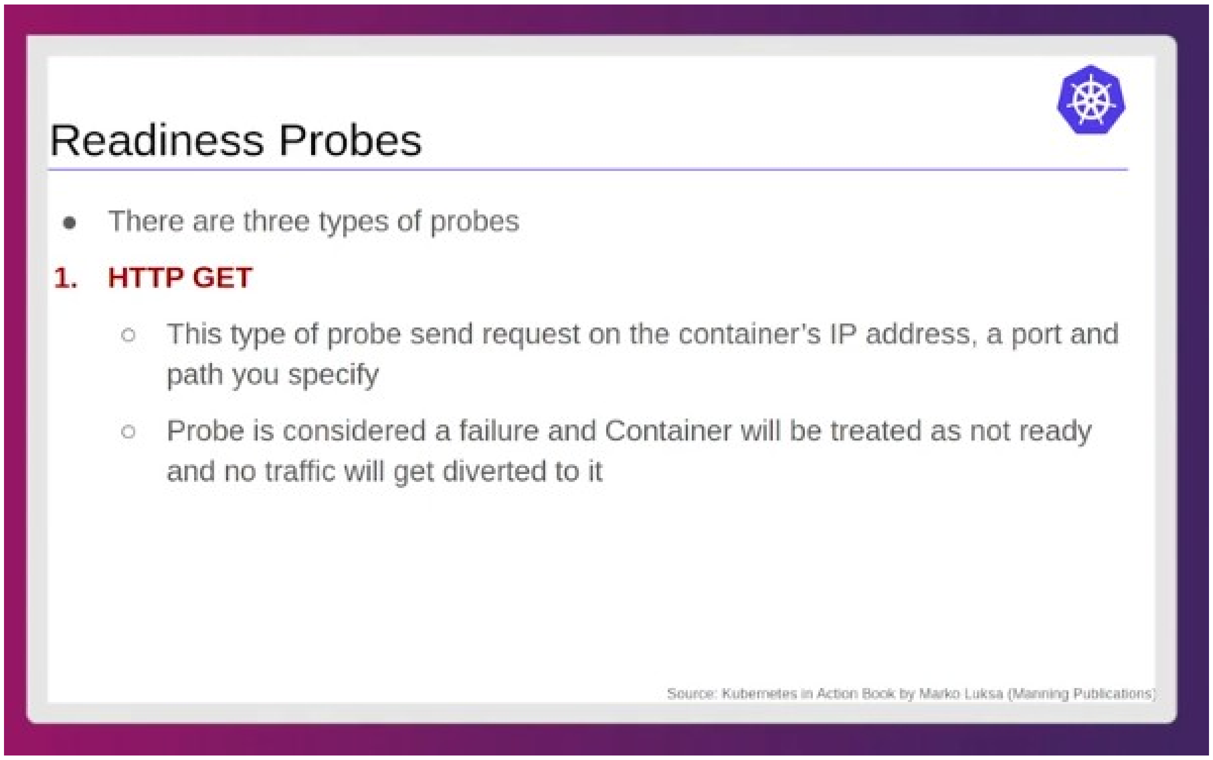
<u>Video 11. "Readiness Probe"</u>

<u>Q. what will be action that when the liveness and readiness work at same time or same container ??</u>

<u>This video is practical based.</u>

<u>I left lot of blank space on this page due adjustments of the content.</u>

Video 12. “Readiness Probe Types”



