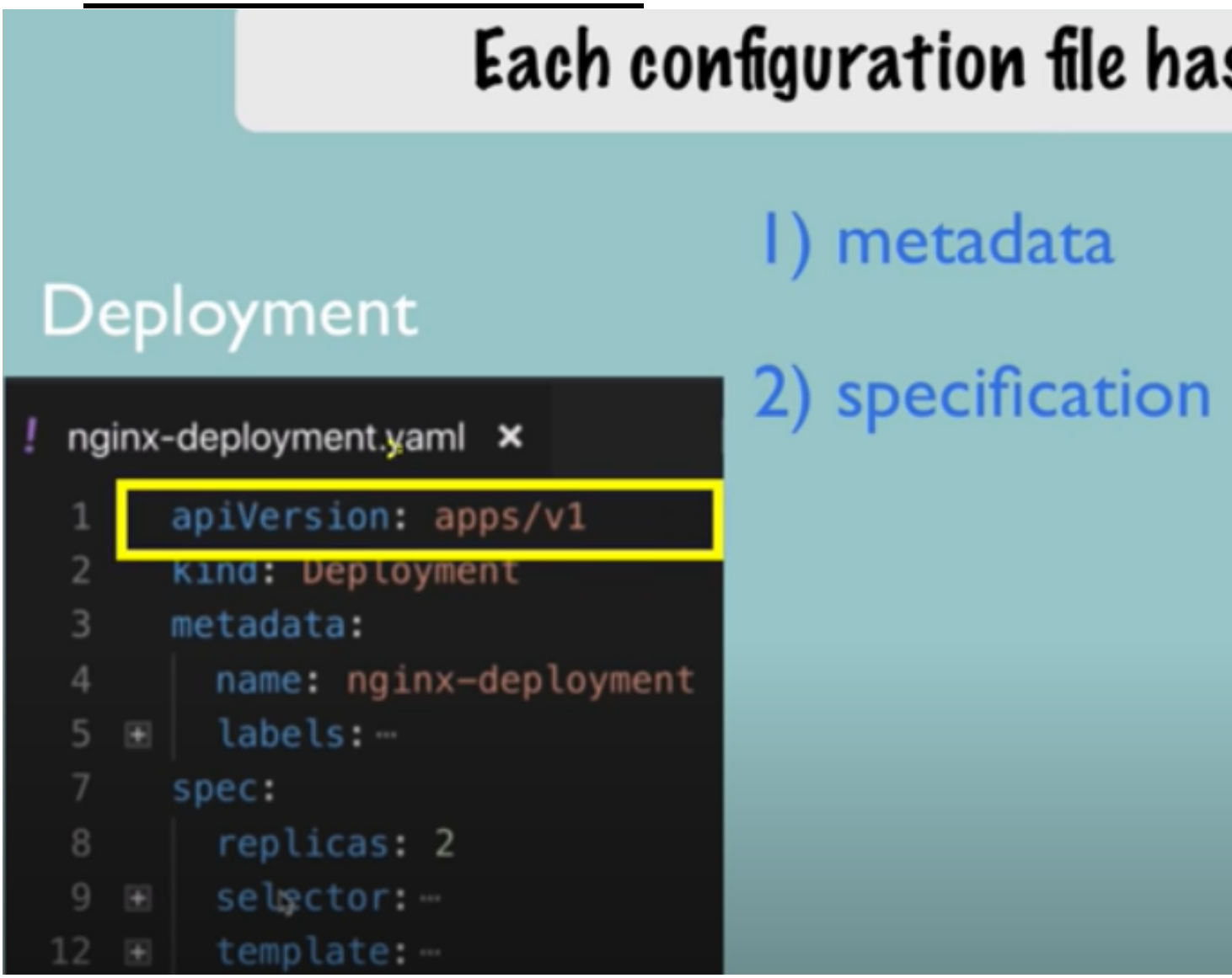
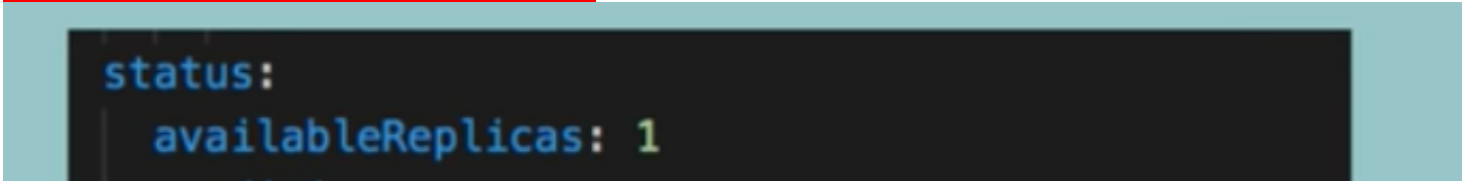
Kubernetes

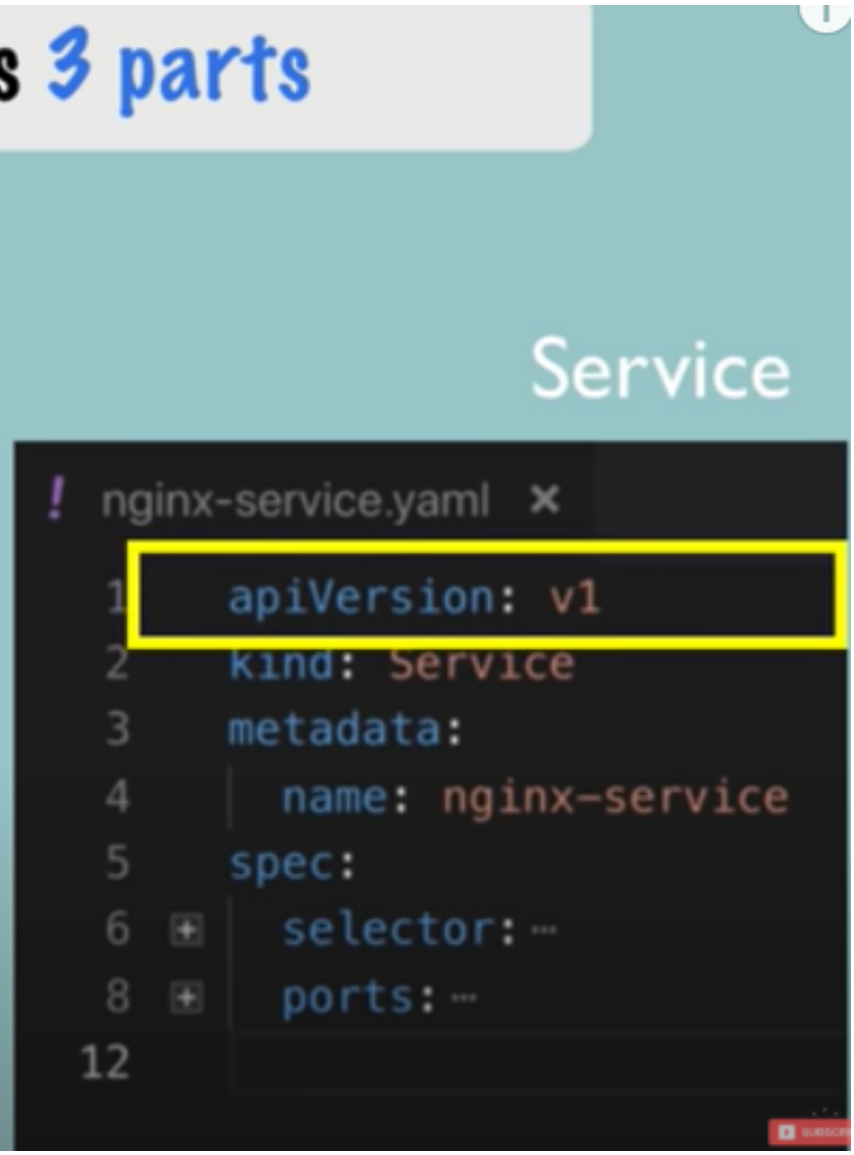
12 March 2021 12:57

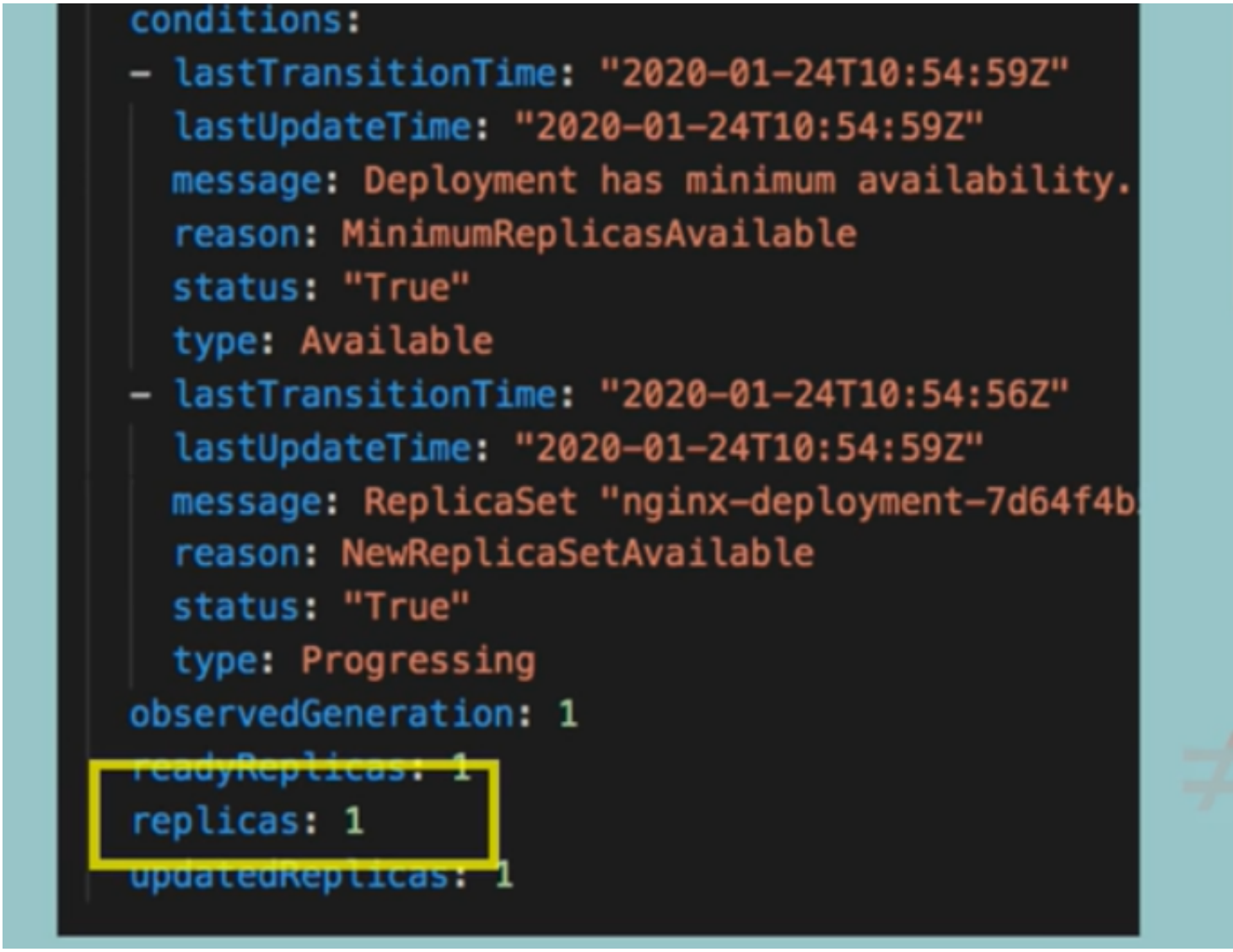
17) ========== YAML Configuration File ==========::



Here total 3 parts are there. And 3rd Part is called Status:Status is automatically generated and added by Kubernetes.

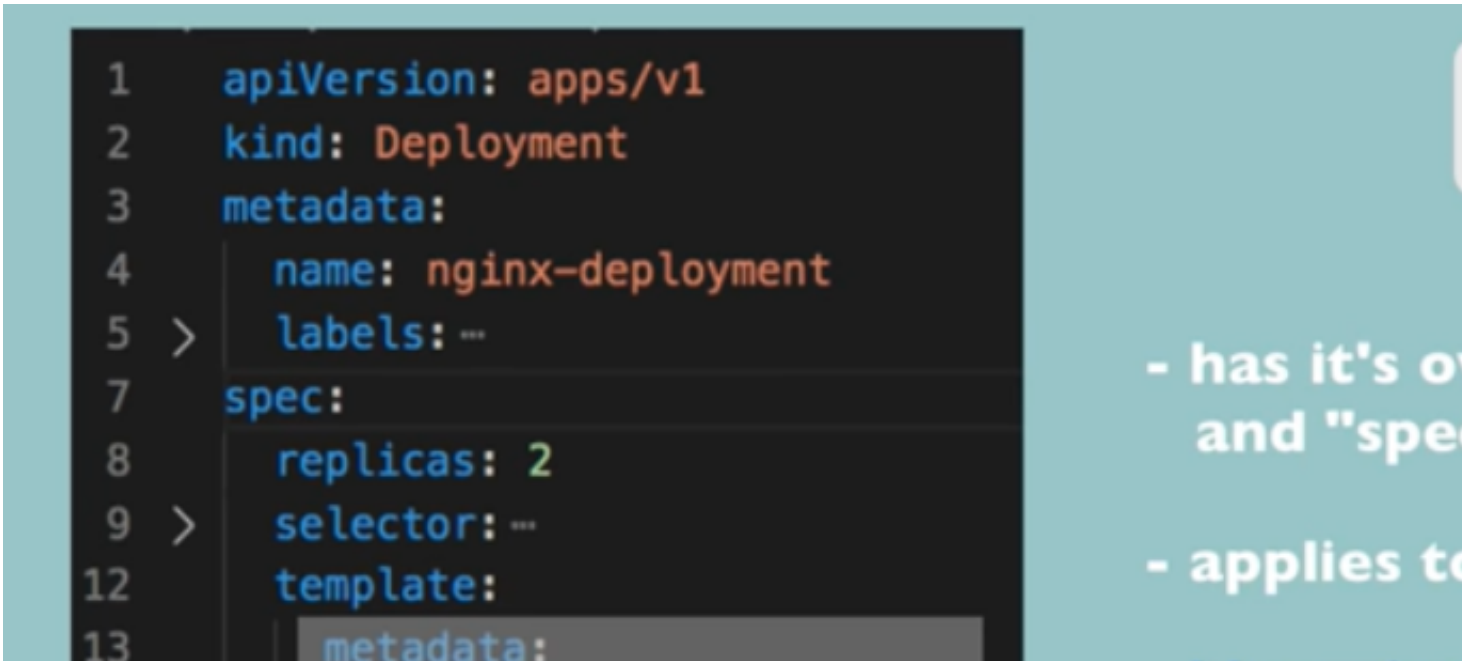


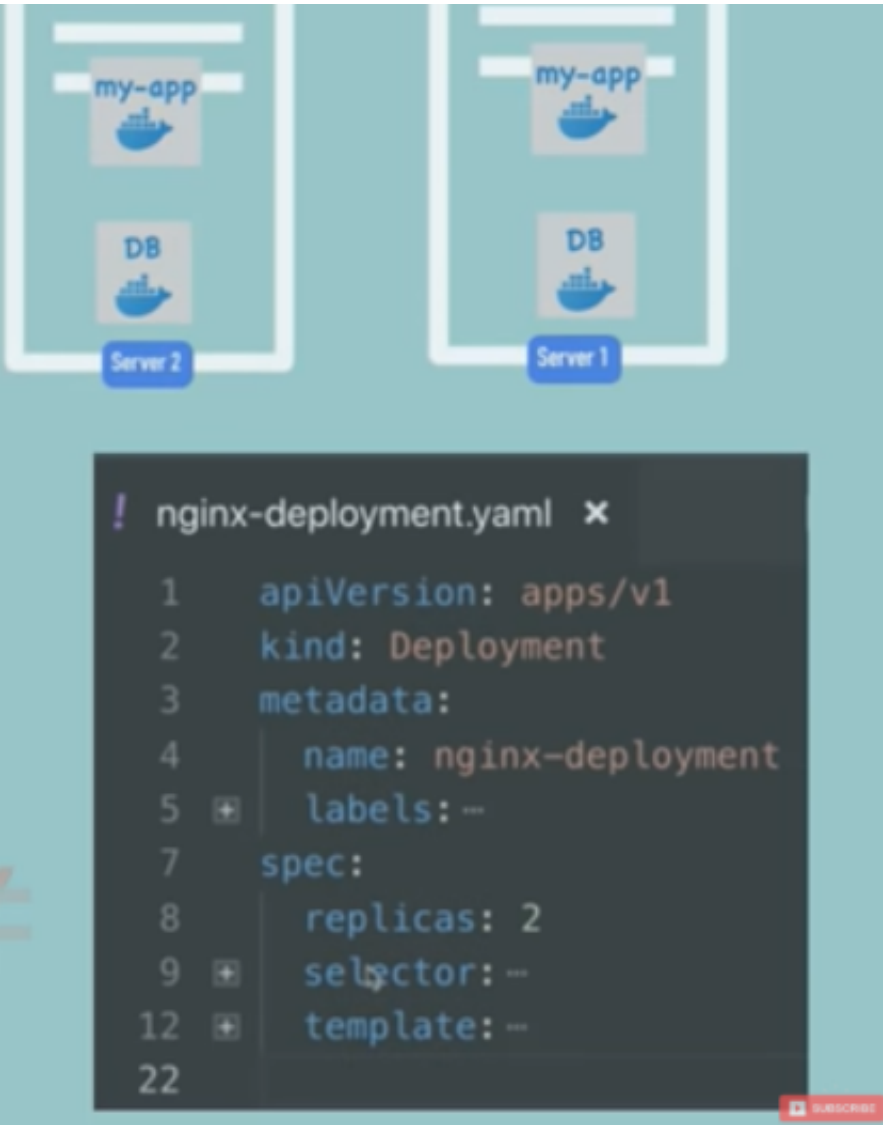

s 3 parts
Service
nginx-service.yaml
1 apiVersion: v1
2 kind: Service
3 metadata:
4 name: nginx-service
5 spec:
6 selector:
8 ports:
12

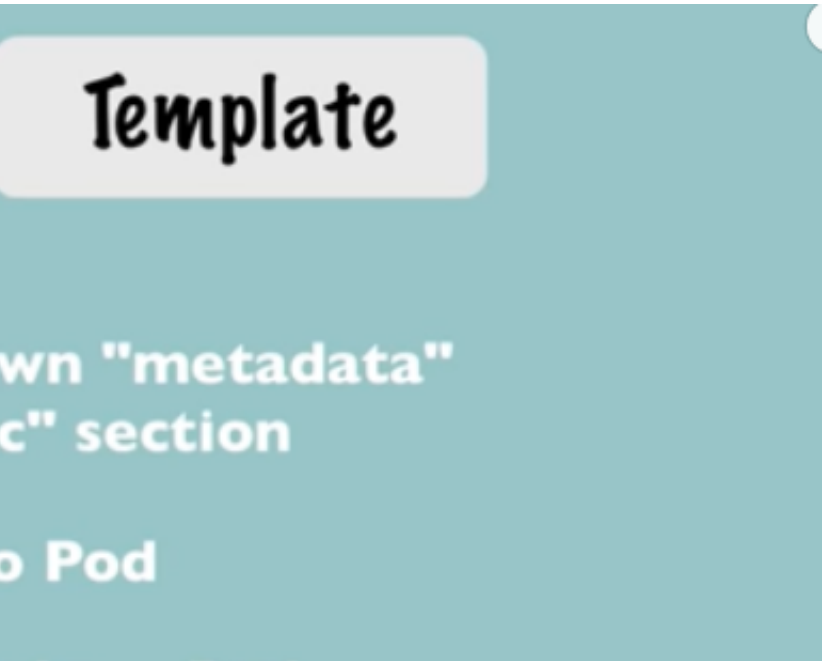


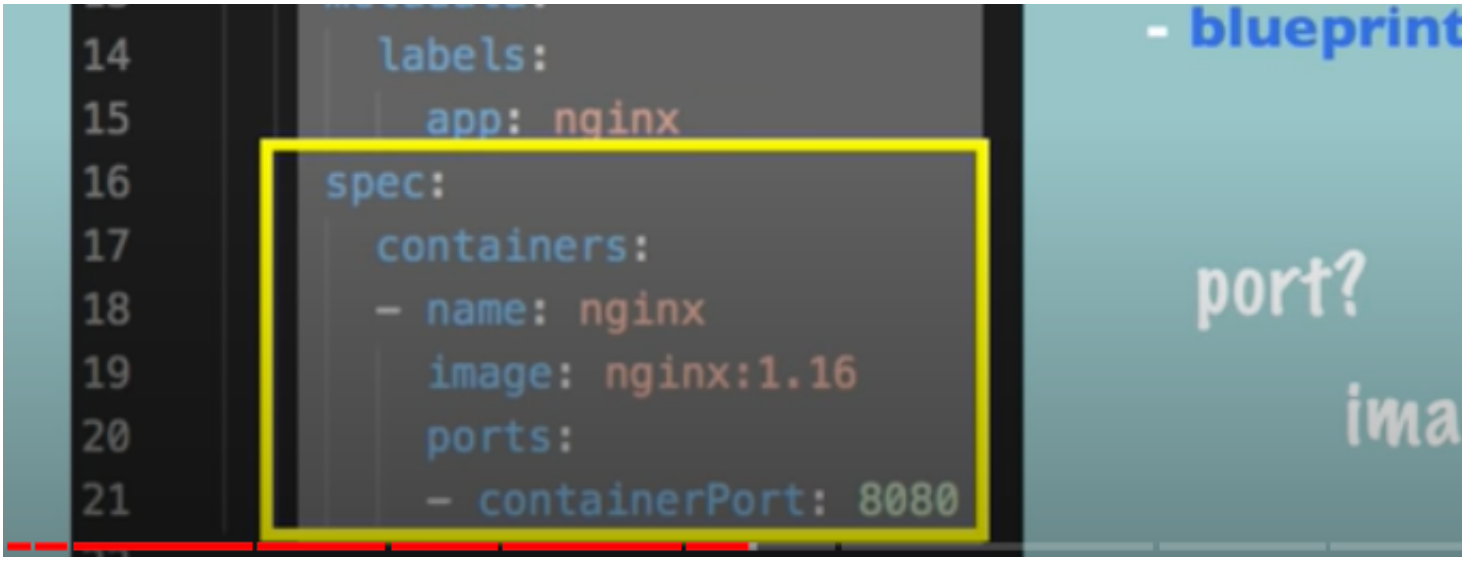
ETCD holds the CURRENT STATUS of K8s.

18)

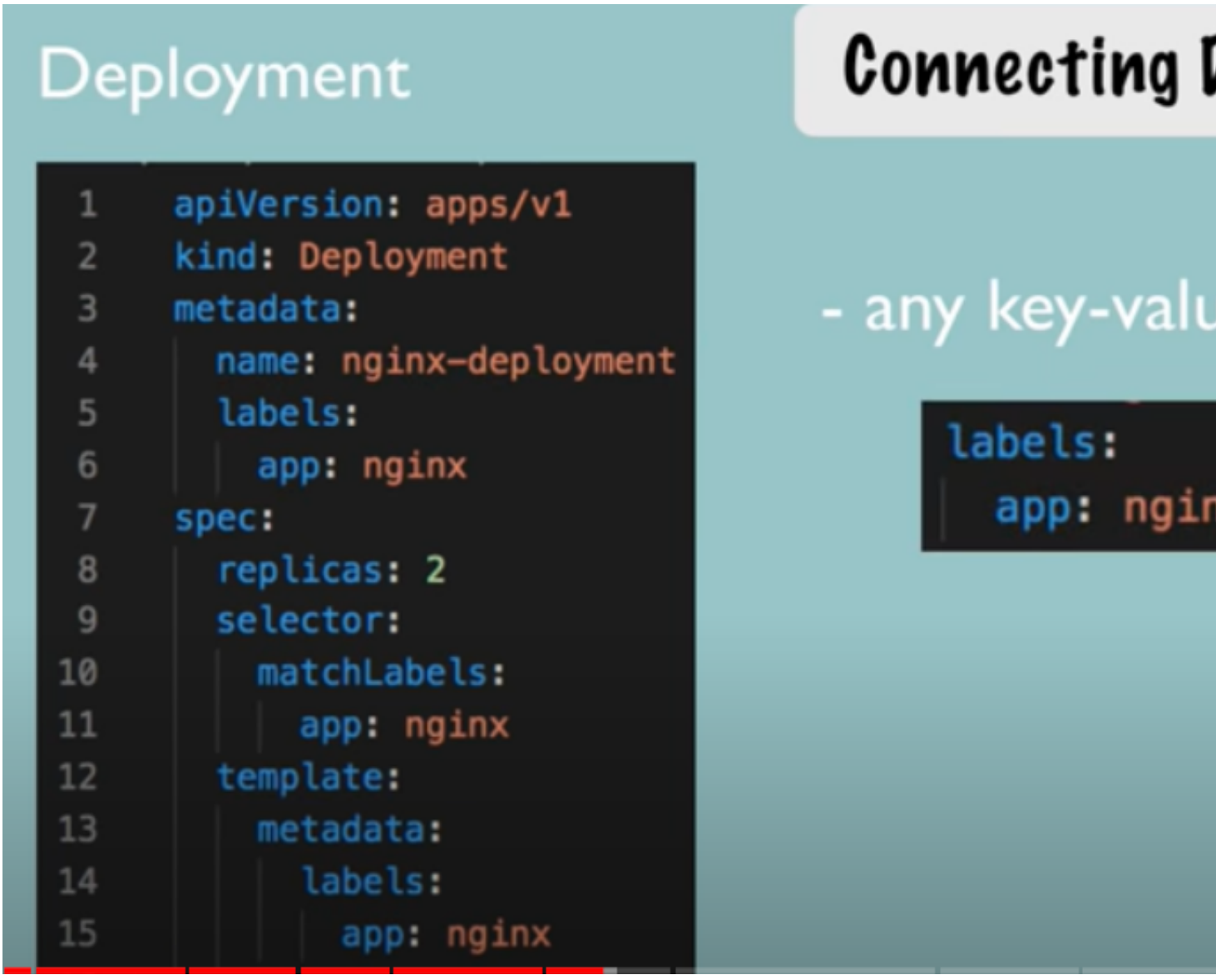



my-app
DB
Server 2
my-app
DB
Server 1
nginx-deployment.yaml
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4 name: nginx-deployment
5 labels:
7 spec:
8 replicas: 2
9 selector:
12 template:
22

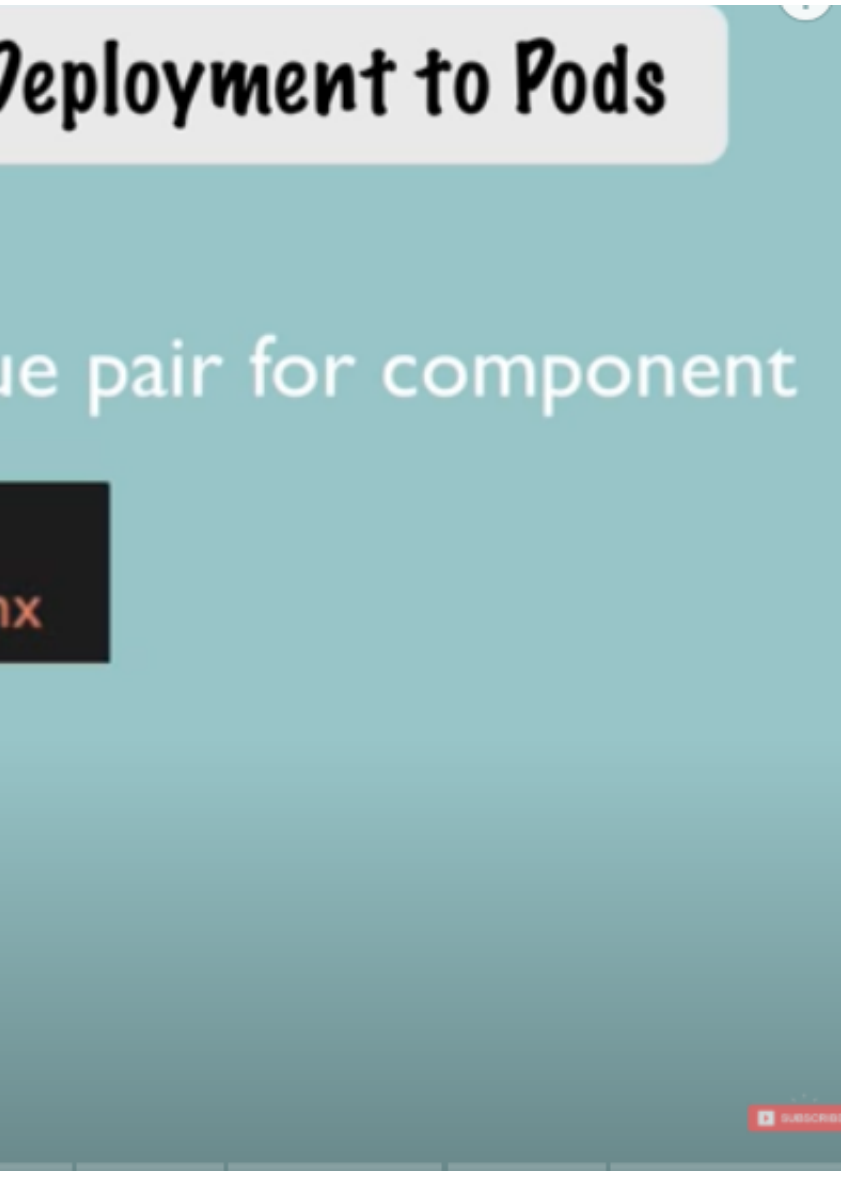

Template
wn "metadata"
c" section
o Pod


labels:
app: nginx
spec:
containers:
- name: nginx
image: nginx:1.16
ports:
- containerPort: 8080
- blueprint
port?

19)


Deployment
Connecting
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
labels:
app: nginx
spec:
replicas: 2
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
- any key-valu
labels:
app: ngin


for a Pod
name?
ge?


Deployment to Pods
ue pair for component
nx

Connecting Deployment

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec: …
```

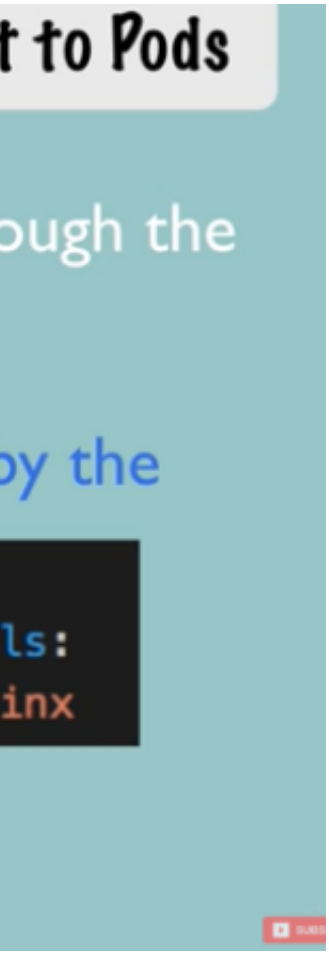
- Pods get the label thro
  template blueprint

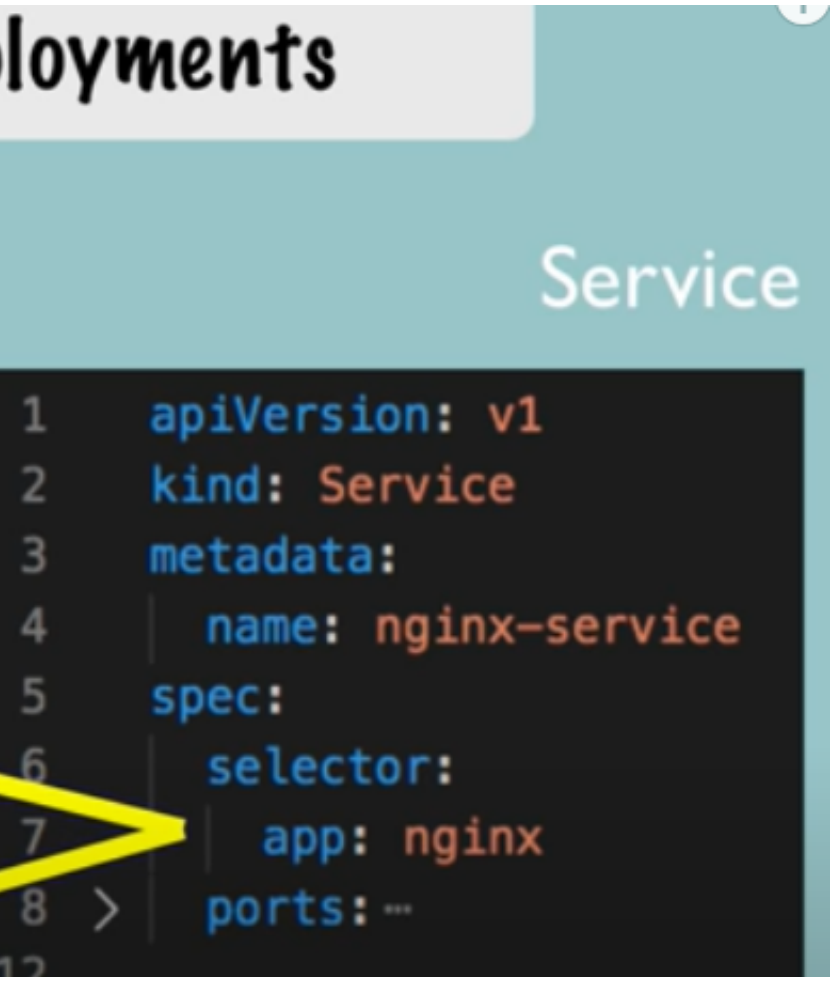
- This label is matched b
  selector

```yaml
selector:
  matchLabe
    app: ng
```

Connecting Services to Dep

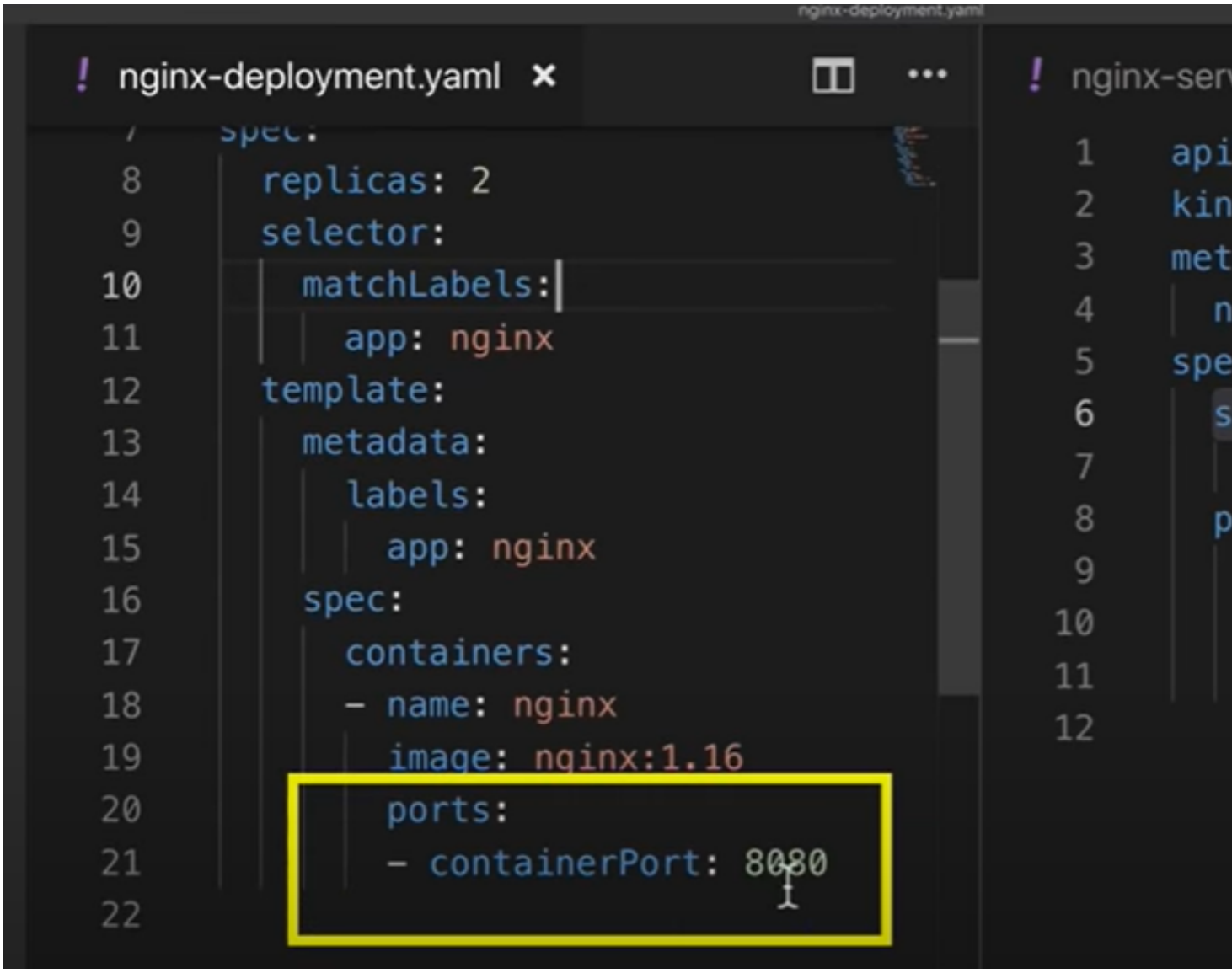
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
```


t to Pods
ough the
by the
ls:
inx

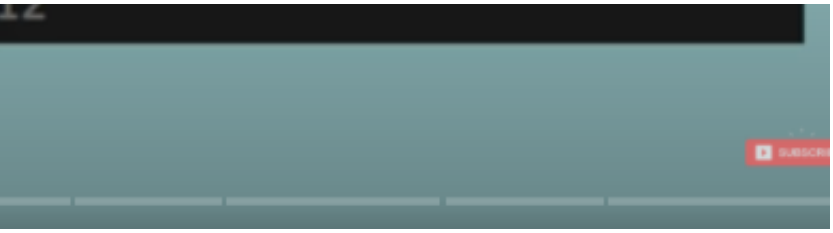

loyments
Service
1 apiVersion: v1
2 kind: Service
3 metadata:
4 name: nginx-service
5 spec:
6 selector:
7 app: nginx
8 ports:


13 metadata:
14 labels:
15 app: nginx

20)

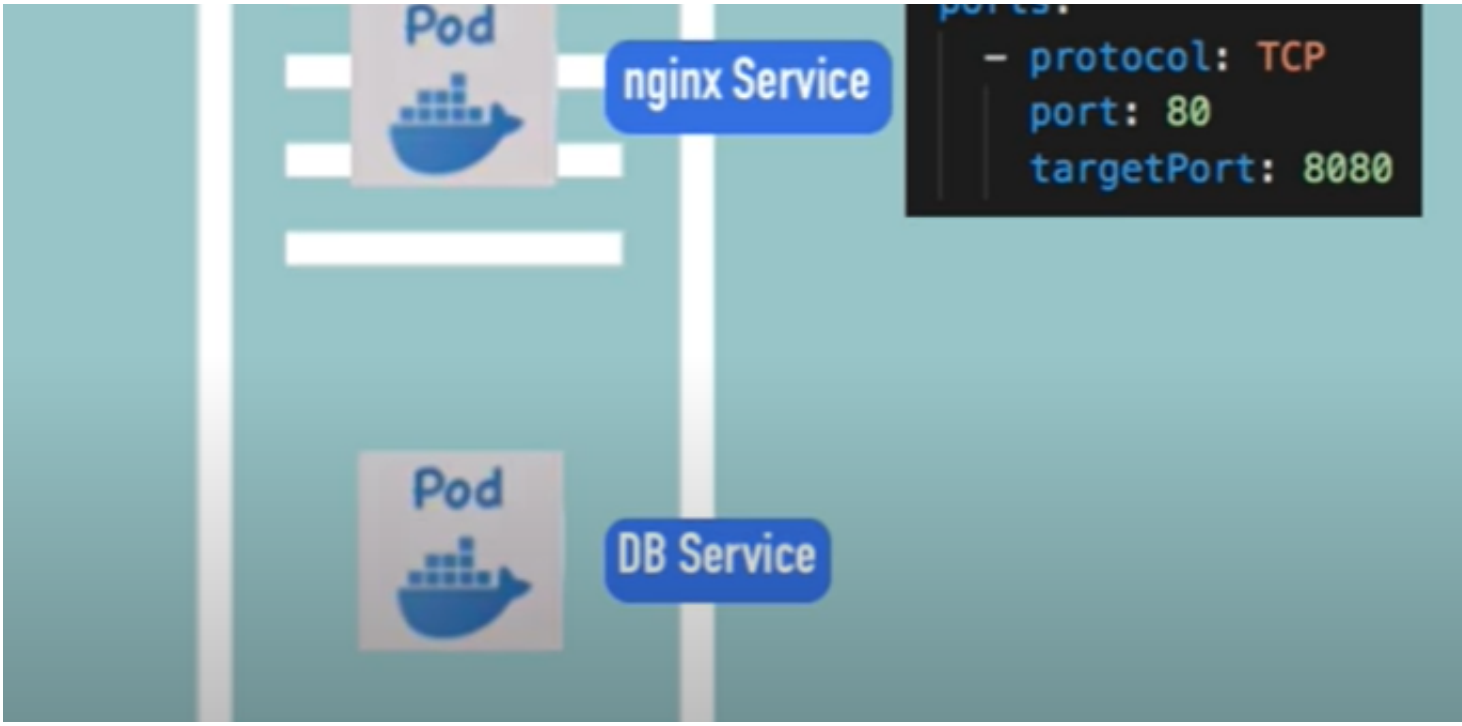

nginx-deployment.yaml
8 replicas: 2
9 selector:
10 matchLabels:
11 app: nginx
12 template:
13 metadata:
14 labels:
15 app: nginx
16 spec:
17 containers:
18 - name: nginx
19 image: nginx:1.16
20 ports:
21 - containerPort: 8080
22

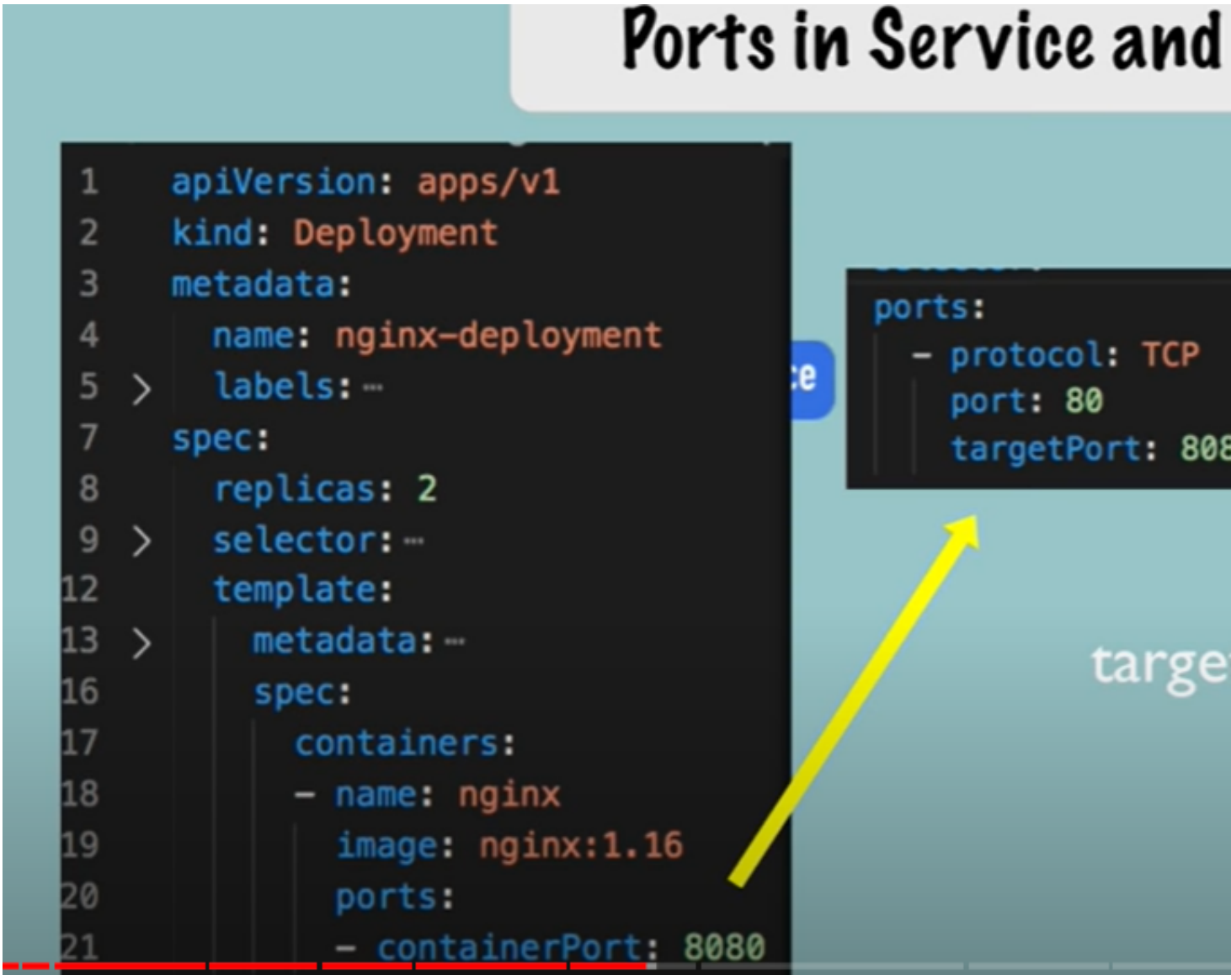

Ports in Service and Po

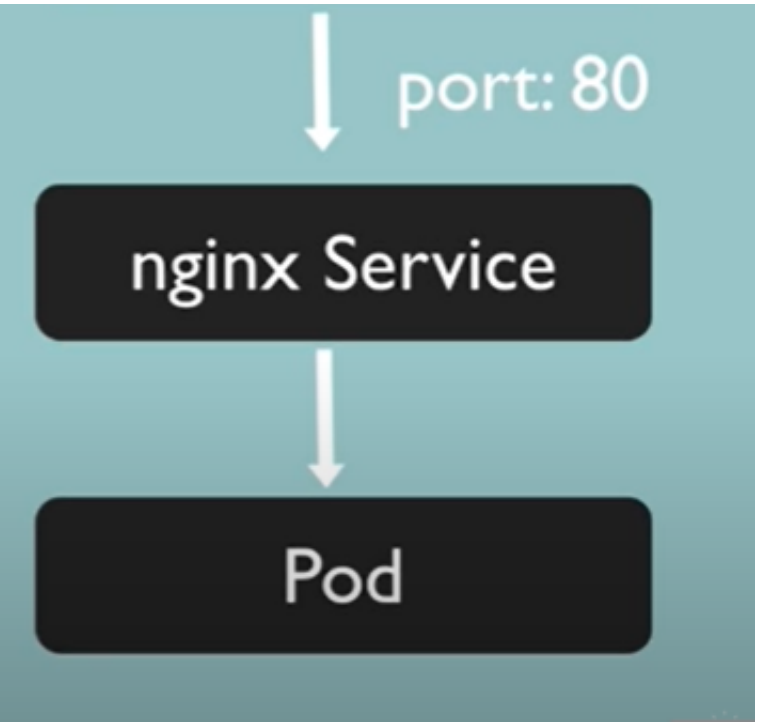


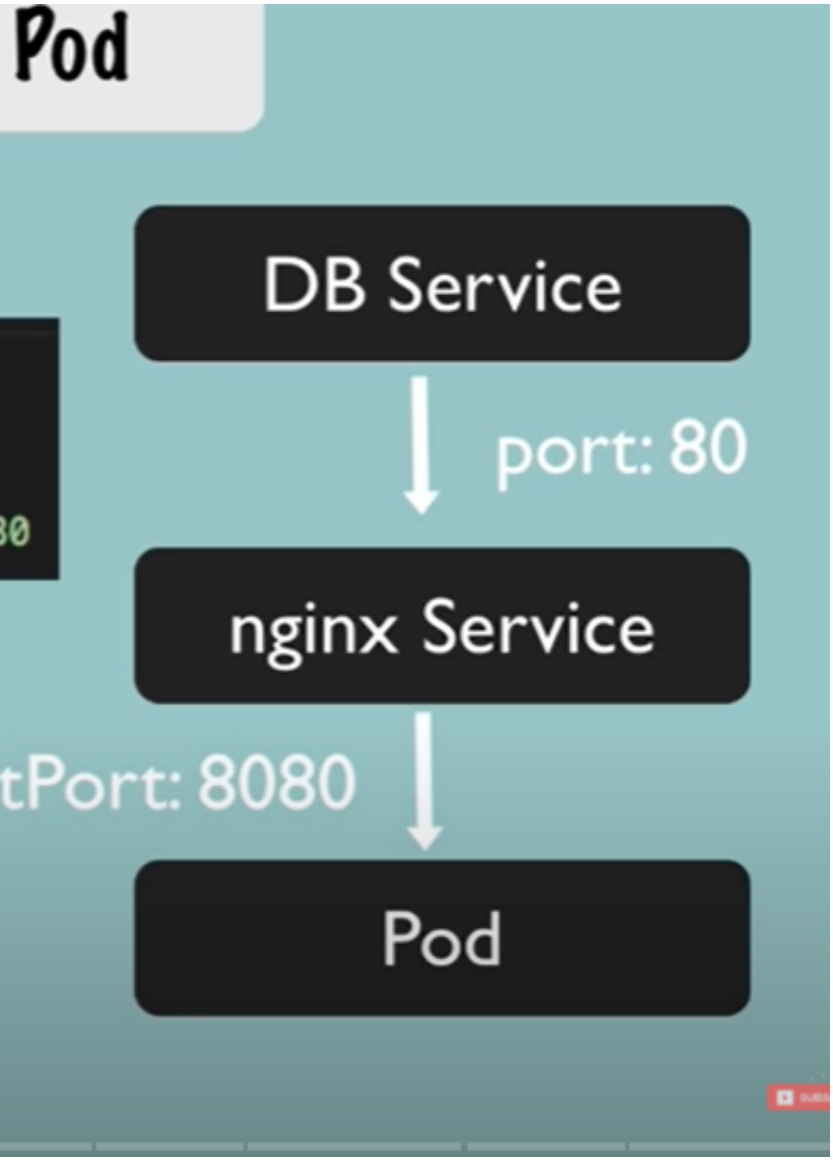
vice.yaml ×

```
Version: v1
d: Service
adata:
ame: nginx-service
c:
elector:
  app: nginx
orts:
  - protocol: TCP
    port: 80
    targetPort: 8080
```


Pod
nginx Service
- protocol: TCP
port: 80
targetPort: 8080
Pod
DB Service


Ports in Service and
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
labels:
spec:
replicas: 2
selector:
template:
metadata:
spec:
containers:
- name: nginx
image: nginx:1.16
ports:
- containerPort: 8080
ports:
- protocol: TCP
port: 80
targetPort: 808


port: 80
nginx Service
Pod

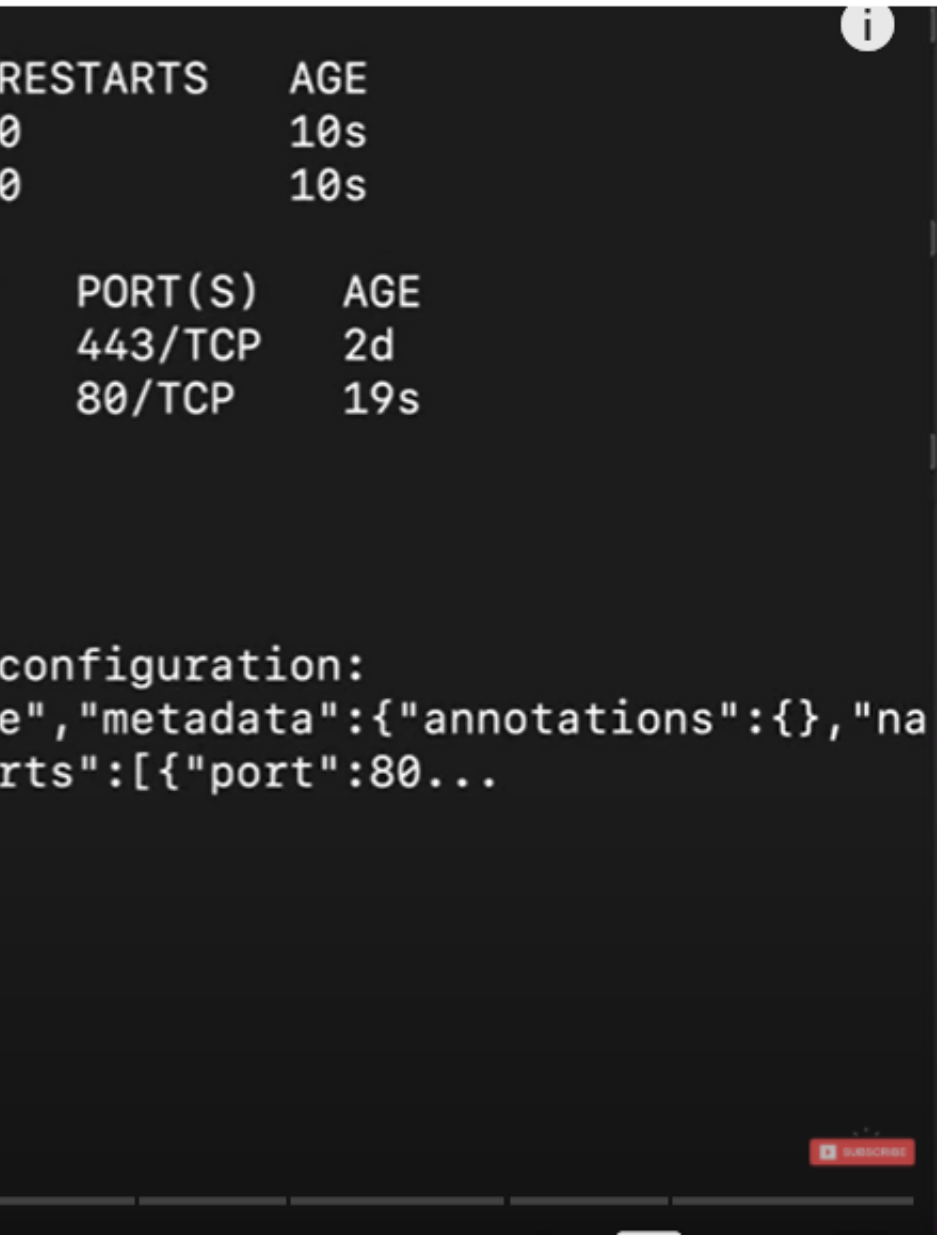

Pod
DB Service
port: 80
nginx Service
tPort: 8080
Pod

21)

```
[Documents]$ kubectl get pod
NAME                                READY   STATUS
nginx-deployment-7d64f4b574-fklxj   1/1     Running
nginx-deployment-7d64f4b574-v7mwj   1/1     Running
[Documents]$ kubectl get service
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP
kubernetes      ClusterIP   10.96.0.1       <none>
nginx-service   ClusterIP   10.96.25.229    <none>
[Documents]$ kubectl describe service nginx-service
Name:              nginx-service
Namespace:         default
Labels:            <none>
Annotations:       kubectl.kubernetes.io/last-applied-
                     {"apiVersion":"v1","kind":"Servic
me":"nginx-service","namespace":"default"},"spec":{"po
Selector:          app=nginx
Type:              ClusterIP
IP:                10.96.25.229
Port:              <unset>  80/TCP
TargetPort:        8080/TCP
Endpoints:         172.17.0.6:8080,172.17.0.7:8080
Session Affinity:  None
Events:            <none>
```

```
Events:            <none>
[Documents]$ kubectl get pod -o wide
NAME                                READY   STATUS
E       NOMINATED NODE   READINESS GATES
nginx-deployment-7d64f4b574-fklxj   1/1     Running
ikube    <none>          <none>
nginx-deployment-7d64f4b574-v7mwj   1/1     Running
ikube    <none>          <none>
```

Get Updated details of Deployment i.e. check how STATUS is getting saved in to .yaml file..


