Kubernetes Namespaces

What is namespace?

- Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called namespaces.
- Namespaces are intended for use in environments with many users spread across multiple teams, or projects.
- For clusters with a few to tens of users, you should not need to create or think about namespaces at all.
- **Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces.**
- **Using a Kubernetes namespace could isolate namespaces for different environments (dev, staging, preprod, prod) in the same cluster.**

Viewing namespaces

$ kubectl get namespace
NAME STATUS AGE
default Active 1d
kube-system Active 1d
kube-public Active 1d

Below image from my cluster

```
ubuntu@ip-172-31-38-166:~$ kubectl get namespace
NAME                   STATUS   AGE
default                Active   4d16h
kube-node-lease        Active   4d16h
kube-public            Active   4d16h
kube-system            Active   4d16h
kubernetes-dashboard   Active   4d16h
```

Kubernetes starts with three initial namespaces:

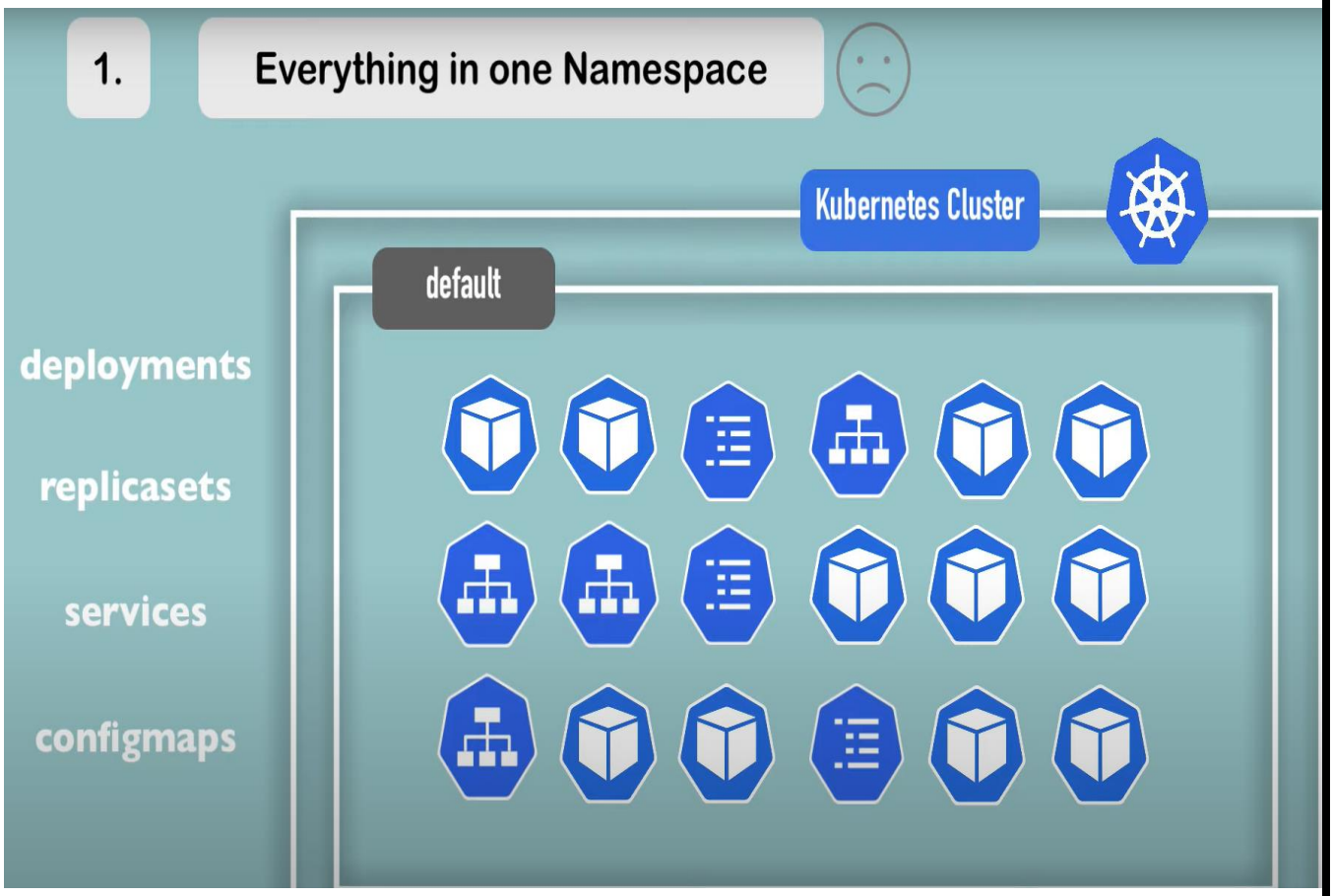
default The default namespace resource you create are located here.