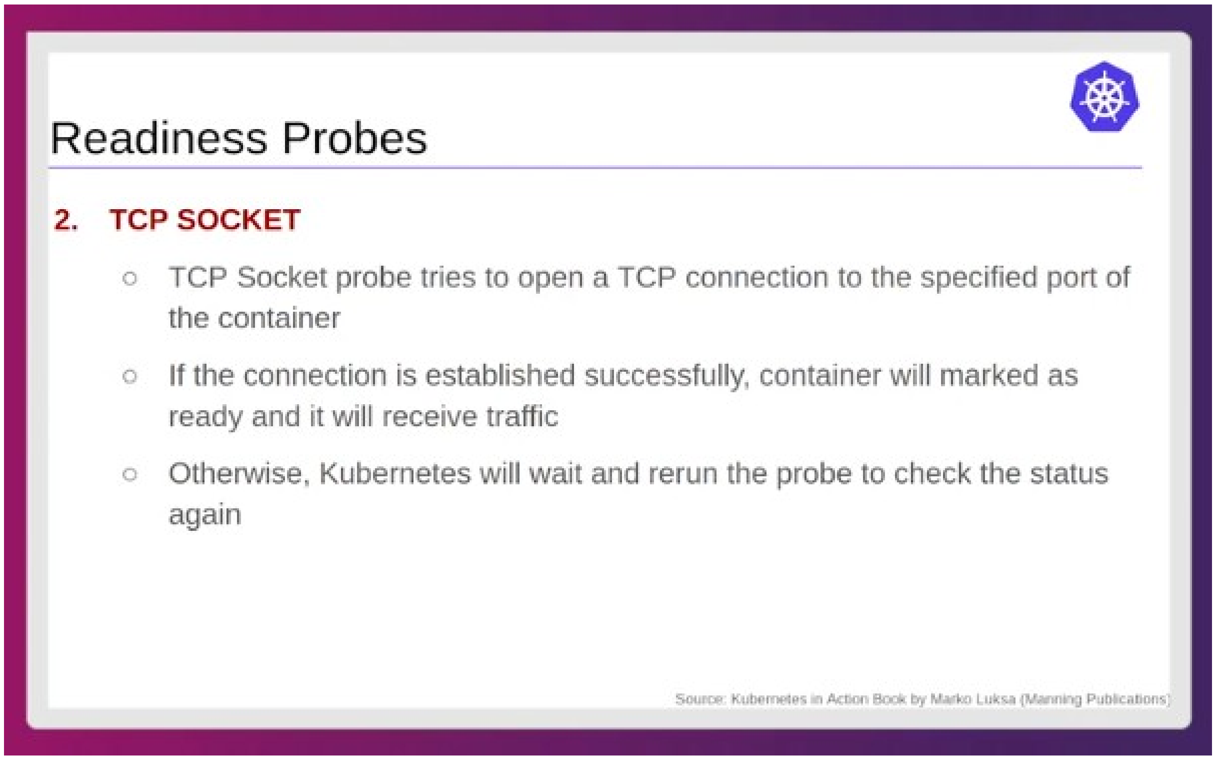
Readiness Probes

- There are three types of probes

1. HTTP GET

- This type of probe send request on the container’s IP address, a port and path you specify
- Probe is considered a failure and Container will be treated as not ready and no traffic will get diverted to it

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

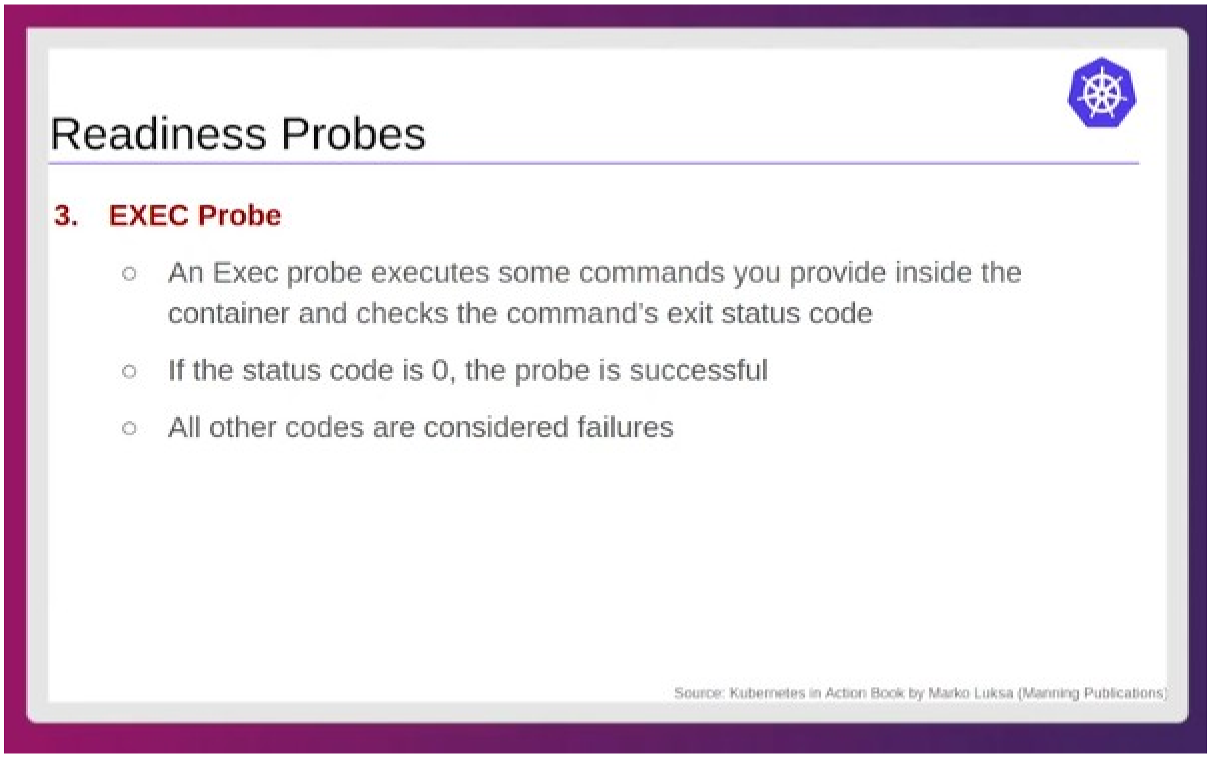


Readiness Probes

2. TCP SOCKET

- TCP Socket probe tries to open a TCP connection to the specified port of the container
- If the connection is established successfully, container will marked as ready and it will receive traffic
- Otherwise, Kubernetes will wait and rerun the probe to check the status again

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)



Readiness Probes

3. EXEC Probe

- An Exec probe executes some commands you provide inside the container and checks the command's exit status code
- If the status code is 0, the probe is successful
- All other codes are considered failures

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Video 13. "Readiness Probe Example"

```
apiVersion: v1
kind: Pod
metadata:
  name: rpodnew
spec:
  containers:
  - name: container1
    image: aamirpinger/hi
    ports:
    - containerPort: 80
    readinessProbe:
      httpGet:
        port: 90
        path: /
```

This video is practical based.

The Lesson 4 Of Kubernetes Completed.

The Services & Probs.

Thanks For Reading !

This Document Is Created By "Rizwan Sheikh"
Email: rizwansheikh7071@gmail.com
Docker ID: rizwansheikh7071
Due To Human Being The Mistake Can Be Done!
Thank You Happy Learning!