RESTARTS AGE
0 10s
0 10s
PORT(S) AGE
443/TCP 2d
80/TCP 19s
configuration:
e","metadata":{"annotations":{},"na
rts":[{"port":80...

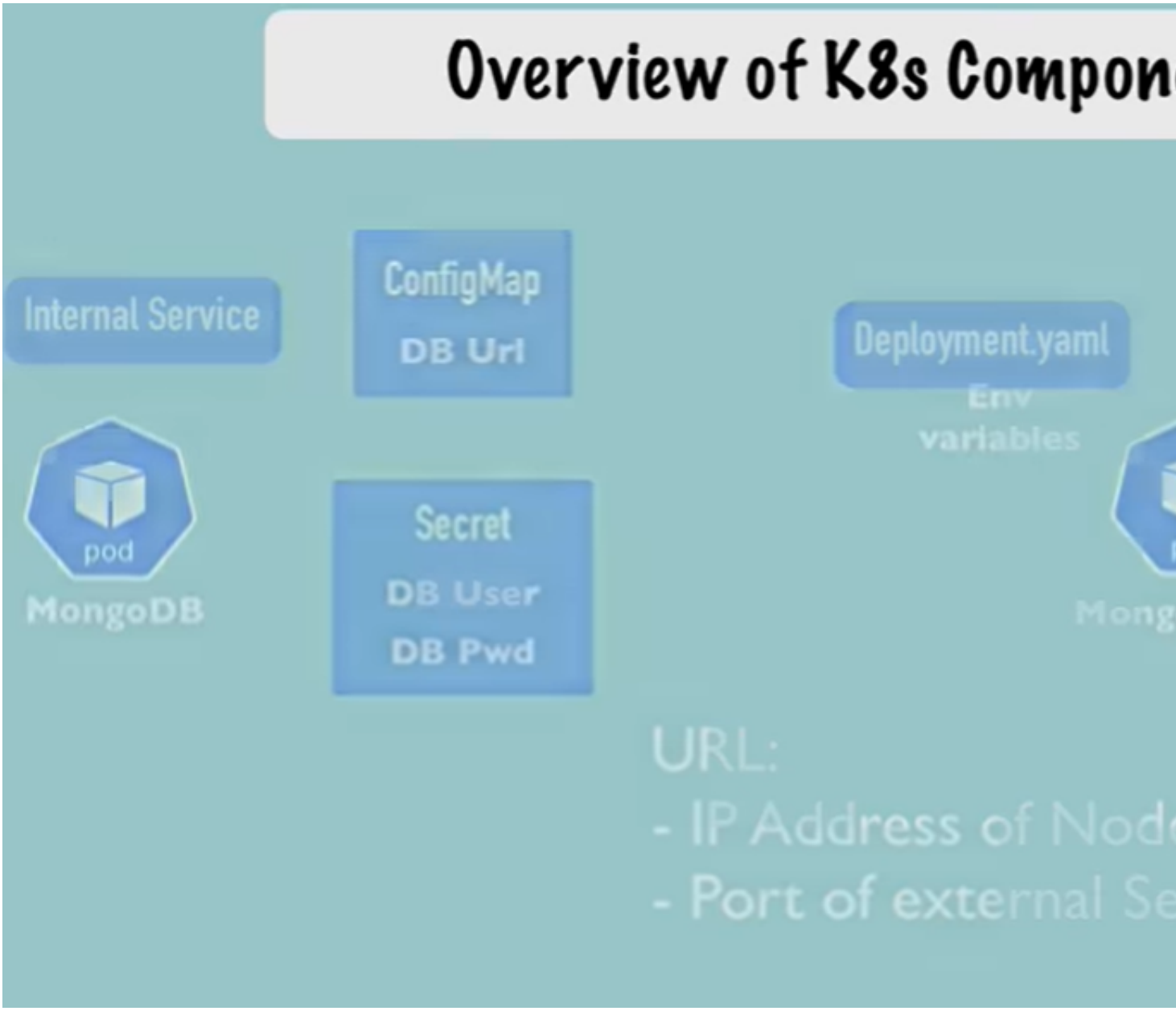

RESTARTS
IP
NOD
0
2m17s
172.17.0.7
min
0
2m17s
172.17.0.6

```
kubectl get deployment nginx-deployment -o yaml > ngin
```

Deployment gets Deleted if we delete config file…..

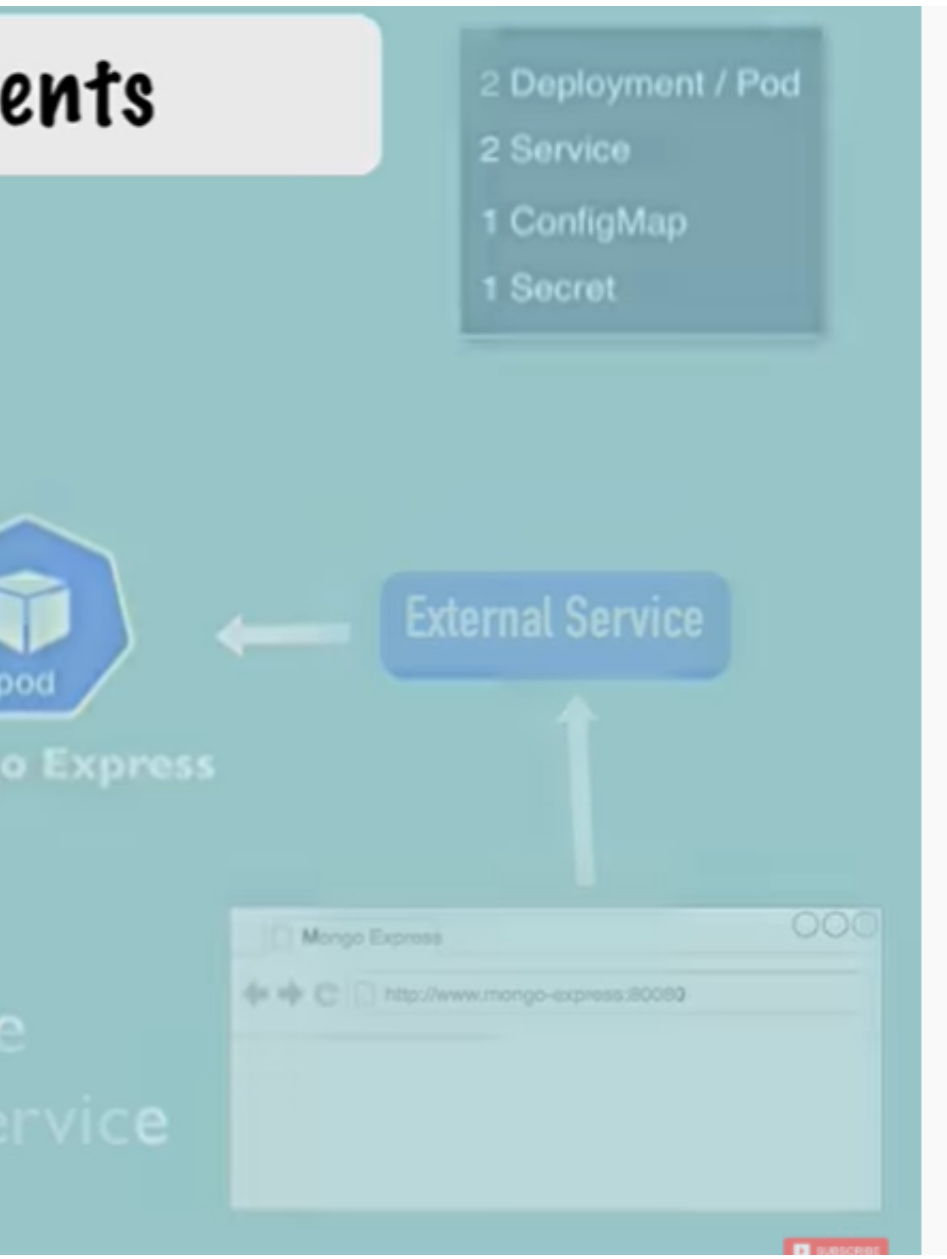
```
[Documents]$ kubectl delete -f nginx-deploymen
deployment.apps "nginx-deployment" deleted
[Documents]$ kubectl delete -f nginx-service.y
service "nginx-service" deleted
```

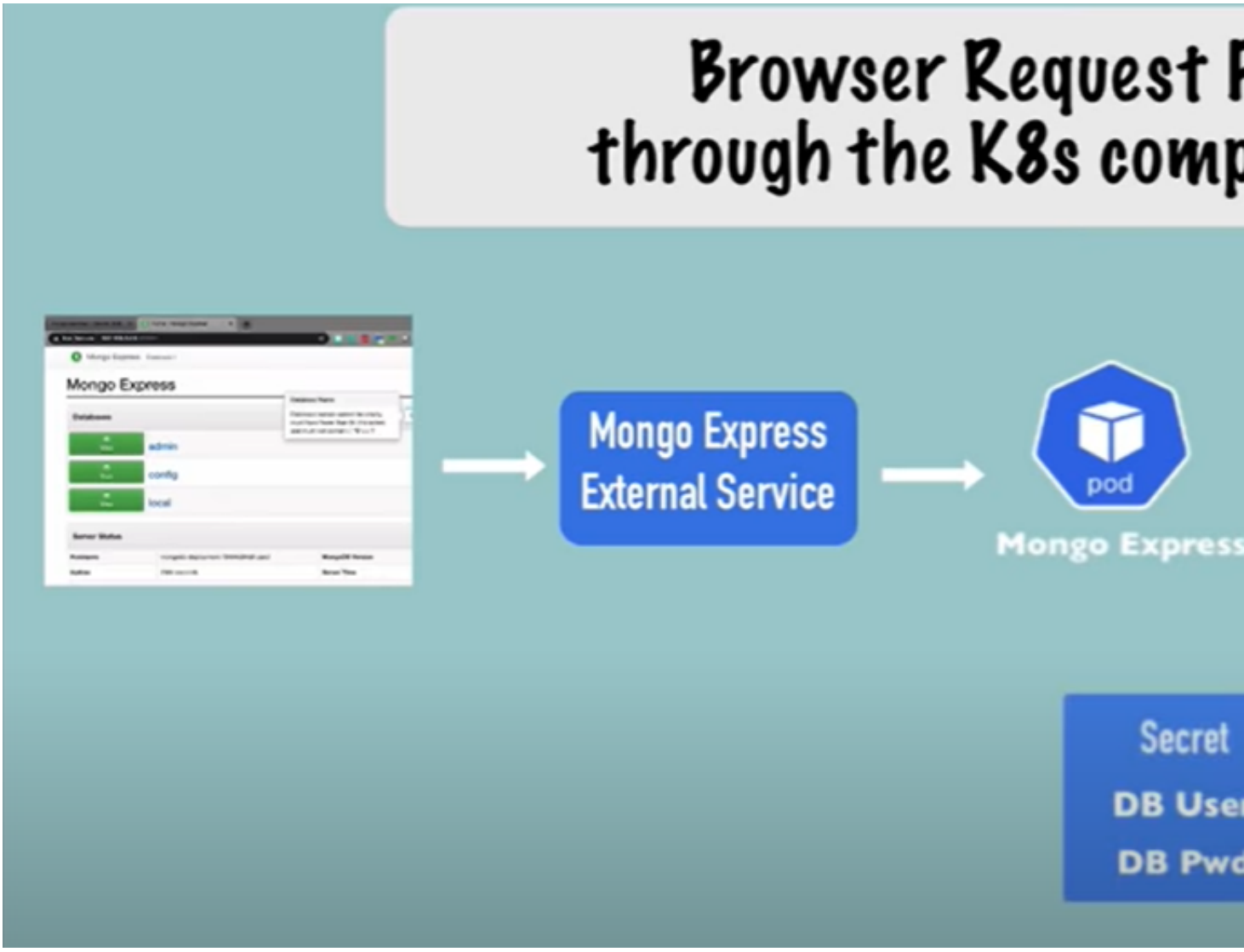
22)
Real time Set uP::



```
x-deployment-result
```

```
nt.yaml

yaml
```

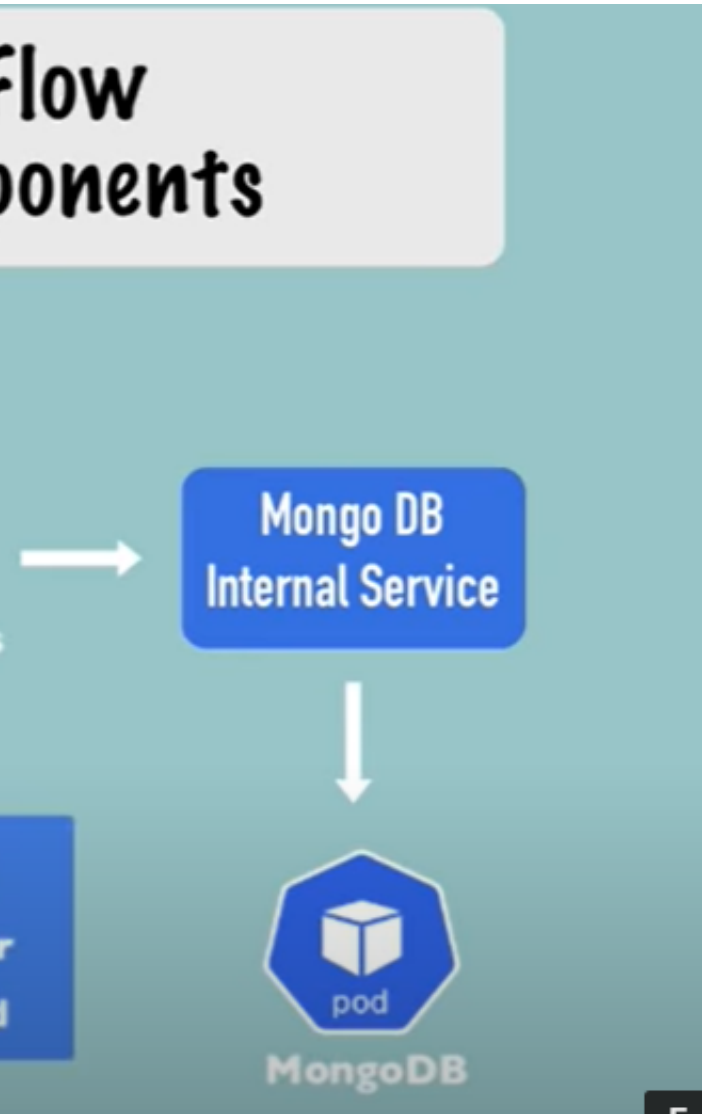




Deployment Set up using config .yaml file:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
```

Pod Blueprint:


Flow
ponents
Mongo DB
Internal Service
pod
MongoDB

```
    spec:
      containers:
      - name: mongodb
        image: mongo
```

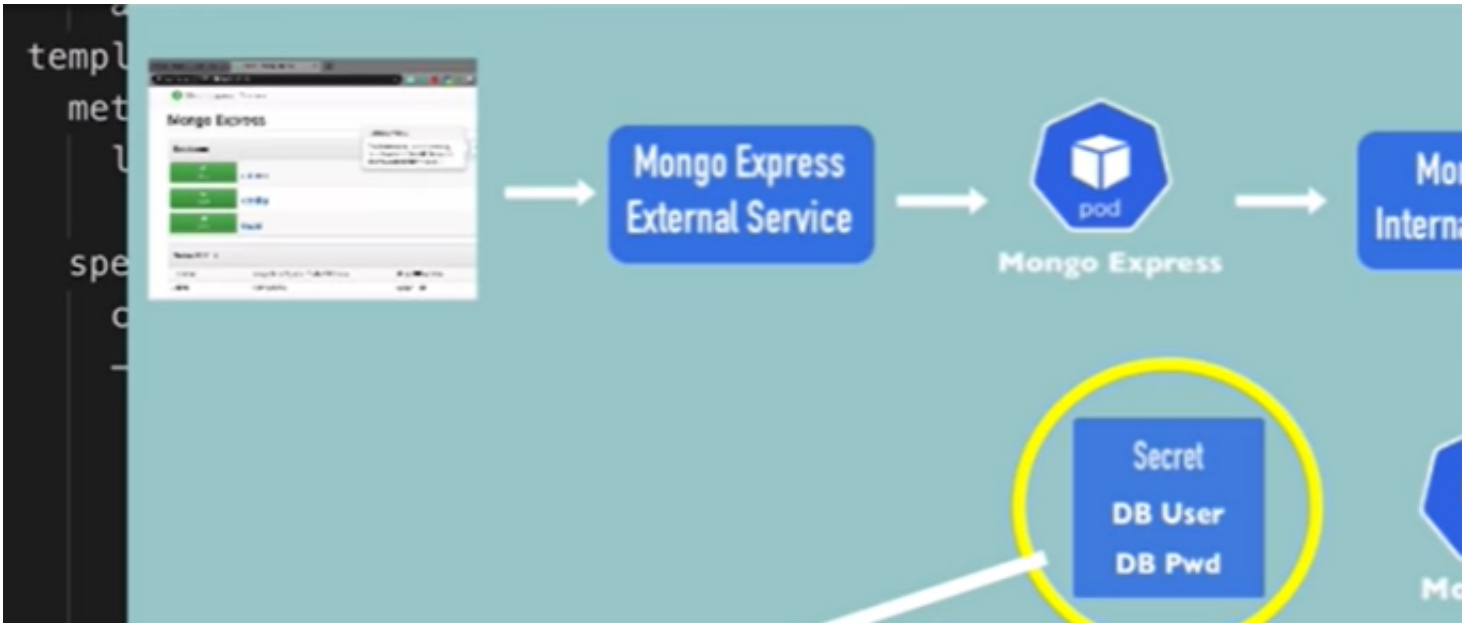
- name: mongodb
- image: mongo

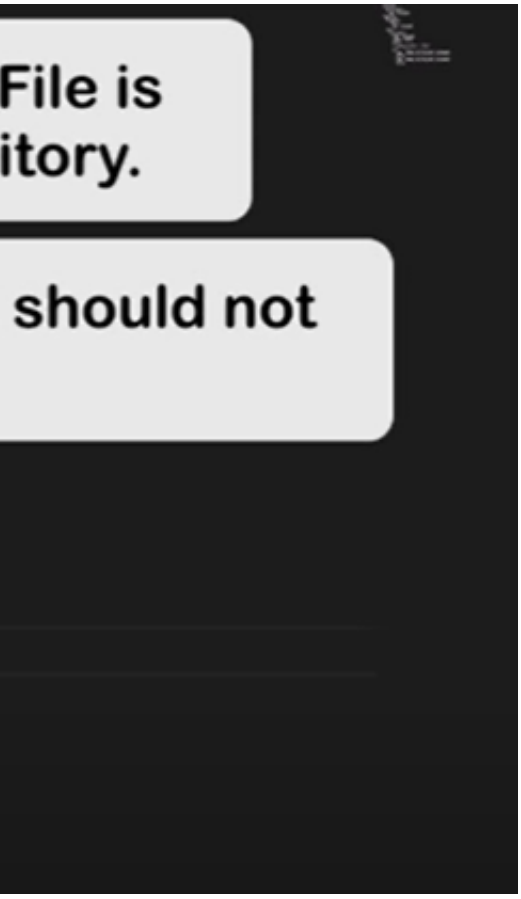
```
template:
  metadata:
    labels:
      app: mongodb
  spec:
    containers:
    - name: mongodb
      image: mongo
      ports:
      - containerPort: 27017
      env:
      - name: MONGO_INITDB_ROOT_USERNAME
        value:
      - name: MONGO_INITDB_ROOT_PASSWORD
        value:
```

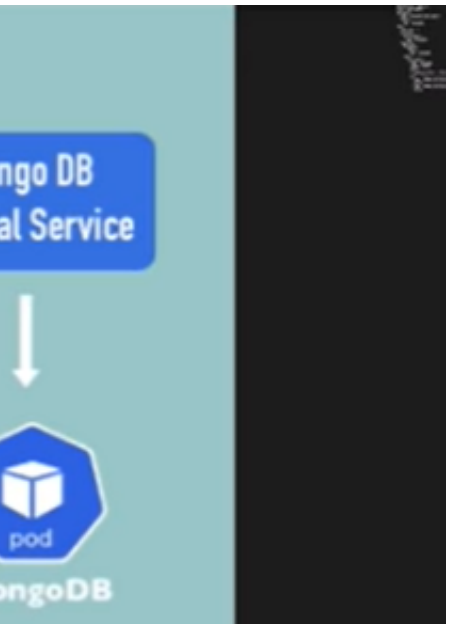
Deployment Config checked into repos

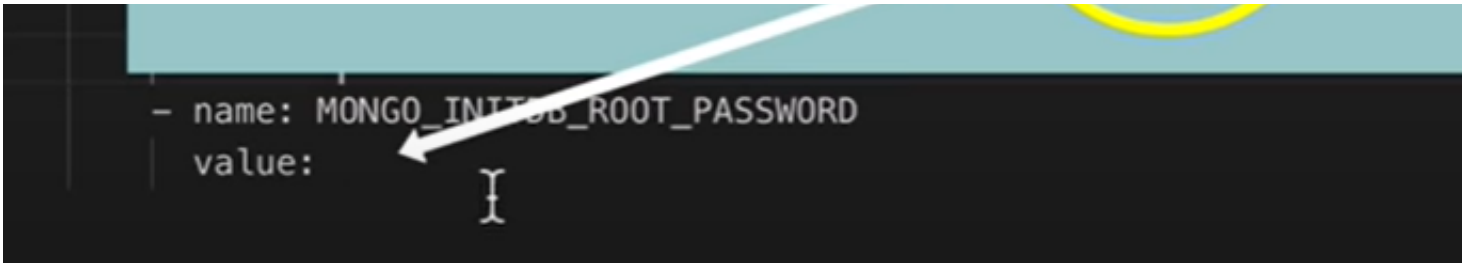
Username and Password go here!

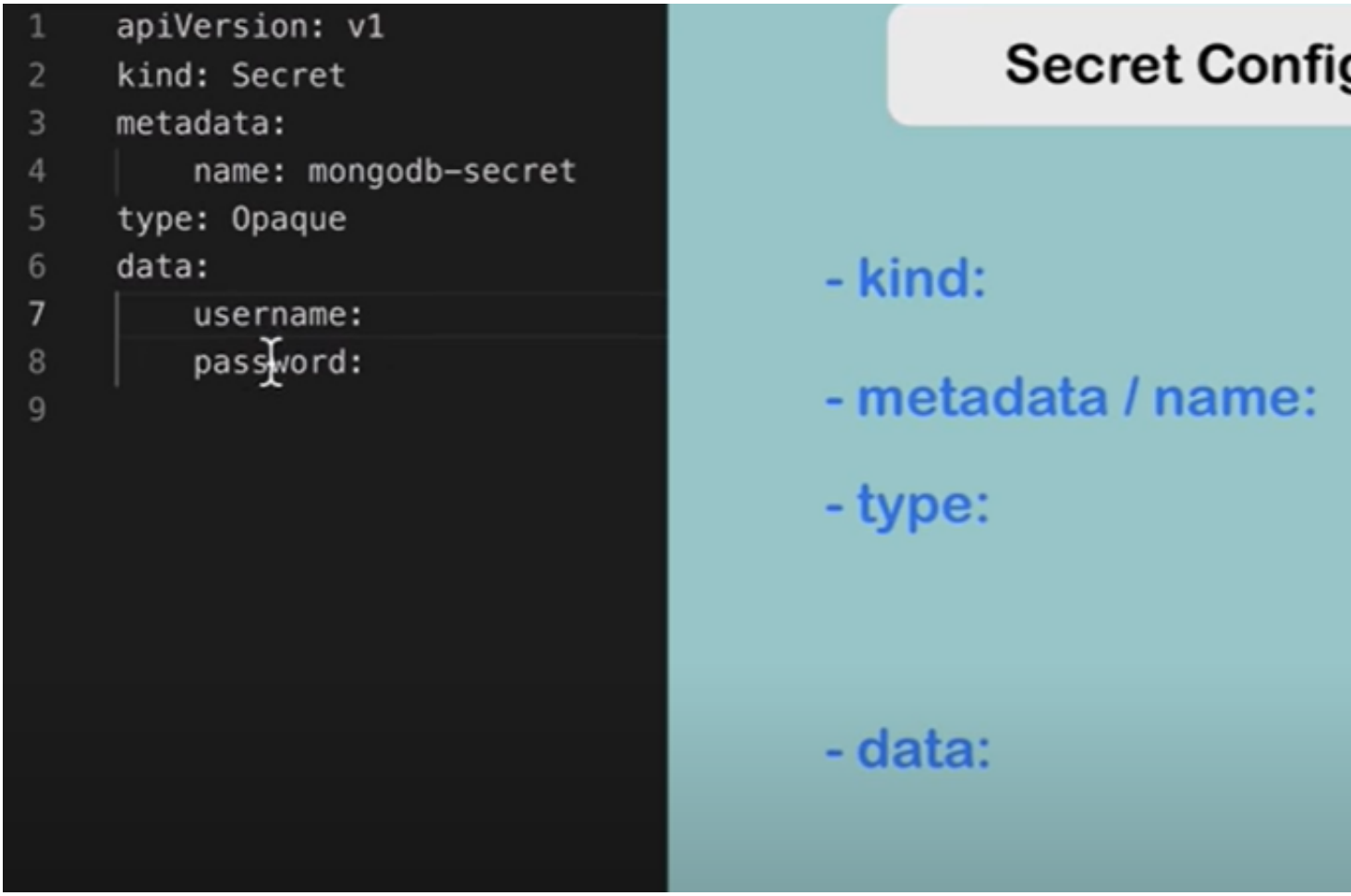
Secret Lives in K8s and not in Repository…



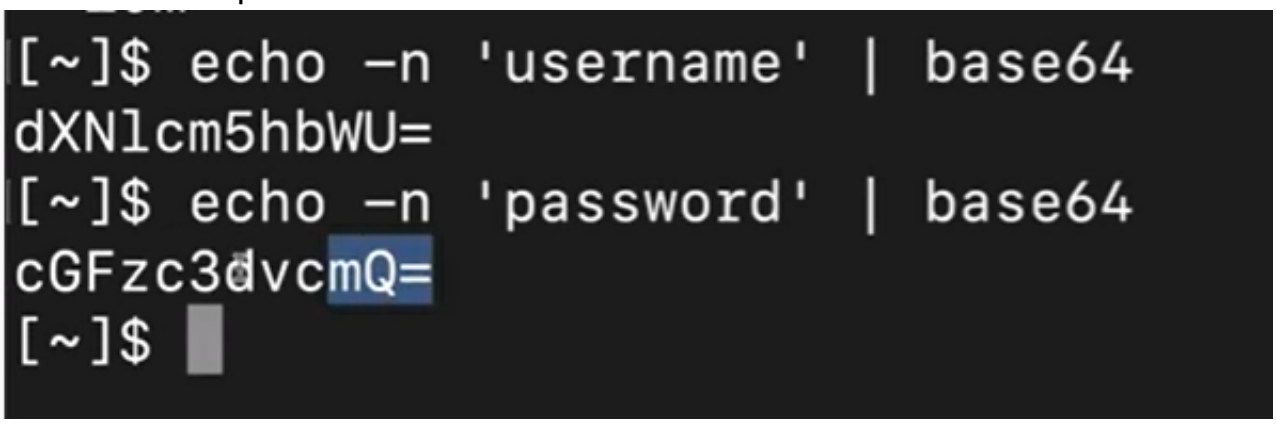

File is
itory.
should not

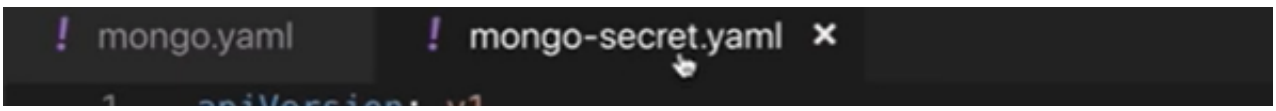

ngo DB
al Service
pod
ongoDB

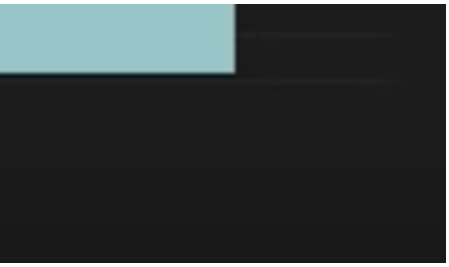


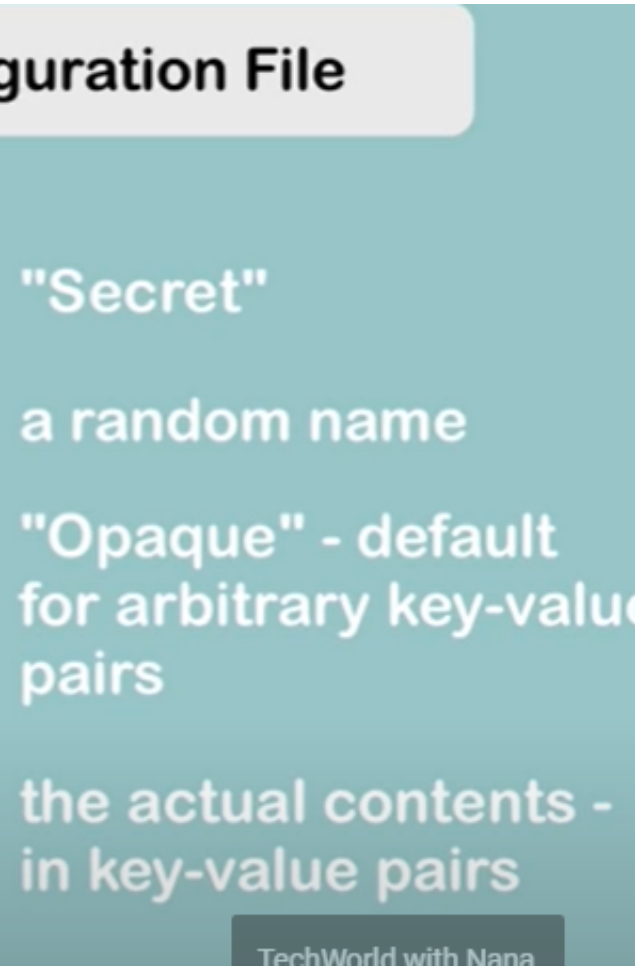


Convert plain text into base 64 encrypted text so ideally we should convert username and password in to base64.







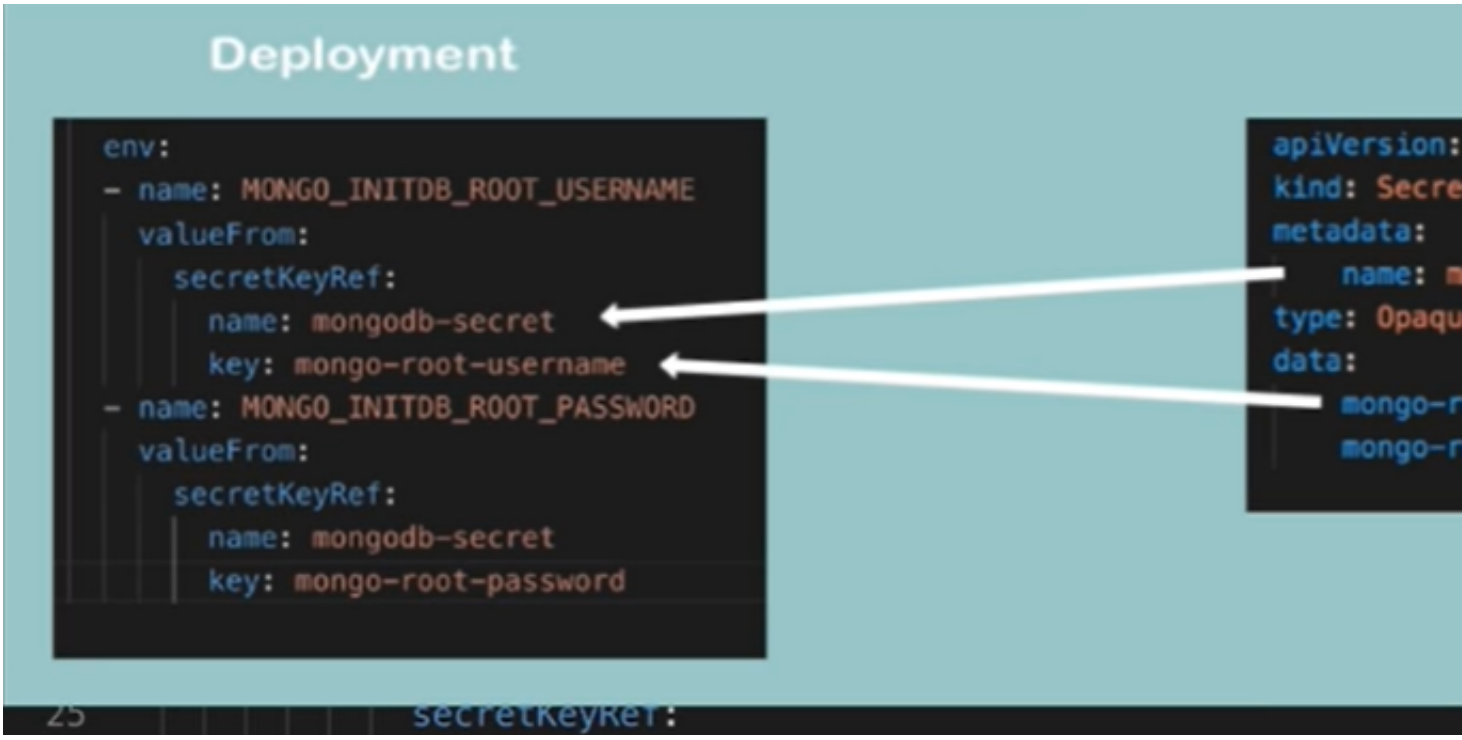

guration File
"Secret"
a random name
"Opaque" - default
for arbitrary key-valu
pairs
the actual contents -
in key-value pairs
TechWorld with Nana

```yaml
apiVersion: v1
kind: Secret
metadata:
    name: mongodb-secret
type: Opaque
data:
    mongo-root-username: dXNlcm5hbWU=
    mongo-root-password: cGFzc3dvcmQ=
```

Apply secret :

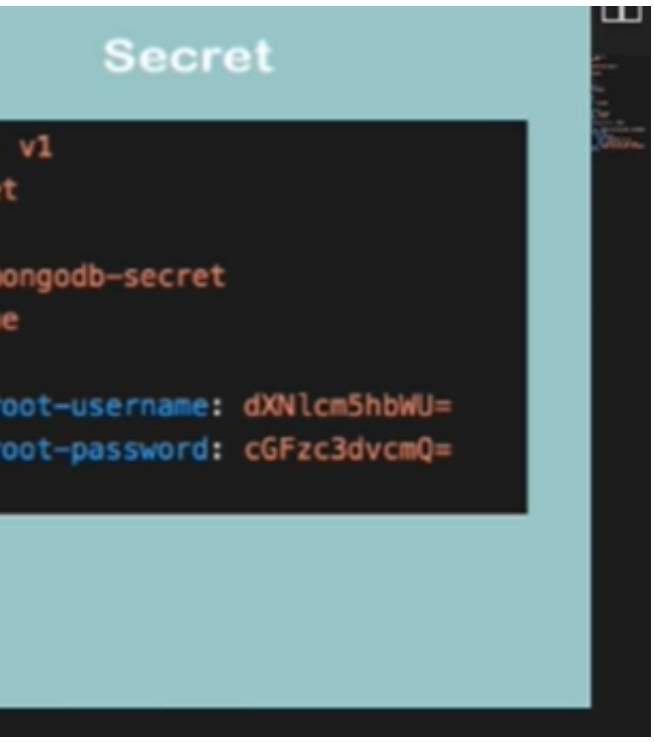
```
[~]$ cd k8s-configuration/
[k8s-configuration]$ ls
mongo-secret.yaml       mongo.yaml
[k8s-configuration]$ kubectl apply -f mong
secret/mongodb-secret created
```

```
[k8s-configuration]$ kubectl get secret
NAME                  TYPE
default-token-4clzs   kubernetes.io/service-
mongodb-secret        Opaque
```



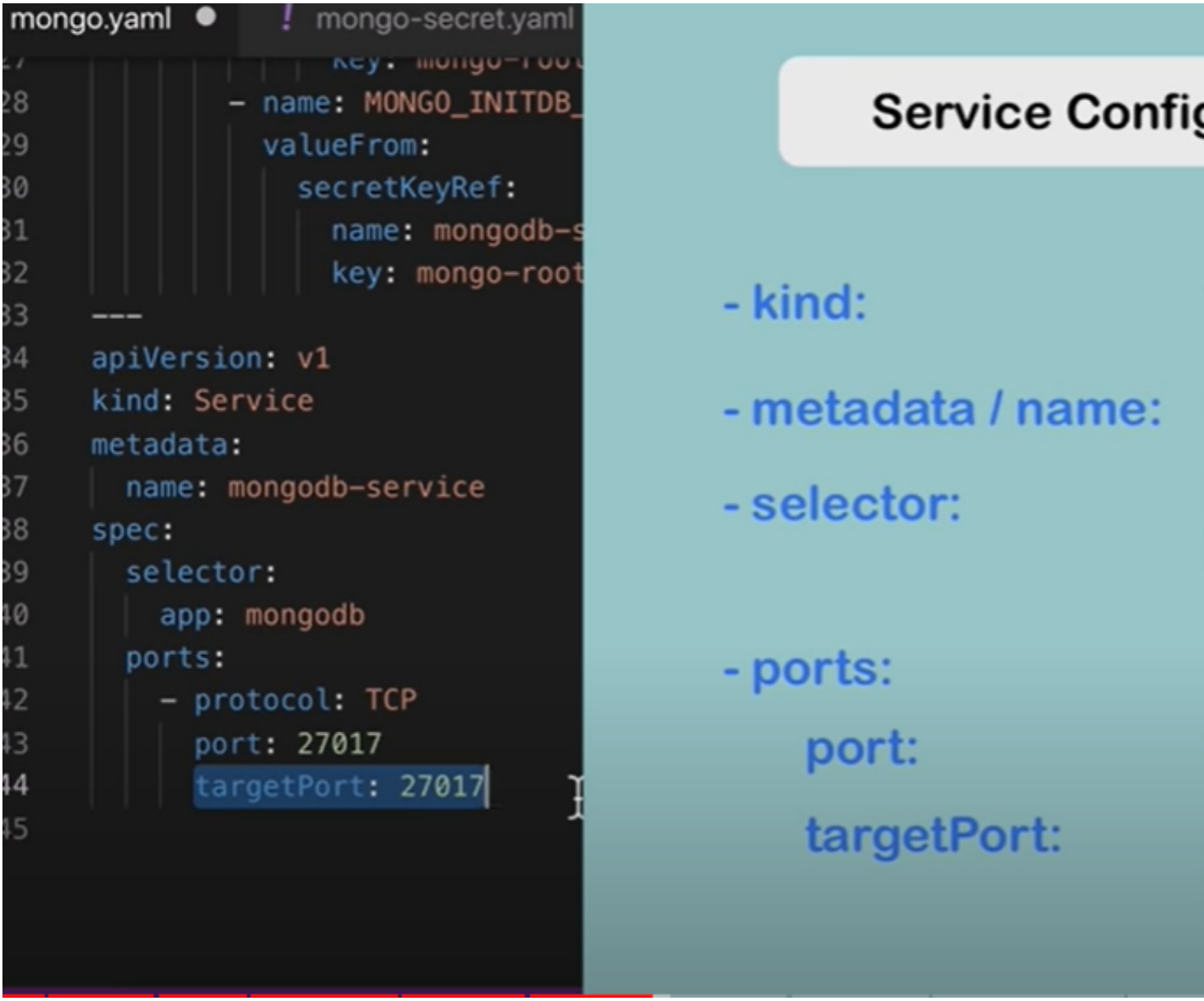
```
go-secret.yaml
```

```
                    DATA
-account-token      3
                    2
```



```
26                  name: mongodb-secret
27                  key: mongo-root-username
28            - name: MONGO_INITDB_ROOT_PASSWORD
29              value:
```

23)Lets Create Internal Service related Config in same .yaml file :
--- ===> 3 dot is seperator of another .yaml file in to same deployment.




