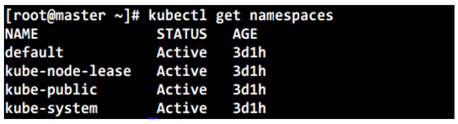
Namespace

12 January 2021 11:40

- Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called namespac es.
- Namespaces are intended for use in environments with many users spread across multiple teams, or projects.
- Namespaces are a way to divide cluster resources between multiple users.

Viewing namespaces

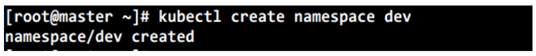


```
[root@master ~]# kubectl get namespaces
NAME              STATUS   AGE
default           Active   3d1h
kube-node-lease   Active   3d1h
kube-public       Active   3d1h
kube-system       Active   3d1h
```

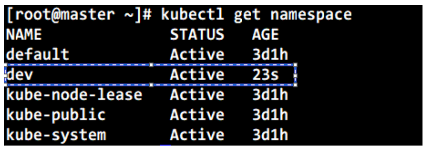
default: Default name space with no other namespace.
kube-node-lease: For the lease objects associated with each node which improves the performance of the node heartbeats as the cluster scales.
kube-public: Created automatically and is readable by all users.
Mostly reserved for cluster usage, in case that some resources should be visible and readable publicly throughout the whole cluster.
kube-system: Namespace for objects created by the Kubernetes system

Create New Namespace

Imperative way:

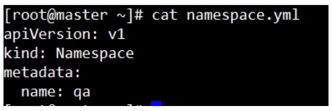


```
[root@master ~]# kubectl create namespace dev
namespace/dev created
```



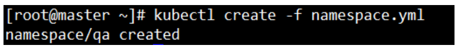
```
[root@master ~]# kubectl get namespace
NAME              STATUS   AGE
default           Active   3d1h
dev               Active   23s
kube-node-lease   Active   3d1h
kube-public       Active   3d1h
kube-system       Active   3d1h
```

Alternatively create Yaml file



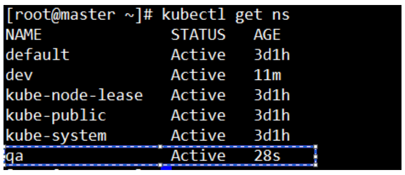
```
[root@master ~]# cat namespace.yml
apiVersion: v1
kind: Namespace
metadata:
  name: qa
```

# kubectl create -f namespace.yml



```
[root@master ~]# kubectl create -f namespace.yml
namespace/qa created
```

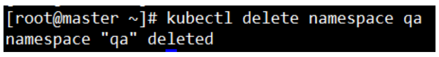
# kubectl get ns



```
[root@master ~]# kubectl get ns
NAME              STATUS   AGE
default           Active   3d1h
dev               Active   11m
kube-node-lease   Active   3d1h
kube-public       Active   3d1h
kube-system       Active   3d1h
qa                Active   28s
```

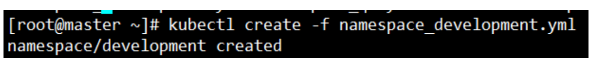
Delete namespace

** delete namespace will delete all the resources also inside it

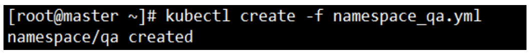


```
[root@master ~]# kubectl delete namespace qa
namespace "qa" deleted
```

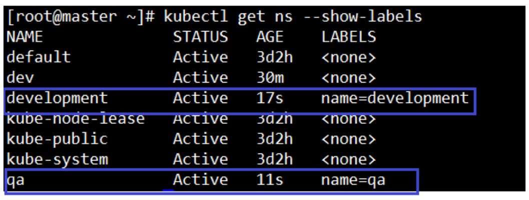
Create namespaces with labels



```
[root@master ~]# kubectl create -f namespace_development.yml
namespace/development created
```