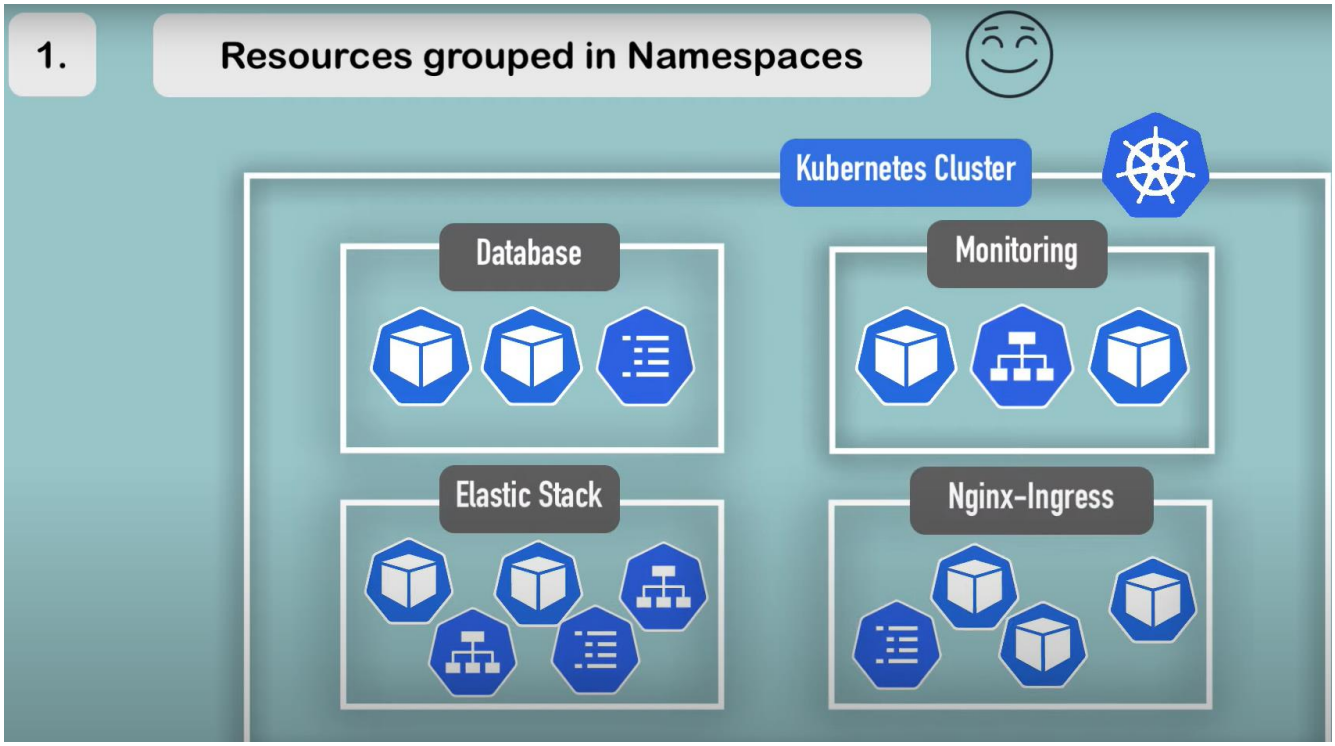
kube-system : Do not create or modify in kube-system.System process, master and kubectl process