```
[k8s-configuration]$ kubectl get service
NAME                TYPE        CLUSTER-IP      EXTE
kubernetes          ClusterIP   10.96.0.1       <nor
mongodb-service     ClusterIP   10.96.86.105    <nor
```

guration File

"Service"

a random name

to connect to Pod
through label

Service port

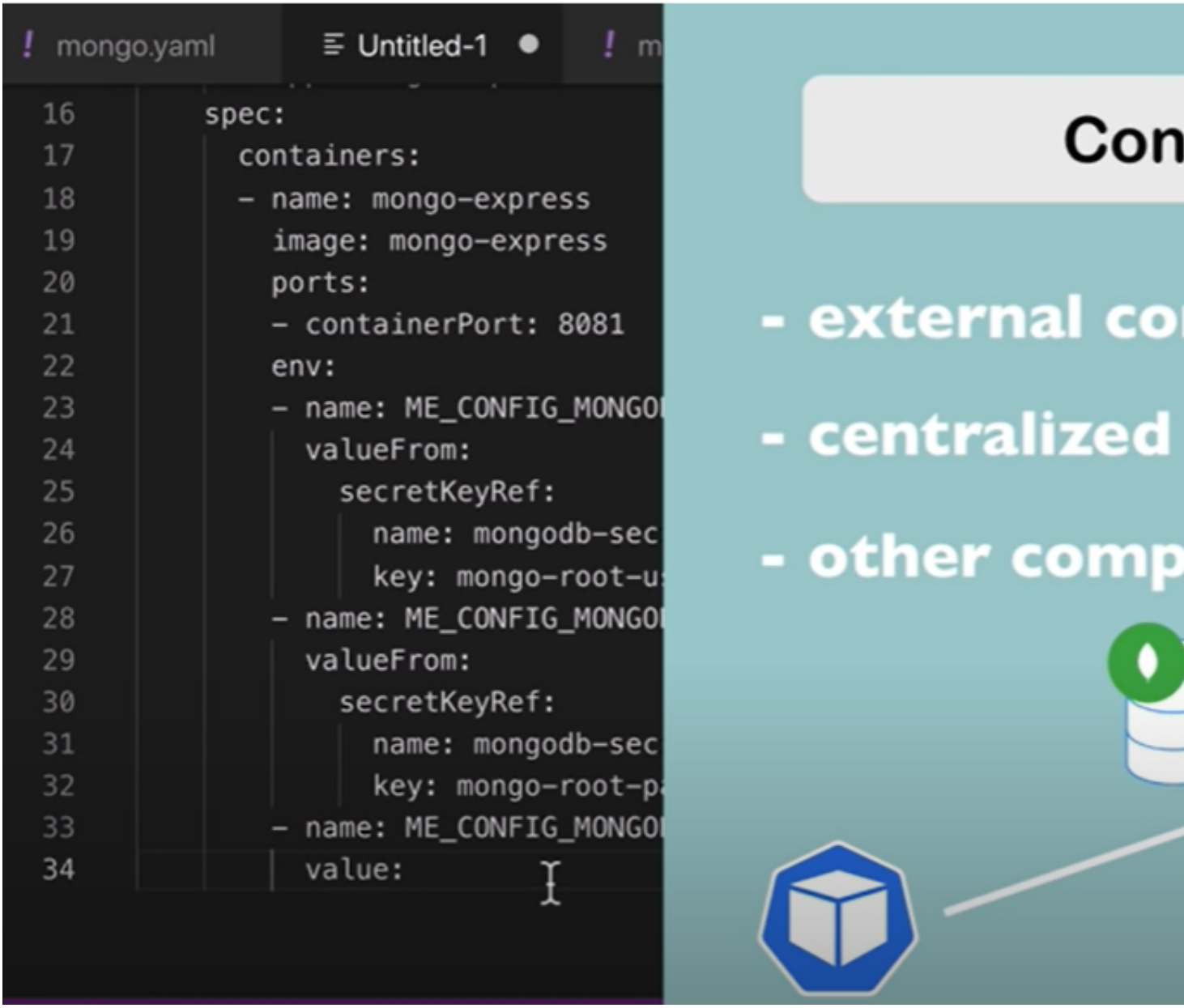
containerPort of
Deployment

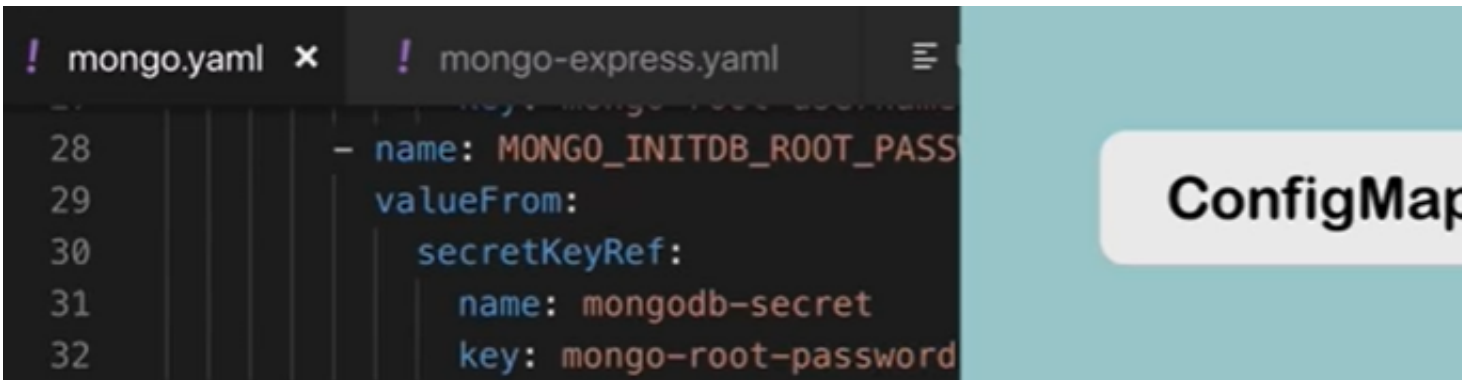
```
ERNAL-IP    PORT(S)      AGE
ne>         443/TCP      36m
ne>         27017/TCP    51s
```

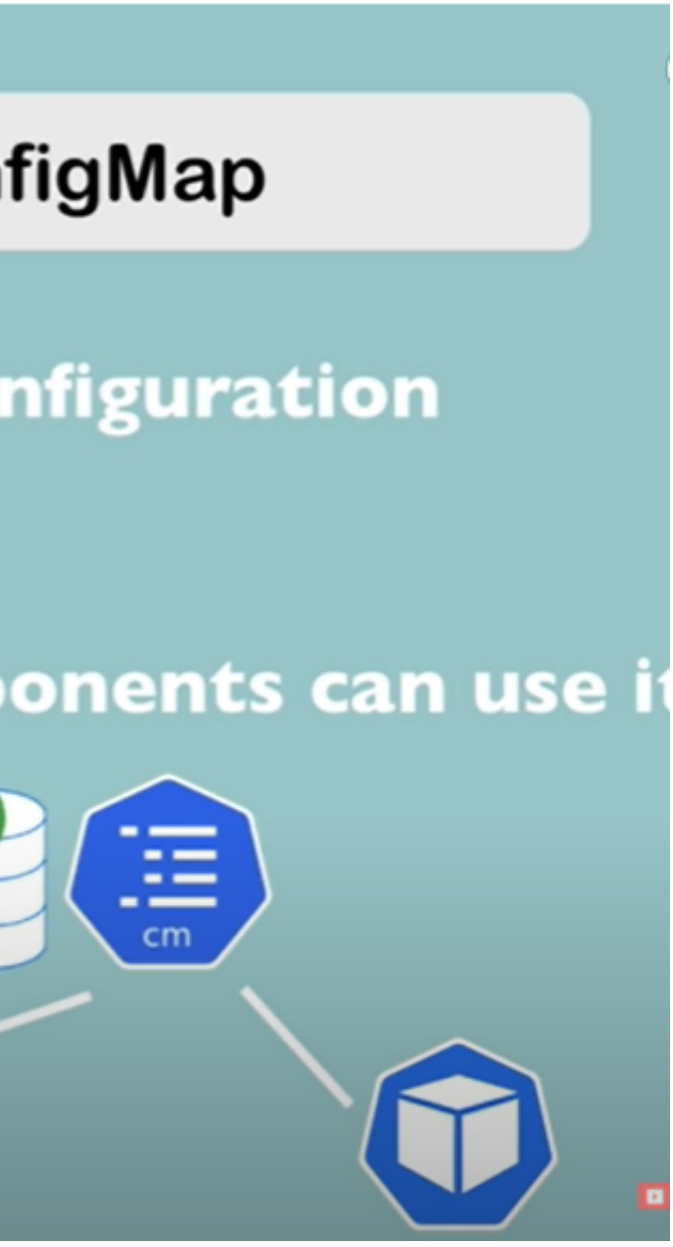
Get all components and filter them using Mongo..

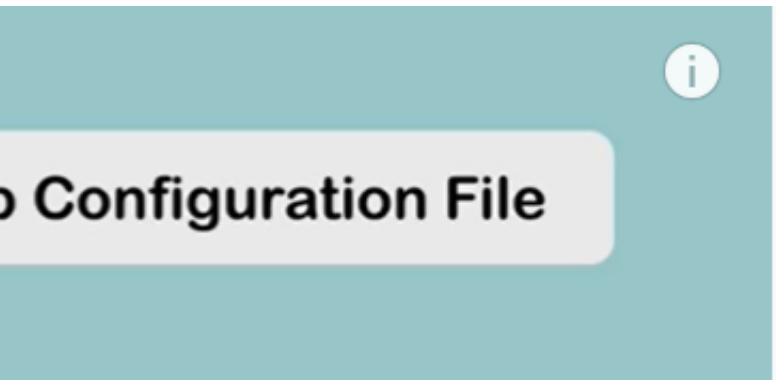
24) Now create the mongo express server which can be accessed from external service…

External detaisl we will put in config-map..






figMap
nfiguration
onents can use it
cm


Configuration File

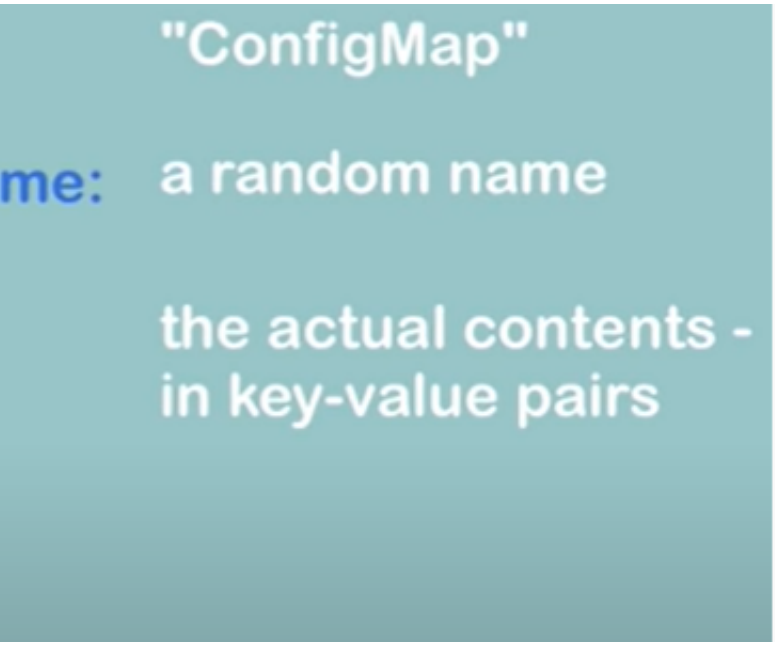
```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

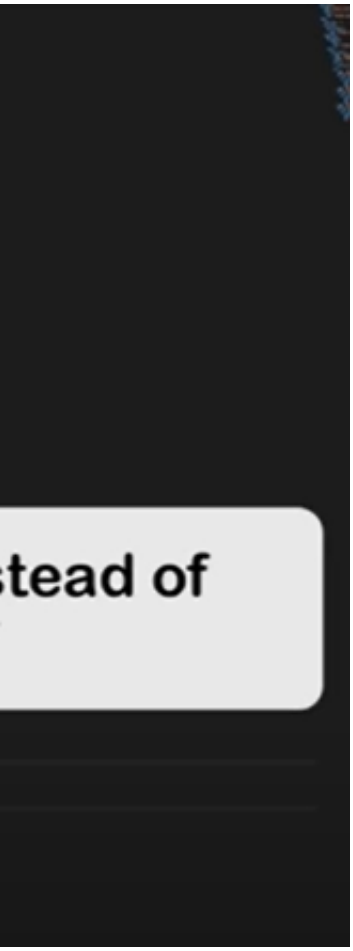
- kind:
- metadata / na
- data:

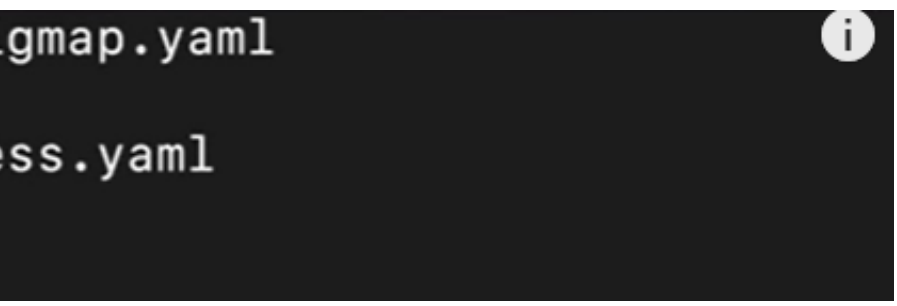
```yaml
ports:
- containerPort: 8081
env:
- name: ME_CONFIG_MONGODB_ADMINUSERNAME
  valueFrom:
    secretKeyRef:
      name: mongodb-secret
      key: mongo-root-username
- name: ME_CONFIG_MONGODB_ADMINPASSWORD
  valueFrom:
    secretKeyRef:
      name: mongodb-secret
      key: mongo-root-password
- name: ME_CONFIG_MONGODB_SERVER
  valueFrom:
    configMapKeyRef:
      name: mongodb-secret
      key: mongo-root-password
```

configMapKeyRef ins
secretKeyRef

```
[k8s-configuration]$ kubectl apply -f mongo-confi
configmap/mongodb-configmap created
[k8s-configuration]$ kubectl apply -f mongo-expre
deployment.apps/mongo-express created
[k8s-configuration]$ kubectl get pod
```


"ConfigMap"
me: a random name
the actual contents -
in key-value pairs

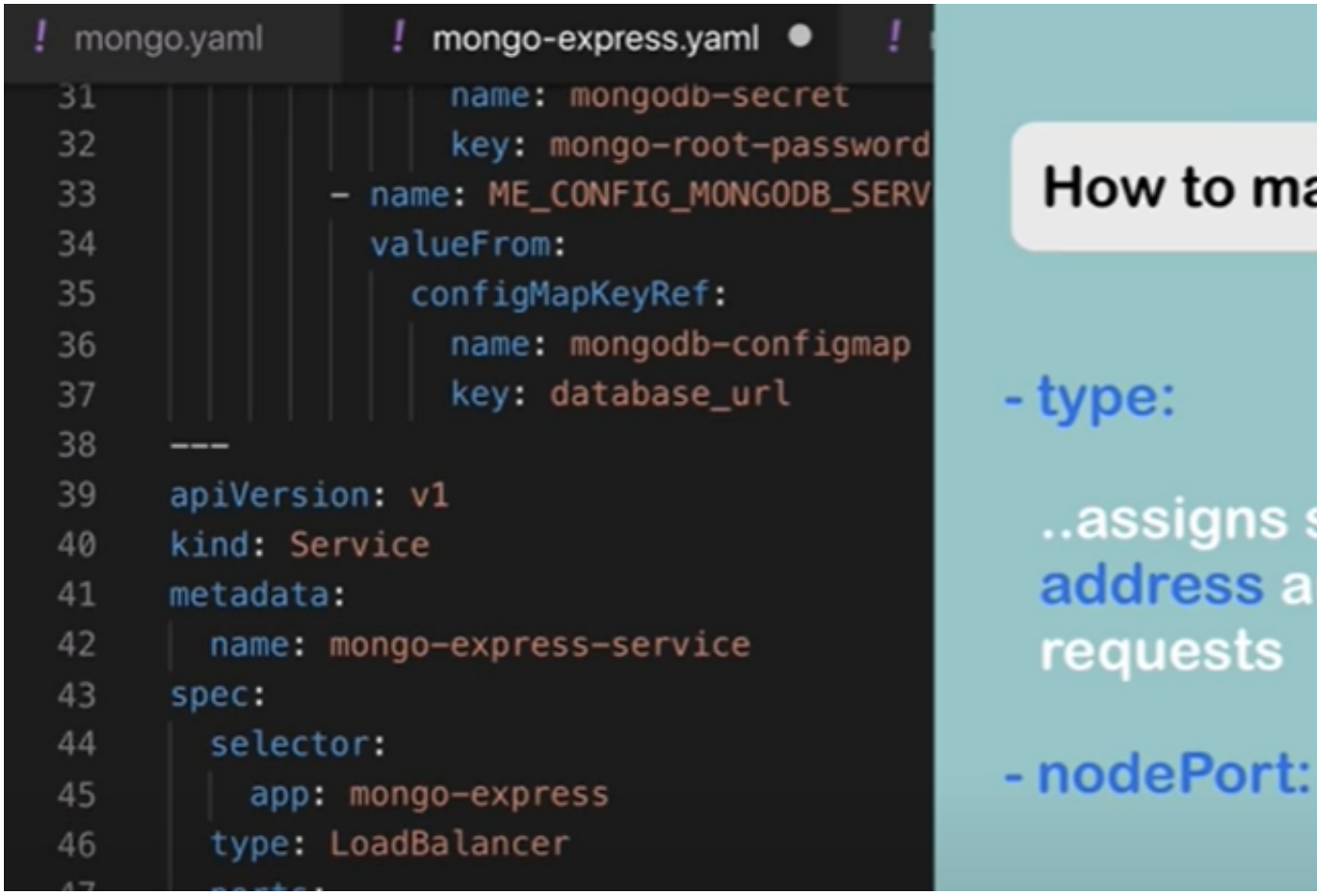

stead of


gmap.yaml
ss.yaml

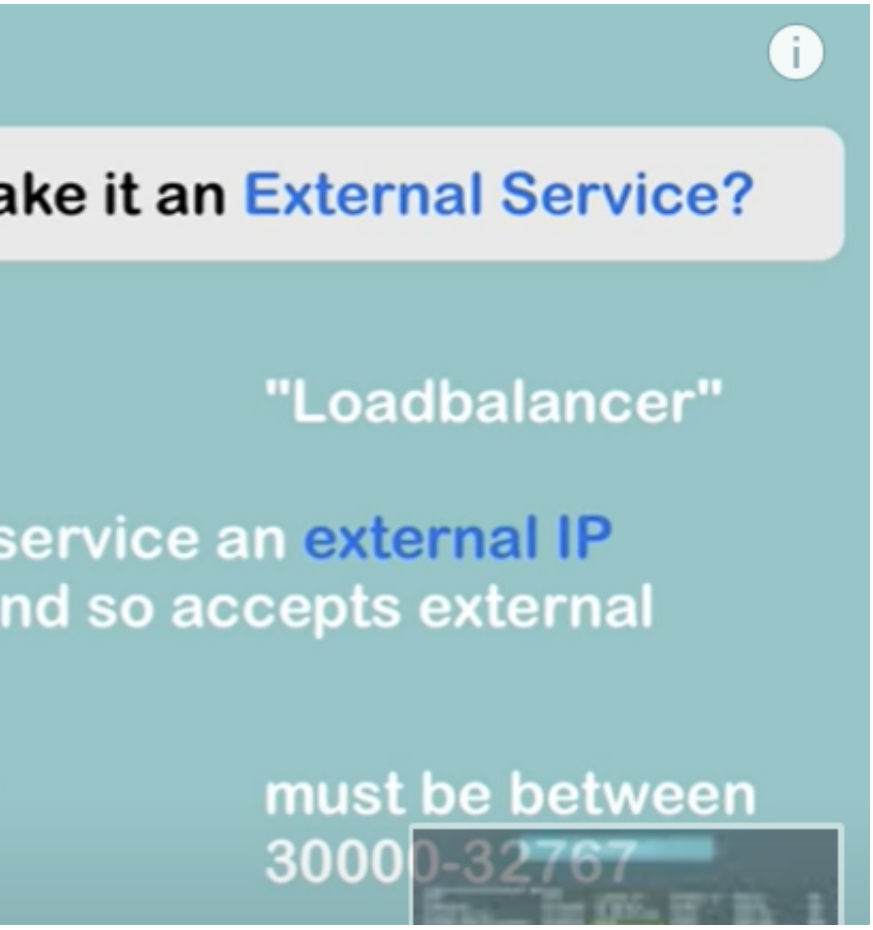
```
NAME                                  READY   STA
E
mongo-express-797845bd97-p9grr        0/1     Con
mongodb-deployment-78444d94d6-zsrcl   1/1     Run
m
[k8s-configuration]$ kubectl get pod
NAME                                  READY   STA
E
mongo-express-797845bd97-p9grr        0/1     Con
s
mongodb-deployment-78444d94d6-zsrcl   1/1     Run
m
[k8s-configuration]$
```

25) Now Create external Service which could access Mongo Express.

Node Port :: Used for external Service.



```
TUS                  RESTARTS   AG

tainerCreating   0          5s
ning             0          23


TUS                  RESTARTS   AG

tainerCreating   0          19

ning             0          24
```

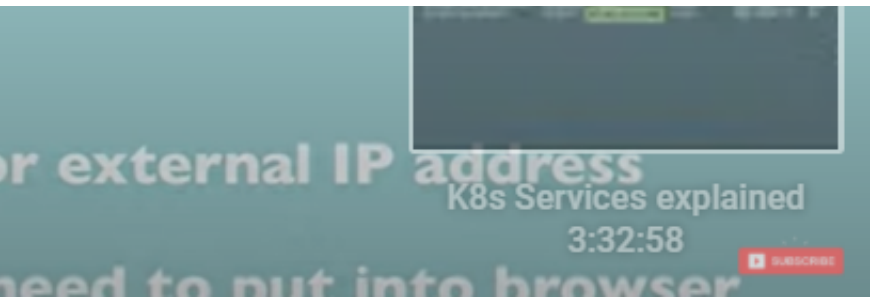


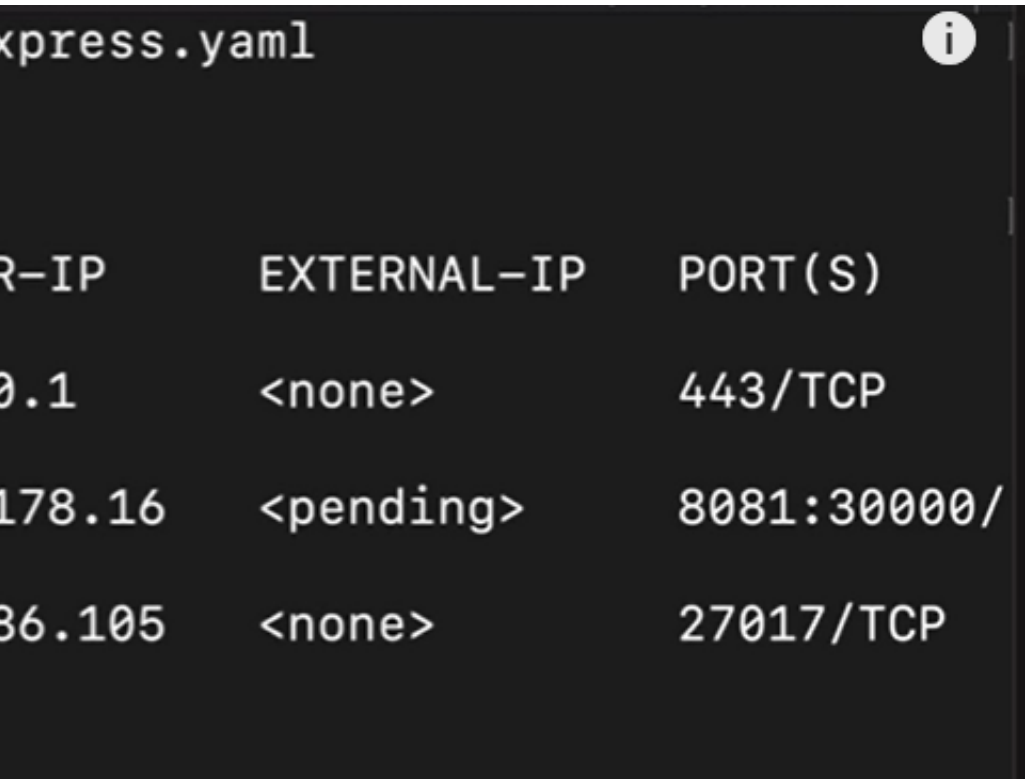
```
48      - protocol: TCP
49        port: 8081
50        targetPort: 8081
51        nodePort: 30
```

```
---
apiVersion: v1
kind: Service
metadata:
  name: mongo-express-service
spec:
  selector:
    app: mongo-express
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 8081
      targetPort: 8081
      nodePort: 30000
```

===> External Service Node Creation.

```
[k8s-configuration]$ kubectl apply -f mongo-ex
deployment.apps/mongo-express unchanged
service/mongo-express-service created
[k8s-configuration]$ kubectl get service
NAME                    TYPE           CLUSTER
      AGE
kubernetes              ClusterIP      10.96.0
      62m
mongo-express-service   LoadBalancer   10.96.1
TCP   6s
mongodb-service         ClusterIP      10.96.8
      26m
[k8s-configuration]$
```

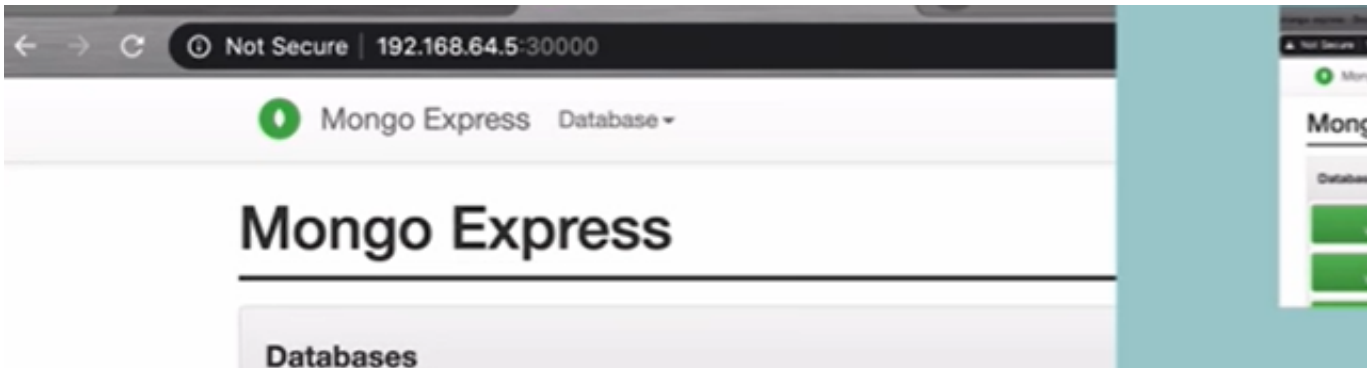

r external IP address
K8s Services explained
3:32:58


xpress.yaml
R-IP EXTERNAL-IP PORT(S)
0.1 <none> 443/TCP
178.16 <pending> 8081:30000/
86.105 <none> 27017/TCP

```
[k8s-configuration]$ kubectl get service
NAME                      TYPE           CLUSTE
      AGE
kubernetes                ClusterIP      10.96.
      67m
mongo-express-service     LoadBalancer   10.96.
TCP   5m7s
mongodb-service           ClusterIP      10.96.
      32m
[k8s-configuration]$ minikube service mongo-e
```

```
      67m
mongo-express-service     LoadBalancer   10.96.
TCP   5m7s
mongodb-service           ClusterIP      10.96.
      32m
[k8s-configuration]$ minikube service mongo-e
|-----------|-------------------------|---------
| NAMESPACE |          NAME           | TARGET
|-----------|-------------------------|---------
| default   | mongo-express-service   |
|-----------|-------------------------|---------
🎉  Opening service default/mongo-express-ser
[k8s-configuration]$
```

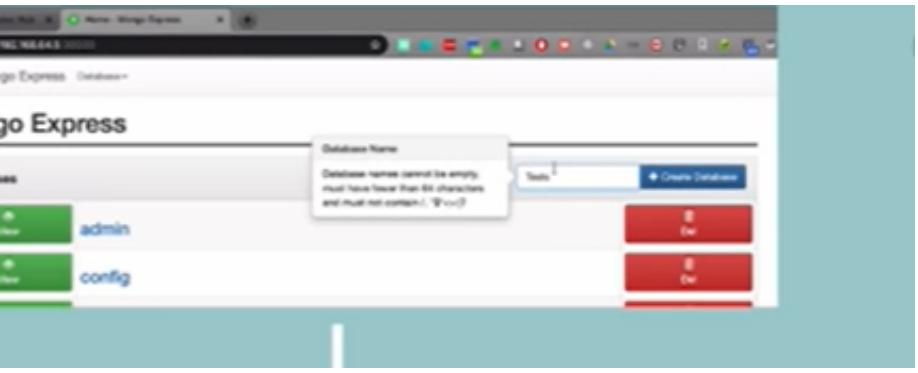
Full Working:

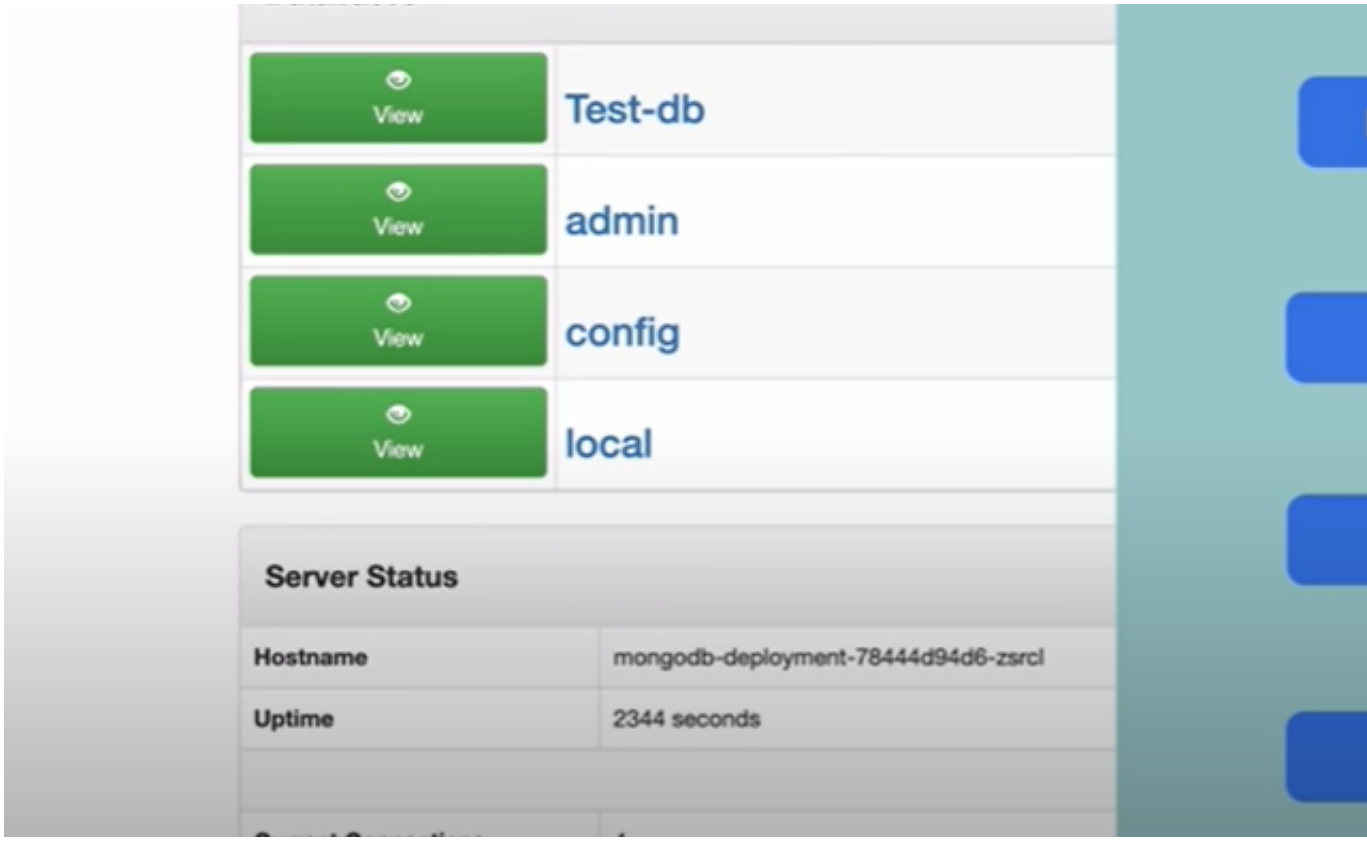


```
R-IP      EXTERNAL-IP    PORT(S)
0.1       <none>         443/TCP
178.16    <pending>      8081:30000/
86.105    <none>         27017/TCP

xpress-service
```

```
178.16    <pending>      8081:30000/
86.105    <none>         27017/TCP

xpress-se
------|-----------------------------|
PORT  |             URL             |
------|-----------------------------|
      | http://192.168.64.5:30000   |
------|-----------------------------|
vice in default browser...
```

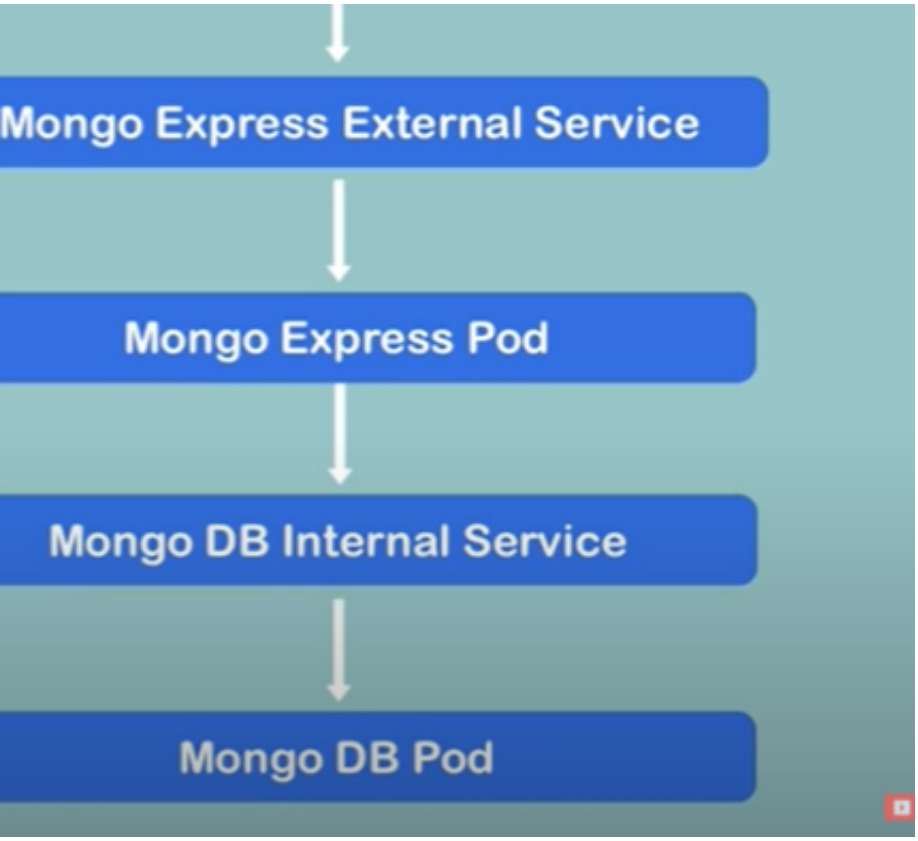
nodePort





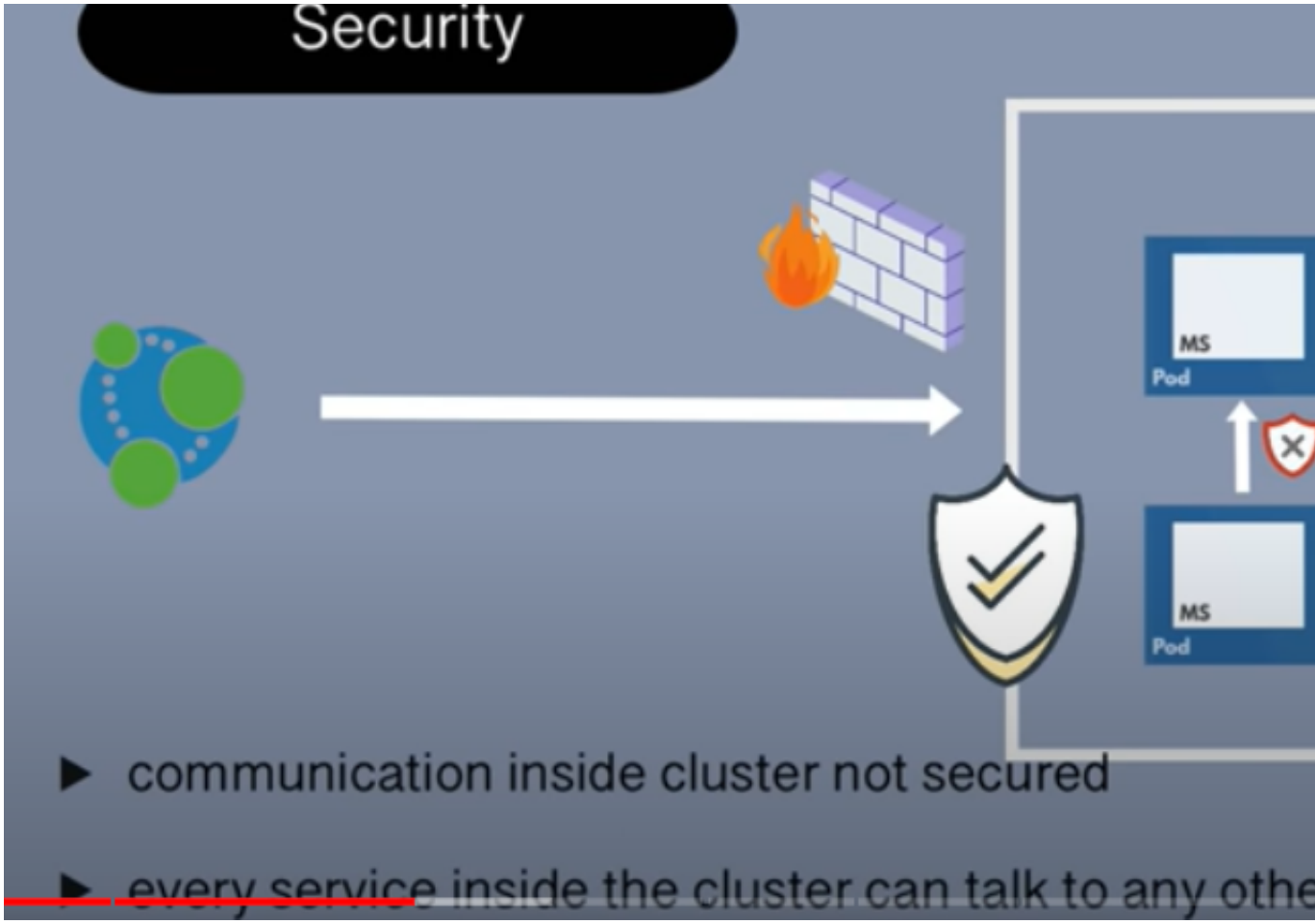
26)

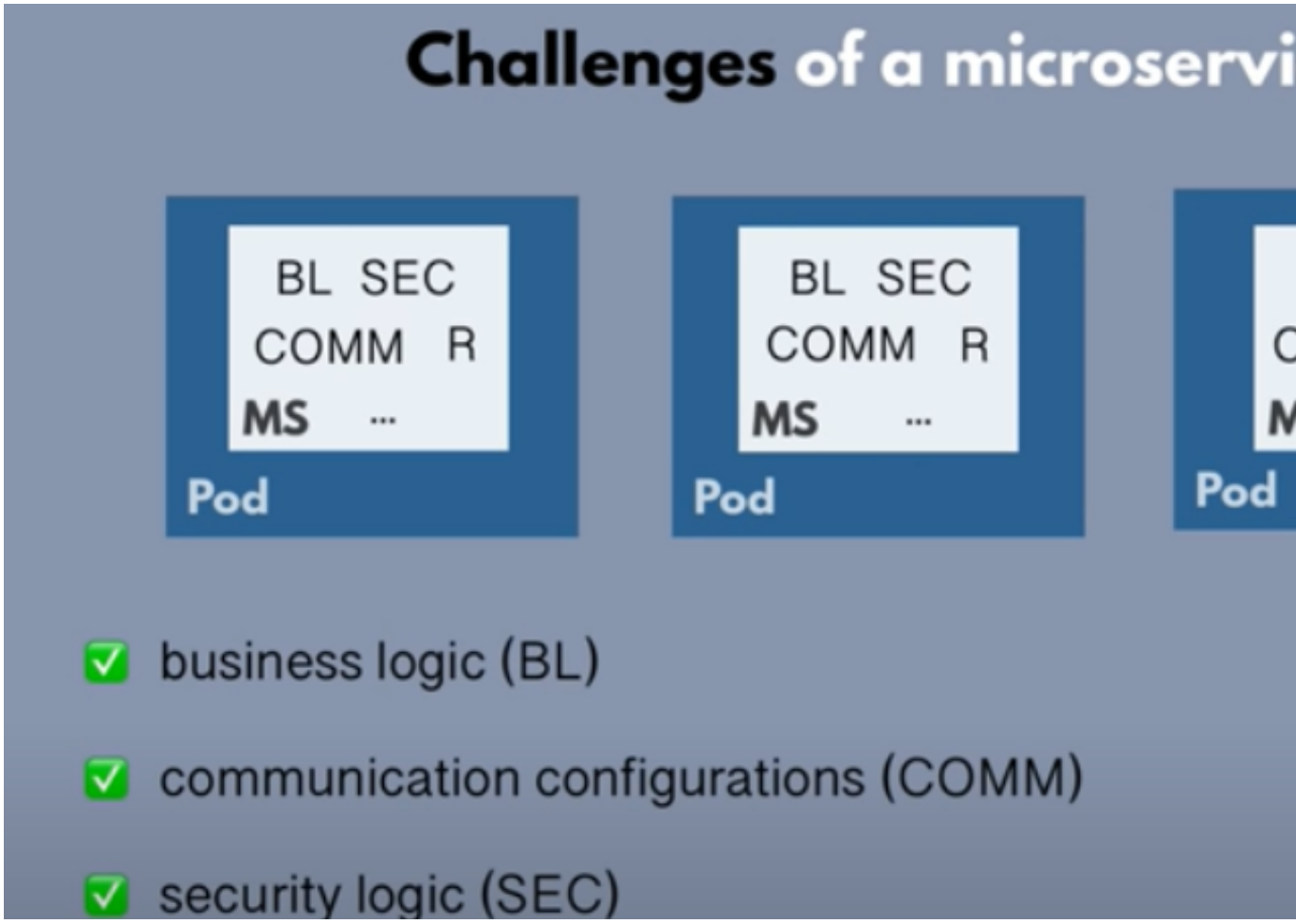
========== Istio and Service Mesh==========

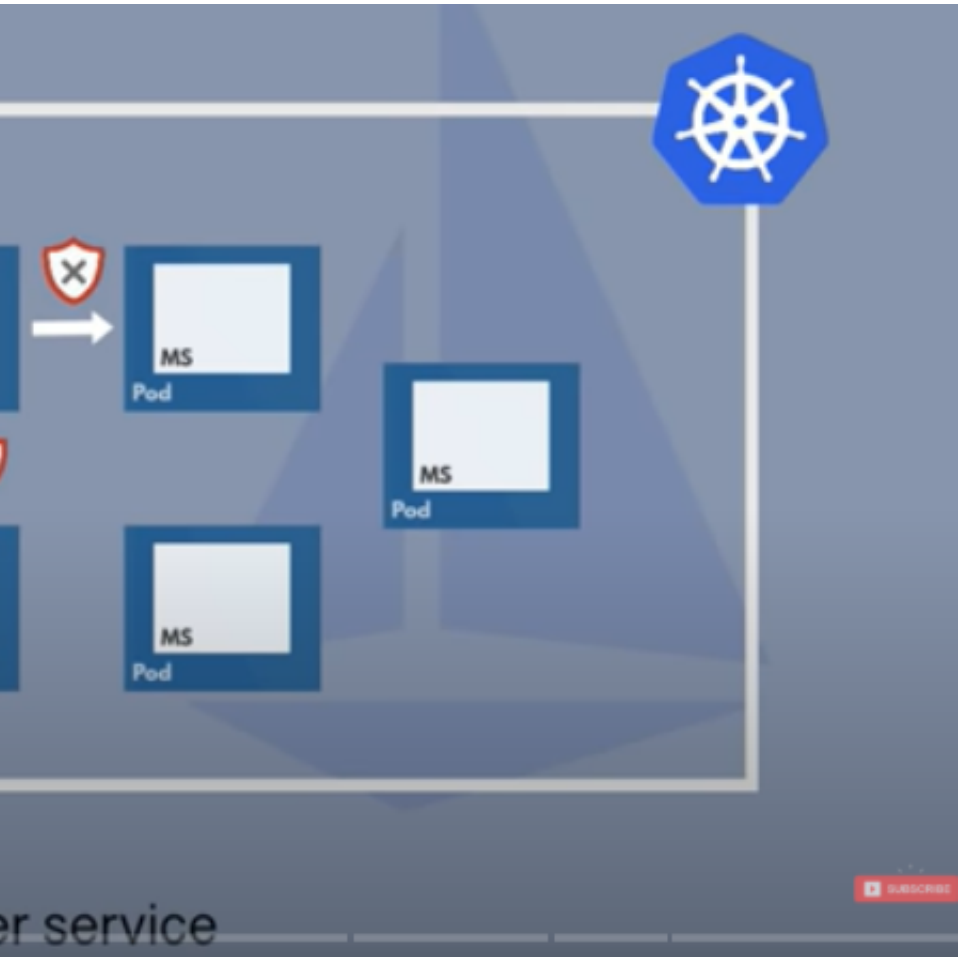

Mongo Express External Service
Mongo Express Pod
Mongo DB Internal Service
Mongo DB Pod

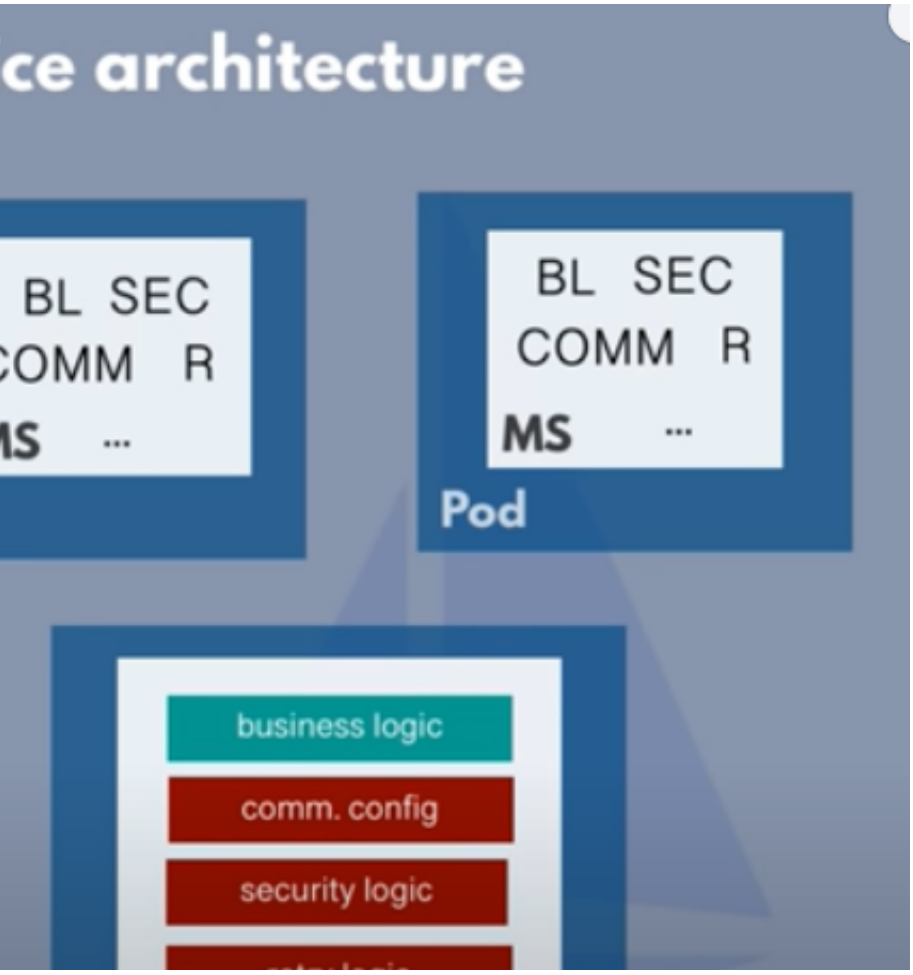

ervice Mesh?
between microservices


ce architecture


Security
MS
Pod
MS
Pod
communication inside cluster not secured
every service inside the cluster can talk to any othe

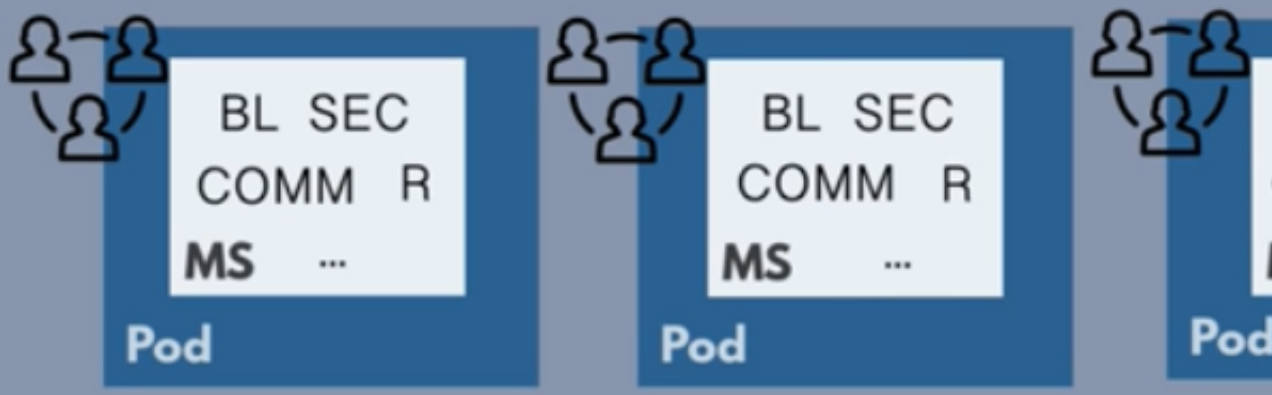

Challenges of a microservi
BL SEC
COMM R
MS ...
Pod
BL SEC
COMM R
MS ...
Pod
Pod
business logic (BL)
communication configurations (COMM)
security logic (SEC)


MS
Pod
MS
Pod
MS
Pod
er service


ce architecture
BL SEC
COMM R
MS ...
BL SEC
COMM R
MS ...
Pod
business logic
comm. config
security logic

✅ retry logic (R)

Challenges of a microserv



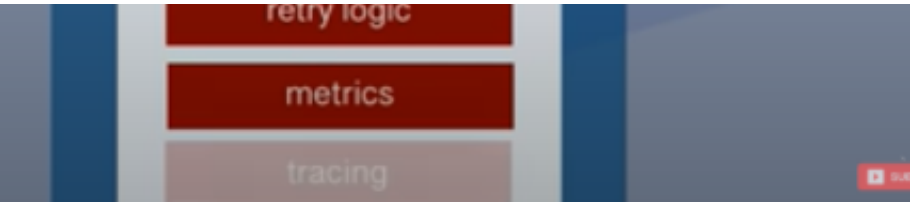
❌ These **NON business logic** must be added to **each application**

