# 05: Virtual Clusters with Namespaces

**Namespaces are a native way to divide a single Kubenetes cluster into multiple virtual clusters.**

## Use cases for Namespaces

- Easy way to apply quotas and policies to groups of objects
- If no target Namespace is spacified the default Namespace will be used to deployment.
- `kubectl api-resources`
- a good way of sharing a single cluster among different departments and environments. For example Dev, Test and QA, each one with they own sets of users and permisions, and unique resource quotes

Namespaces ar not good for isolating hostile workloads. To **strong isolation** the correct method is to use multiple clusters.

Every Kubernetes cluster has a set of a pre-created Namespaces.

```
kbuectl get namespaces
NAME STATUS AGE
default ACTIVE ...
kube-system ...
kube-public ...
kube-node-lease ...
```

- default: newly crated objects with no specified Namespace
- Kube-system: for DNS, metrics server and control plane components
- Kube-public: for objects that need to be readable by anyone
- kube-node-lease: node heartbeat and managing leases

```
kubectl describe ns default
kubectl describe ns -n my-namespace
```

## Creating and Mananging Namespaces

```
git clone https://github.com/nigelpoulton/TheK8sBook.git
cd namespaces
```

```
kubectl create ns hydra
namespace/hydra created
```

```
# shield-ns.yml
kind: Namespace
apiVersion: v1
metadata:
    name: shield
    labels:
        env: marvel
```

```
kubectl apply -f shield-ns.yml
namespace/shield created
```

```
kubectl get ns

kubectl delete ns hydra
```

#### Configuring kubectl to use a specific NS

```
kubectl config set-context --current --namespace shield
```

### Deploying to Namespaces

- imperative method: required to add -n or --namespace flag to commands.
- declarative method: specifies the Namespace in YAML manifest file.

```
apiVersion: v1
kind: ServiceACcount
metadata:
    namespace: shield
    name: default
---
apiVersion: v1
kind: Service
metadata:
    namespace: shield
    name: the-bus
spec:
    ports:
    - nodePort: 31112
      port: 8080
      targetPort: 8080
    selector:
      env: marvel
---
```

```
apiVersion: v1
kind: Pod
metadata:
    namespace: shield
    name: trisklion
```

```
kubectl apply -f shield-app.yml
serviceaccount/default ocnfigured
service/the-bus configured
pod/triskelion created
```

```
kubectl get pods -n shield
kubectl get svc -n shield
```

```
curl localhost:31112
```

## Clean-up

```
kubectl delete -f shield-app.yml
kubectl delete ns shield

kubectl config set-context --current --namespace default
```