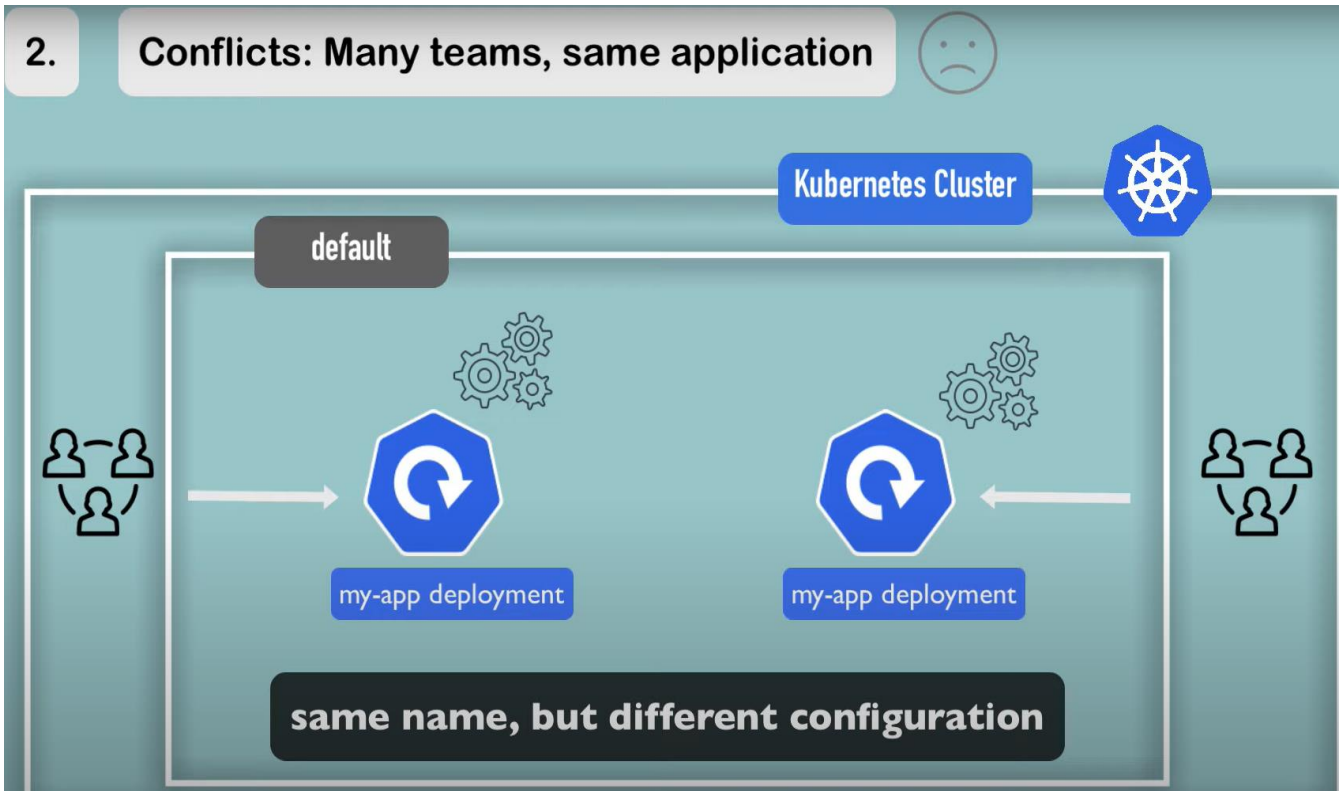
kube-public This namespace is created automatically and is readable by all users (including those not authenticated). This namespace is mostly reserved for cluster usage, in case that some resources should be visible and readable publicly throughout the whole cluster. The public aspect of this namespace is only a convention, not a requirement.

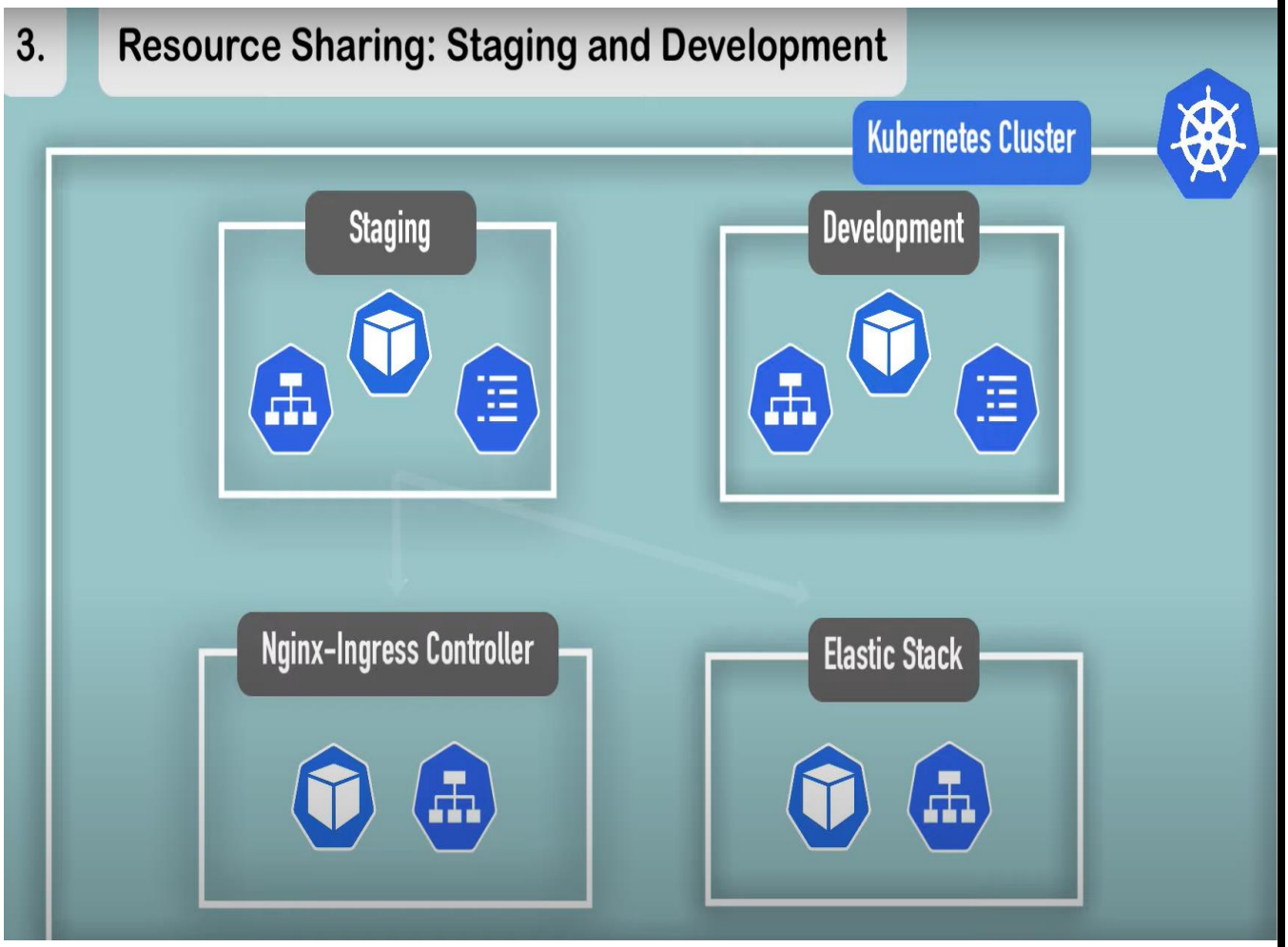
Kube-node-relase: Heart beats nodes , determines the availability of Nodes.