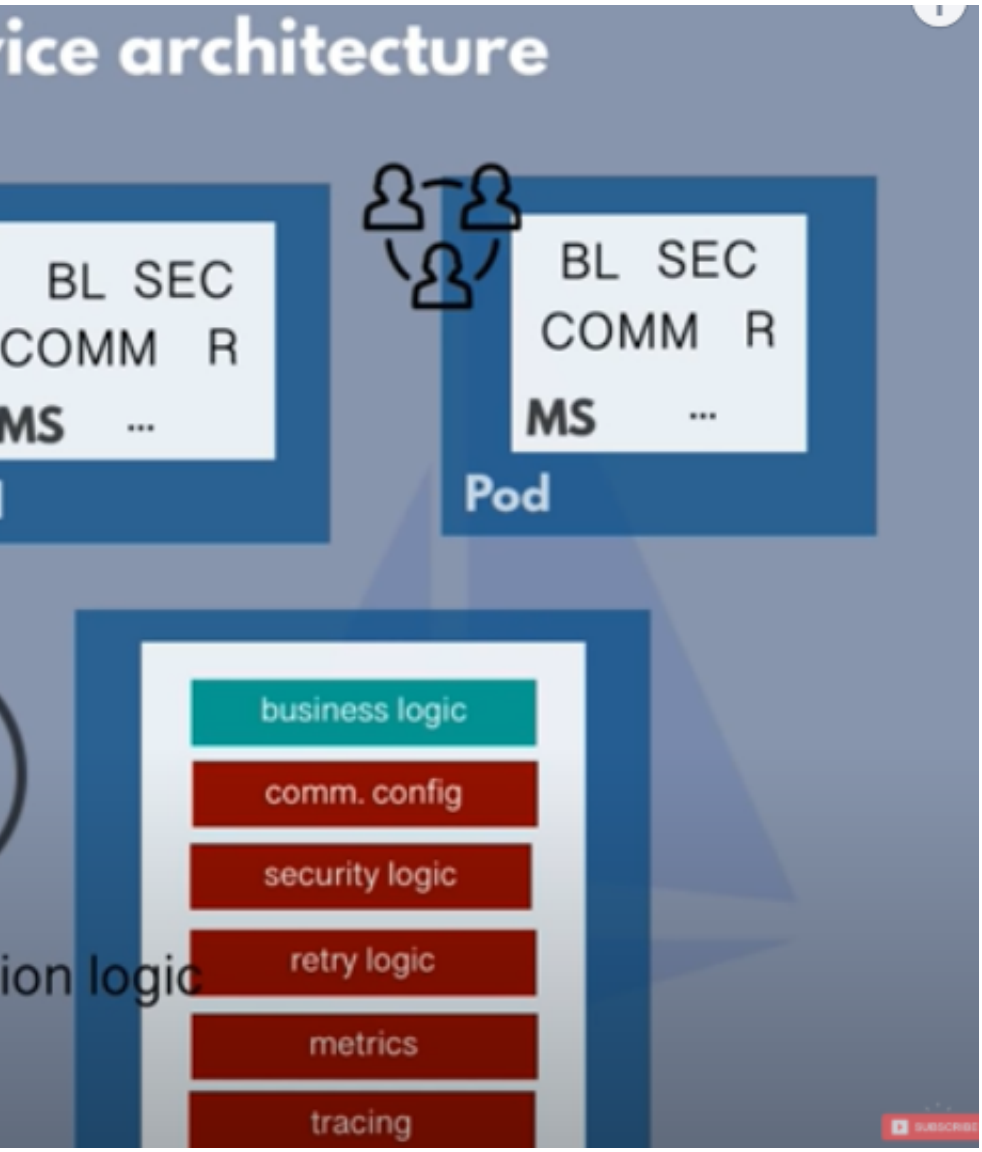
❌ Developers don't work on actual service/applicat

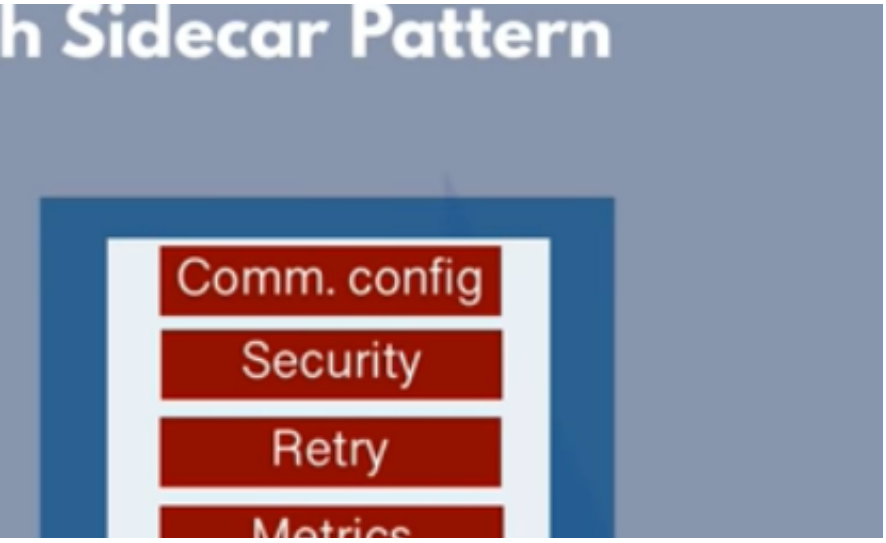
Solution: Service Mesh wit

Sidecar Proxy

- handles these networking logic
- acts as a **Proxy**


metrics
tracing


ice architecture
BL SEC
COMM R
MS ...
BL SEC
COMM R
MS ...
Pod
business logic
comm. config
security logic
retry logic
metrics
tracing
ion logic

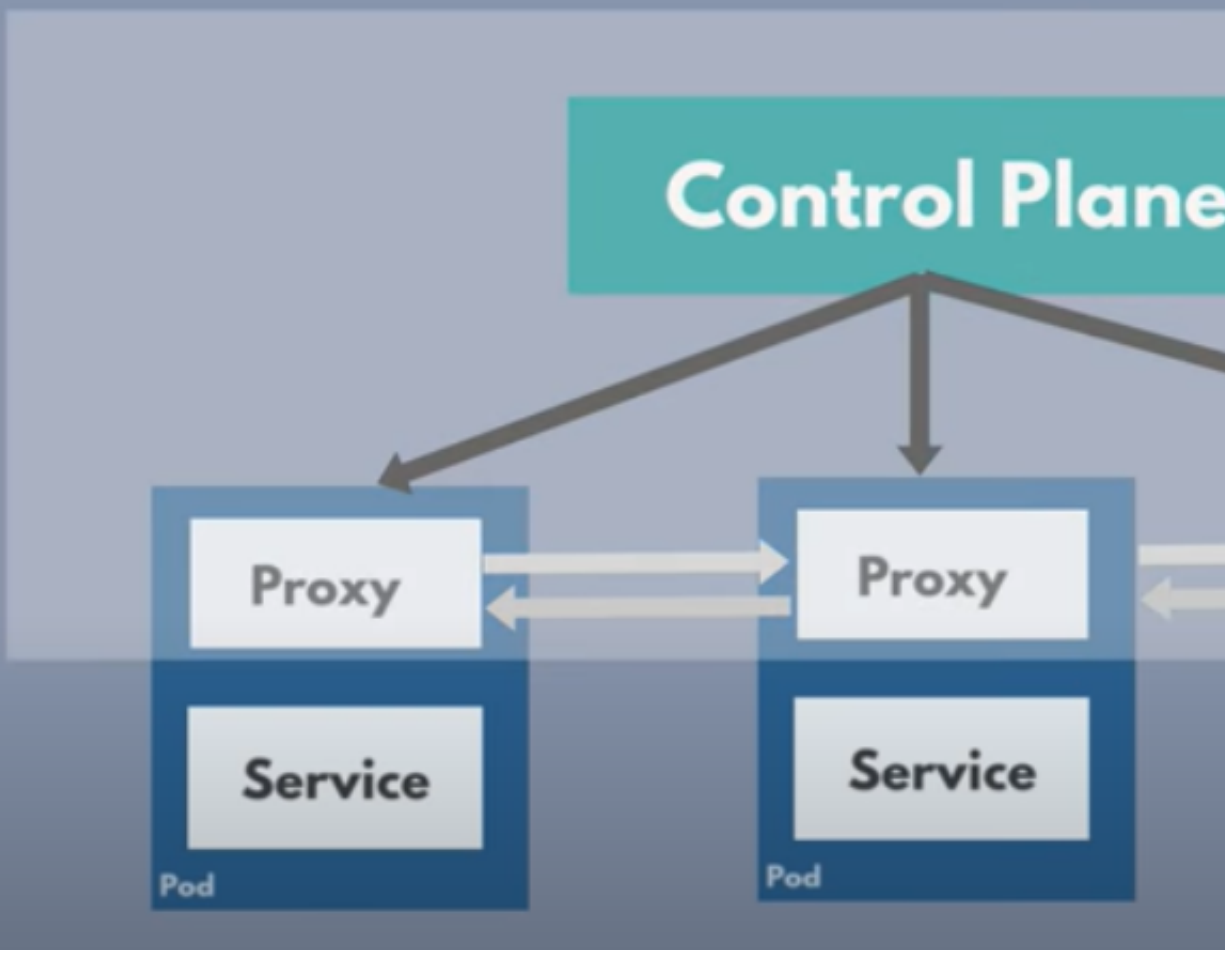

h Sidecar Pattern
Comm. config
Security
Retry
Metrics

- third-party application
- cluster operators can configure it easily

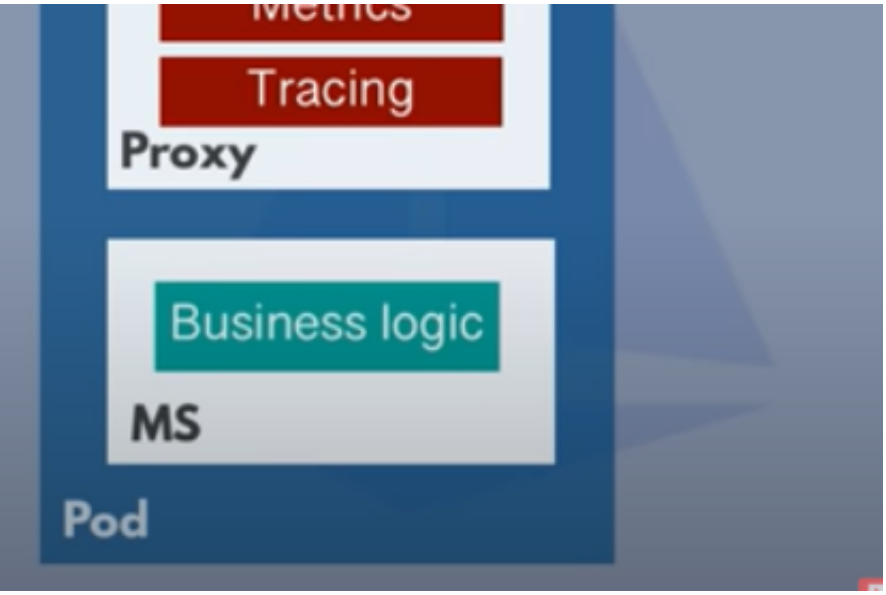
Control Plane

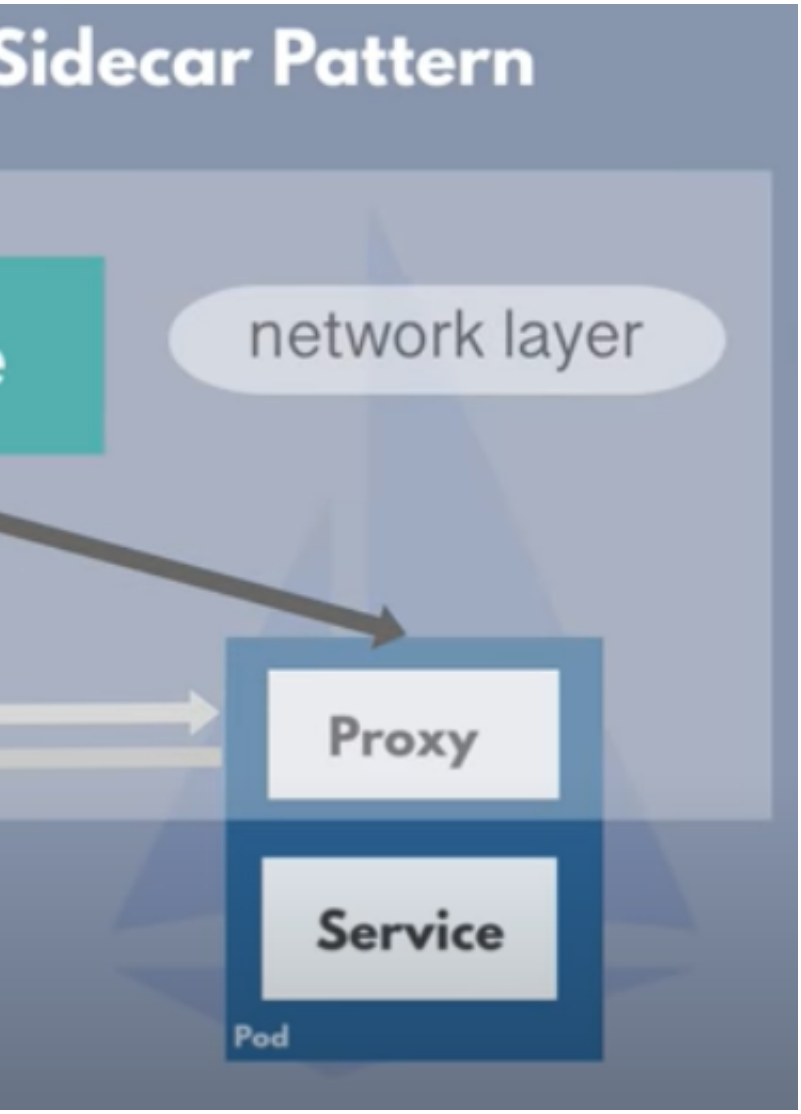
- Control Plane injects the Sidecar Proxy

Solution: Service Mesh with S

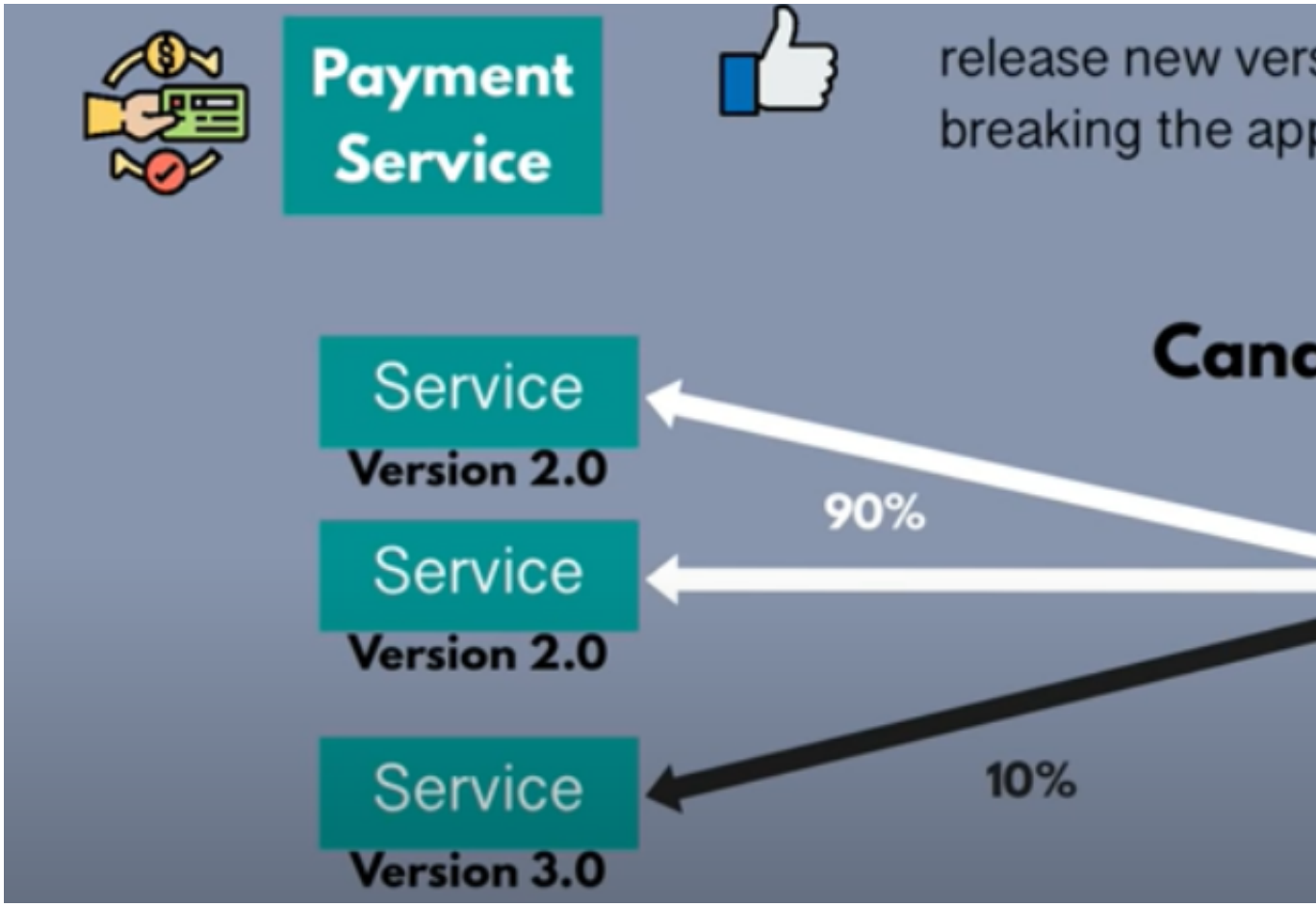


Core feature: Traffic Sp

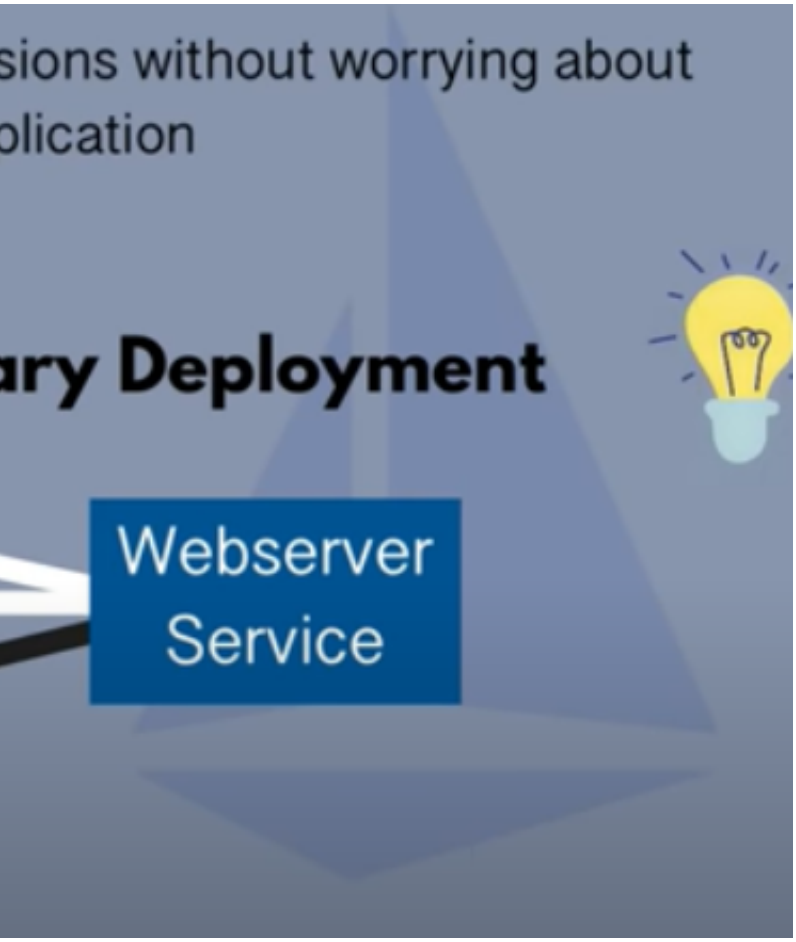

Tracing
Proxy
Business logic
MS
Pod


Sidecar Pattern
network layer
Proxy
Service
Pod

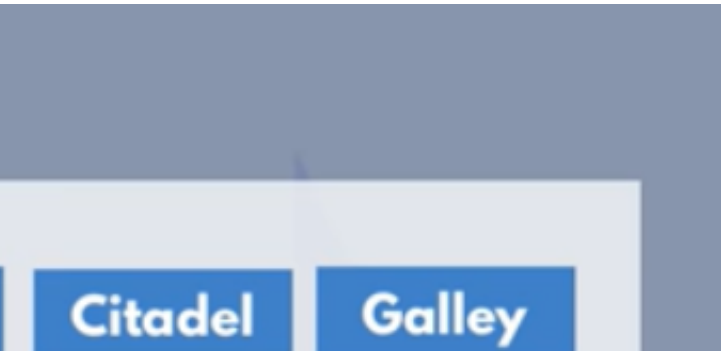

olitting

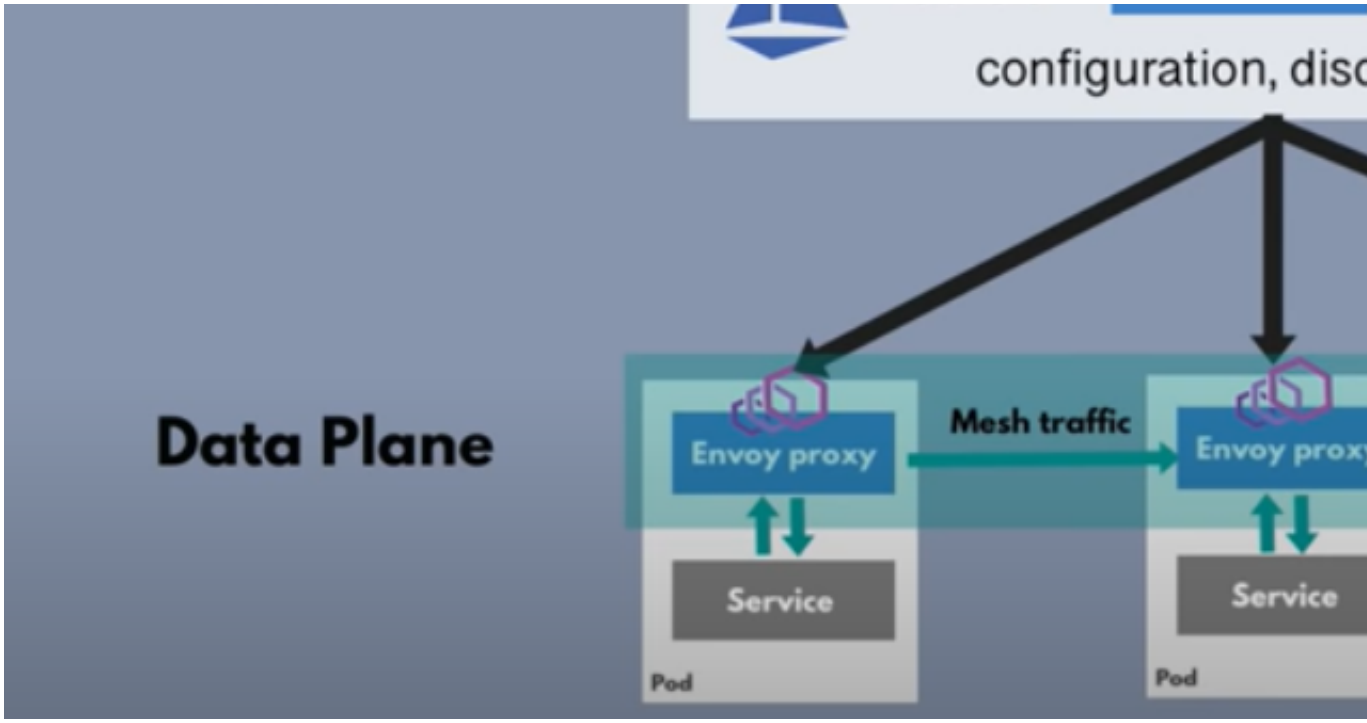


26) ISTIO Architecture ==========>


sions without worrying about
plication
ary Deployment
Webserver
Service


e
ementation


Citadel
Galley



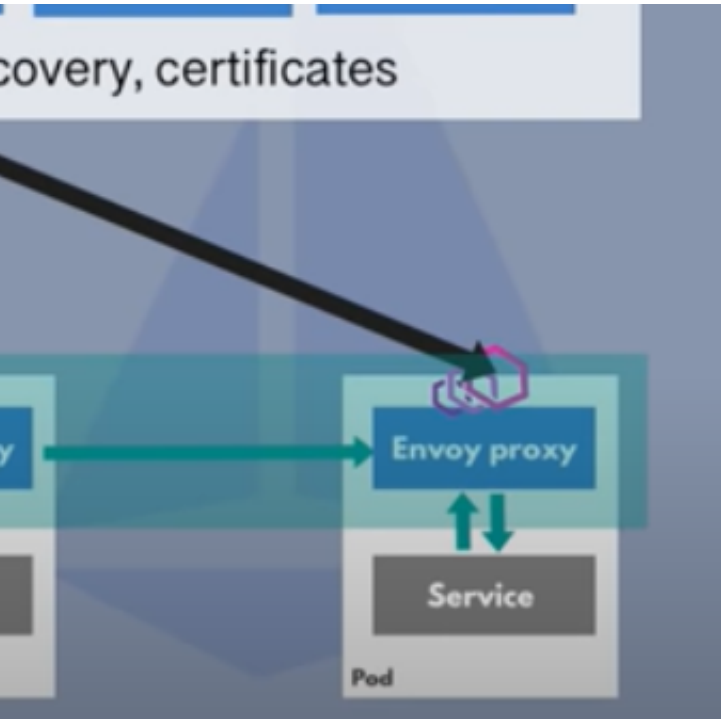
How to configure Is

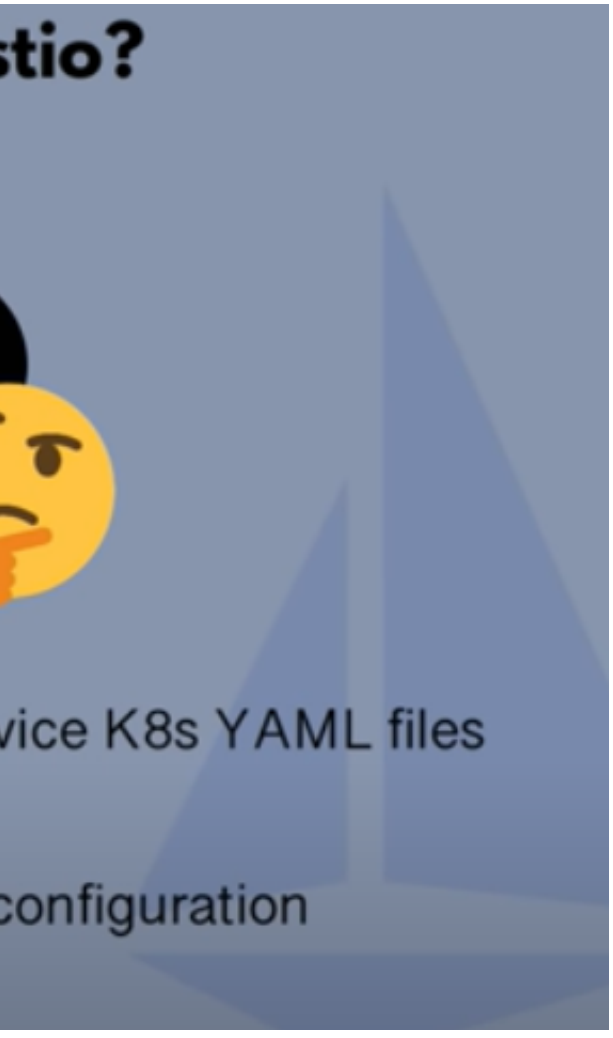
How do we configure all these features for our microservice application?

you don't have to adjust Deployment and Serv

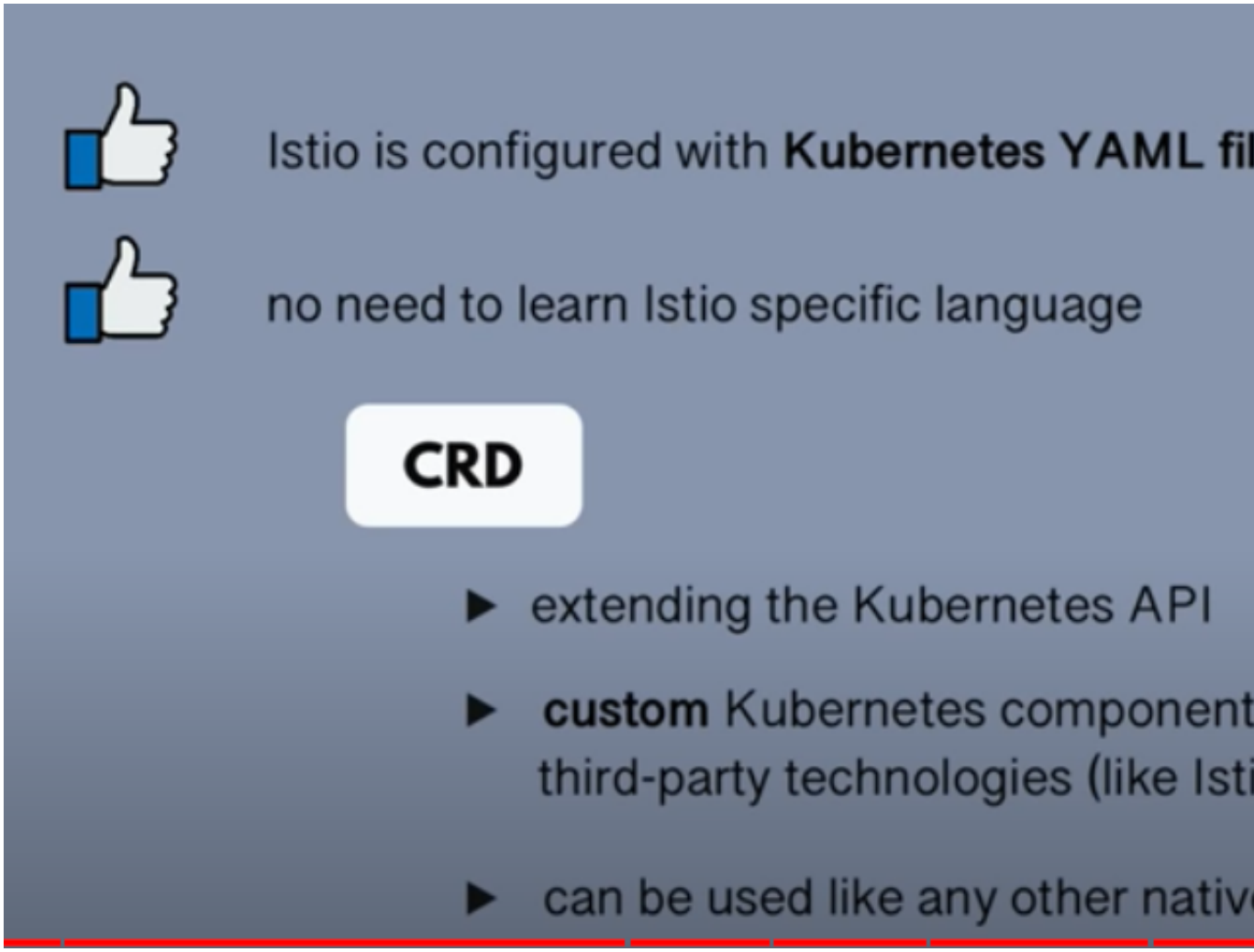
Istio configuration **separate** from application c

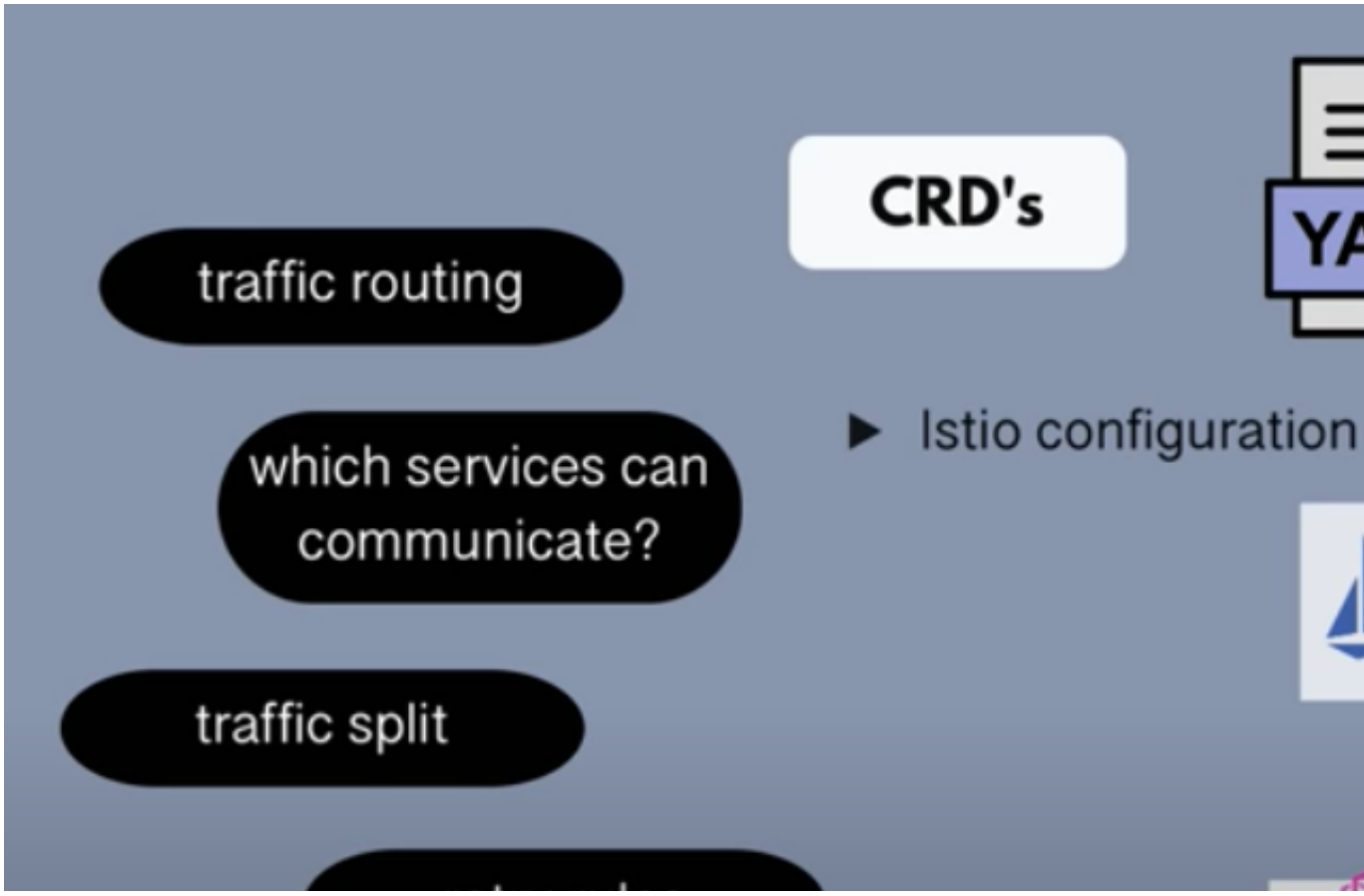
How to configure Is

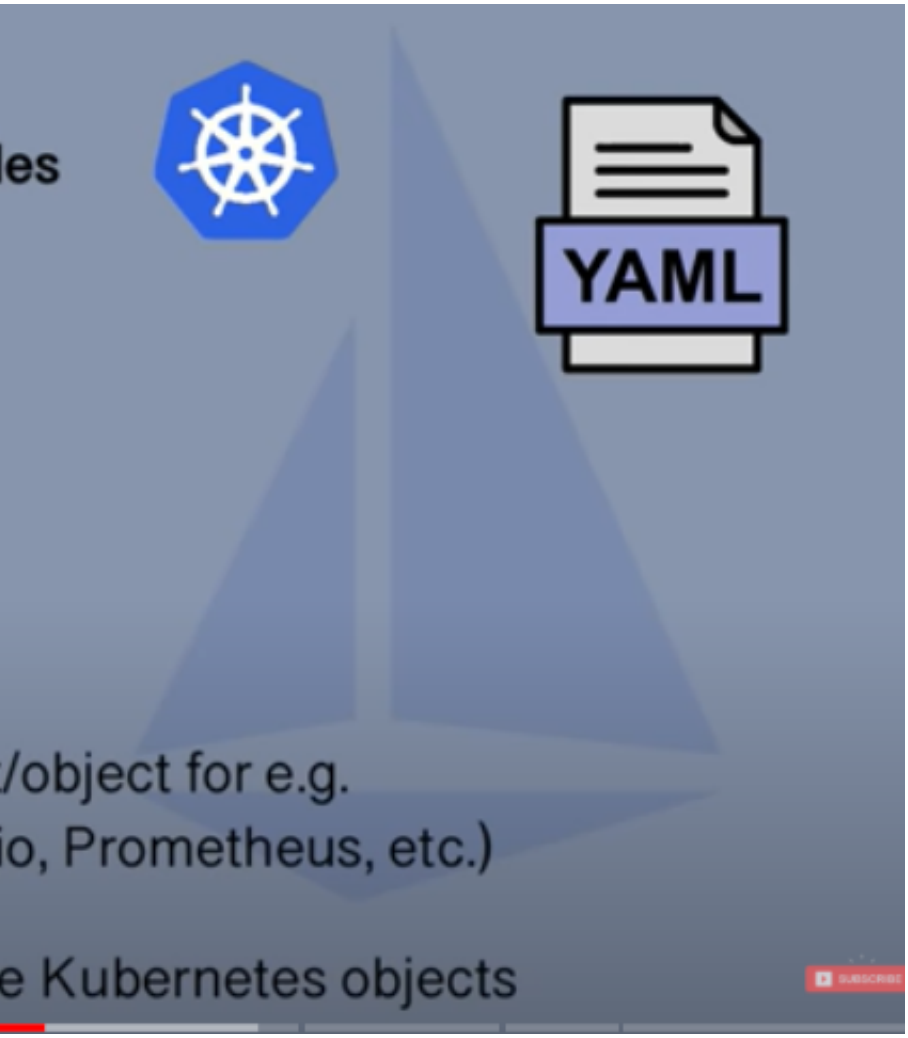

covery, certificates
Envoy proxy
Service
Pod

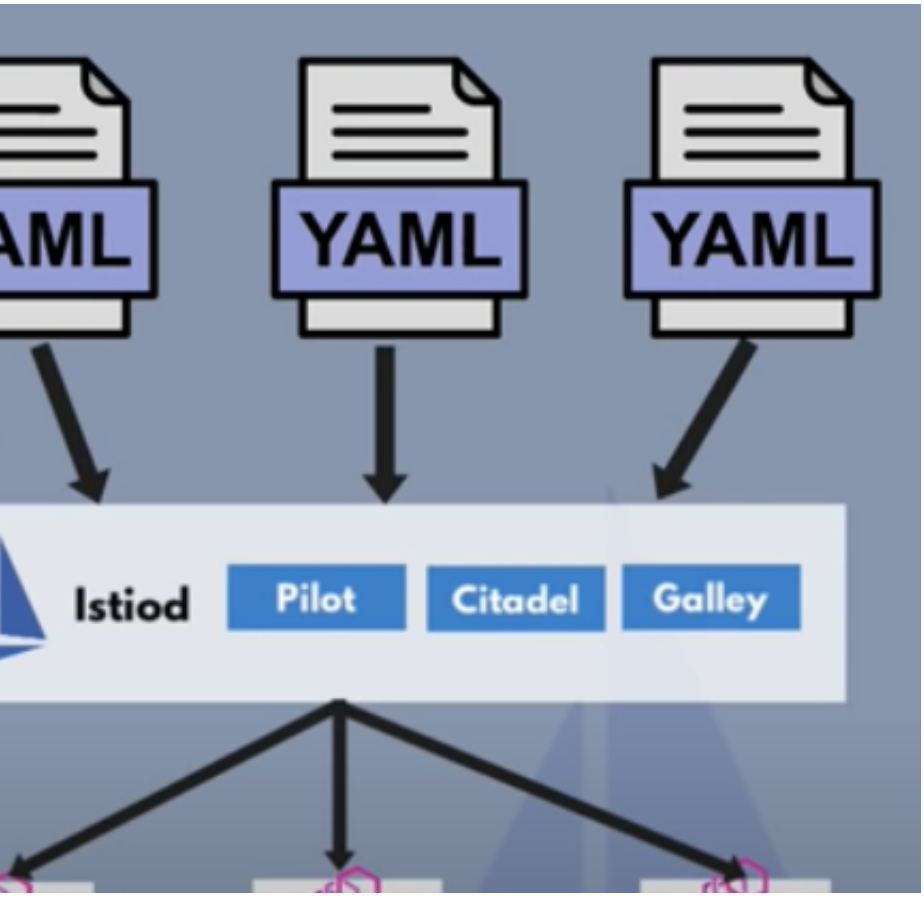

stio?
vice K8s YAML files
configuration


stio?

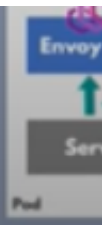

Istio is configured with **Kubernetes YAML fil**
no need to learn Istio specific language
CRD
extending the Kubernetes API
custom Kubernetes component
third-party technologies (like Isti
can be used like any other native


CRD's
YA
traffic routing
Istio configuration
which services can communicate?
traffic split


les
YAML
/object for e.g.
io, Prometheus, etc.)
e Kubernetes objects


AML
YAML
YAML
Istiod
Pilot
Citadel
Galley

retry rules

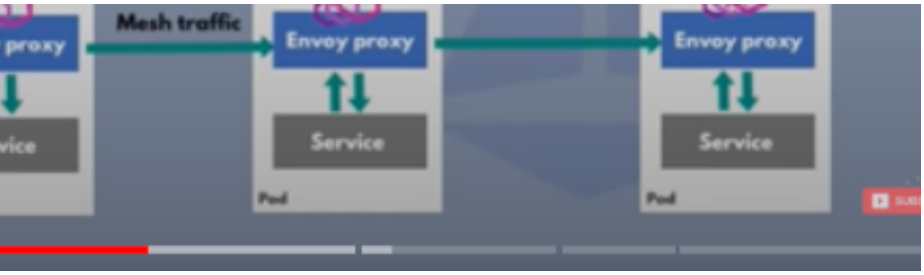


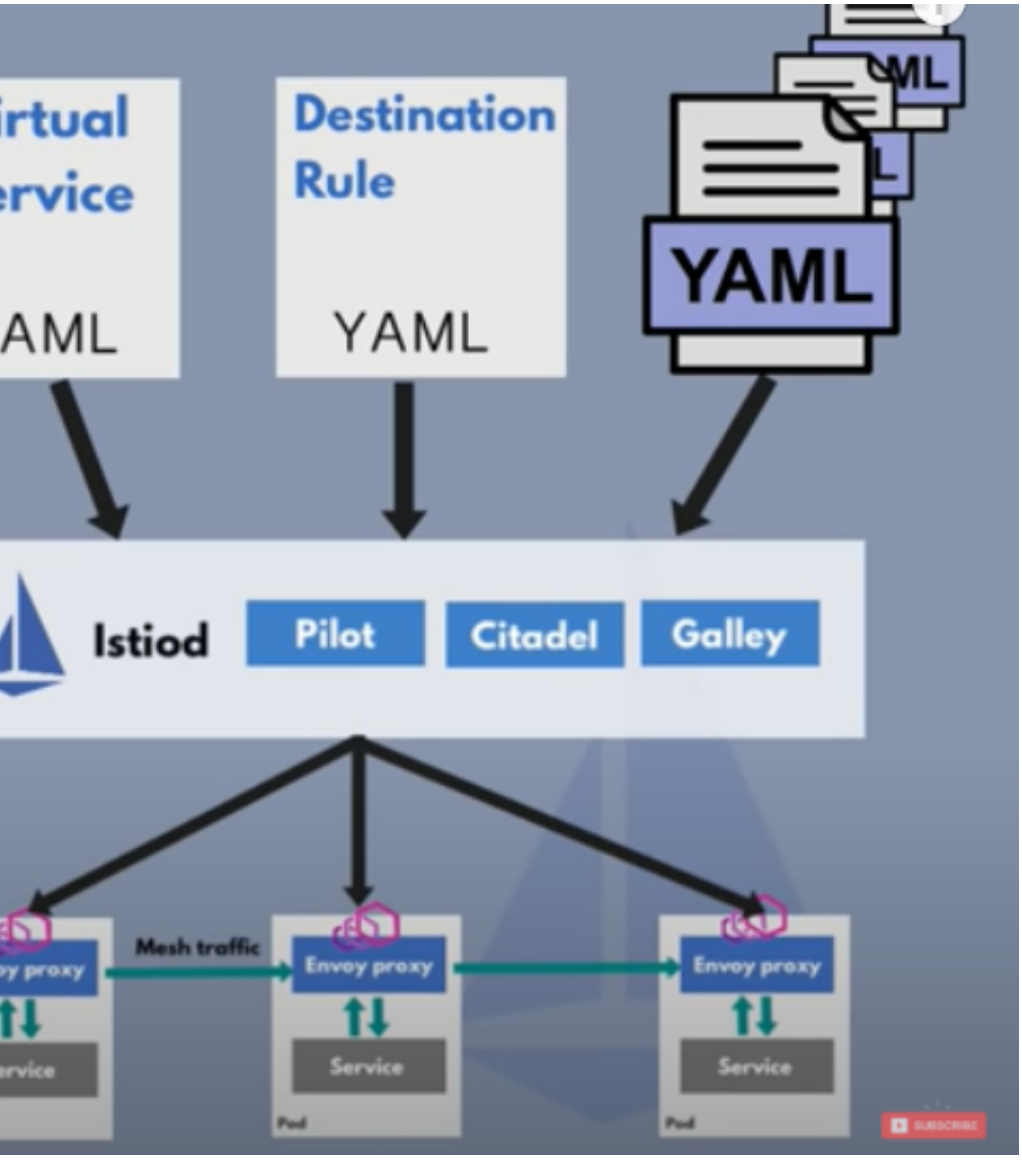
- We create CRD's

- Istiod **converts** these high level routing rules **into Envoy-specific configurations**

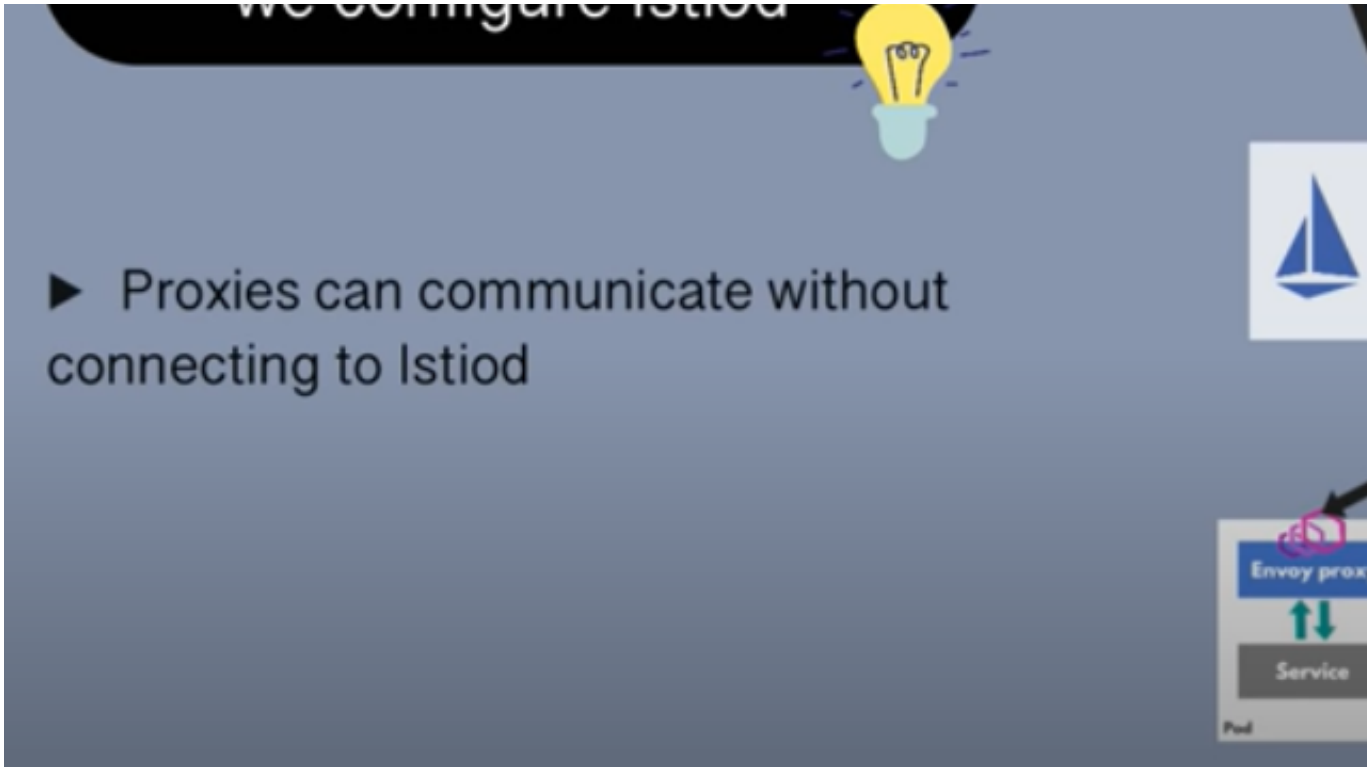
- Configuration is **propagated into Proxy sidecars**