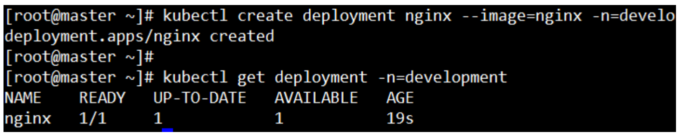
```
[root@master ~]# kubectl create -f namespace_qa.yml
namespace/qa created
```



```
[root@master ~]# kubectl get ns --show-labels
NAME              STATUS   AGE    LABELS
default           Active   3d2h   <none>
dev               Active   30m    <none>
development       Active   17s    name=development
kube-node-lease   Active   3d2h   <none>
kube-public       Active   3d2h   <none>
kube-system       Active   3d2h   <none>
qa                Active   11s    name=qa
```

Create Deployment/Pod in each namespace

# kubectl create deployment nginx --image=nginx -n=development

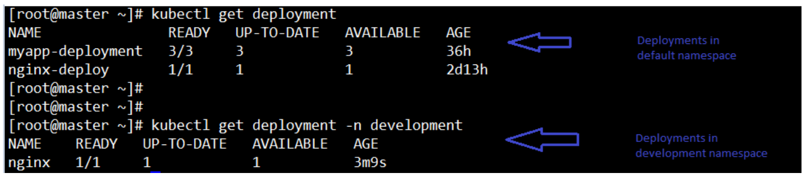


```
[root@master ~]# kubectl create deployment nginx --image=nginx -n=develo
deployment.apps/nginx created
[root@master ~]#
[root@master ~]# kubectl get deployment -n=development
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           19s
```

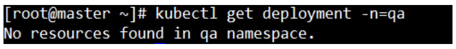
Check Deployment in each namespace i.e. default, development and qa

# kubectl get deployment -n=development

# kubectl get deployment -n=qa

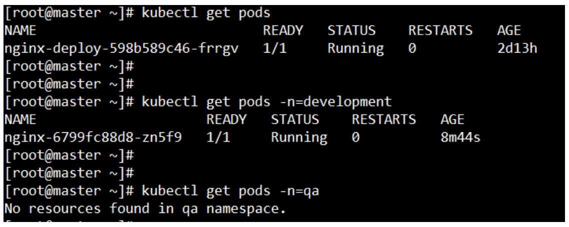

Deployments in default namespace
Deployments in development namespace

```
[root@master ~]# kubectl get deployment
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
myapp-deployment   3/3     3            3           36h
nginx-deploy       1/1     1            1           2d13h
[root@master ~]#
[root@master ~]#
[root@master ~]# kubectl get deployment -n development
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
nginx   1/1     1            1           3m9s
```



```
[root@master ~]# kubectl get deployment -n=qa
No resources found in qa namespace.
```

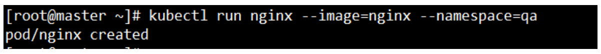
Check Pod in each namespace i.e. default, development and qa



```
[root@master ~]# kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
nginx-deploy-598b589c46-frrgv  1/1     Running   0          2d13h
[root@master ~]#
[root@master ~]#
[root@master ~]# kubectl get pods -n=development
NAME                     READY   STATUS    RESTARTS   AGE
nginx-6799fc88d8-zn5f9   1/1     Running   0          8m44s
[root@master ~]#
[root@master ~]#
[root@master ~]# kubectl get pods -n=qa
No resources found in qa namespace.
```

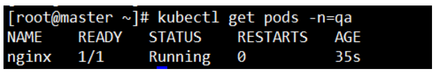
Create Pod in qa namespace and check status

# kubectl run nginx --image=nginx --namespace=qa



```
[root@master ~]# kubectl run nginx --image=nginx --namespace=qa
pod/nginx created
```

# kubectl get pods -n=qa



```
[root@master ~]# kubectl get pods -n=qa
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          35s
```

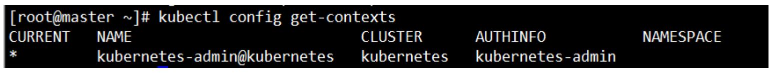
Context :

- A kubernetes context is just a set of access parameters that contains
  - **a Kubernetes cluster, a user, and a namespace.**
- Kubernetes Context is essentially the configuration that you use to access a particular cluster & namespace with a user account.