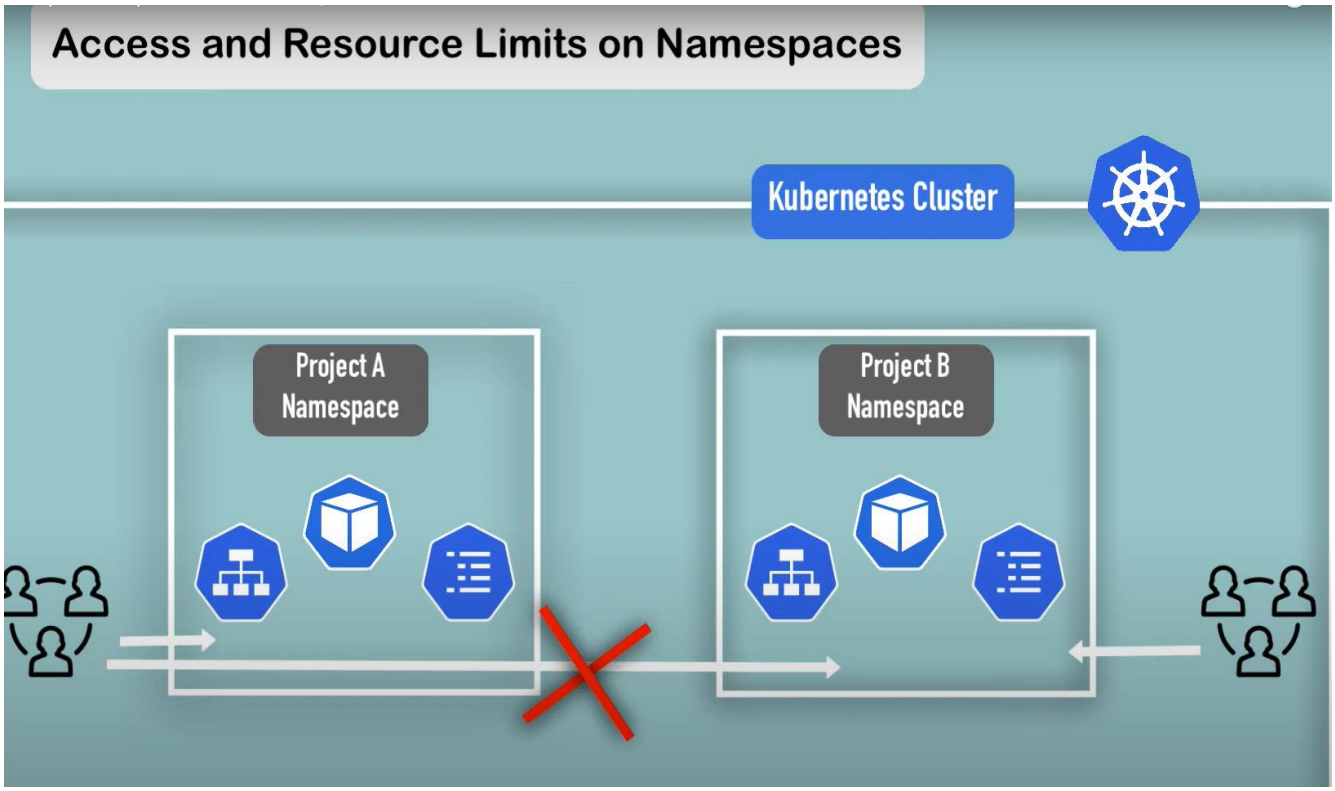
The name of a namespace must be a DNS label and follow the following rules:
At most 63 characters
Matching regex [a-z0-9]([-a-z0-9]*[a-z0-9])