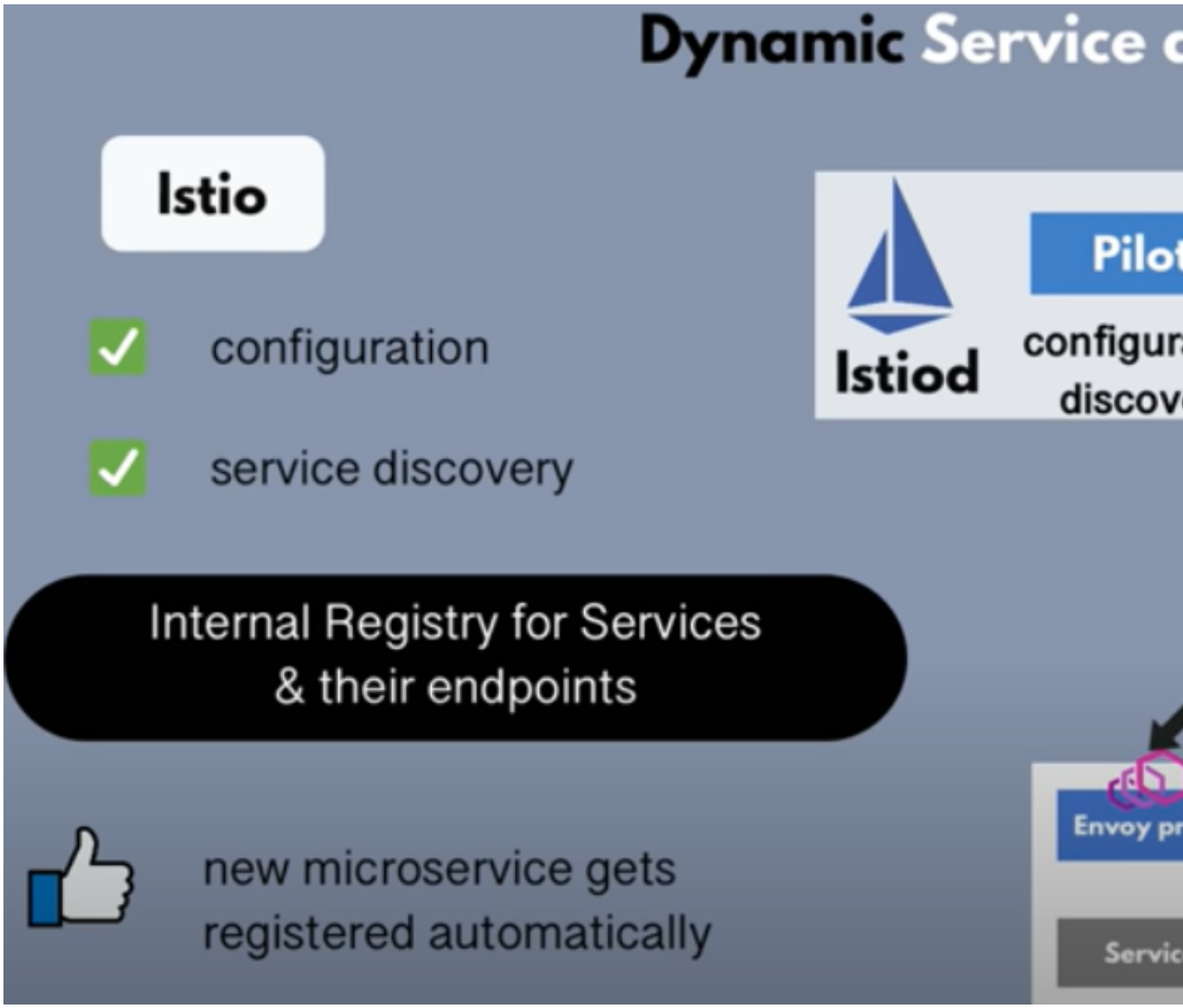
We don't configure Proxies, we configure Istiod

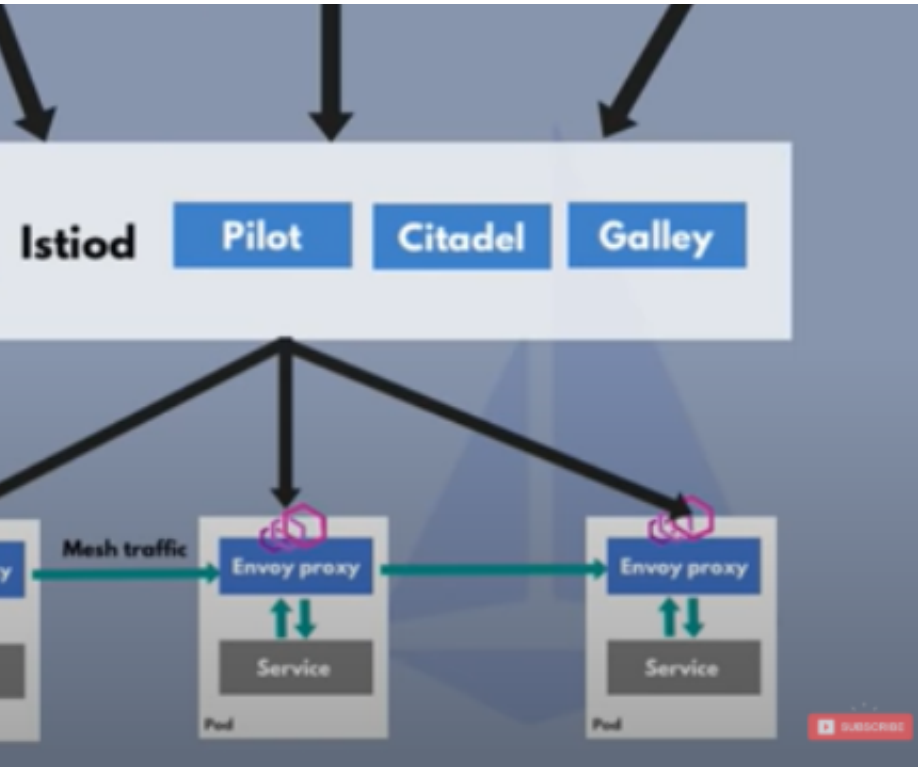

Mesh traffic
Envoy proxy
Service
Pod
Envoy proxy
Service
Pod

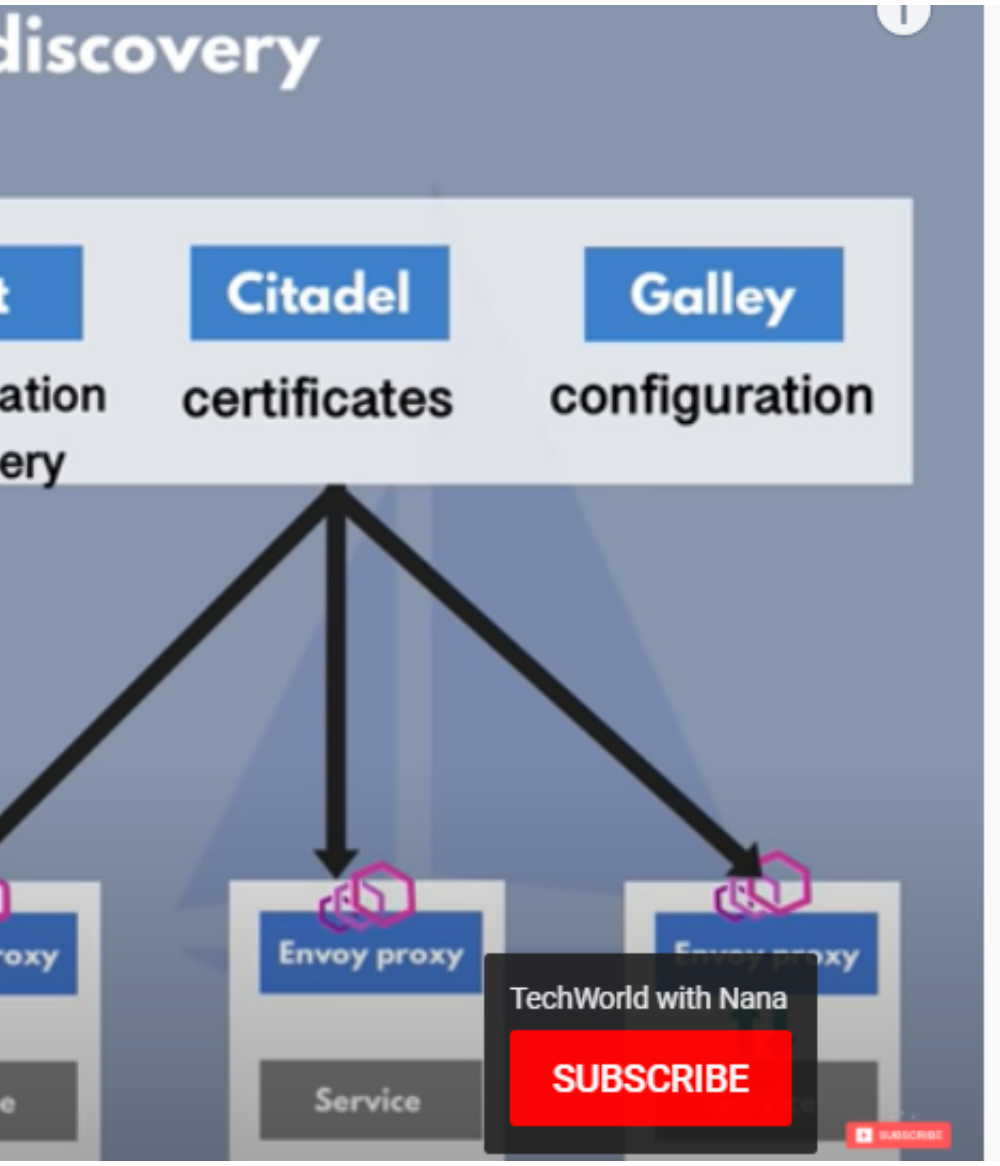

Destination Rule
YAML
YAML
Istiod
Pilot
Citadel
Galley
Mesh traffic
Envoy proxy
Service
Pod
Envoy proxy
Service
Pod

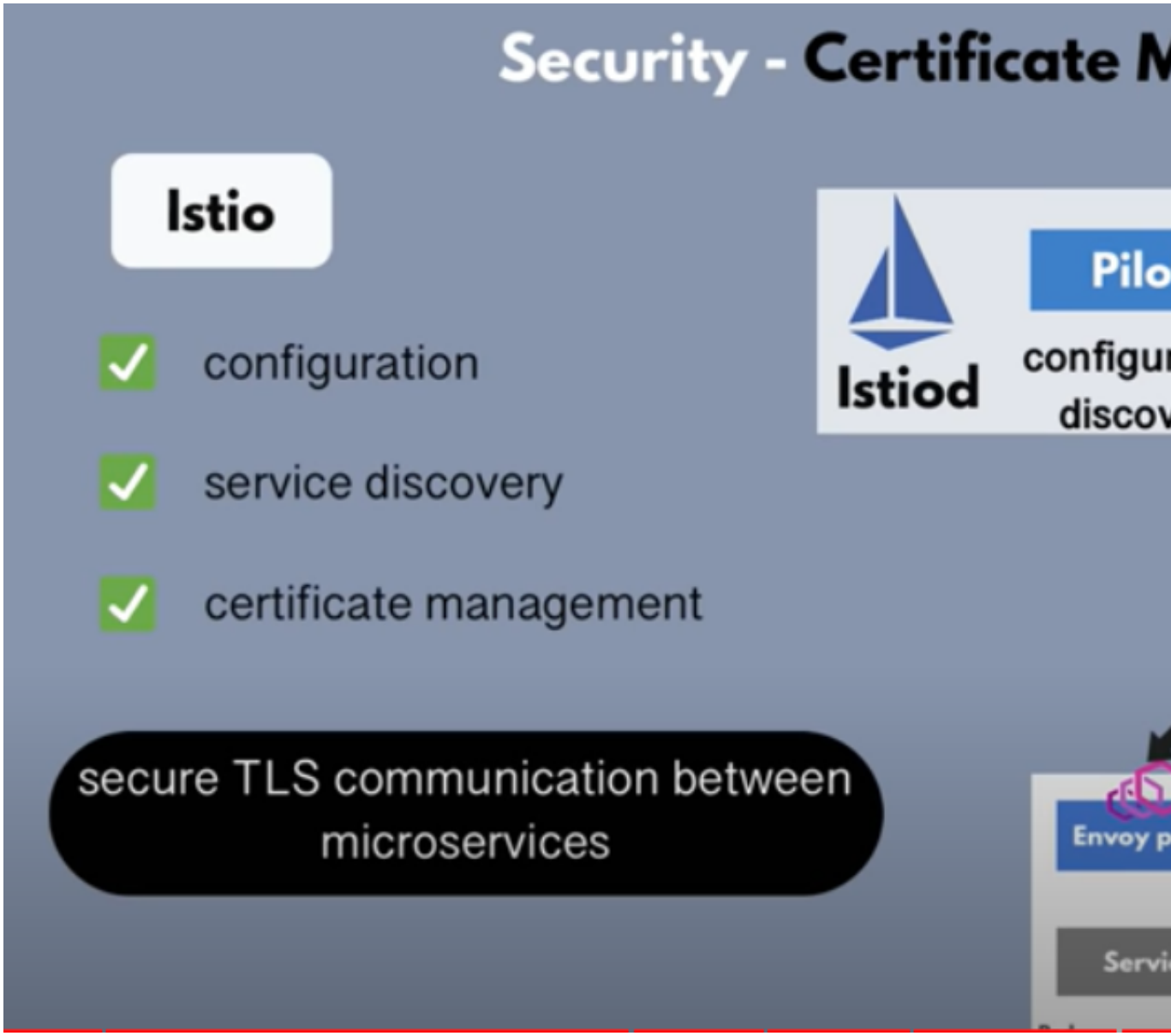

Destination Rule
YAML
YAML


▶ Proxies can communicate without connecting to Istiod
Envoy prox
Service
Pod

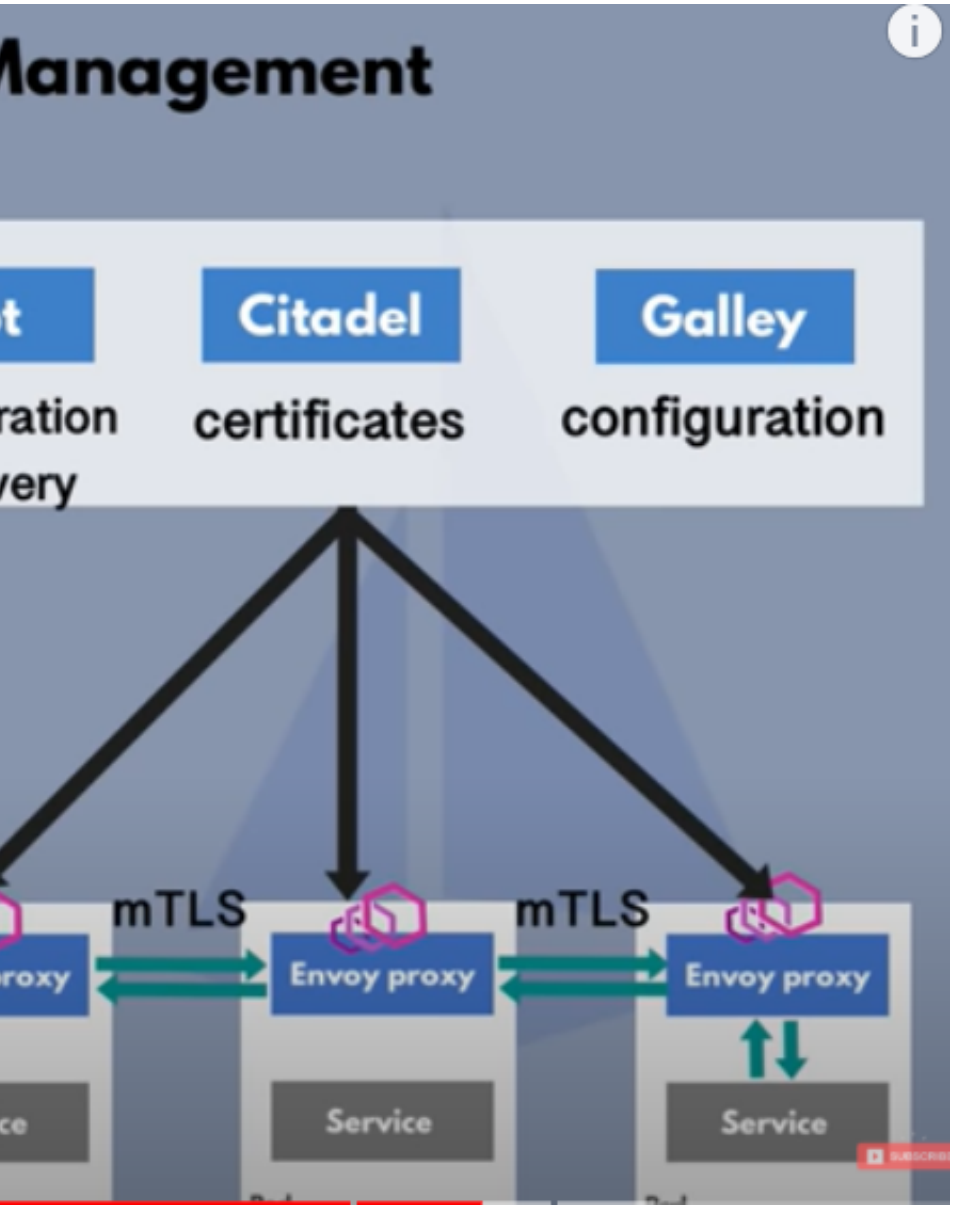

Dynamic Service d
Istio
configuration
service discovery
Istiod
Pilot
configur
discov
Internal Registry for Services
& their endpoints
new microservice gets
registered automatically
Envoy pr
Servic


Istiod
Pilot
Citadel
Galley
Mesh traffic
Envoy proxy
Service
Pod
Envoy proxy
Service
Pod

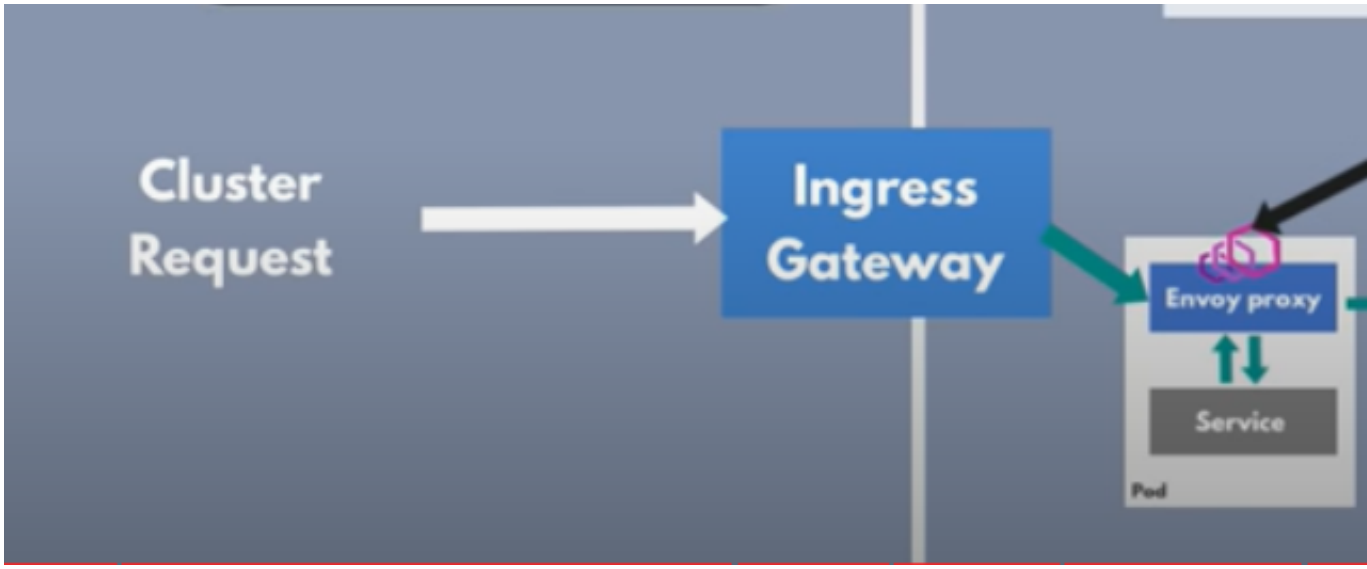

discovery
Citadel
Galley
certificates
configuration
Envoy proxy
Service
TechWorld with Nana
SUBSCRIBE

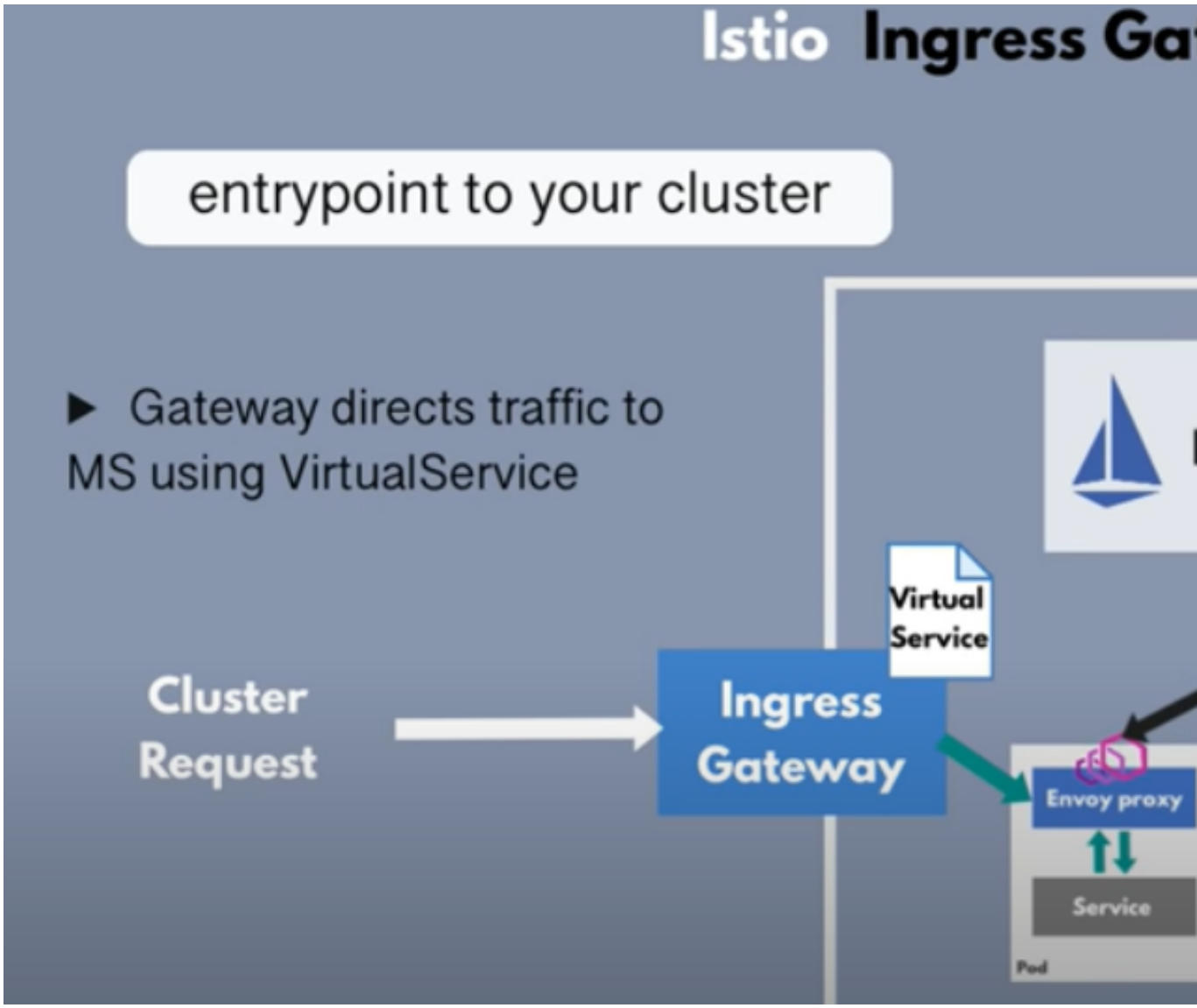

Security - Certificate M
Istio
configuration
service discovery
certificate management
secure TLS communication between microservices
Istiod
Pilo
configu
discov
Envoy p
Servi


Istio Ingress Gat
entrypoint to your cluster
alternative to nginx Ingress Controller

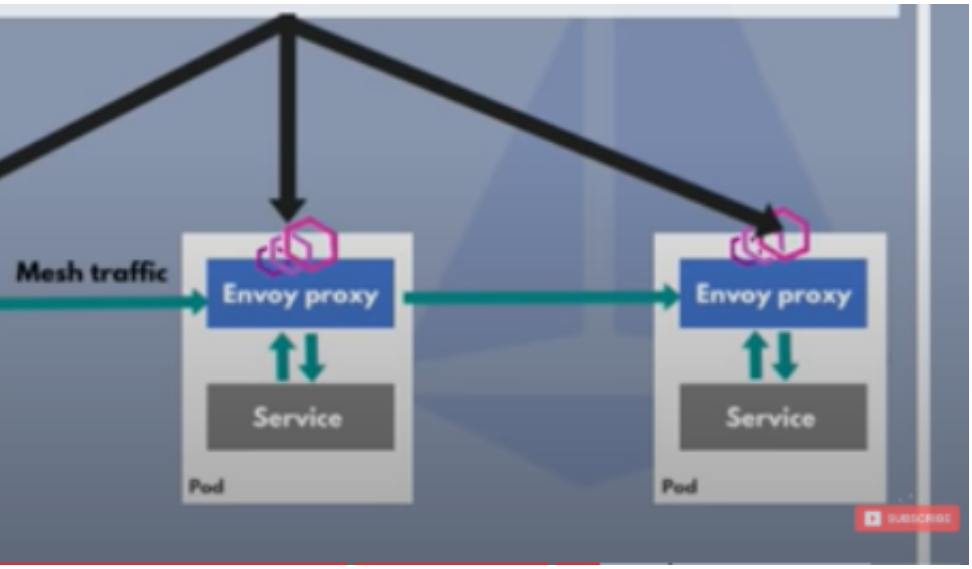

Management
Citadel
Galley
certificates
configuration
mTLS
mTLS
Envoy proxy
Envoy proxy
Service
Service

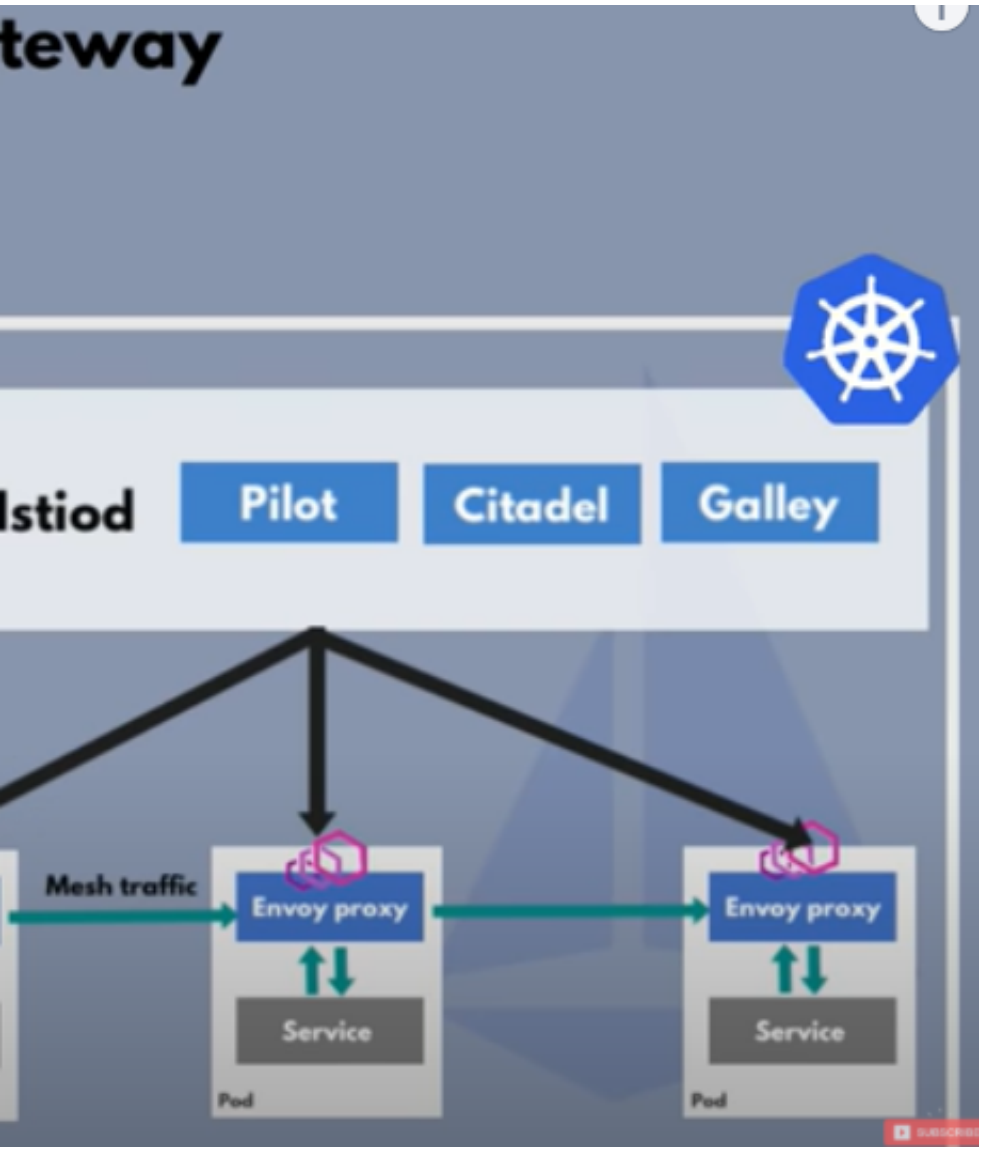

eway
Pilot
Citadel
Galley


Cluster
Request
Ingress
Gateway
Envoy proxy
Service
Pod

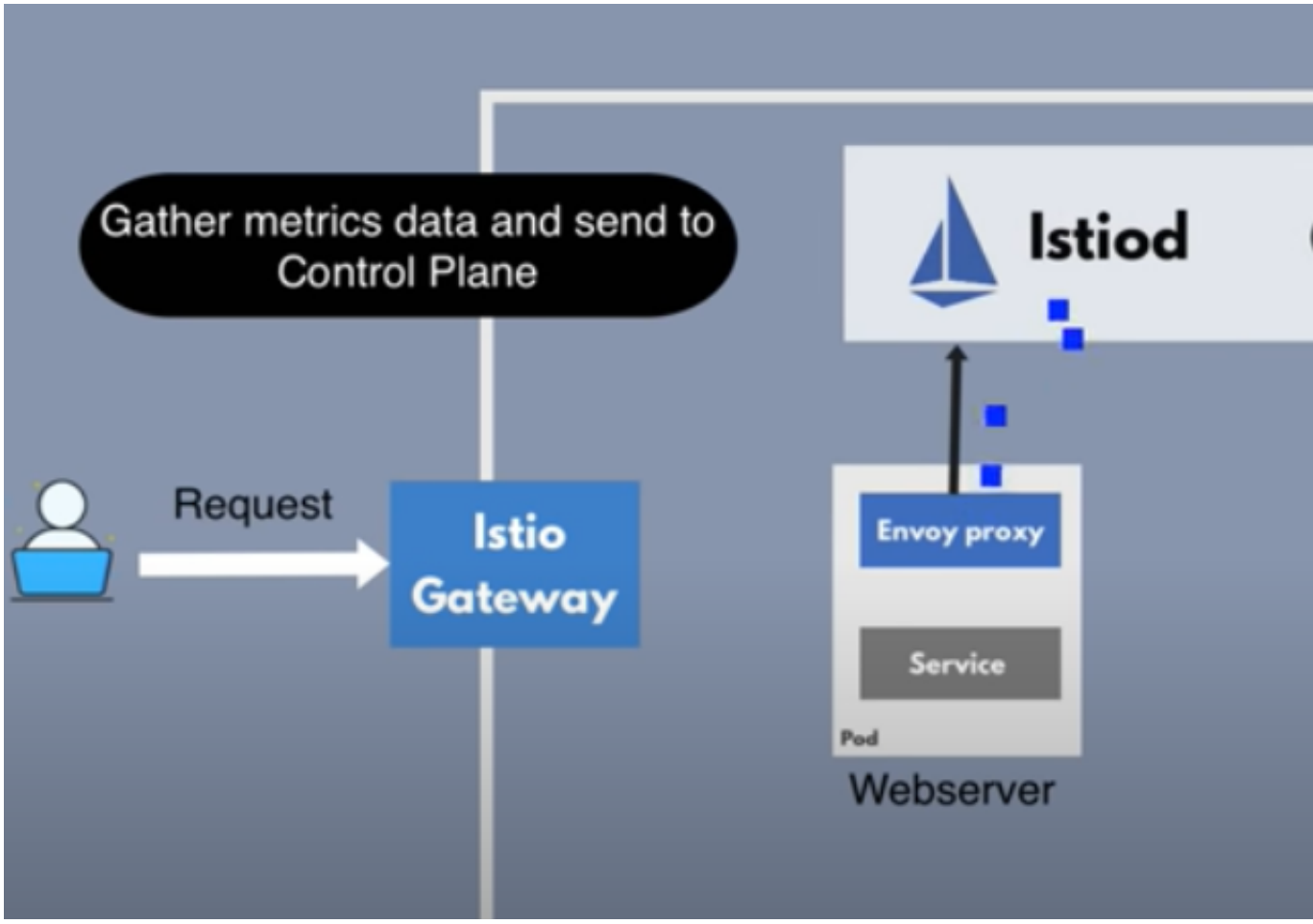

Istio Ingress Ga
entrypoint to your cluster
▶ Gateway directs traffic to
MS using VirtualService
Virtual
Service
Cluster
Request
Ingress
Gateway
Envoy proxy
Service
Pod

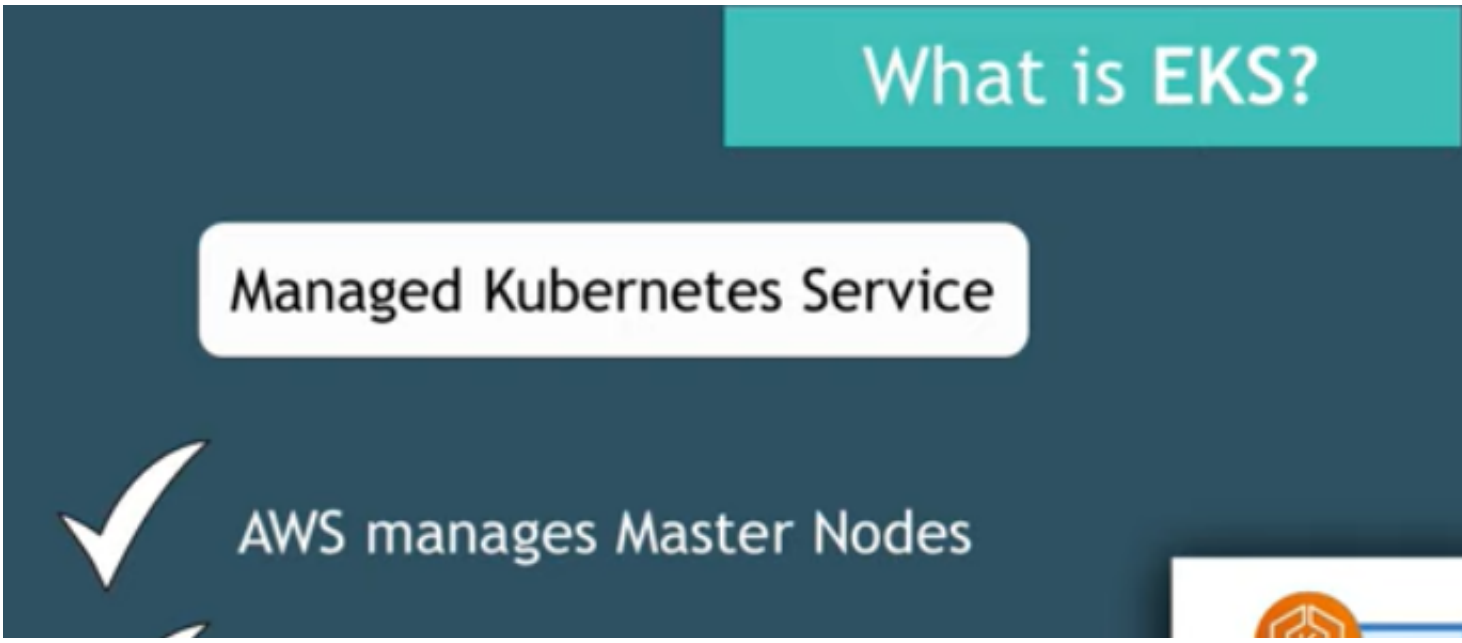

Istio: Traffic F

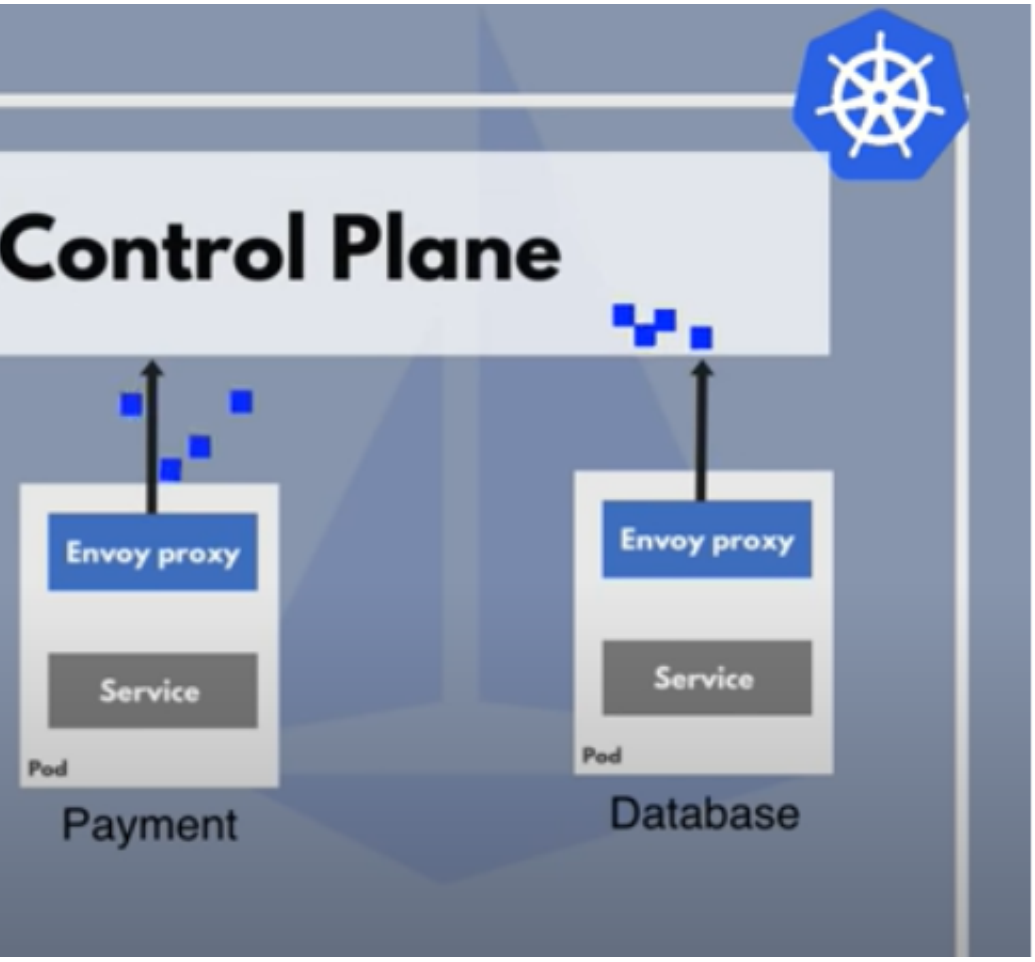

Mesh traffic
Envoy proxy
Service
Pod
Envoy proxy
Service
Pod

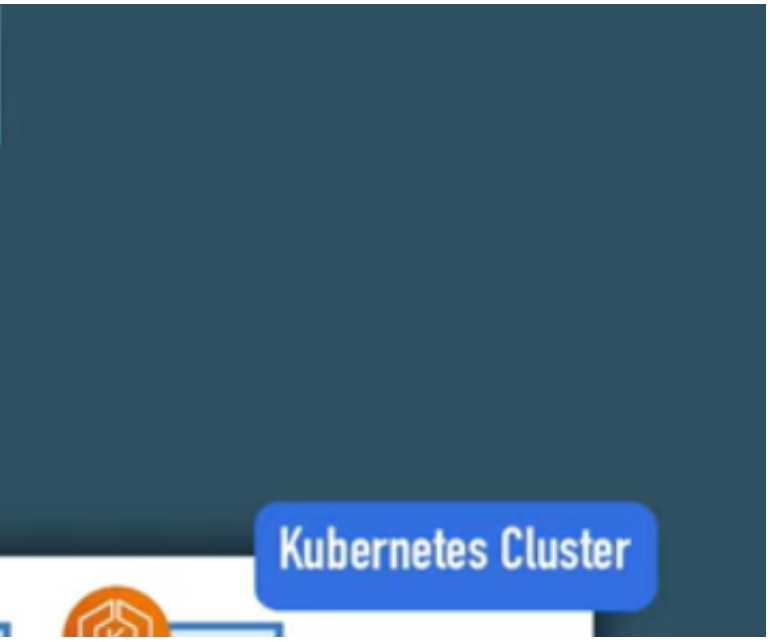

teway
lstiod
Pilot
Citadel
Galley
Mesh traffic
Envoy proxy
Service
Pod
Envoy proxy
Service
Pod

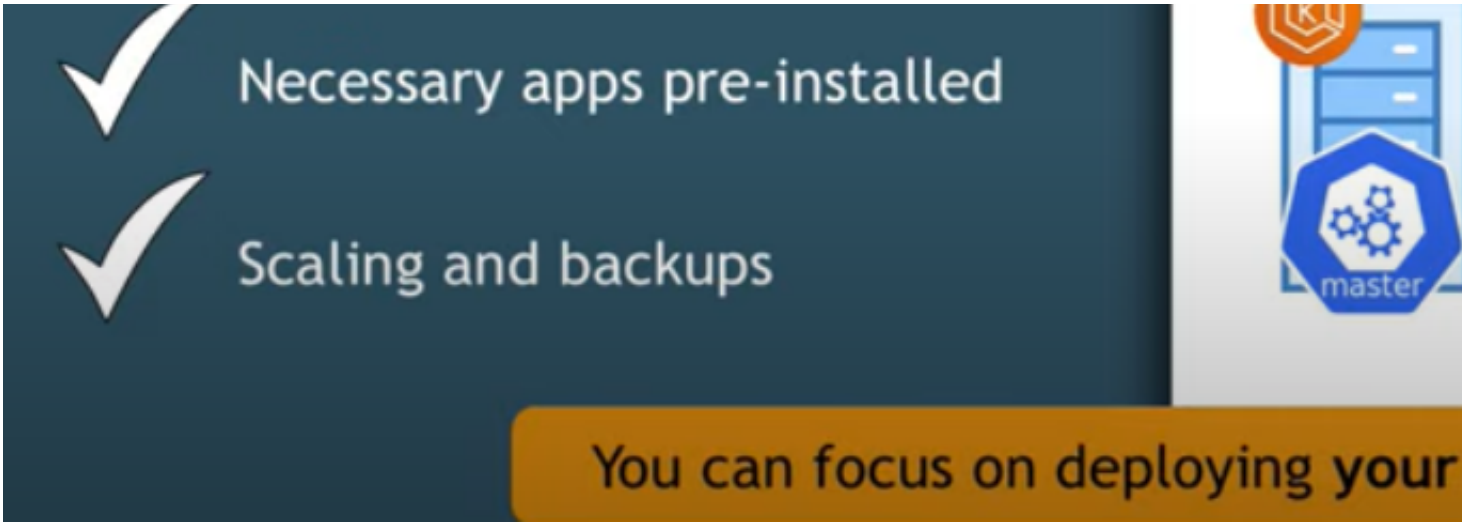

low


Gather metrics data and send to Control Plane
Istiod
Request
Istio Gateway
Envoy proxy
Service
Pod
Webserver

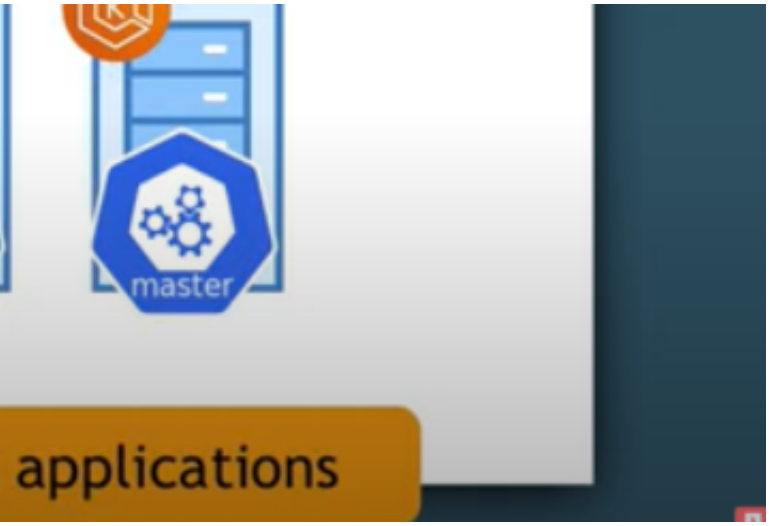

What is EKS?
Managed Kubernetes Service
AWS manages Master Nodes


Control Plane
Envoy proxy
Service
Pod
Payment
Envoy proxy
Service
Pod
Database

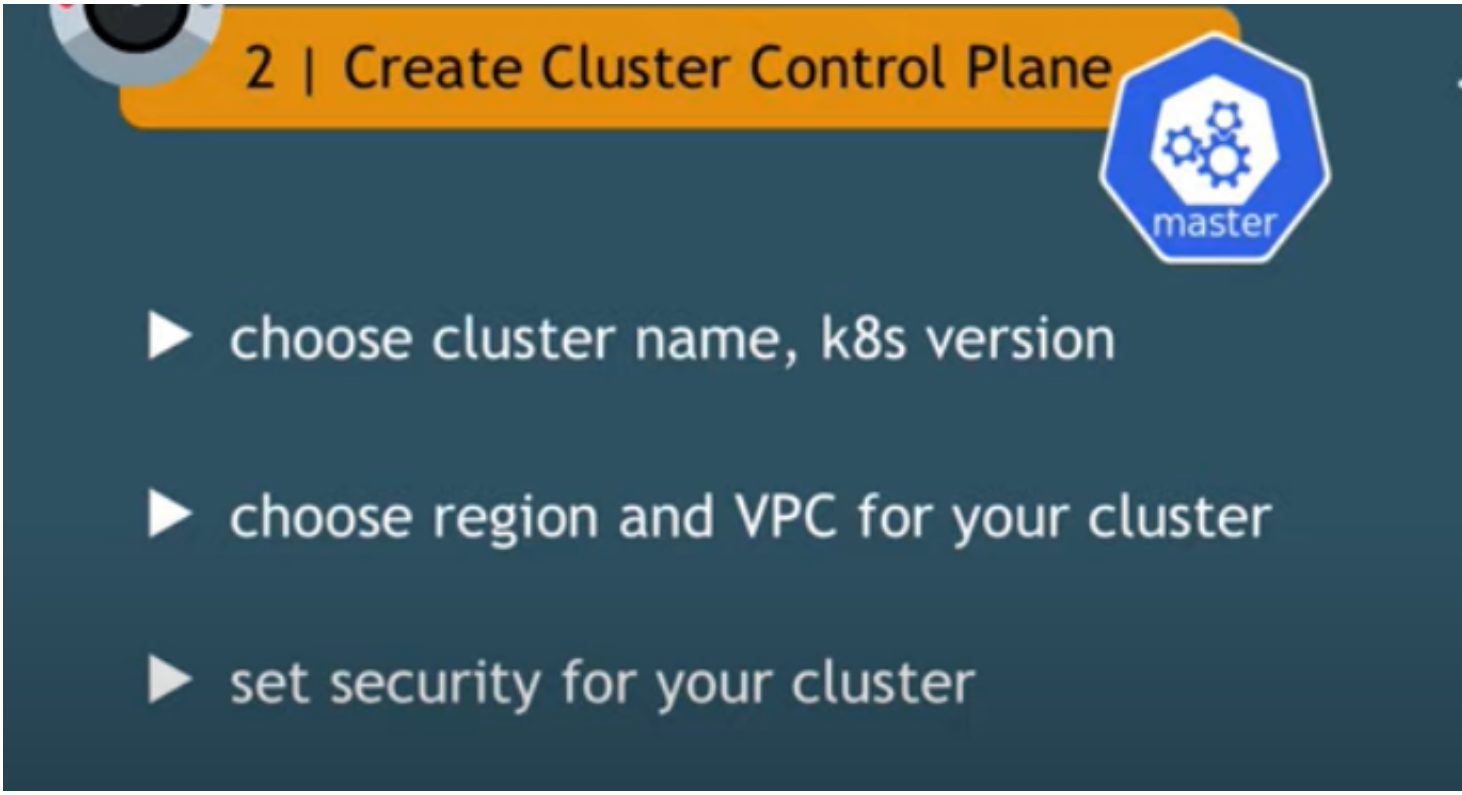

Kubernetes Cluster

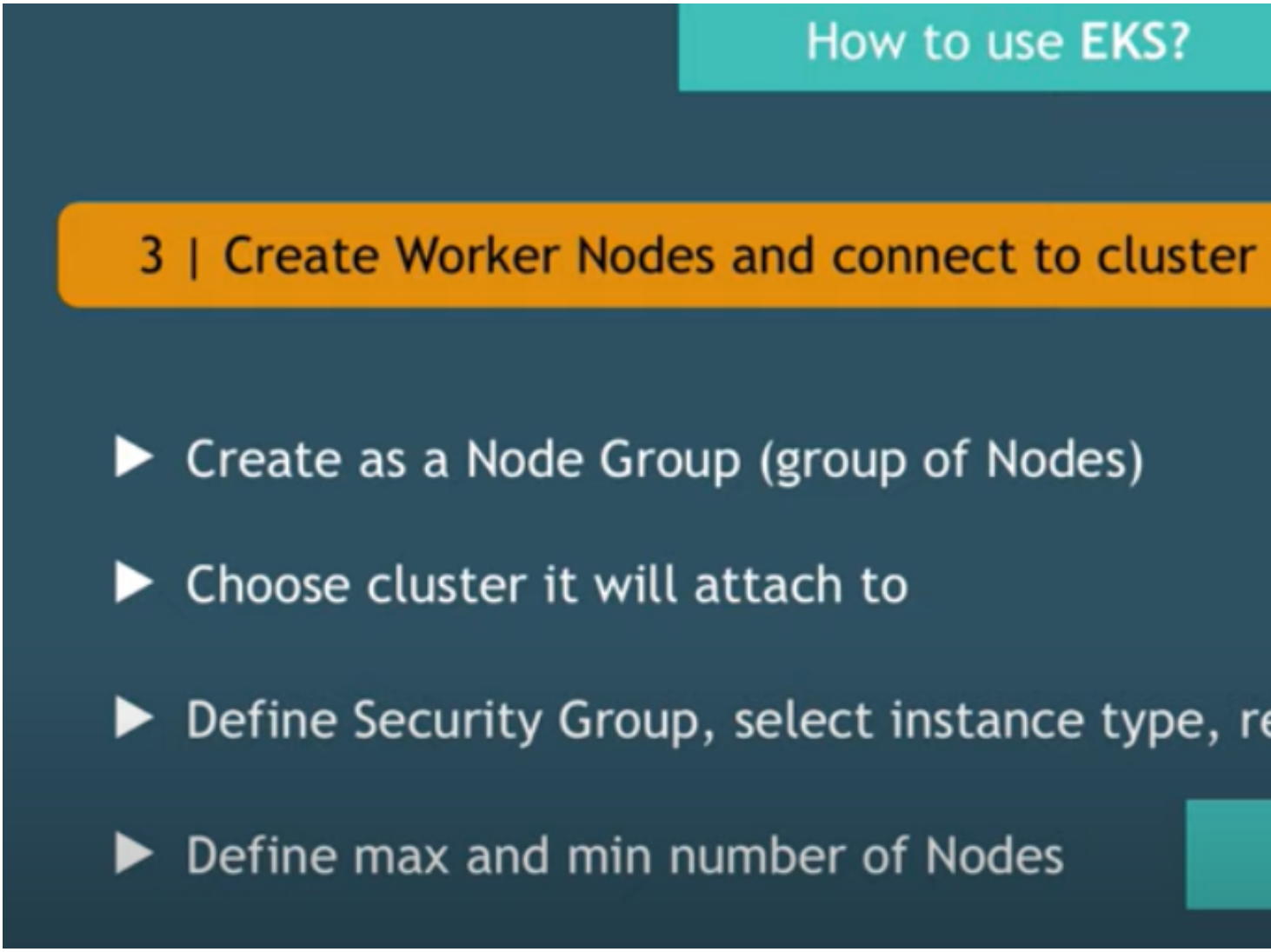


Managed k8 Service :: No need to bother about Master Nodes.. Only our headache is to bother about Worker Nodes.


master
applications

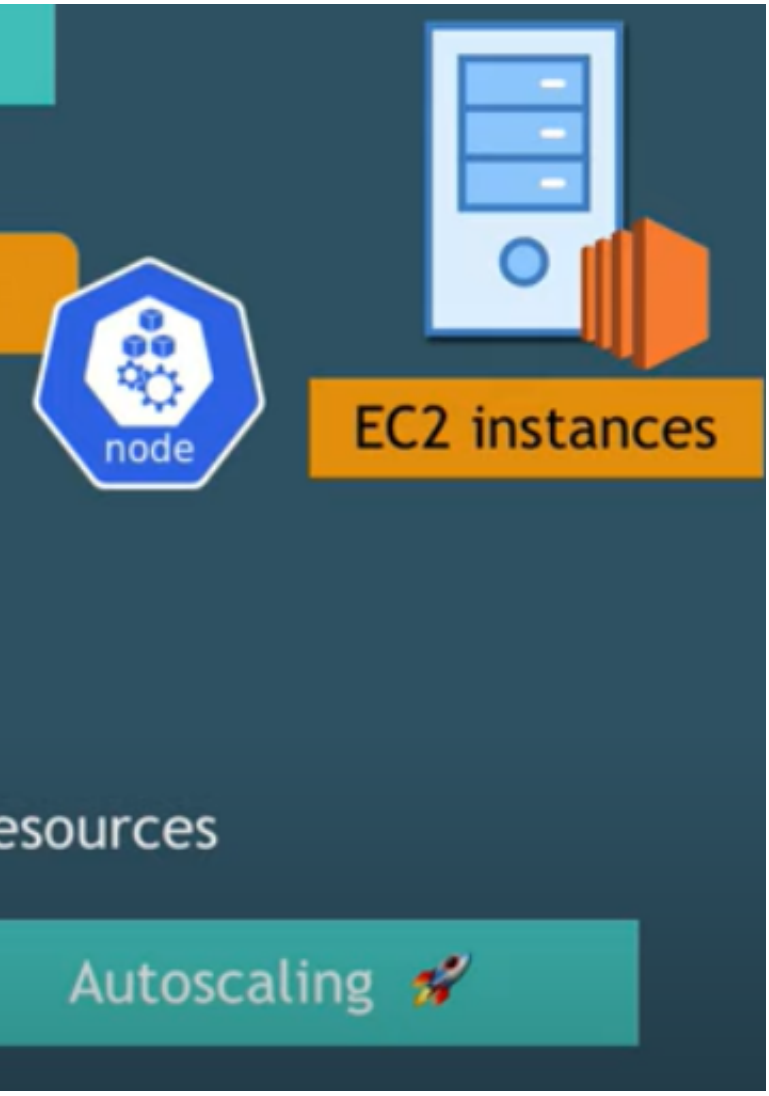

Let's use EKS!
curity Group
of permissions

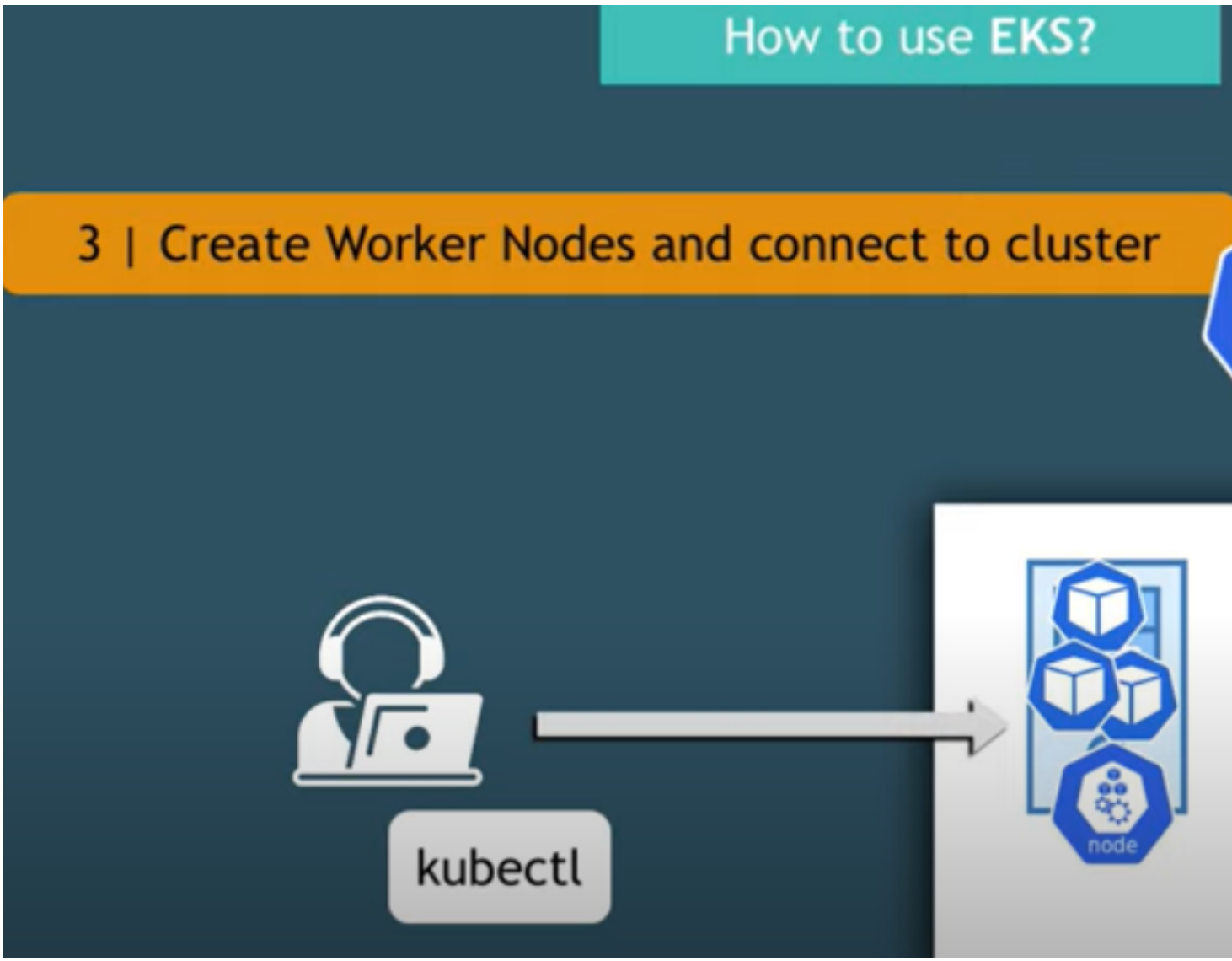




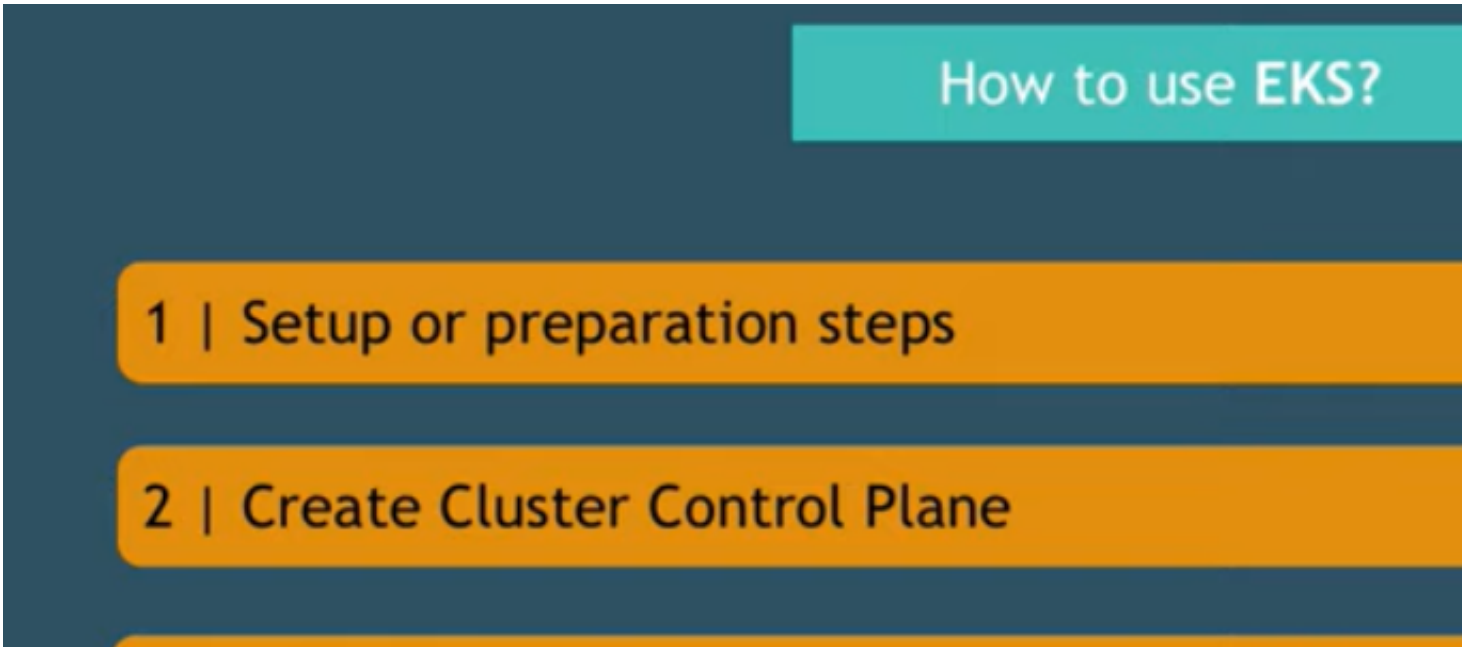
Use Kubectl to connect your local to AWS or K8 Cluster.

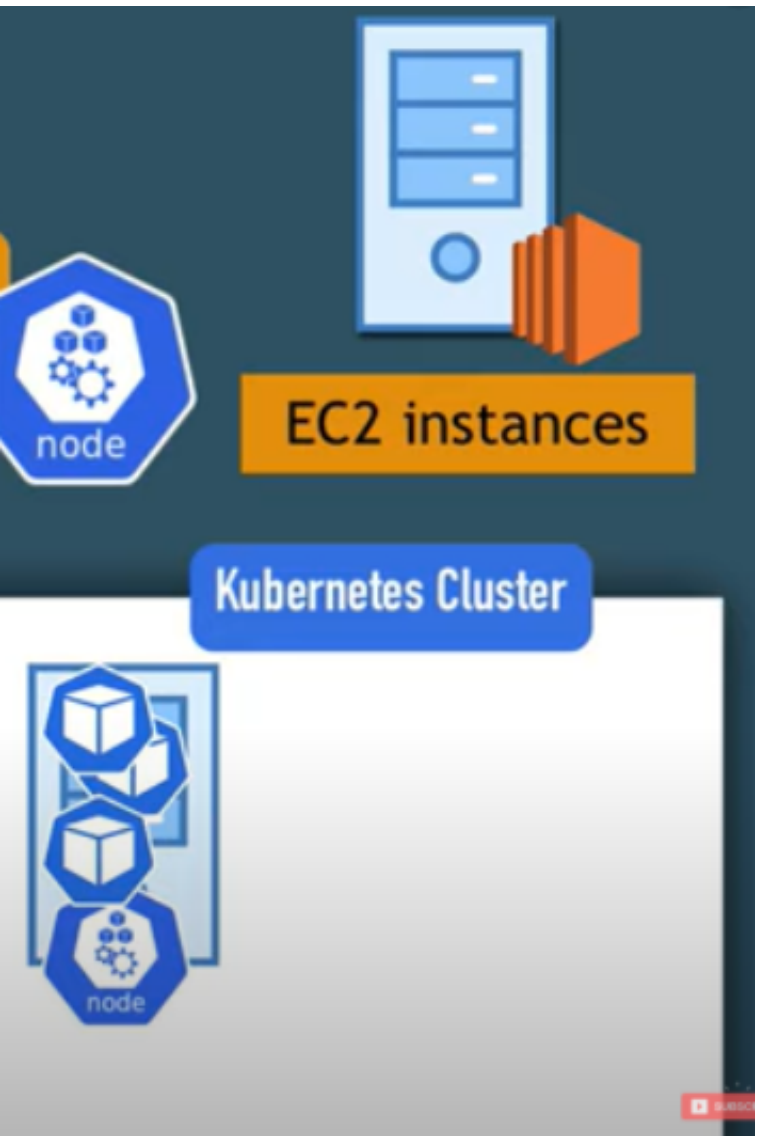
...create with IAM role

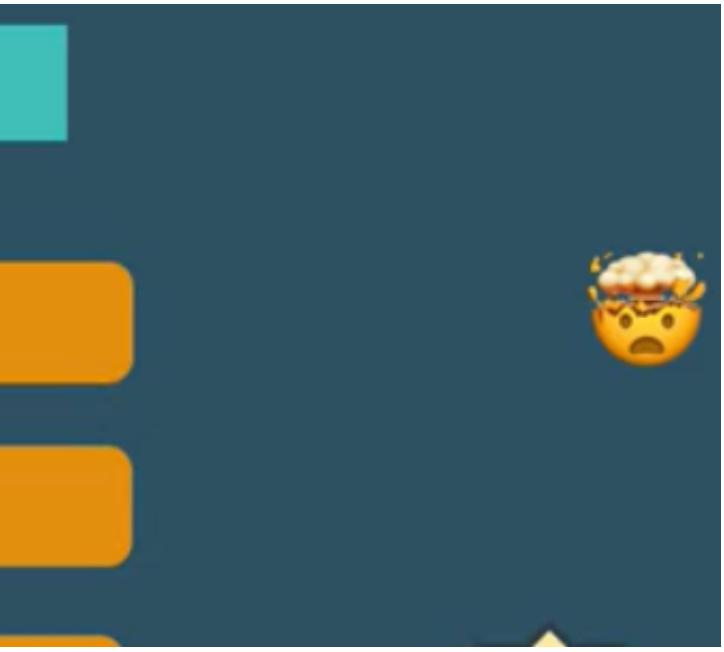


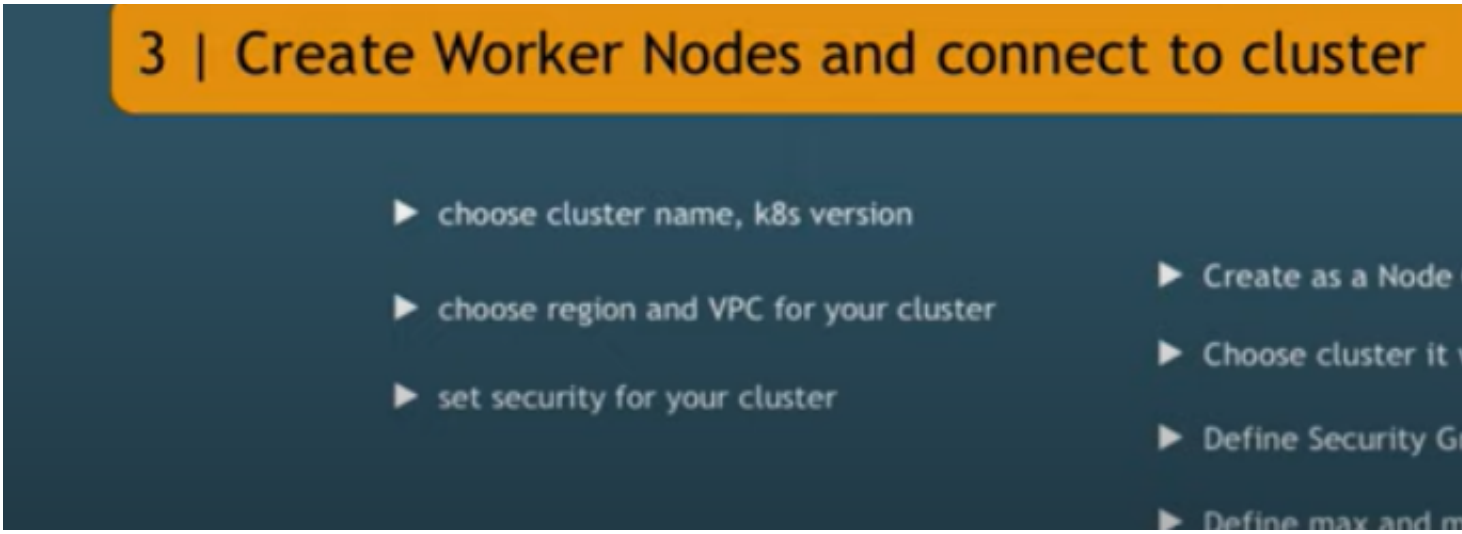


2)
Above steps Summarization:

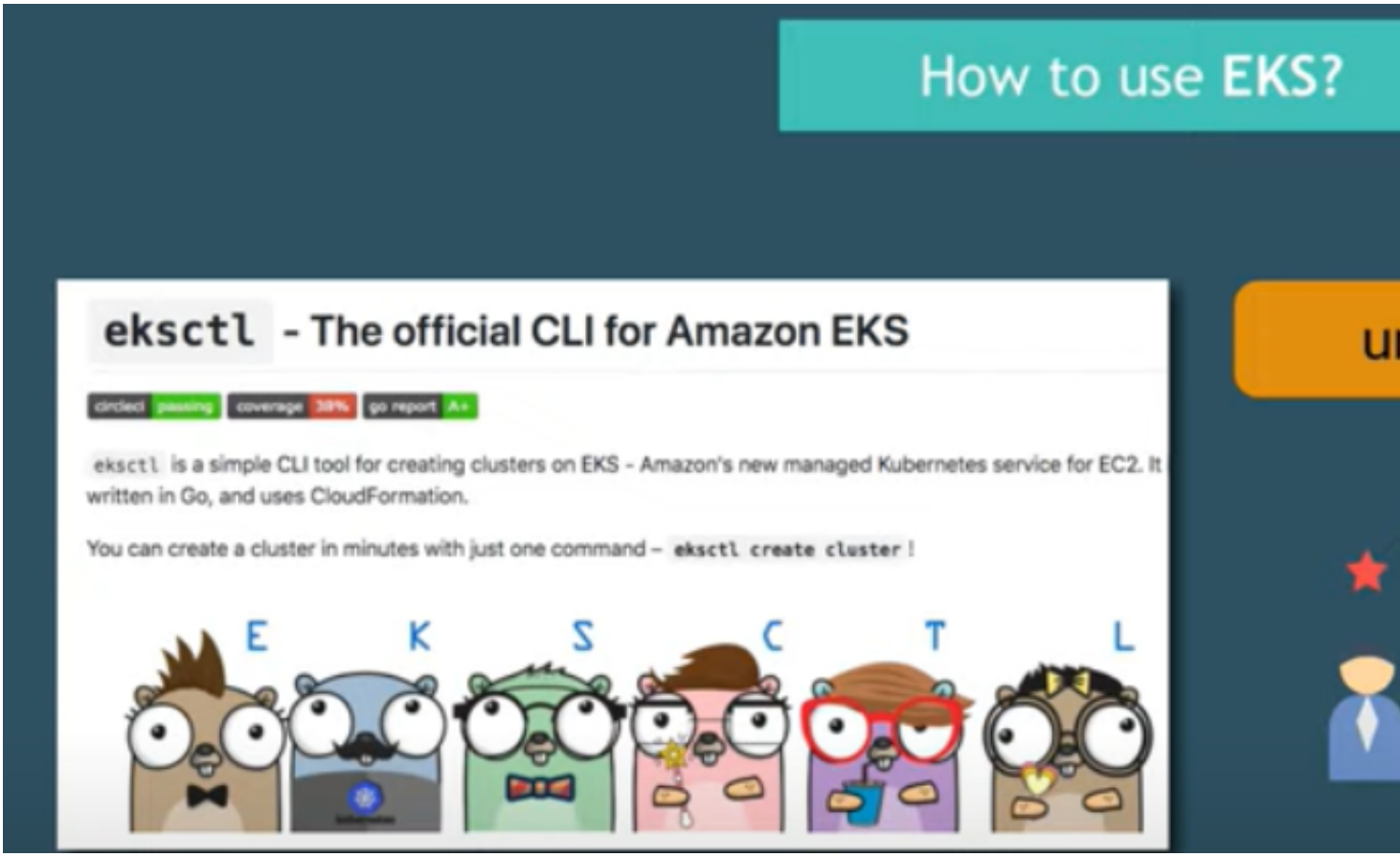


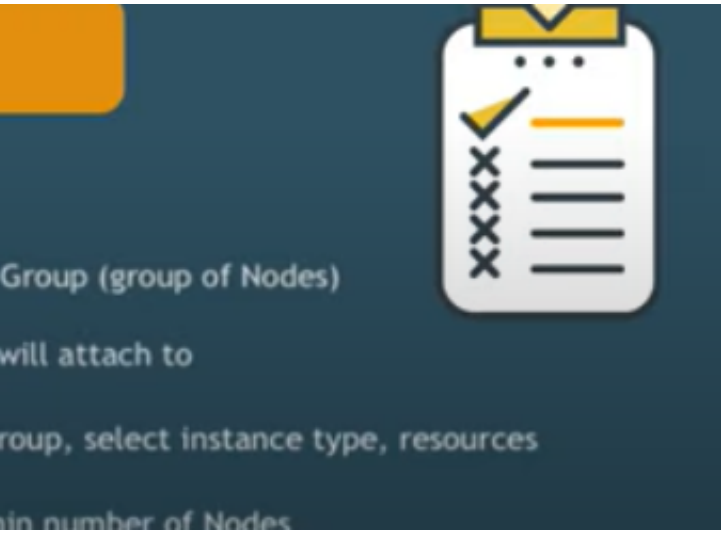

node
EC2 instances
Kubernetes Cluster
node



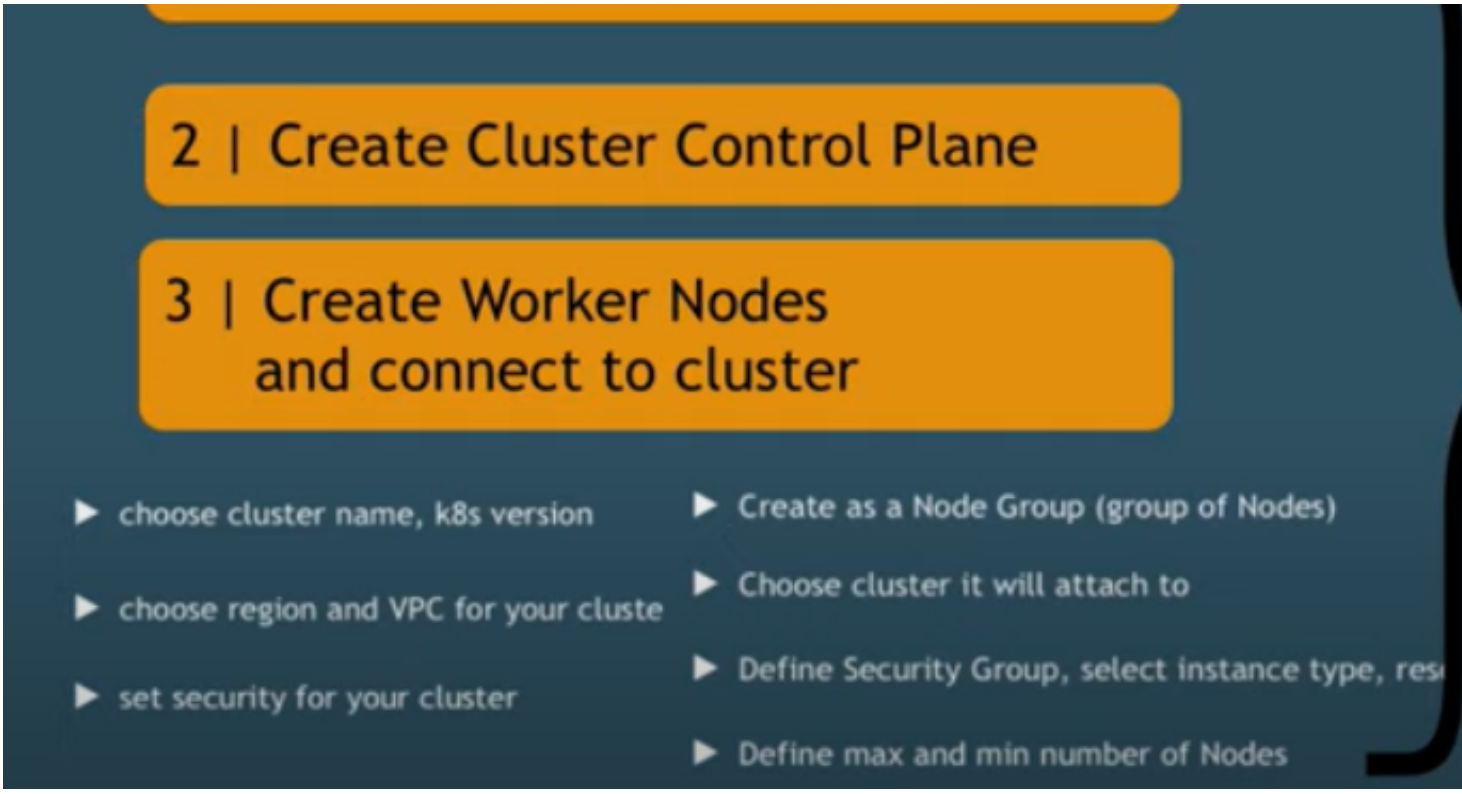


3)
All above mentioned steps seems so much hectic as we need to handle both master and workers and so many other configuration things. But below tool is our savior.




Group (group of Nodes)
will attach to
roup, select instance type, resources


nderstanding the concepts



3.1)

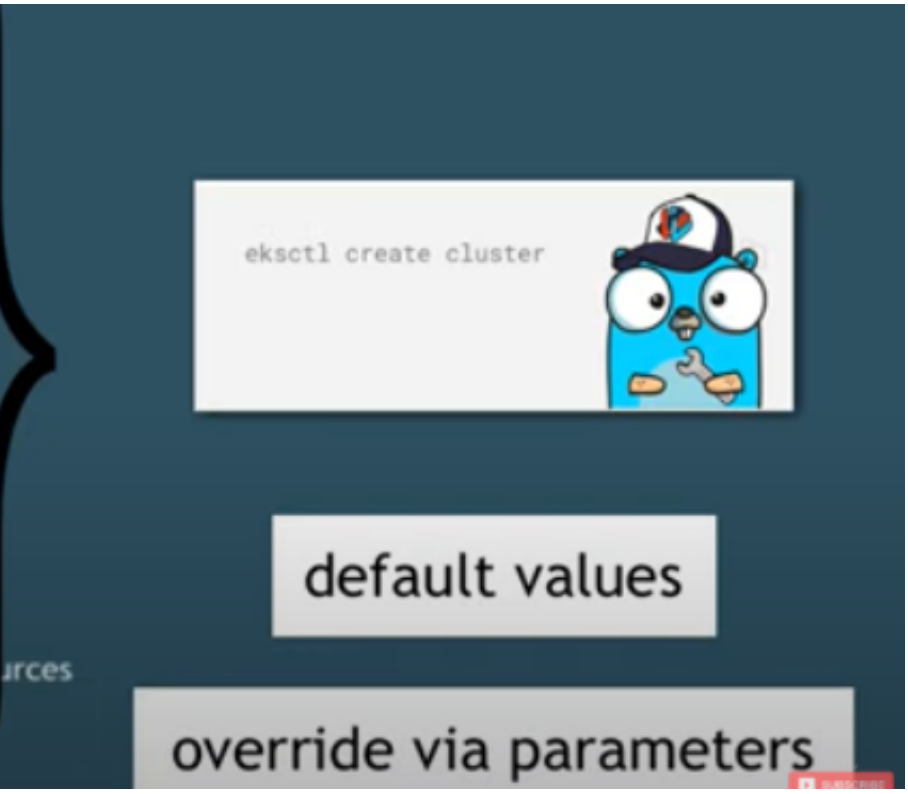
```
[~]$ brew tap weaveworks/tap
Updating Homebrew...
```

```
[~]$ brew install weaveworks/tap/eksctl
```

```
[~]$ eksctl version
0.26.0
```

! eksctl needs to authenticate with AWS

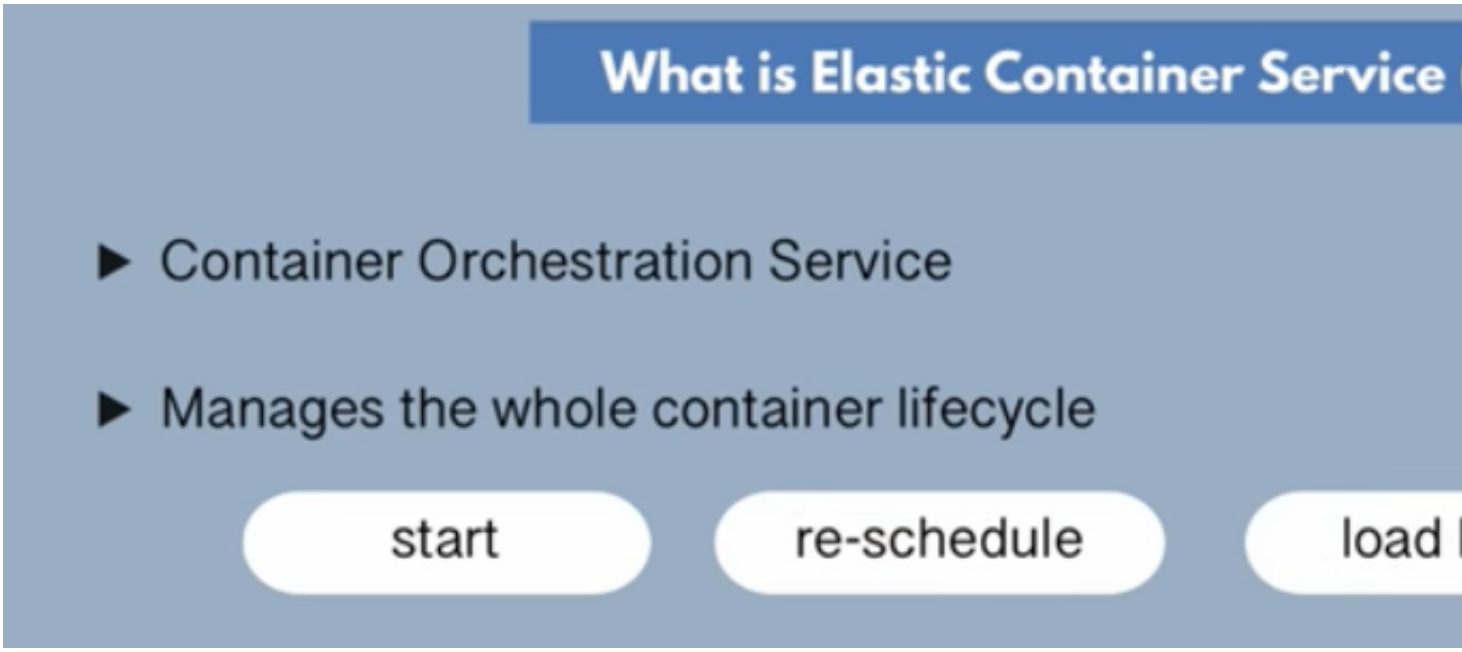
```
[~]$ eksctl create cluster \
> --name test-cluster \
> --version 1.17 \
> --region eu-central-1 \
> --nodegroup-name linux-nodes \
> --node-type t2.micro \
> --nodes 2
[ℹ]  eksctl version 0.26.0
```


eksctl create cluster
default values
override via parameters

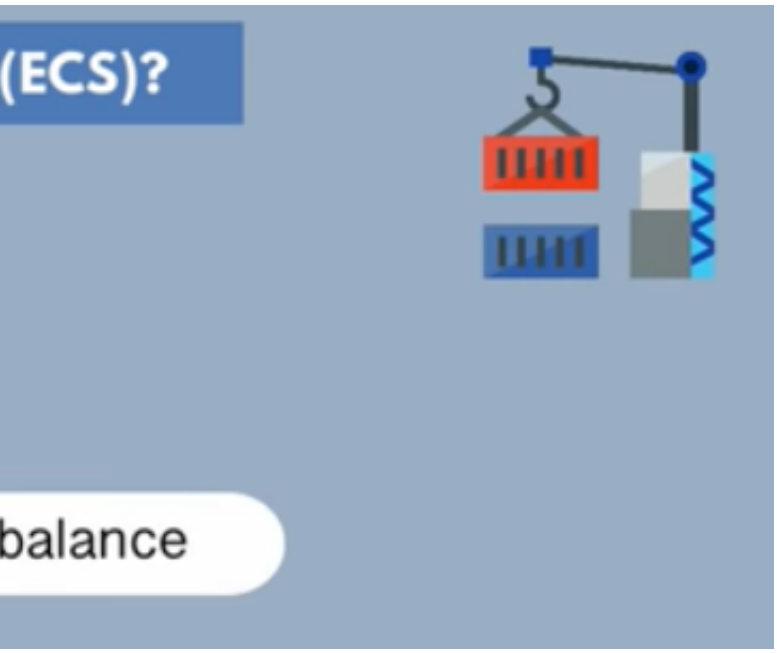
```
i]  waiting for the control plane availability...
✔]  saved kubeconfig as "/Users/nanajanashia/.kube/config"
i]  no tasks
```

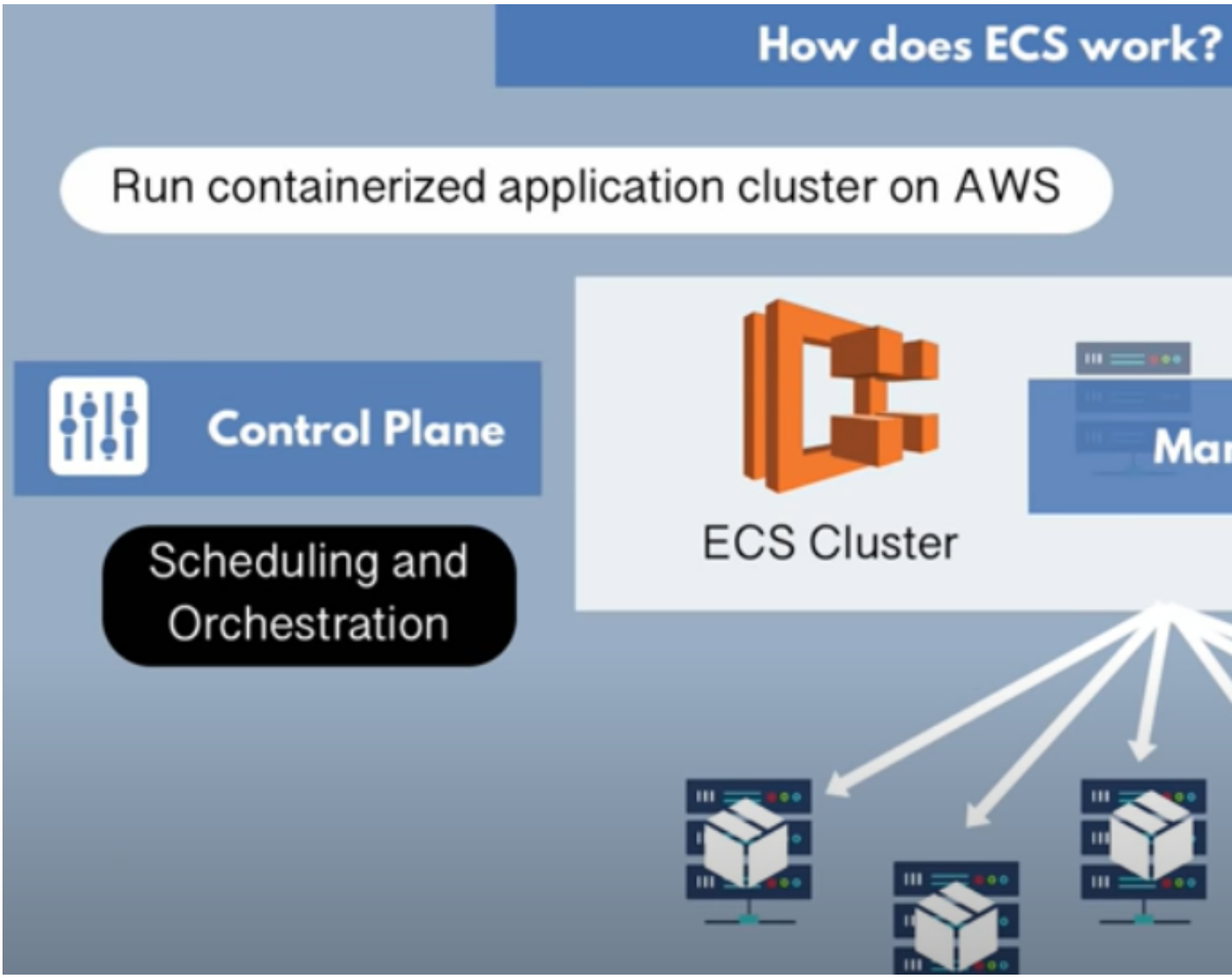
```
~]$ kubectl get nodes
AME                                                STATUS   ROLES
p-192-168-12-111.eu-central-1.compute.internal   Ready    <none
p-192-168-33-61.eu-central-1.compute.internal    Ready    <none
~]$ kubectl get pod
o resources found in default namespace.
~]$ kubectl get ns
AME              STATUS   AGE
efault           Active   10m
ube-node-lease   Active   10m
ube-public       Active   10m
ube-system       Active   10m
~]$ eksctl delete cluster --name test-cluster
i]  eksctl version 0.26.0
i]  using region eu-central-1
i]  deleting EKS cluster "test-cluster"
i]  deleted 0 Fargate profile(s)
✔]  kubeconfig has been updated
i]  cleaning up AWS load balancers created by Kubernetes objec
```

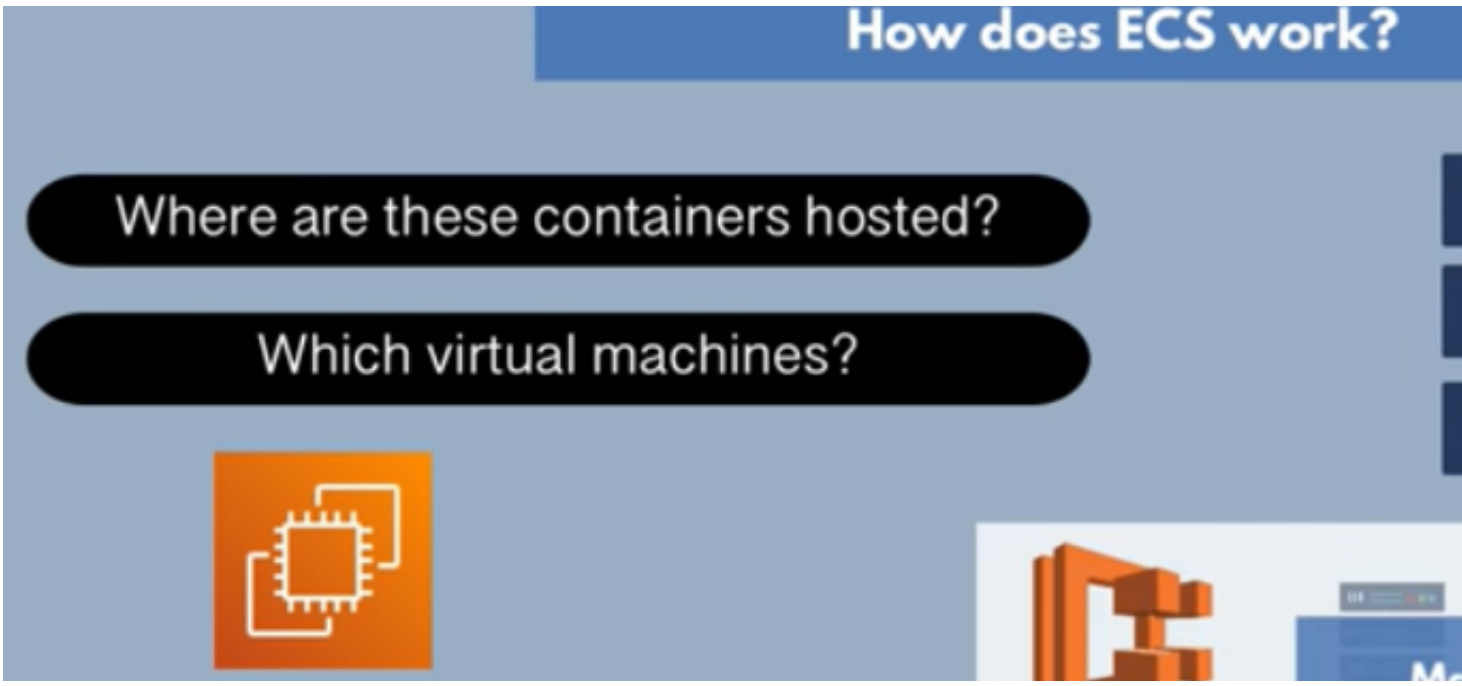
1) =========ECS ==== Equivalent to K8s==========

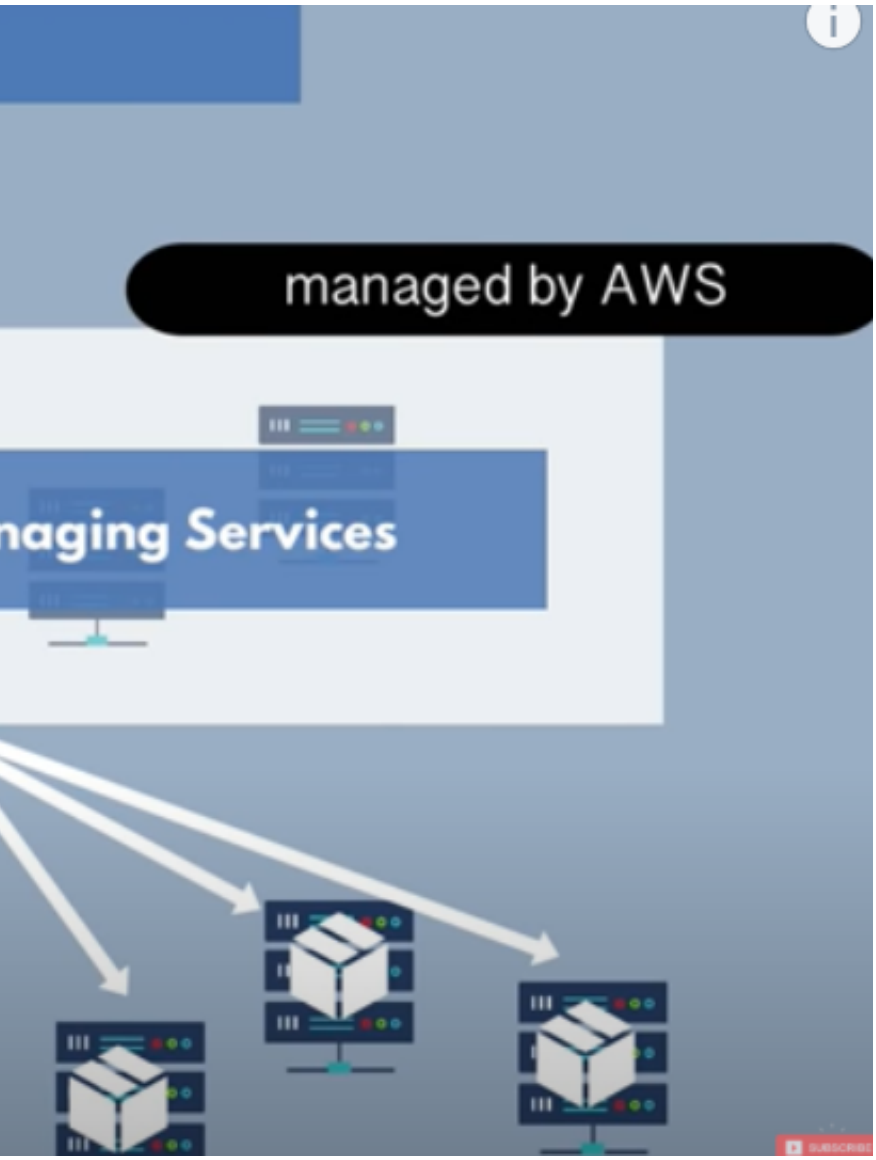


```
S     AGE     VERSION
e>    4m1s    v1.17.9-eks-4c6976
e>    4m1s    v1.17.9-eks-4c6976

ts of Kind Service or Ingress
```



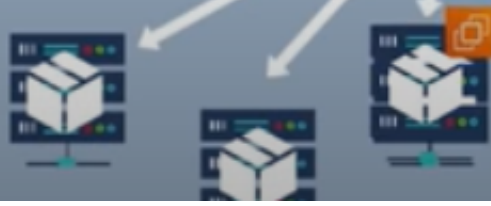

How does ECS work?
Run containerized application cluster on AWS
Control Plane
Scheduling and Orchestration
ECS Cluster


How does ECS work?
Where are these containers hosted?
Which virtual machines?