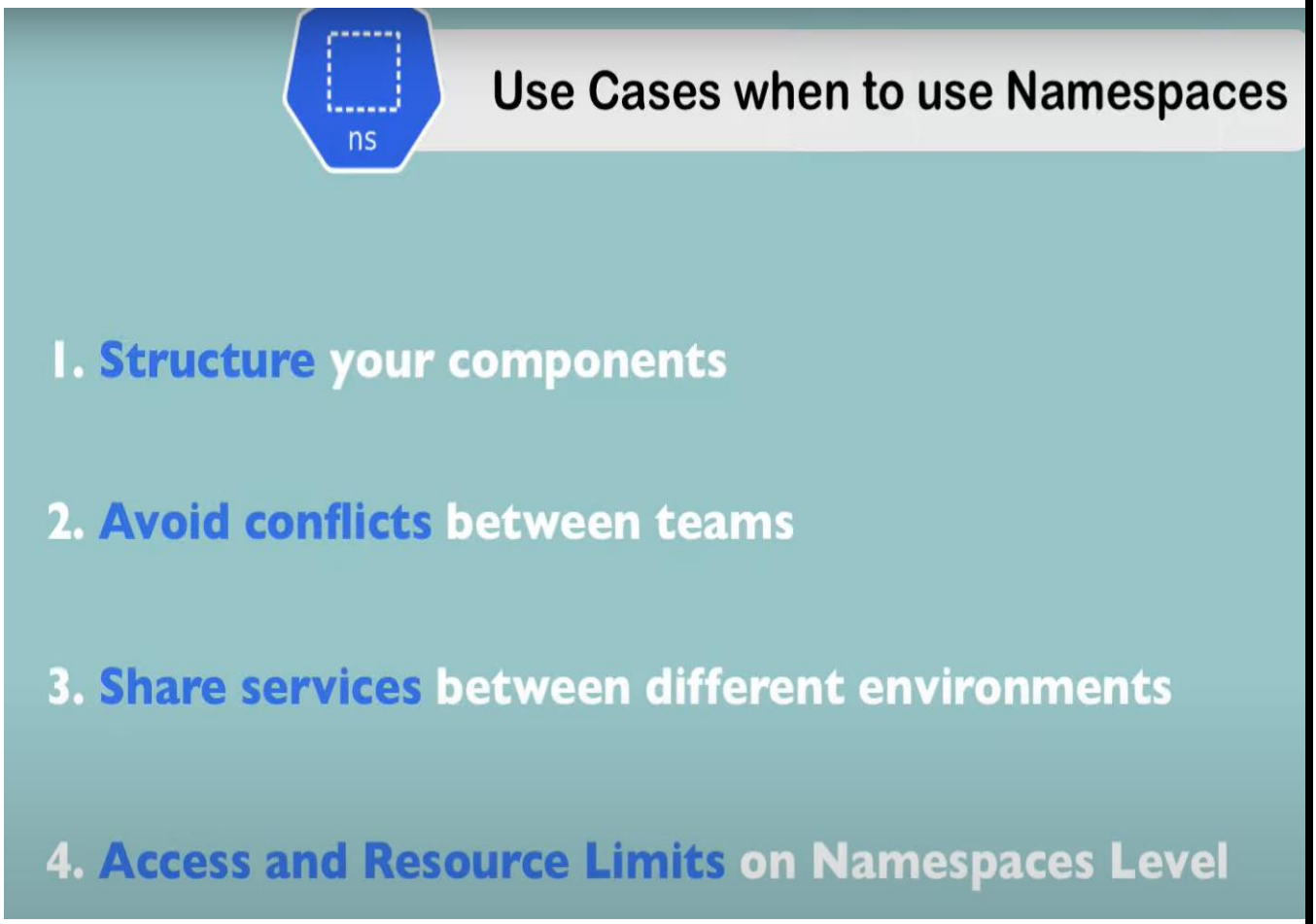


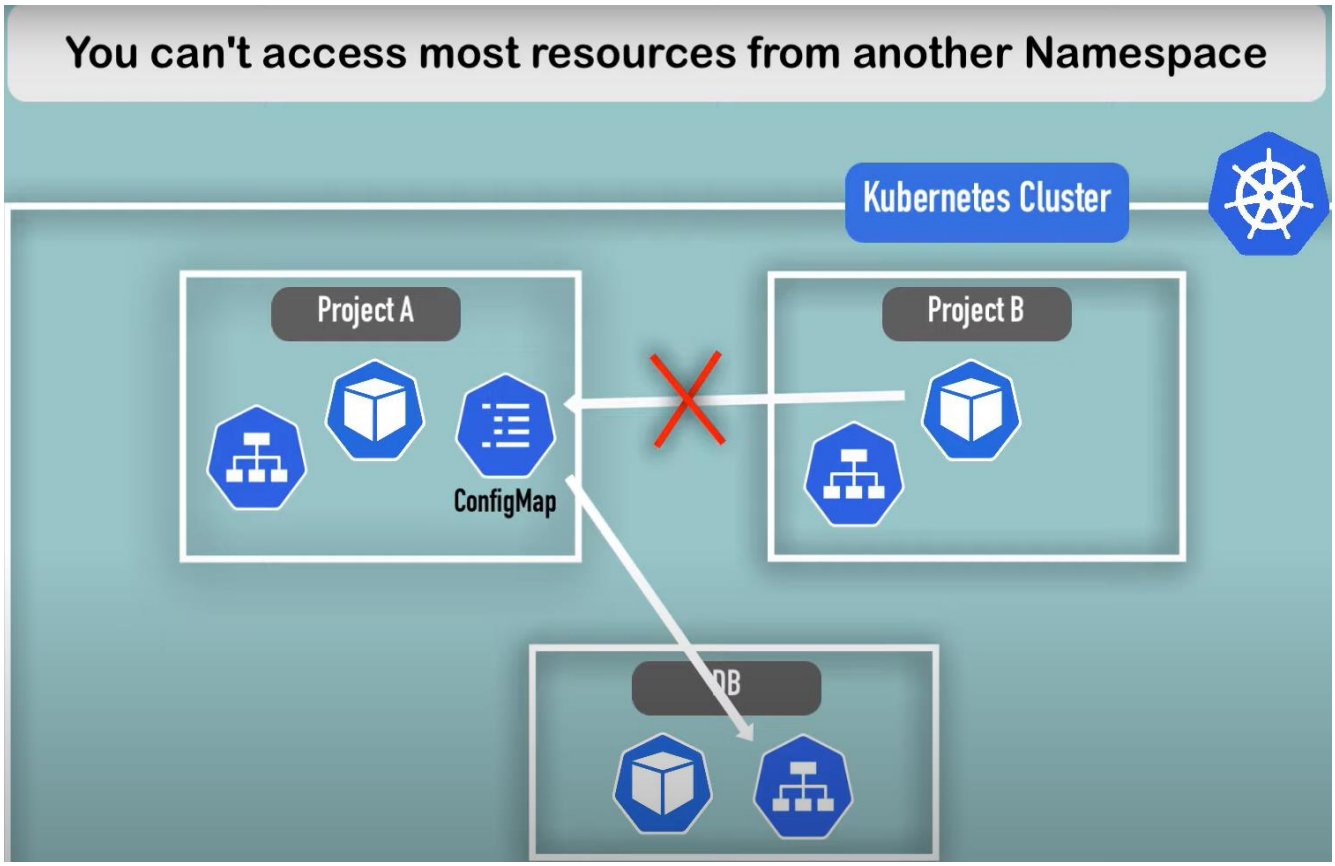


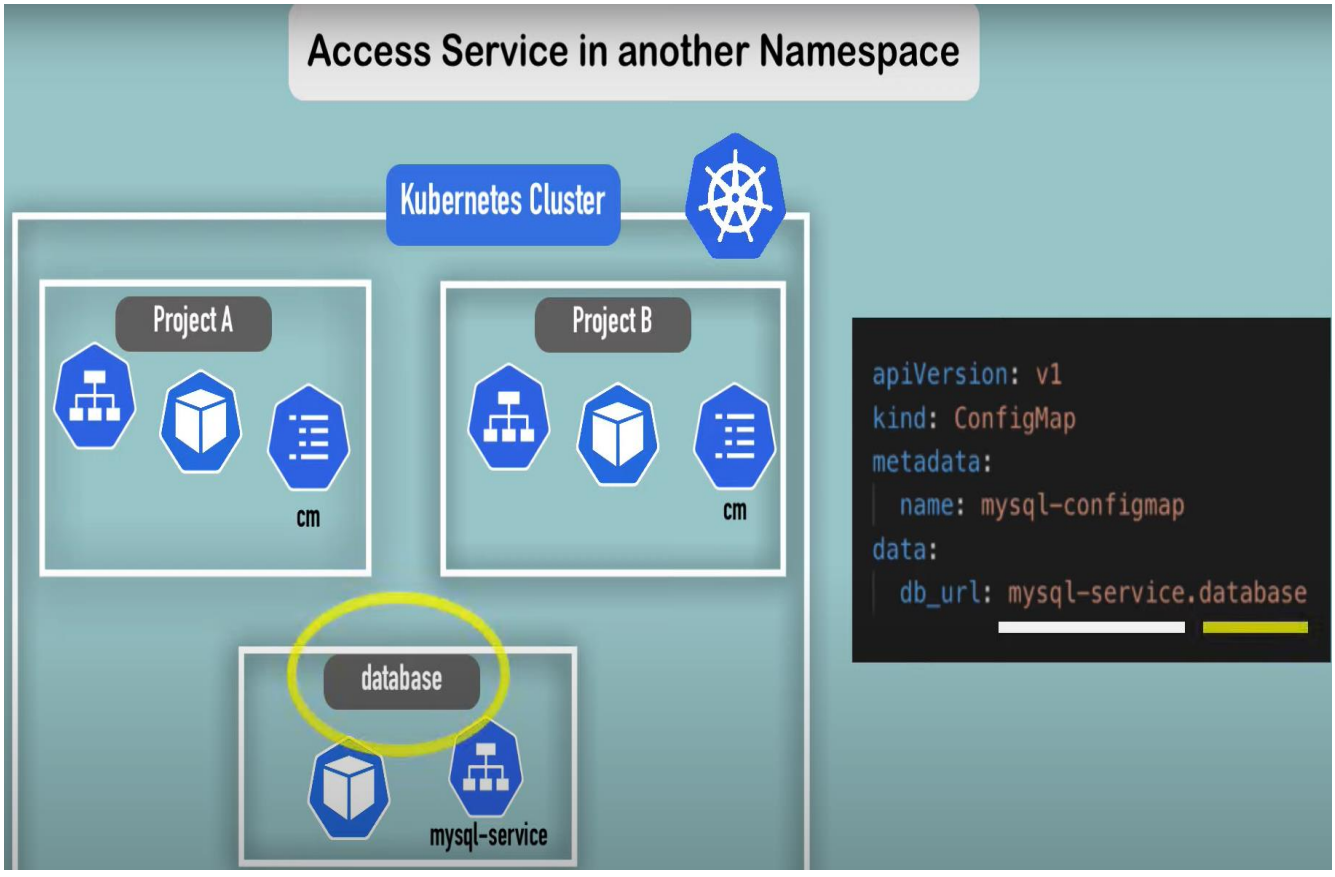

2.
Conflicts: Many teams, same application
Kubernetes Cluster
default
my-app deployment
my-app deployment
same name, but different configuration


3.
Resource Sharing: Staging and Development
Kubernetes Cluster
Staging
Development
Nginx-Ingress Controller
Elastic Stack


Access and Resource Limits on Namespaces
Kubernetes Cluster
Project A Namespace
Project B Namespace


ns
Use Cases when to use Namespaces
1. Structure your components
2. Avoid conflicts between teams
3. Share services between different environments
4. Access and Resource Limits on Namespaces Level


You can't access most resources from another Namespace
Kubernetes Cluster