managed by AWS
naging Services

EC2 Instances

ECS Cluster



Which Services are running on your EC2 Instance

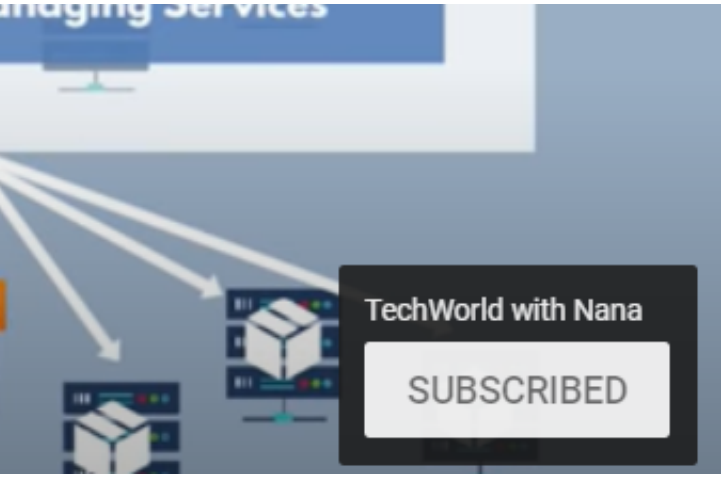
- Container Runtime
- ECS Agent - for Control Plane Communication

- ECS hosted on **EC2 Instances**

✅ manages the **containers**

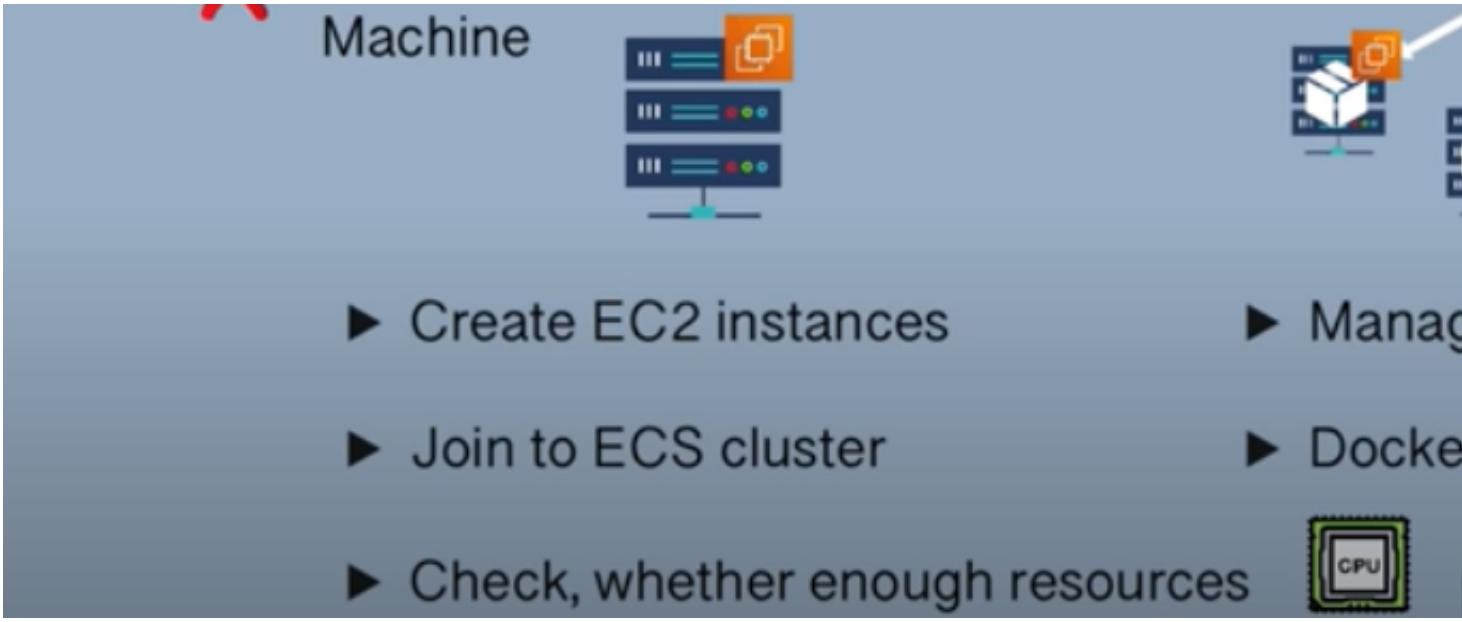
❌ You still need to manage the Virtual

ECS Clust

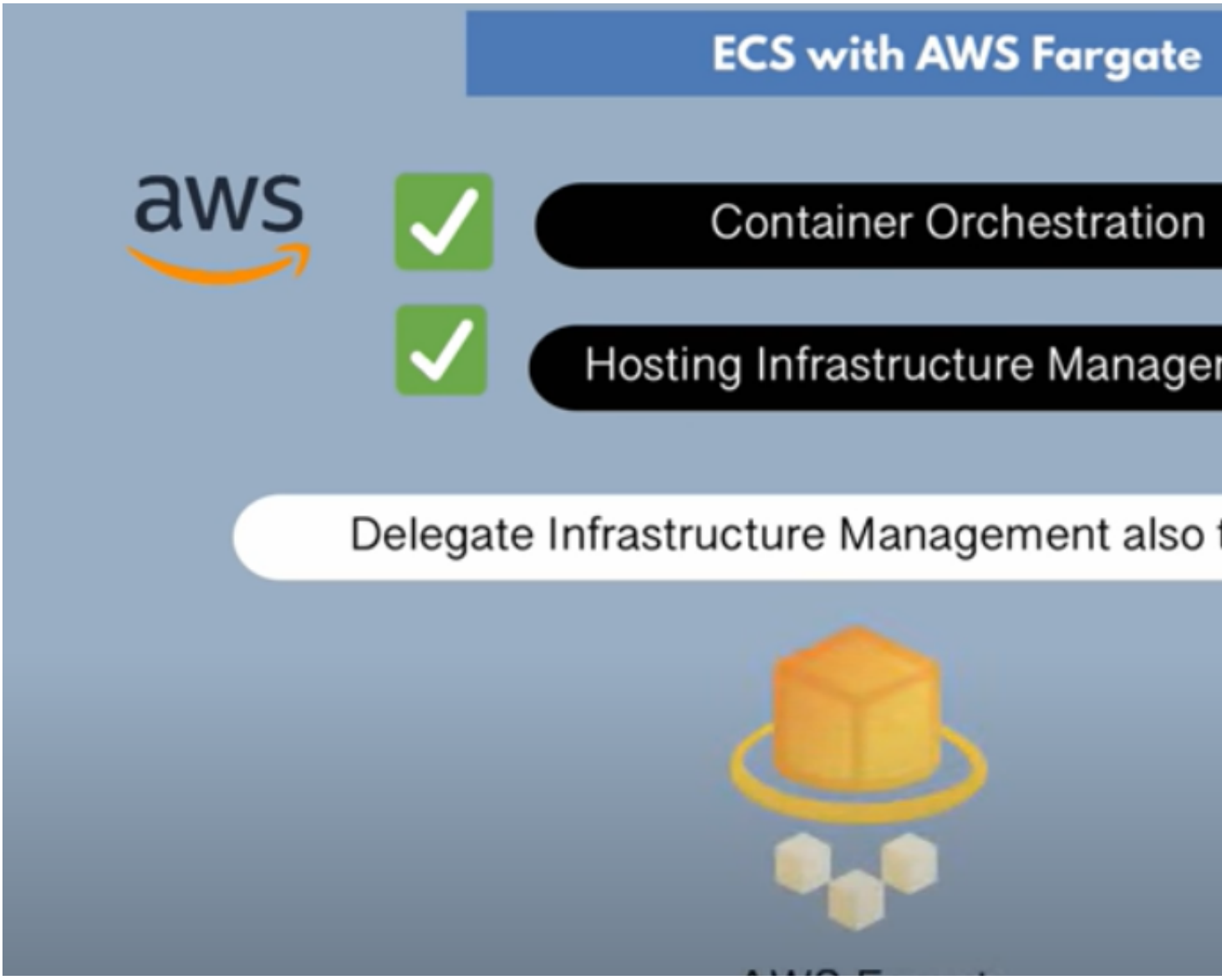

TechWorld with Nana
SUBSCRIBED

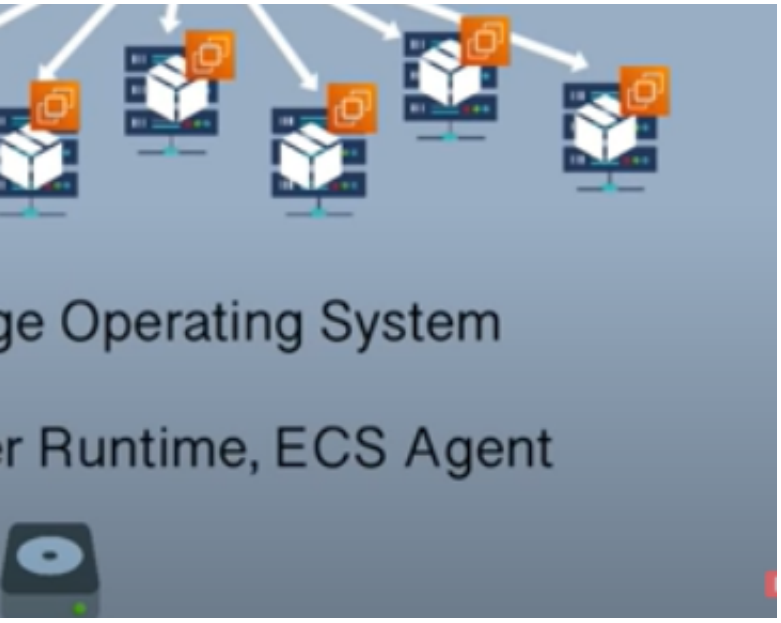

ECS
Agent

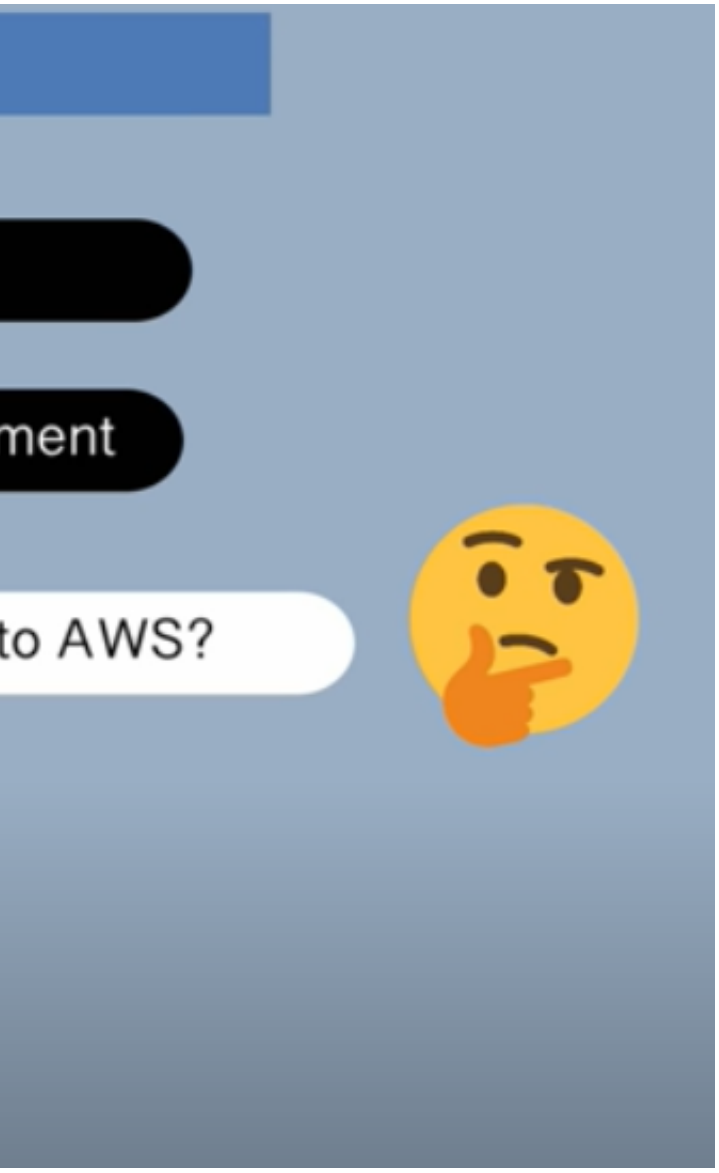

Managing Services



1) ==========AWS with Fargate==========




ge Operating System
r Runtime, ECS Agent


ment
to AWS?

ECS with AWS Fargate

- Alternative to EC2 Instances

How does AWS Fargate wor

- Serverless way to launch containers

"Serverless" meaning:
No Server in your AWS account, Server in AW


rvices


k?
WS account

How does AWS Fargate wo

- Serverless way to launch containers

Launch container with Fargate

ECS Cluster

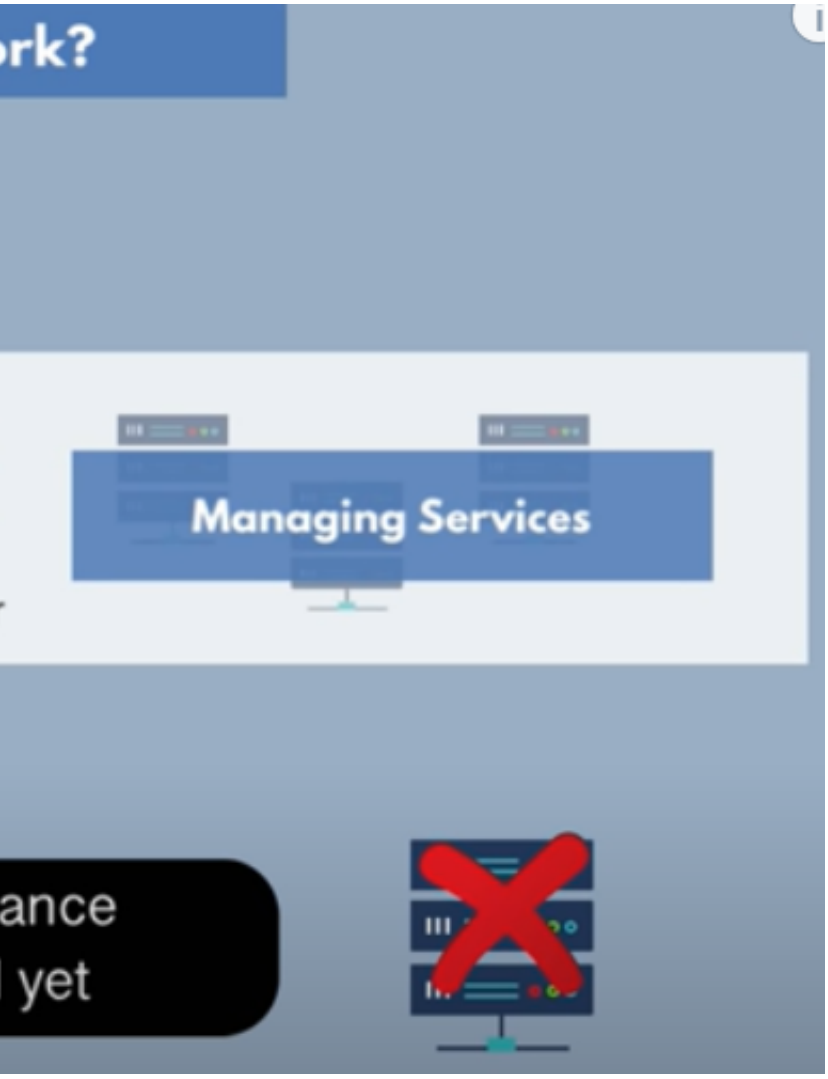
No EC2 Inst
provisioned

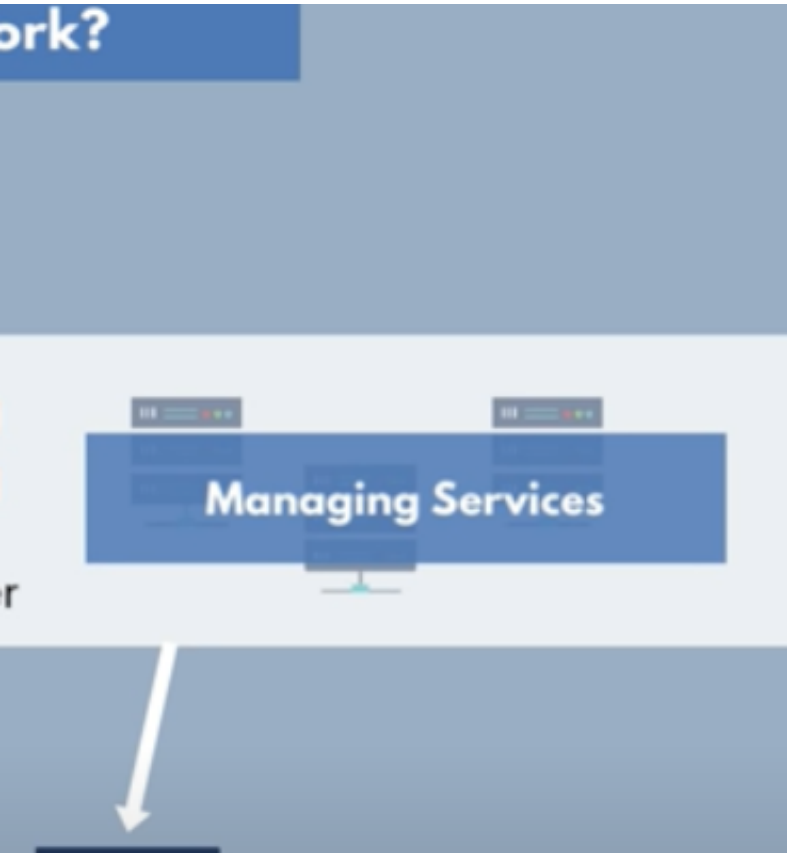
How does AWS Fargate wo

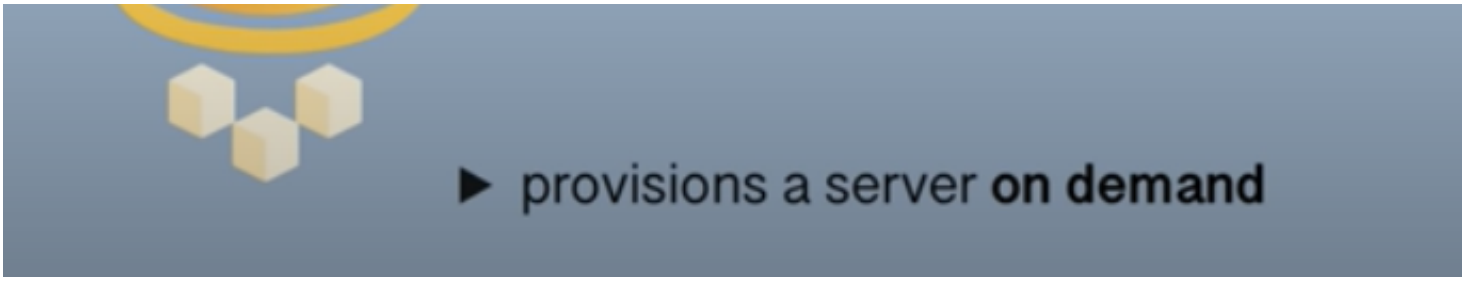
- Serverless way to launch containers

Launch container with Fargate

ECS Cluste


rk?
Managing Services
ance
yet


rk?
Managing Services
r

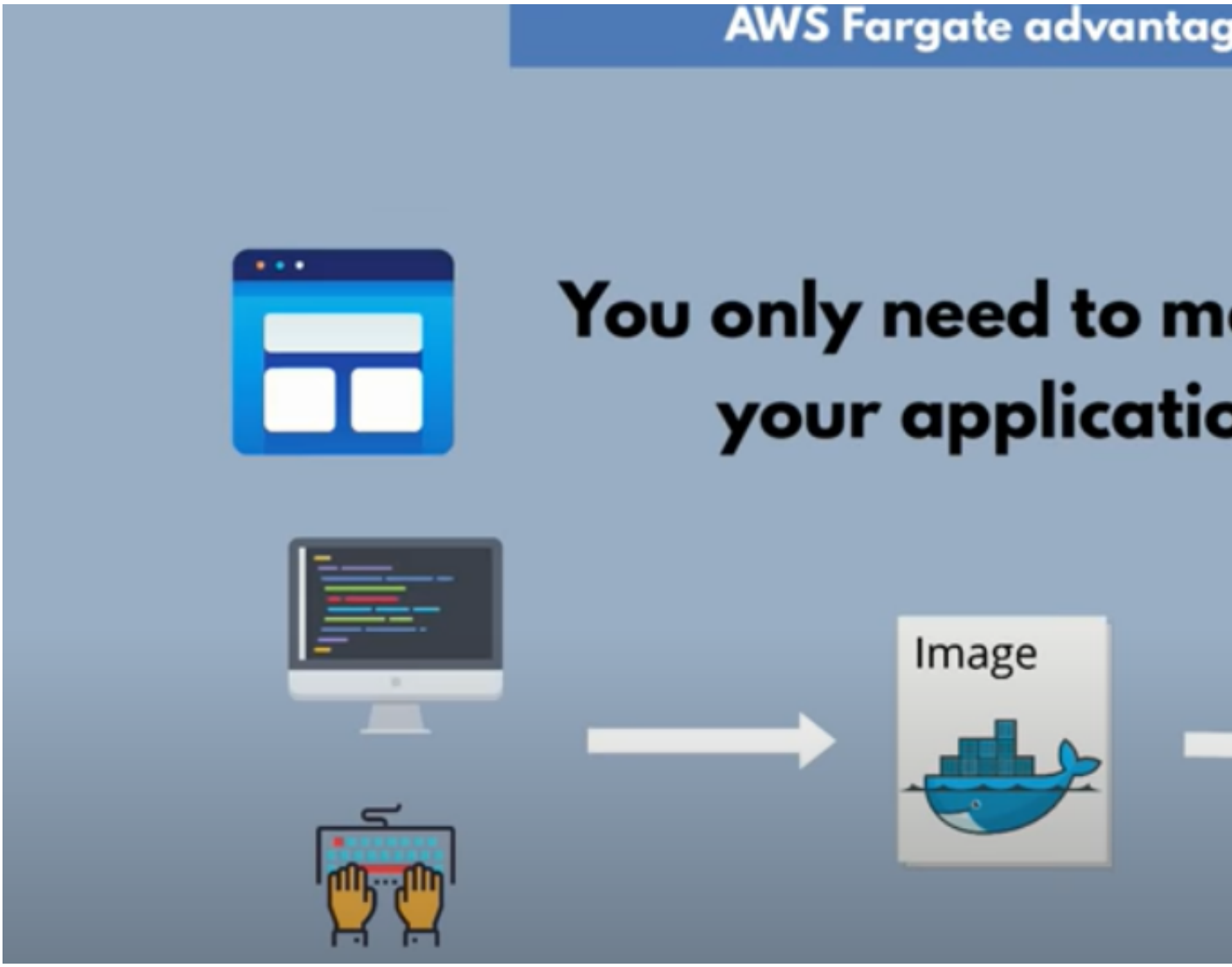

provisions a server **on demand**

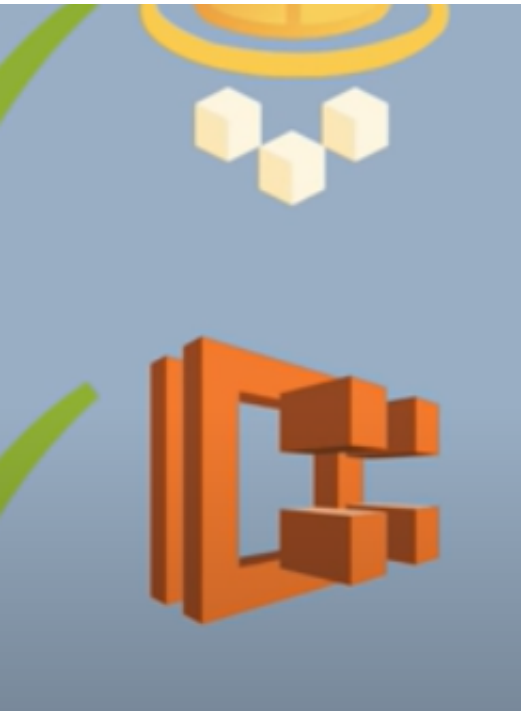

AWS Fargate advantag
ECS with **AWS Fargate**
No need to provision and manage servers
on demand
only the infrastructure resources needed to ru
your containers
pay only for **what you use**
easily **scales up & down without fixed resourc**
defined beforehand


AWS Fargate advantages
Infrastructure


ges
EC2 Instance Pricing
Pay for whole server
un
AWS Fargate Pricing
How long?
How much capacity?
es


managed by AWS
Containers
managed by AWS


AWS Fargate advantag
You only need to m
your applicatio
Image




ges
anage
on
aws

AWS Fargate is great, but..

More control and f
with EC2 Insta

Integration with other AWS Service

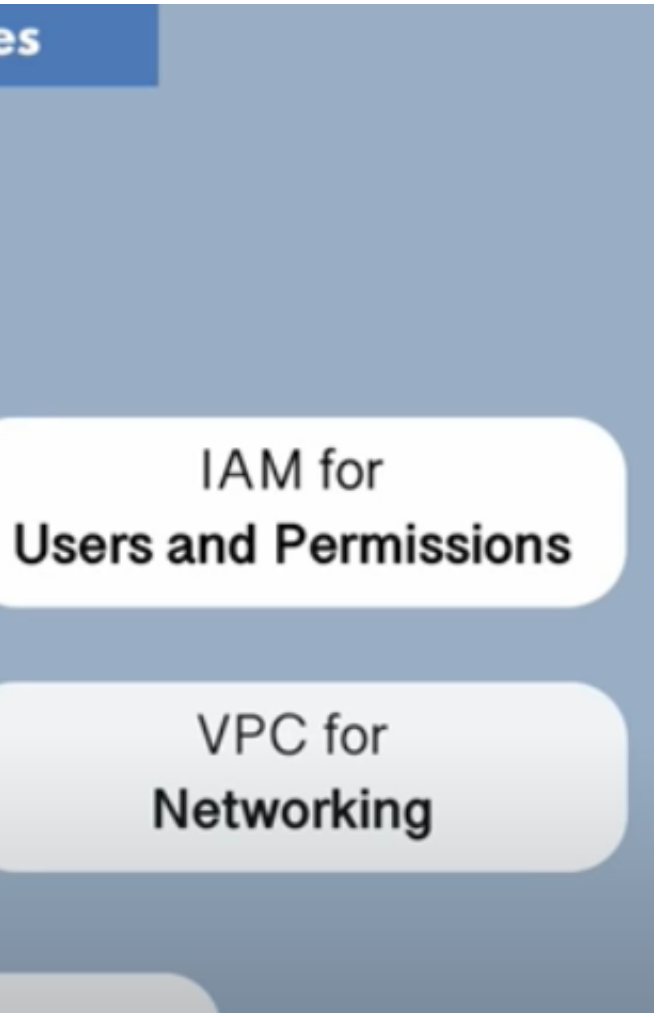
- AWS ecosystem available

aws

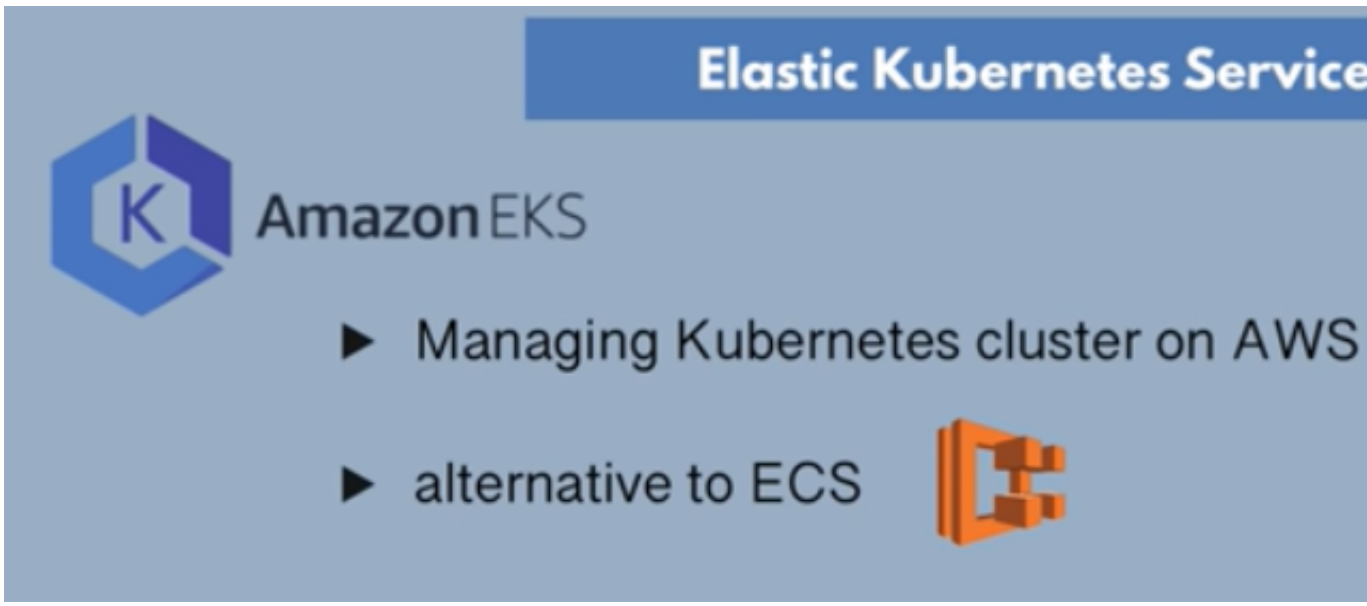
CloudWatch for
Monitoring

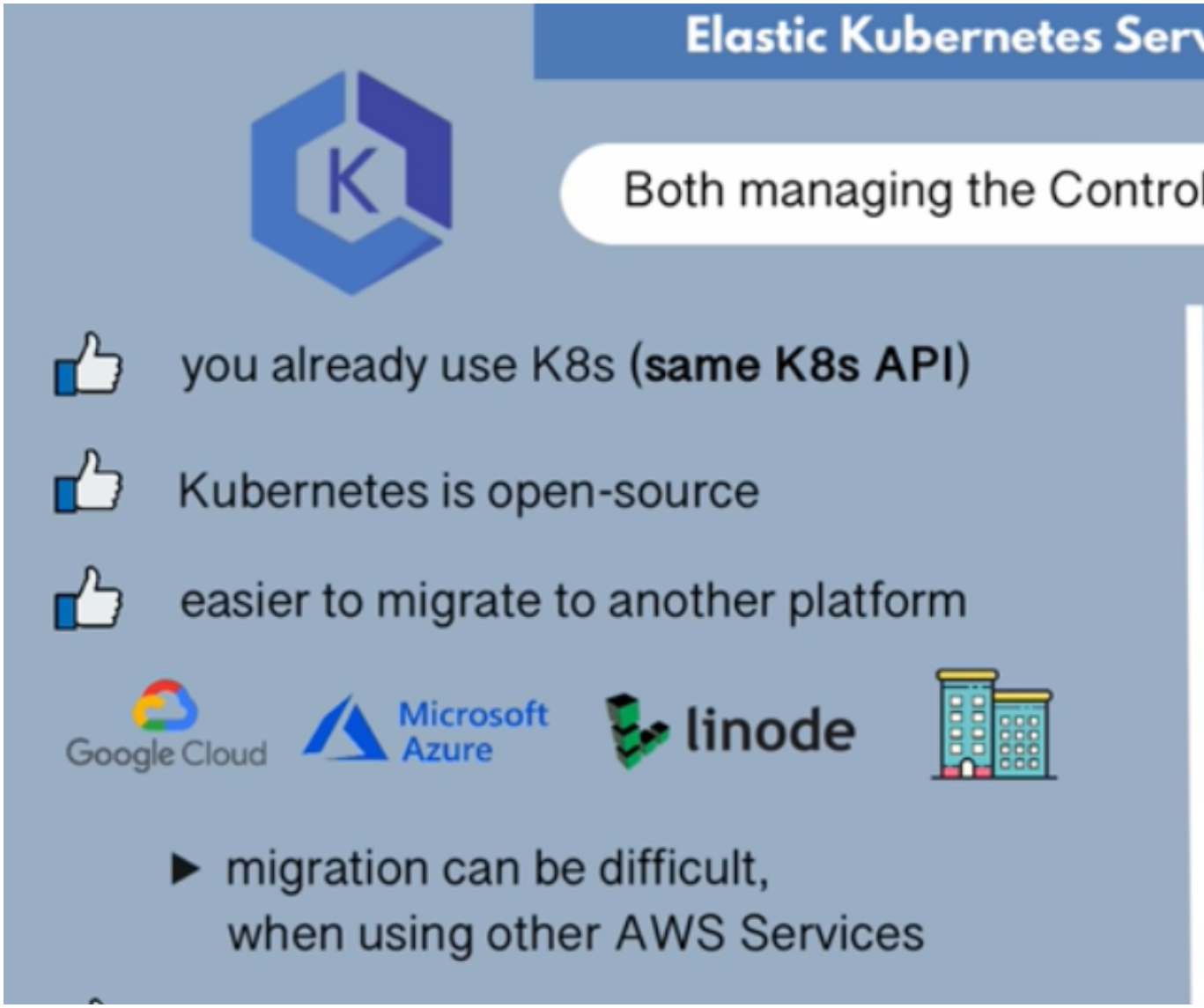
Elastic Load Balancer for
Loadbalancing


lexibility
nces


es
IAM for
Users and Permissions
VPC for
Networking

1) ========== Alternative to ECS === EKS====





(EKS)

infrastructure

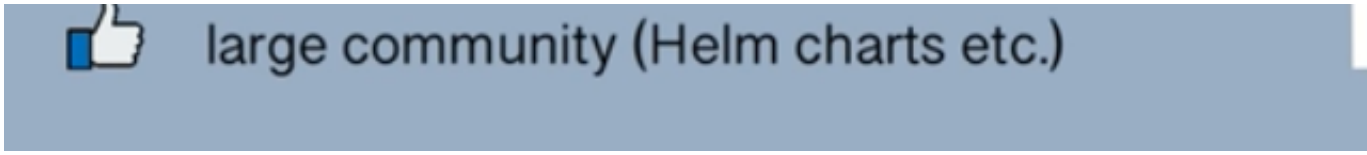
vice (EKS)

l Plane, but:

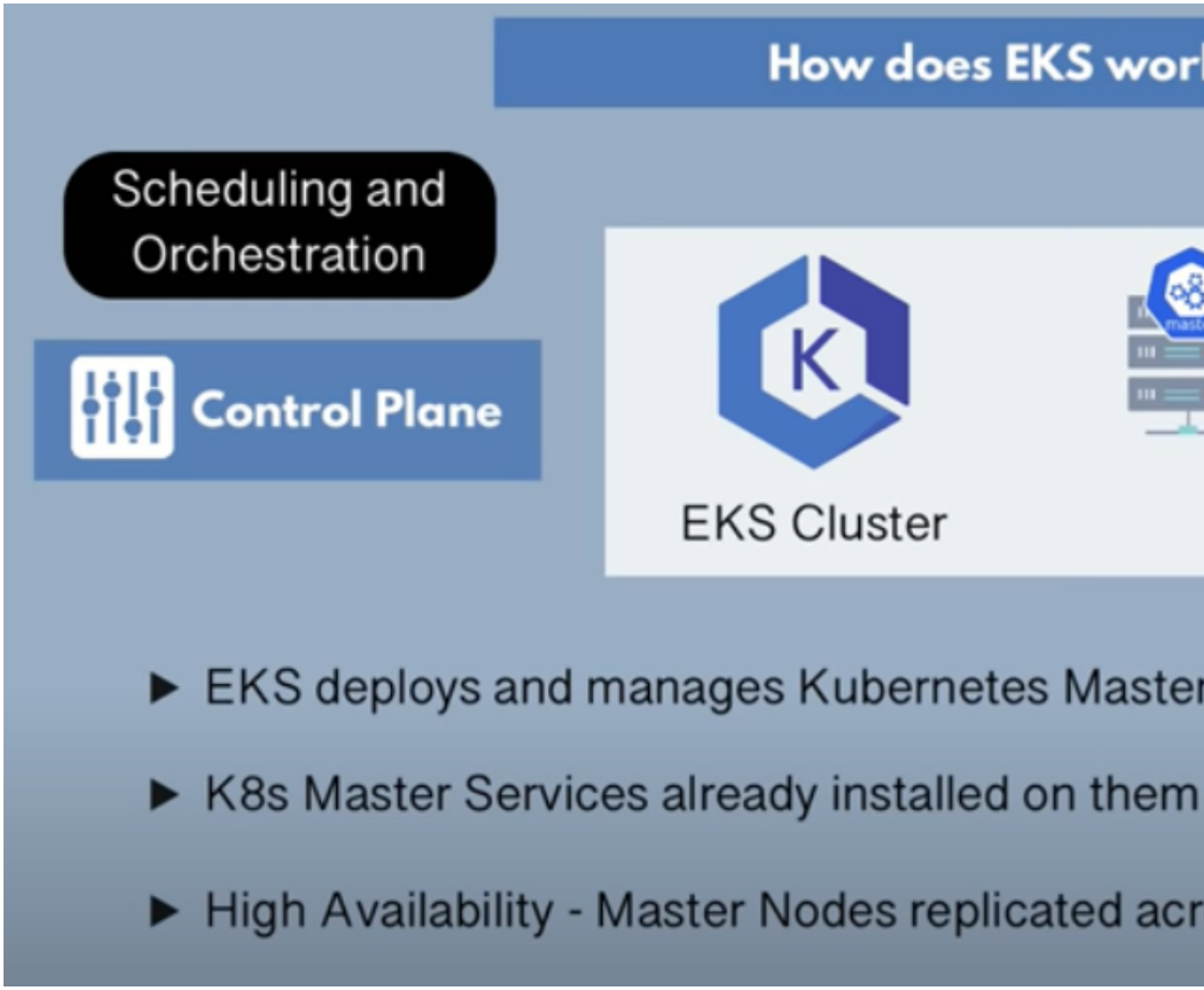
- specific to AWS
- migration difficult

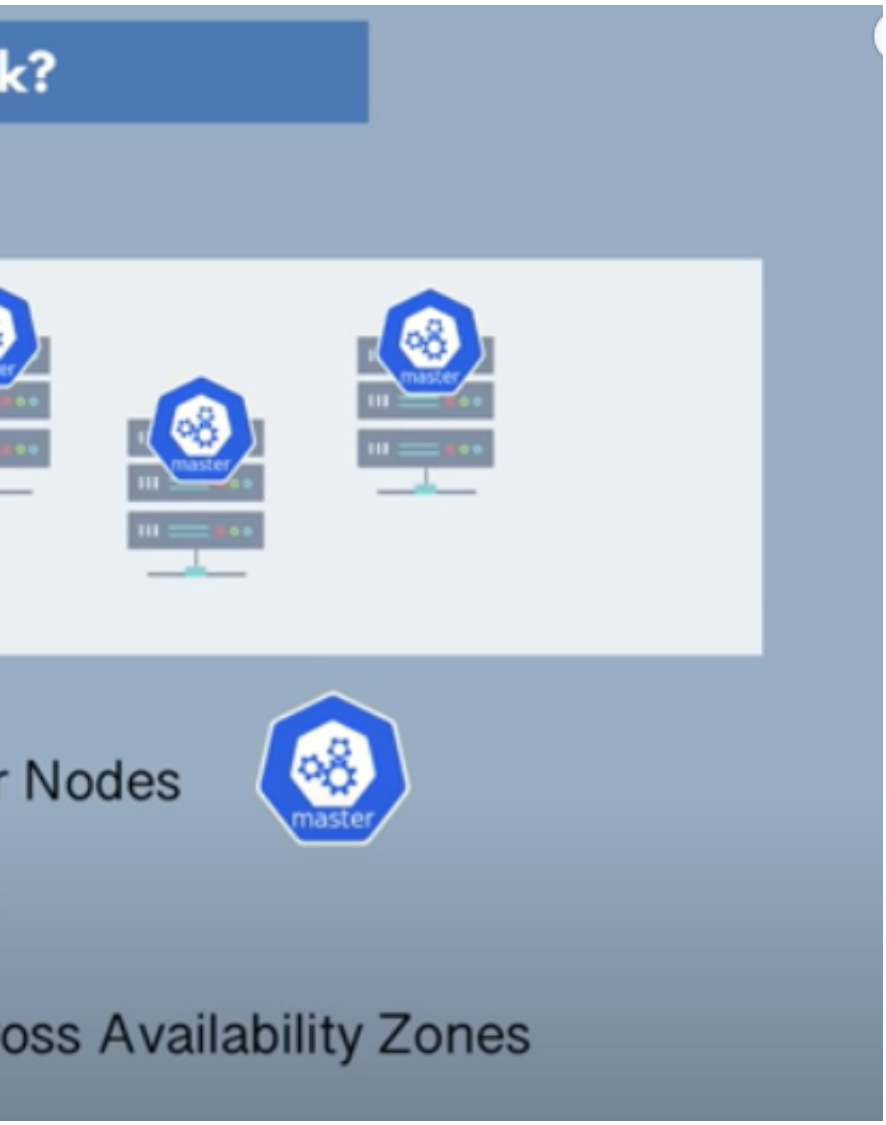
less complex applications

ECS control plane is free

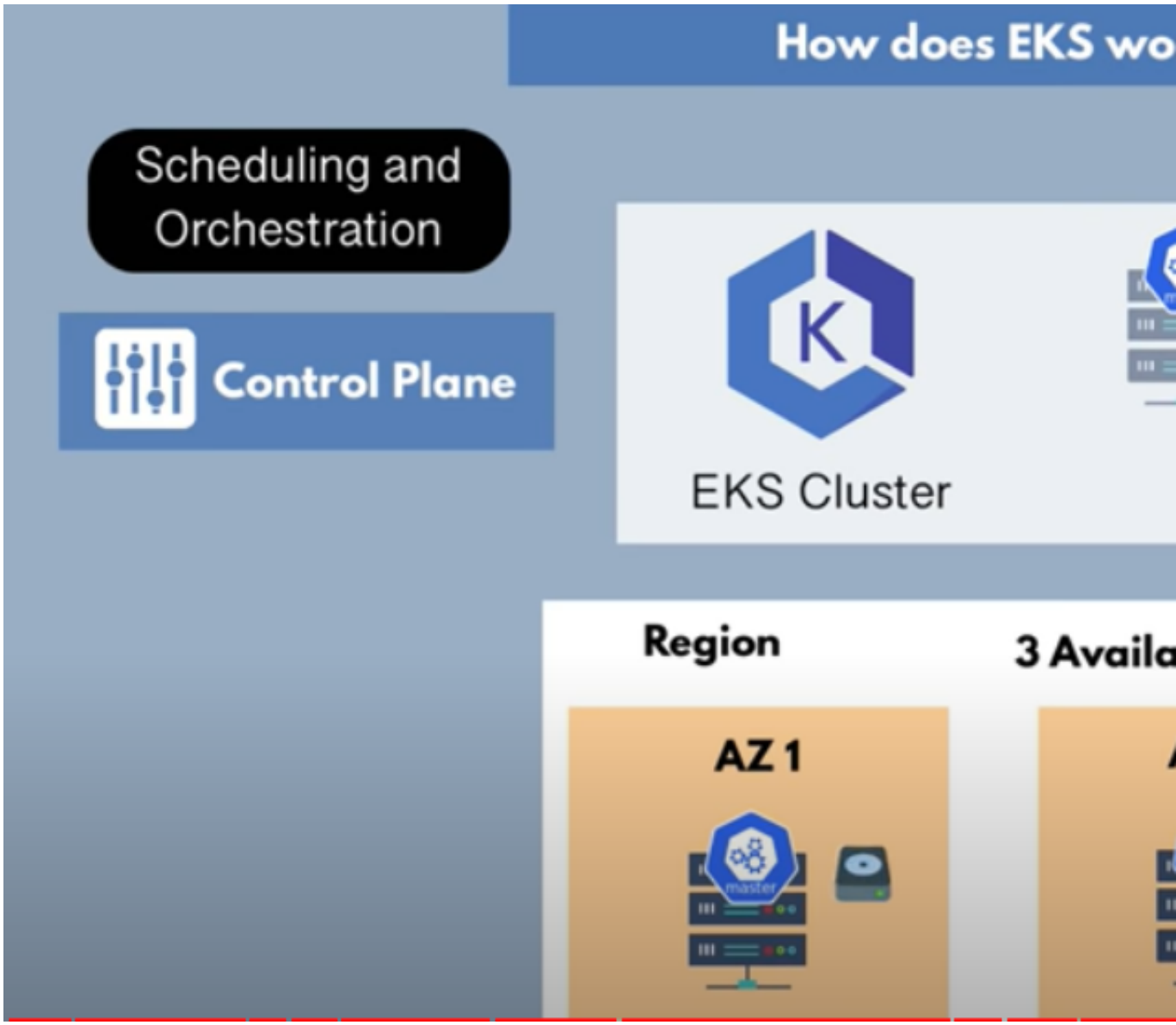


2) ==== Control Plane ==> here we are configuring all MASTER NODES.