Project A
Project B
ConfigMap


Access Service in another Namespace
Kubernetes Cluster
Project A
Project B
cm
cm
database
mysql-service
apiVersion: v1
kind: ConfigMap
metadata:
name: mysql-configmap
data:
db_url: mysql-service.database

Creating Namespace

Type1 :

Create namespace trough command line

$ kubectl create namespace test

```
ubuntu@ip-172-31-38-166:~$ kubectl create namespace test
namespace/test created
ubuntu@ip-172-31-38-166:~$ kubectl get ns
NAME                   STATUS   AGE
default                Active   4d17h
kube-node-lease        Active   4d17h
kube-public            Active   4d17h
kube-system            Active   4d17h
kubernetes-dashboard   Active   4d16h
test                   Active   23s
```

Type2 :

Create namespace trough yaml file

$vi newNamespace.yml

apiVersion: v1

kind: Namespace

metadata:

name: new-namespace

$ kubectl apply -f newNamespace.yml

After the namespace is created successfully, list the namespace again:

```
ubuntu@ip-172-31-38-166:~$ kubectl apply -f newNamespace.yml
namespace/new-namespace created
ubuntu@ip-172-31-38-166:~$ kubectl get ns
NAME                   STATUS   AGE
default                Active   4d20h
kube-node-lease        Active   4d20h
kube-public            Active   4d20h
kube-system            Active   4d20h
kubernetes-dashboard   Active   4d19h
new-namespace          Active   6s
test                   Active   172m
```

Type 1 applying namespace while deploying
Let's run the tutum replication controller , Building Your Own Kubernetes in a new namespace:

$ kubectl run tutum --image=tutum/hello-world --namespace=new-namespace

$ kubectl get pods --namespace=new-namespace

$ sudo kubectl get pods --namespace=default

$ kubectl get pods --all-namespaces

```
ubuntu@ip-172-31-38-166:~$ kubectl get pods --namespace=new-namespace
NAME                     READY   STATUS    RESTARTS   AGE
tutum-f4b45cb8-6h2bh     1/1     Running   0          81s
```

This deployment will create one pod , replicaset and deployment object

```
ubuntu@ip-172-31-38-166:~$ kubectl get pods --namespace=new-namespace
NAME                     READY   STATUS    RESTARTS   AGE
tutum-f4b45cb8-6h2bh     1/1     Running   0          3m41s
ubuntu@ip-172-31-38-166:~$ kubectl get rs --namespace=new-namespace
NAME              DESIRED   CURRENT   READY   AGE
tutum-f4b45cb8    1         1         1       3m46s
ubuntu@ip-172-31-38-166:~$ kubectl get deployment --namespace=new-namespace
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
tutum   1/1     1            1           3m55s
```

Type 2:
You can specify the namespace in yaml file
$vi mytutum.yml

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
  namespace: test
  labels:
    name: mypod
spec:
  containers:
  - name: mypod
    image: nginx
```

```
ubuntu@ip-172-31-38-166:~$ kubectl apply -f mytutum.yml
pod/mytutum created
```

$ kubectl get pods --namespace=test

```
ubuntu@ip-172-31-38-166:~$ kubectl get pods --namespace=test
NAME      READY   STATUS    RESTARTS   AGE
mytutum   1/1     Running   0          2m21s
```

By default, if you don't specify any namespace in the command line, Kubernetes will create the resources in the default namespace. If you want to create resources by configuration file, just simply specify it when doing kubectl create:
kubectl create -f myResource.yaml --namespace=new-namespace

Changing the default namespace

It is possible to switch the default namespace in Kubernetes:

1. Find your current context:
   $ kubectl config view | grep current-context

```
ubuntu@ip-172-31-38-166:~$ kubectl config view | grep current-context
current-context: kubernetes-admin@kubernetes
```

2. No matter whether there is current context or not, using set-context could create a new one or overwrite the existing one.

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

```
ubuntu@ip-172-31-38-166:~$ kubectl config set-context --current --namespace=new-namespace
Context "kubernetes-admin@kubernetes" modified.
ubuntu@ip-172-31-38-166:~$ kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
tutum-f4b45cb8-6h2bh    1/1     Running   0          43m
```

Deleting a namespace

1. Using **kubectl delete** could delete the resources including the namespace. Deleting a namespace will erase all the resources under that namespace:

$ kubectl delete namespaces new-namespace

```
ubuntu@ip-172-31-38-166:~$ kubectl delete namespaces new-namespace
namespace "new-namespace" deleted
```

2. After the namespace is deleted, our tutum pod is gone:

```
ubuntu@ip-172-31-38-166:~$ kubectl get pods
No resources found in new-namespace namespace.
```

3. However, the default namespace in the context is still set as **new-namespace**:

```
ubuntu@ip-172-31-38-166:~$ kubectl config view | grep namespace
    namespace: new-namespace
    namespace: new-namespace
```

Will it be a problem? Yes because we deleted new-namespace we deleted

4. Let's run an nginx replication controller again.

$ kubectl run tutum --image=tutum/hello-world

```
ubuntu@ip-172-31-38-166:~$ kubectl run tutum --image=tutum/hello-world
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future ver
sion. Use kubectl run --generator=run-pod/v1 or kubectl create instead.
Error from server (NotFound): namespaces "new-namespace" not found
```

It will try to create an tutum replication controller and replica pod in the current namespace we just deleted. Kubernetes will throw out an error if the namespace is not found.

5. Let's switch back to the default namespace.
$ kubectl config set-context --current --namespace=""

```
ubuntu@ip-172-31-38-166:~$ kubectl config set-context --current --namespace=""
Context "kubernetes-admin@kubernetes" modified.
ubuntu@ip-172-31-38-166:~$ kuectl get pods
```

Lets run tutum application now it will deploy on default namespace

```
ubuntu@ip-172-31-38-166:~$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
hello-deploy-2tdrg            1/1     Running   1          2d21h
hello-deploy-4lkzm            1/1     Running   1          2d21h
hello-deploy-bmf92            1/1     Running   1          2d21h
hello-deploy-nmsgw            1/1     Running   1          2d21h
hello-deploy-rv7lj            1/1     Running   1          2d21h
hello-pod                     1/1     Running   1          3d21h
nginx-app-89cdc496-smz55      1/1     Running   1          4d20h
```