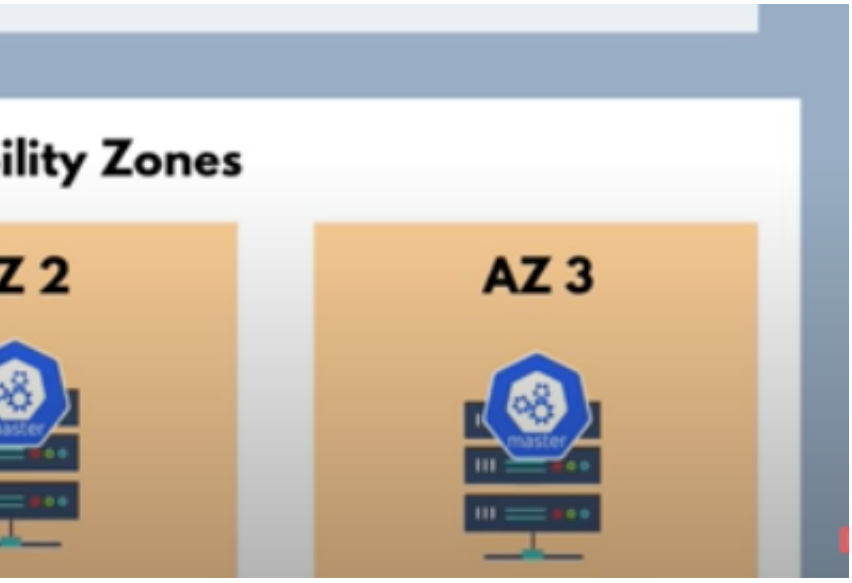


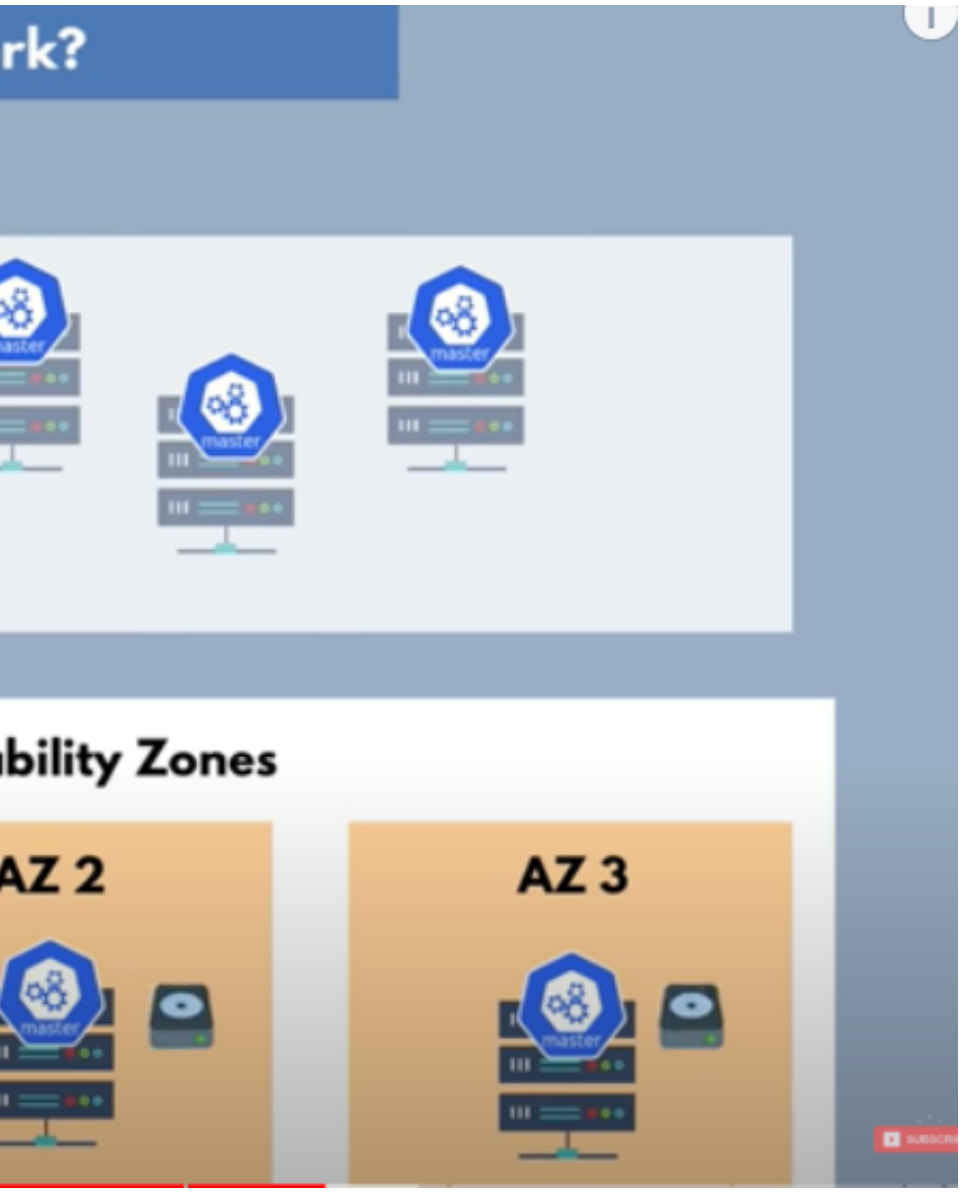

k?
master
master
master
r Nodes
master
oss Availability Zones


k?
master
master
master

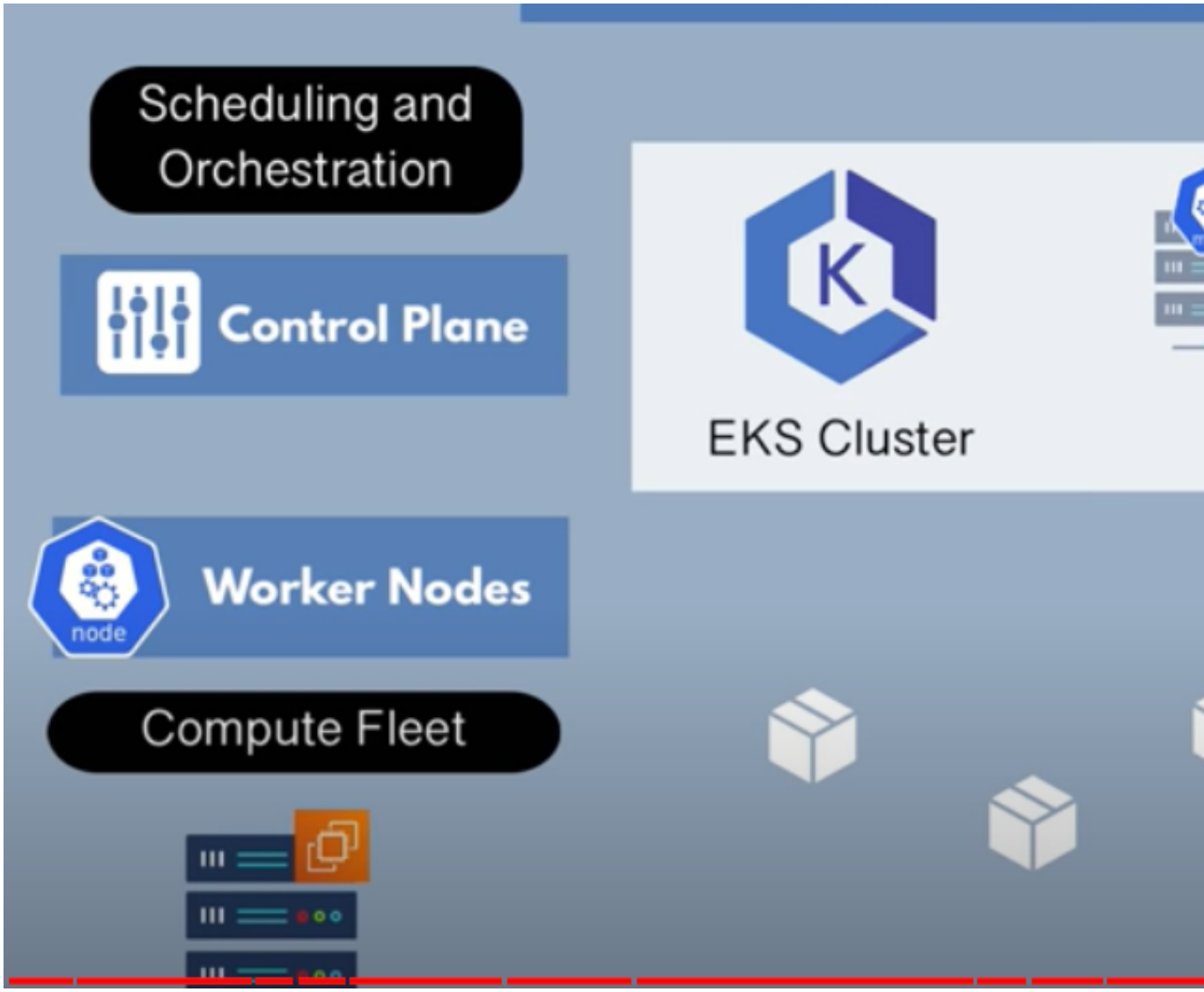


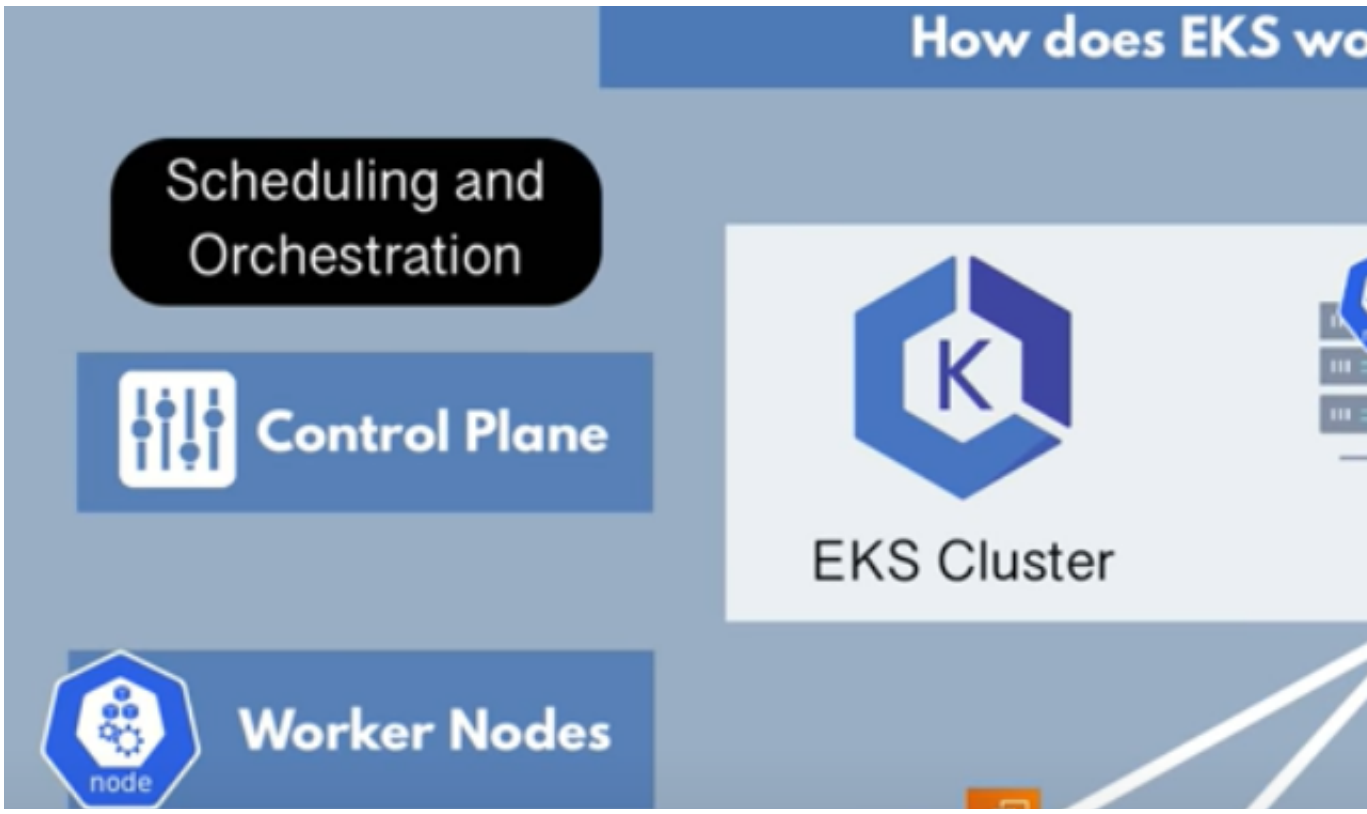
3) Worker Nodes Configuration:: Compute fleet is actually a EC2 Instance.


ility Zones
Z 2
AZ 3
master

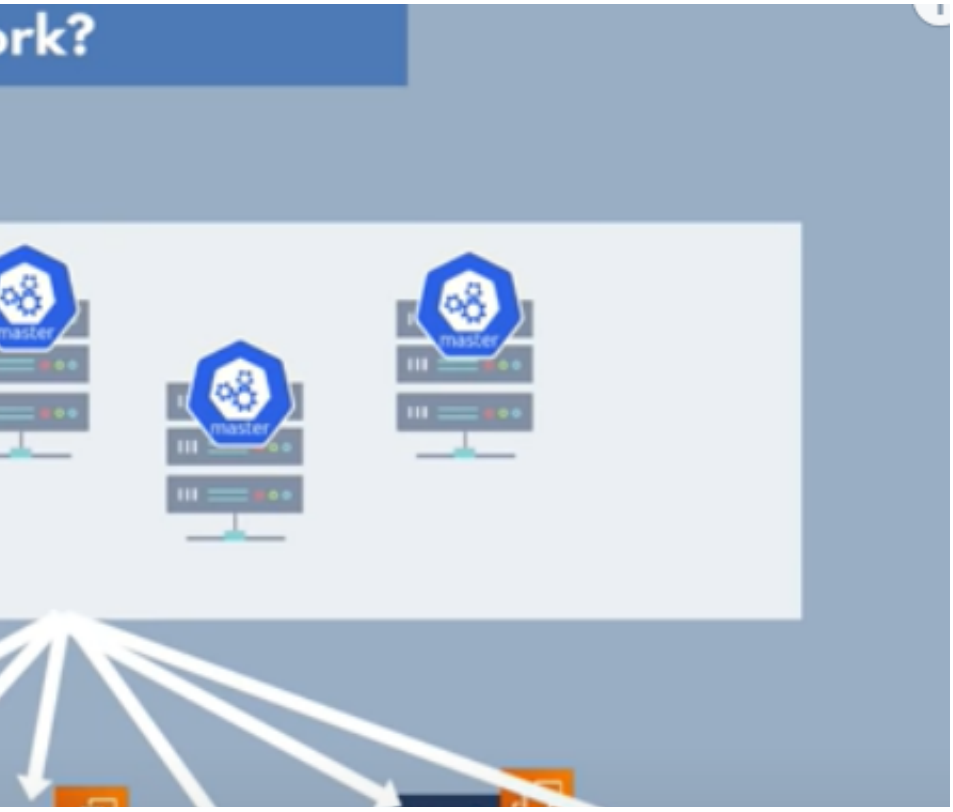

rk?
master
bility Zones
AZ 2
AZ 3
master

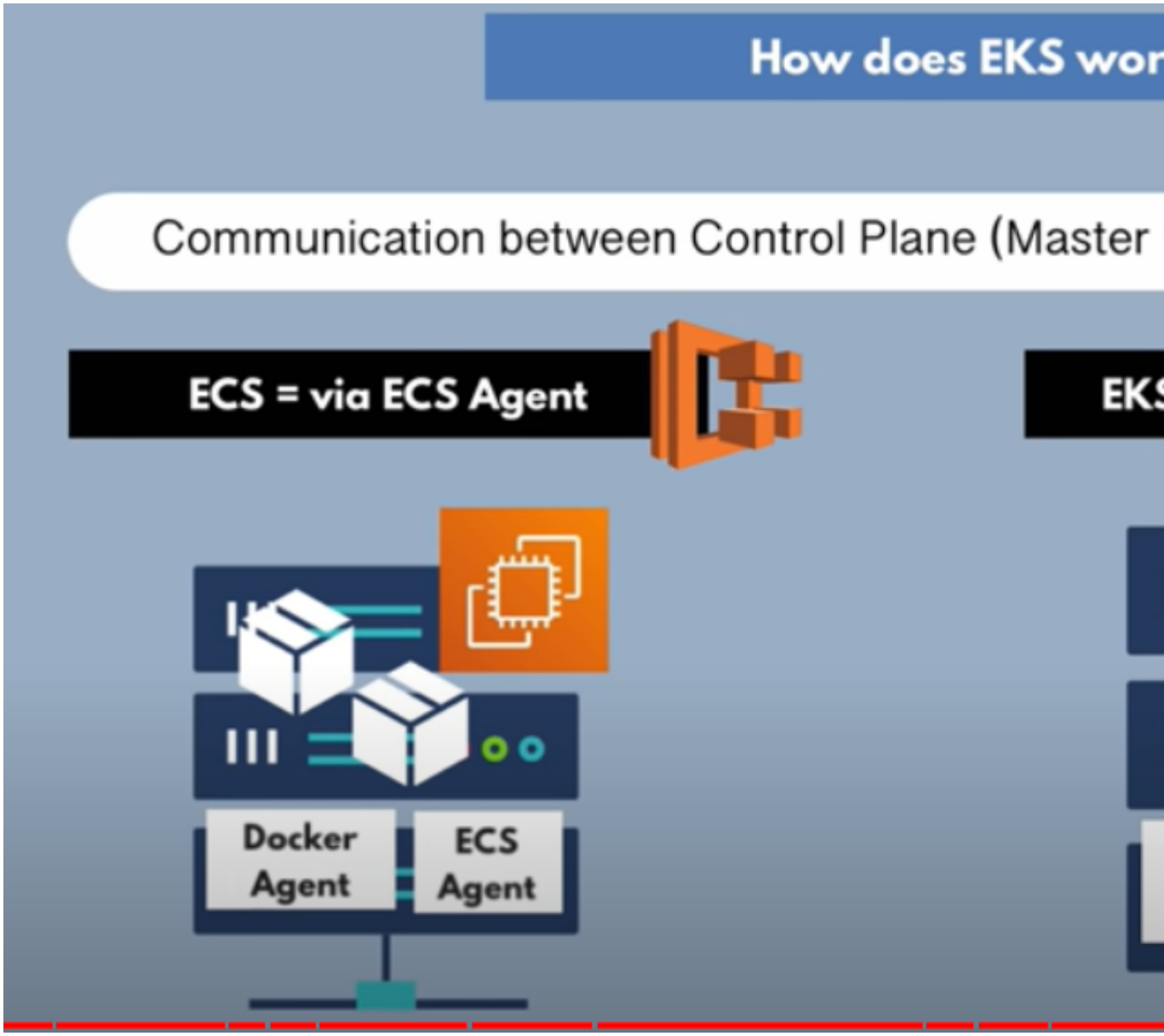

rk?


Scheduling and Orchestration
Control Plane
EKS Cluster
Worker Nodes
node
Compute Fleet

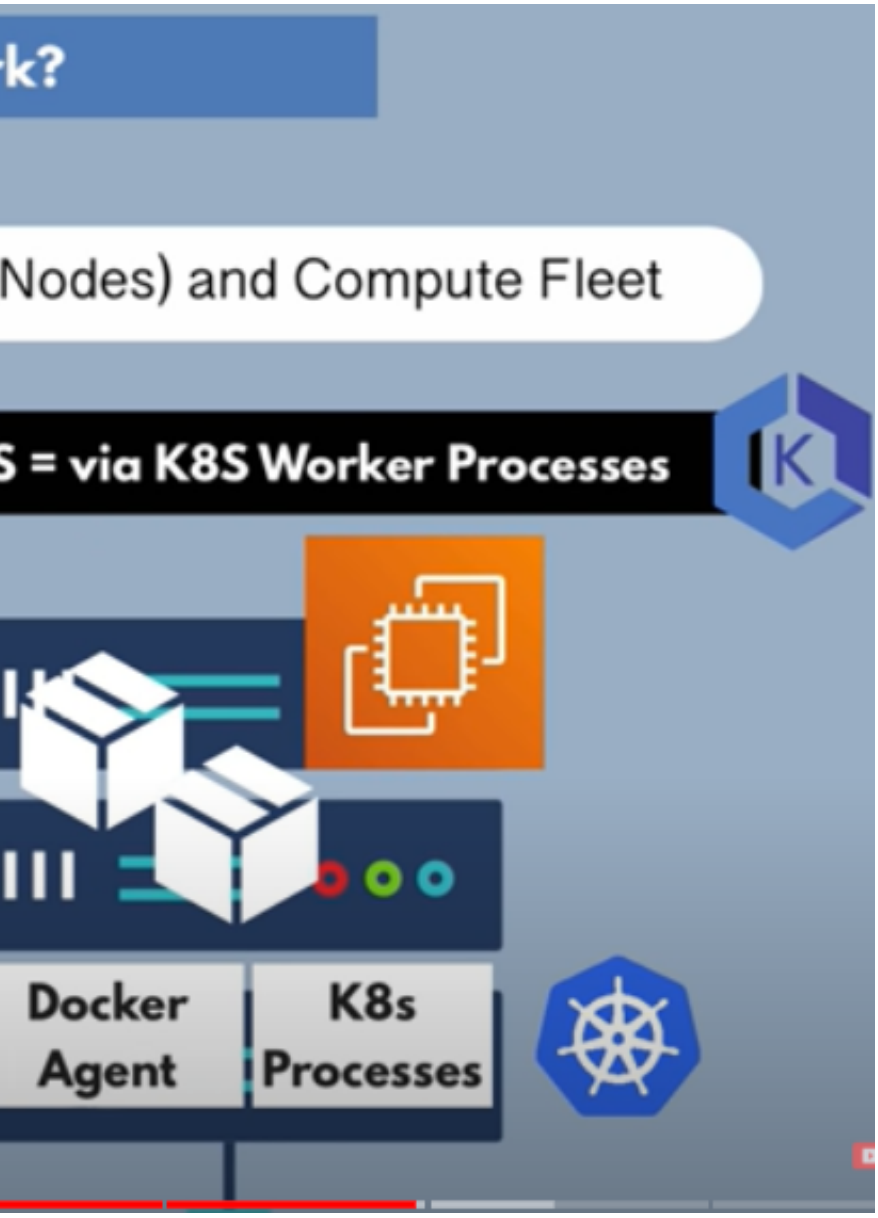

How does EKS wo
Scheduling and Orchestration
Control Plane
EKS Cluster
Worker Nodes
node


master
master
master


rk?
master
master
master


How does EKS wor
Communication between Control Plane (Master
ECS = via ECS Agent
EKS
Docker Agent
ECS Agent


node
Worker Nodes
EC2 vs EC2 Node


k?
Nodes) and Compute Fleet
S = via K8S Worker Processes
Docker Agent
K8s Processes


egroup vs Fargate

EKS with EC2 Instances

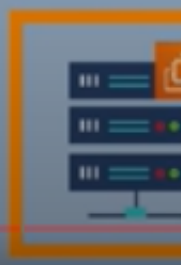
self-managed

- You need to manage the infrastructure for W

EKS with Nodegroup

semi-managed

- Creates, deletes EC2 Instances for you, but you need to configure it



20:55 / 25:09 • Worker Nodes options: EC2 vs Nodegroup vs F

Worker Nodes

EC2 vs EC2 Node

EKS with EC2 Instances

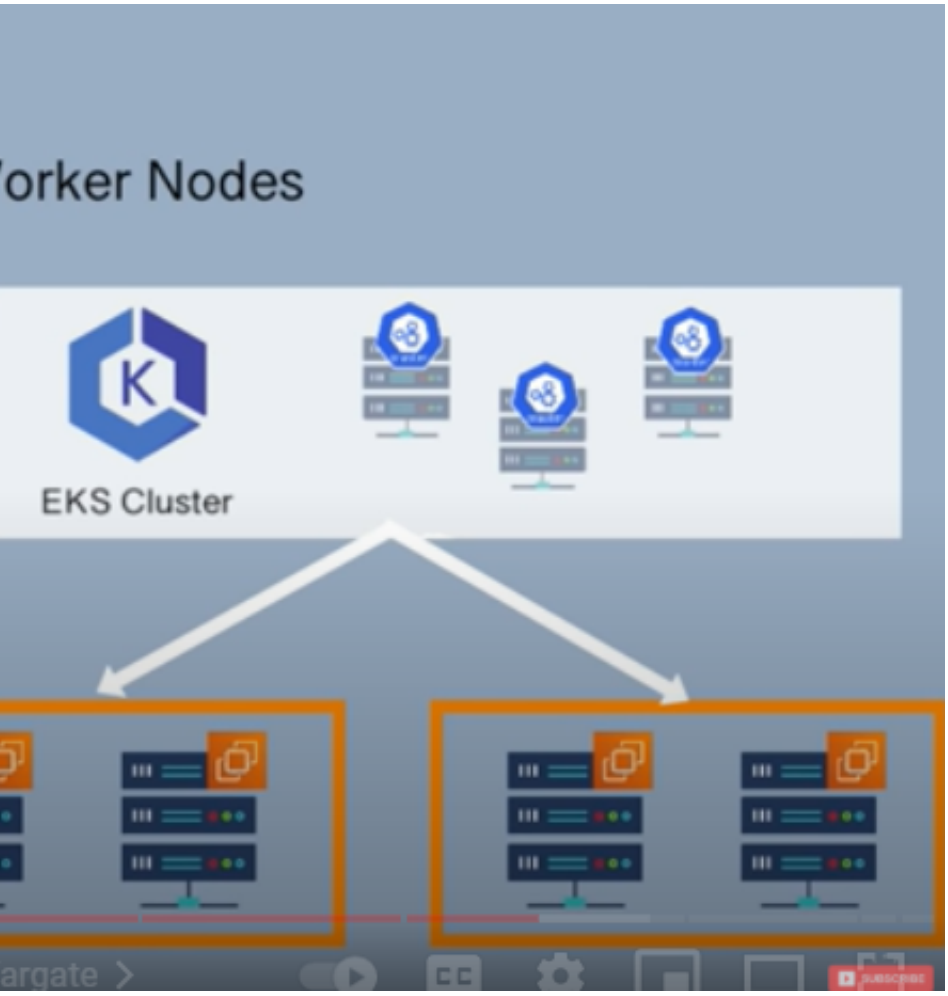
self-managed

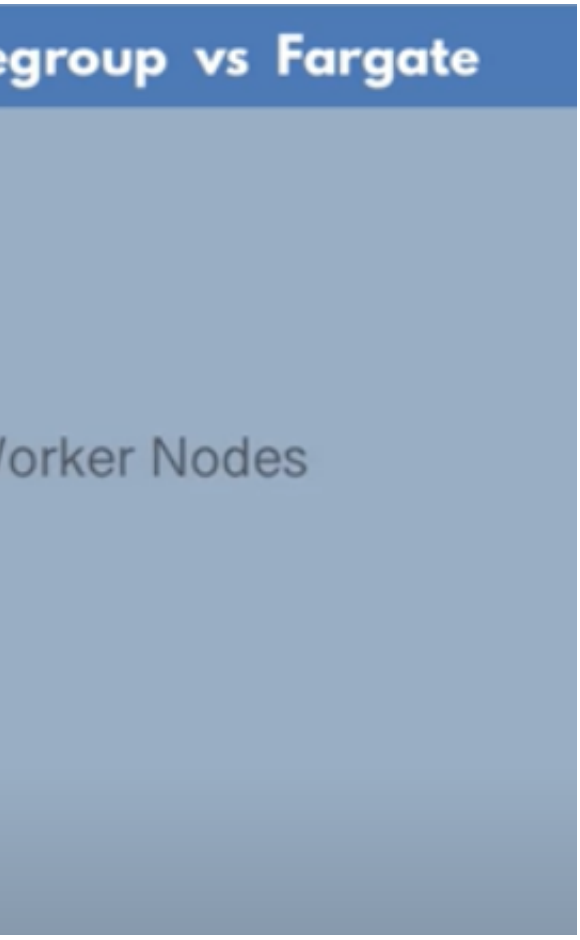
- You need to manage the infrastructure for W

EKS with Nodegroup

semi-managed

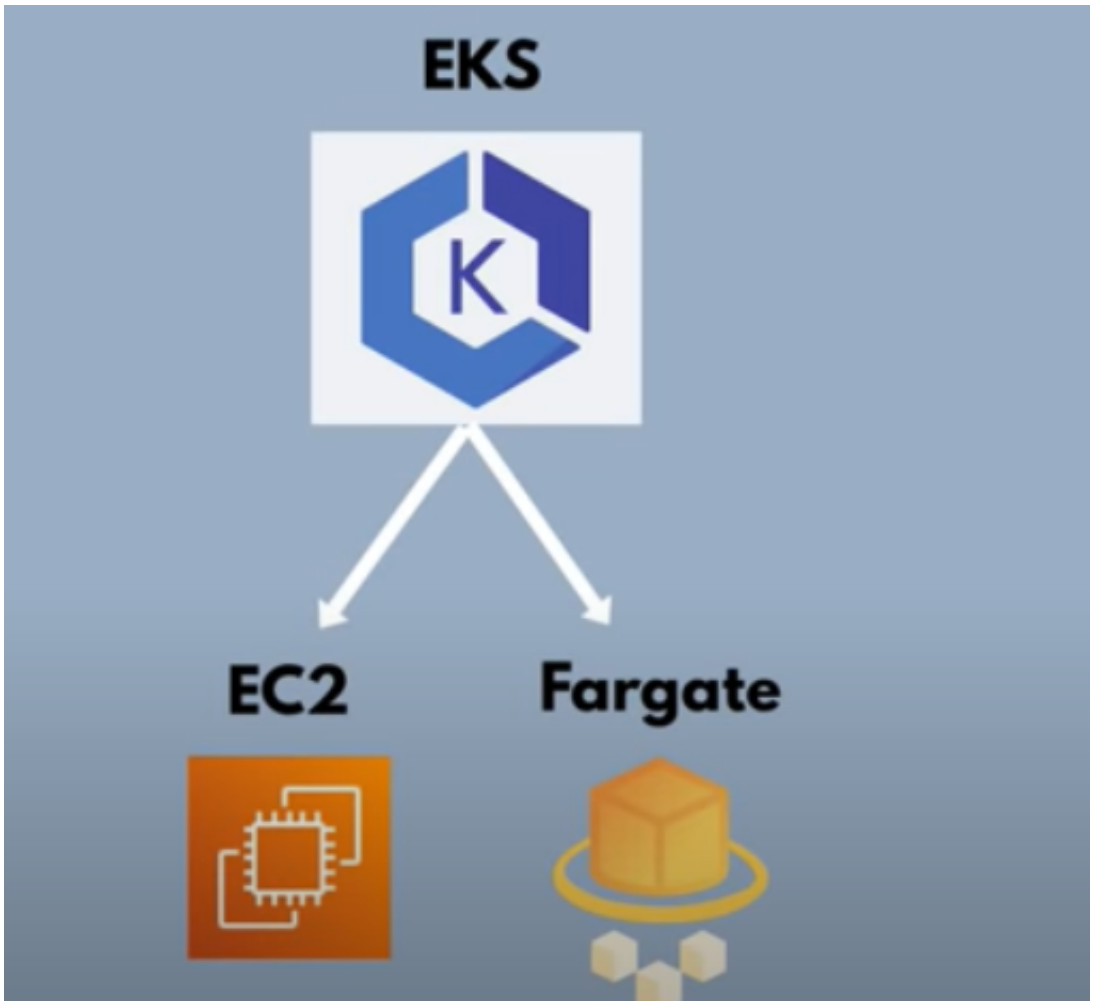
- Creates, deletes EC2 Instances for you, but you need to configure it


orker Nodes
EKS Cluster
argate

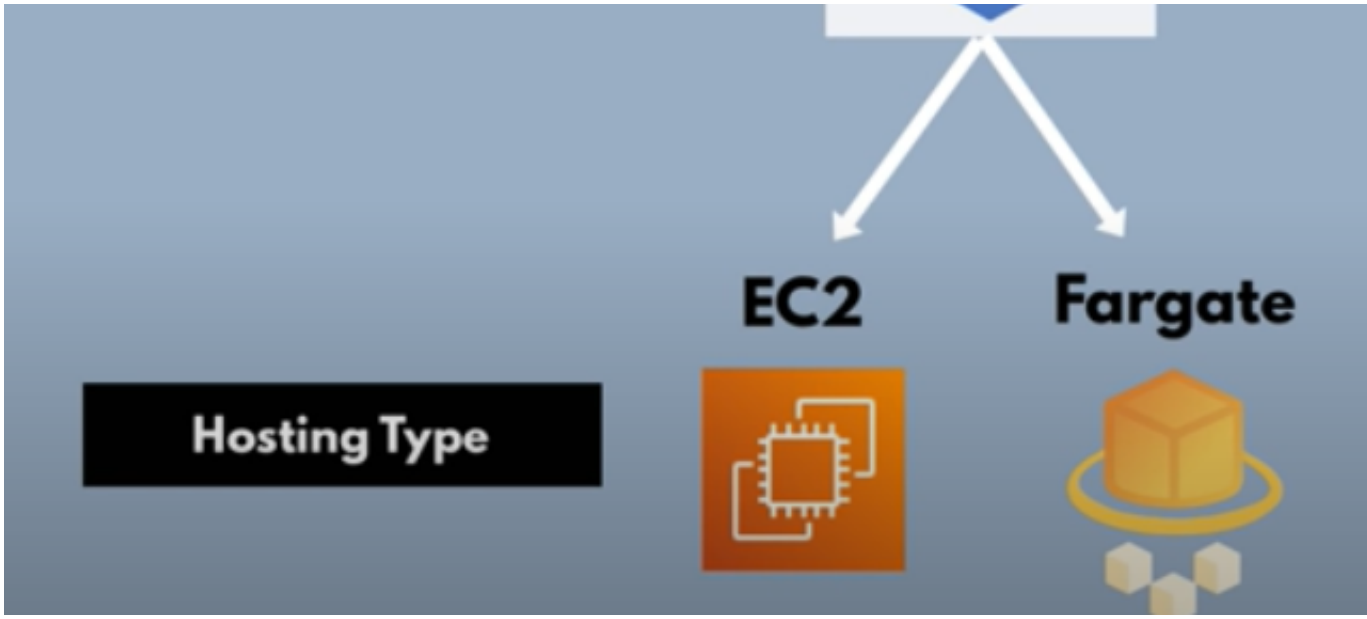

egroup vs Fargate
orker Nodes

EKS with Fargate

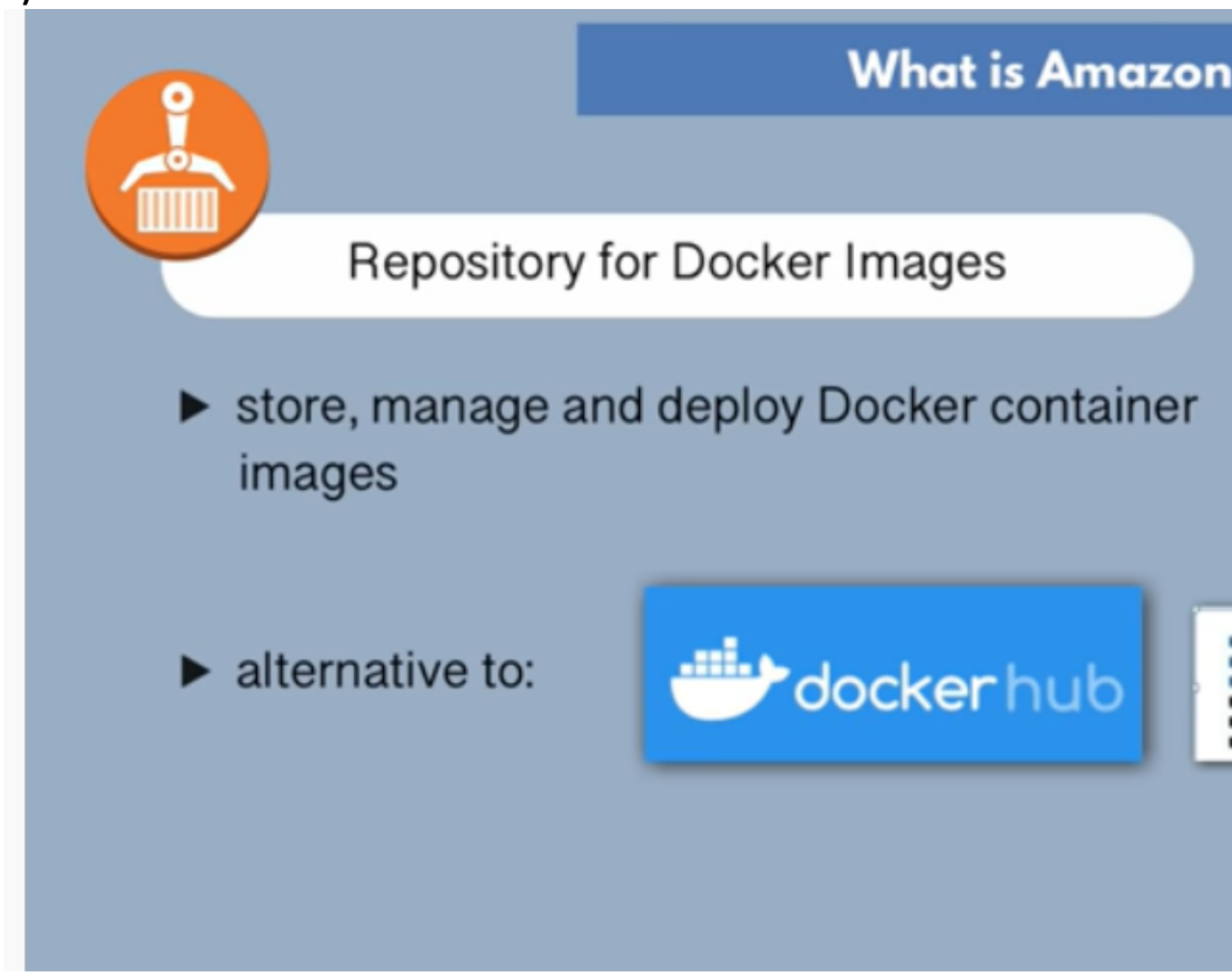
fully-managed



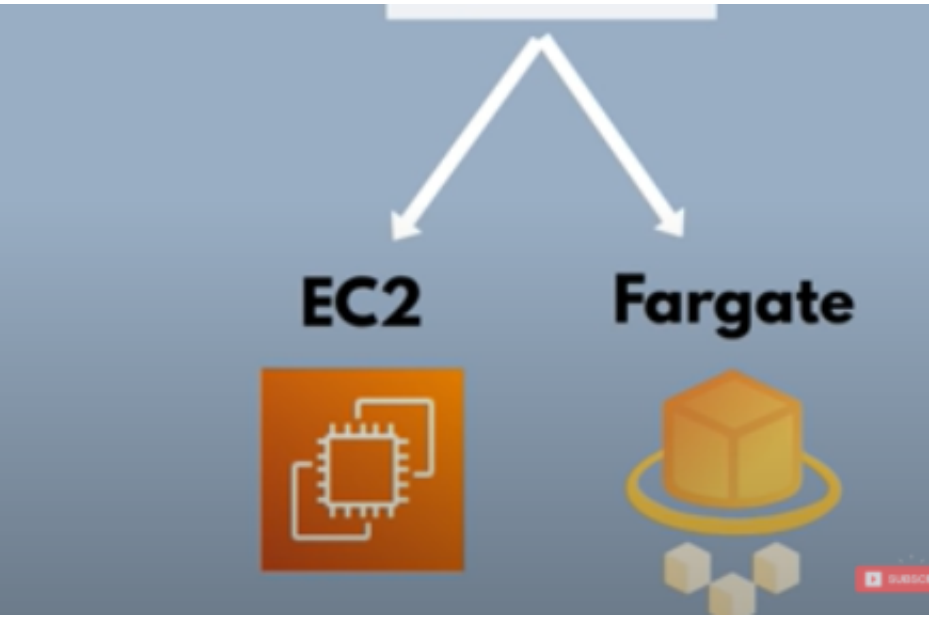

Combinations on AWS
ECS


EC2
Fargate
Hosting Type

9)


What is Amazon
Repository for Docker Images
store, manage and deploy Docker container images
alternative to:
docker hub


What is Amazon EC


EC2
Fargate

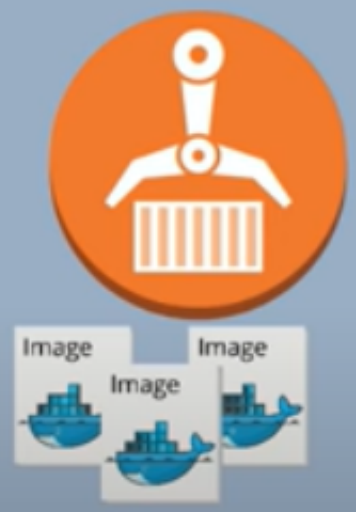

ECR?
Image
Nexus


CR?

integrates well with other AWS Service

notify when new image

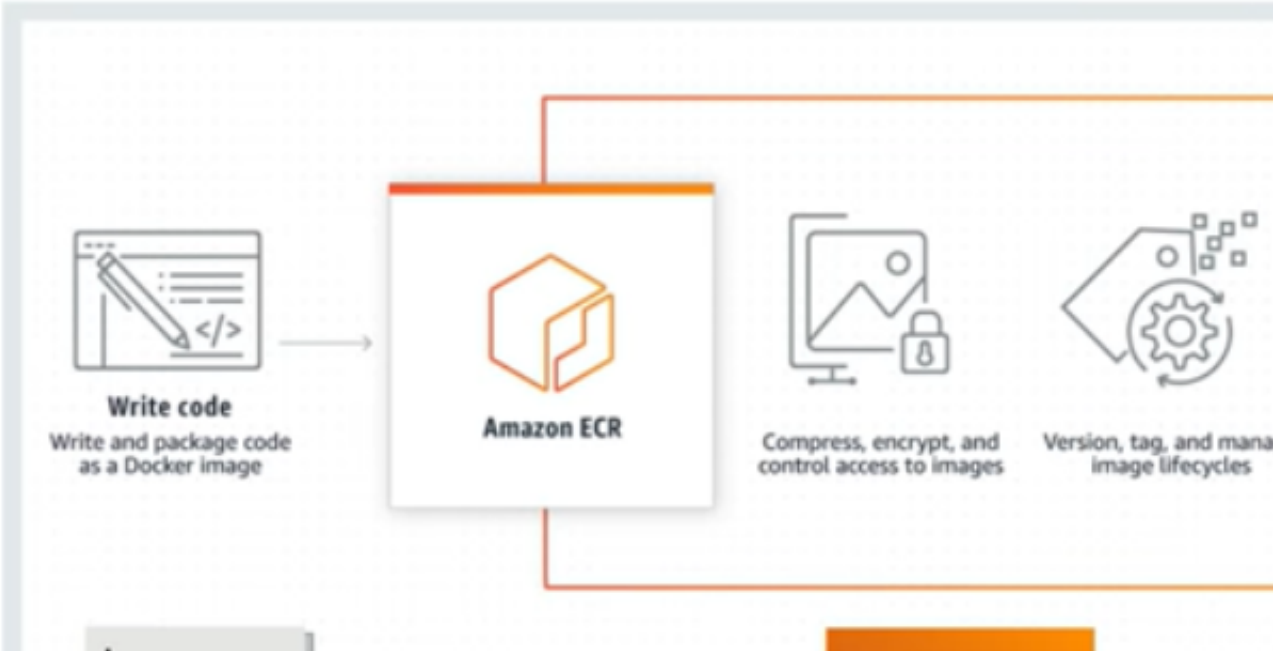
pull images



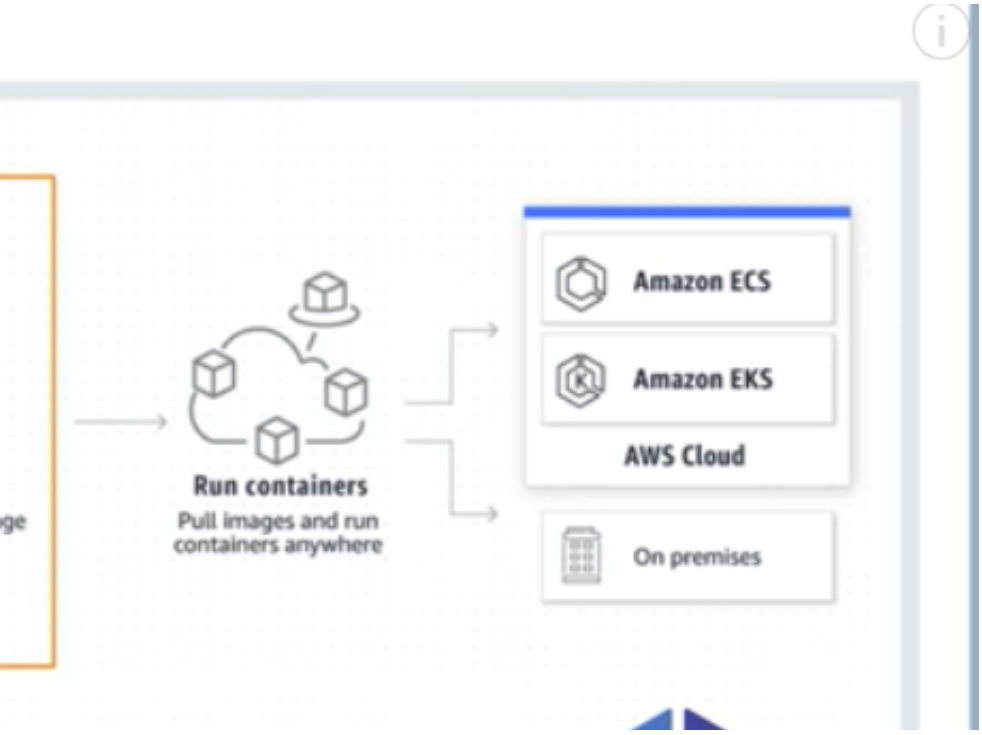
easy to connect and configure

your App Images
stored here

How it works




aws
Image


Amazon ECS
Amazon EKS
AWS Cloud
Run containers
Pull images and run containers anywhere
On premises

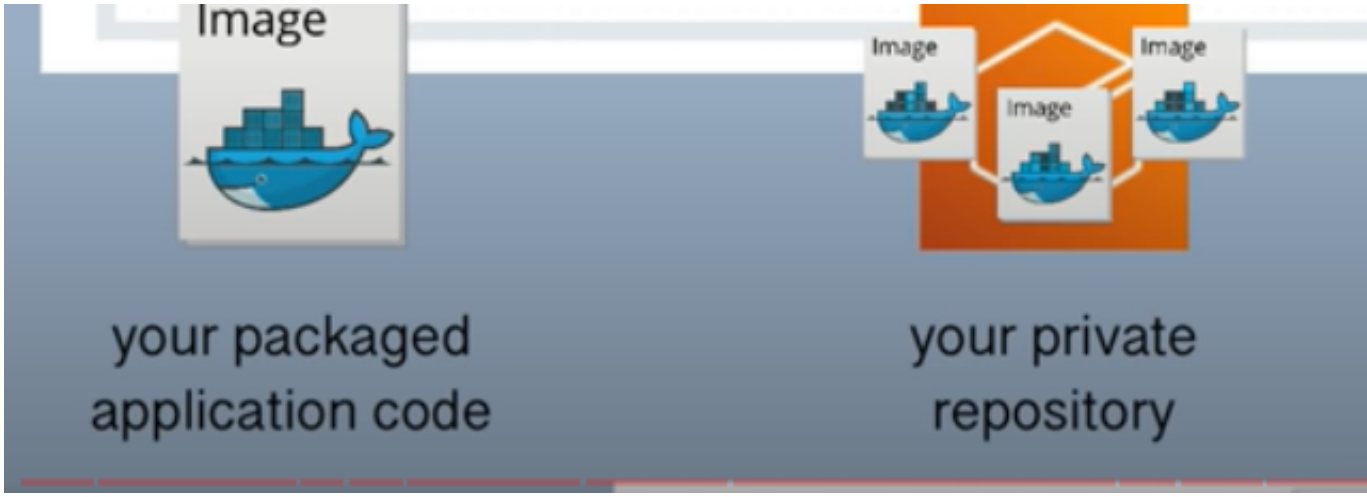
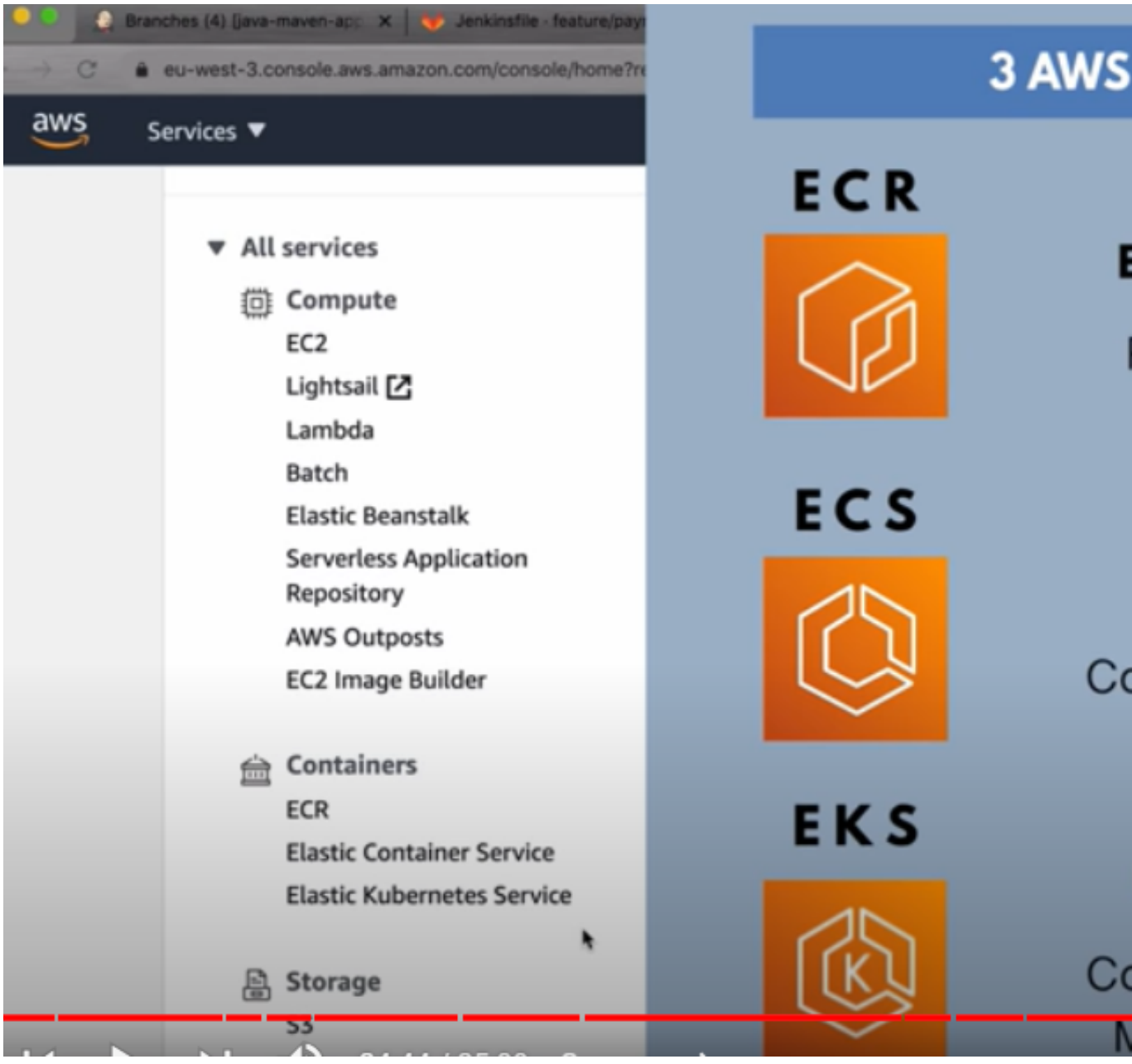


Image
Image
Image
Image
your packaged
application code
your private
repository


Branches (4) [java-maven-app
Jenkinsfile - feature/pay
eu-west-3.console.aws.amazon.com/console/home?re
aws
Services
All services
Compute
EC2
Lightsail
Lambda
Batch
Elastic Beanstalk
Serverless Application Repository
AWS Outposts
EC2 Image Builder
Containers
ECR
Elastic Container Service
Elastic Kubernetes Service
Storage
S3
3 AWS
ECR
ECS
EKS

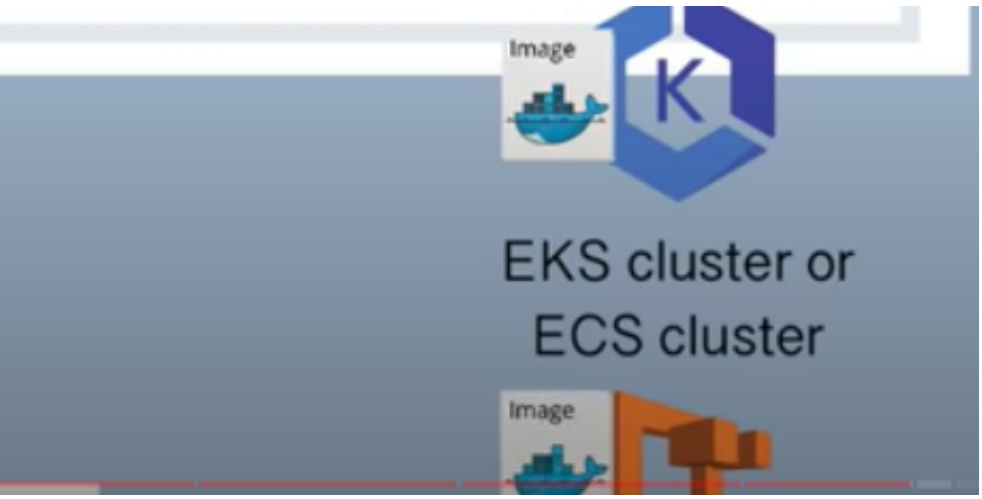


Image
EKS cluster or
ECS cluster
Image


Container Services
Elastic Container Registry
Private Docker Repository
Elastic Container Service
ontainer Orchestration Service
Elastic Kubernetes Service
ontainer Orchestration Service
Managed Kubernetes Service