List Context :

# kubectl config get-contexts

- "*" shows current context
- **Namespace blank field shows default Namespace**

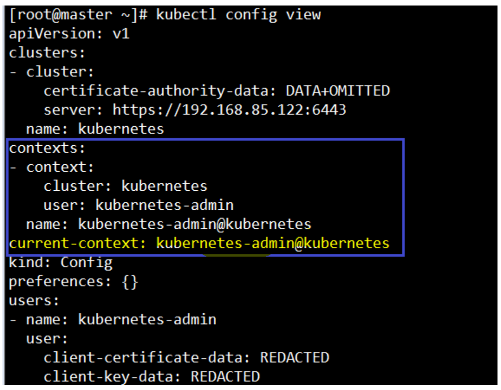


```
[root@master ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin
```

- **Cluster : kubernetes**
- **User: kubernetes-admin**
- **Namespace: <default>**
- **Name of Context: kubernetes-admin@kubernetes**

View context configuration

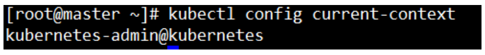
# kubectl config view



```
[root@master ~]# kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://192.168.85.122:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: REDACTED
    client-key-data: REDACTED
```

View current context

# kubectl config current-context



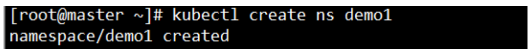
```
[root@master ~]# kubectl config current-context
kubernetes-admin@kubernetes
```

Change Namespace in current context

1. Step one - create scenario

Firstly create new namespace named **demo1**

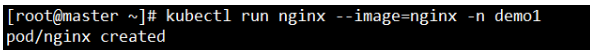
# kubectl create ns demo1



```
[root@master ~]# kubectl create ns demo1
namespace/demo1 created
```

Create POD nginx in namespace demo1

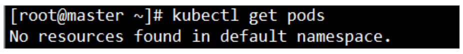
# kubectl run nginx --image=nginx -n demo1



```
[root@master ~]# kubectl run nginx --image=nginx -n demo1
pod/nginx created
```

Check status of Pod. There will be no pod shown in output as we are in default namespace

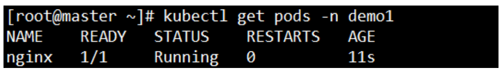
# kubectl get pods



```
[root@master ~]# kubectl get pods
No resources found in default namespace.
```

Check status of POD in namespace demo1. It will show one pod named nginx in running state.

# kubectl get pods -n demo1



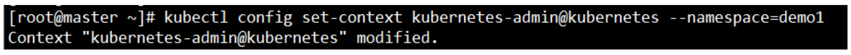
```
[root@master ~]# kubectl get pods -n demo1
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          11s
```

2. Change default NS

Set-context

5) Set namesapce demo1 in current context

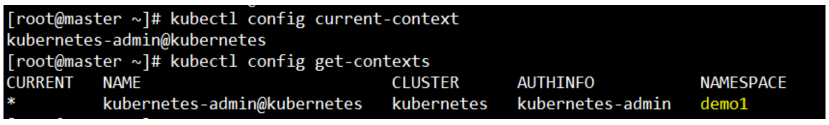
# kubectl config set-context kubernetes-admin@kubernetes --namespace=demo1



```
[root@master ~]# kubectl config set-context kubernetes-admin@kubernetes --namespace=demo1
Context "kubernetes-admin@kubernetes" modified.
```

6) View current context and get context. Under namespace , there will be demo1 namespace

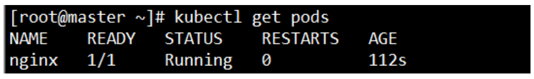
# kubectl config current-context



```
[root@master ~]# kubectl config current-context
kubernetes-admin@kubernetes
[root@master ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin   demo1
```

7) View pod without giving namespace name. You will be able to pod from demo1 namespace. This is because we have set namespace demo1 in current context

# kubectl get pods

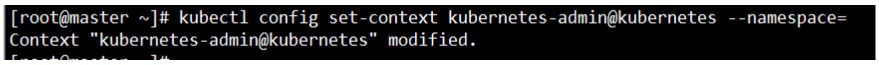


```
[root@master ~]# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          112s
```

3. Reset to default NS

8) Change namespace to default namespace in current context

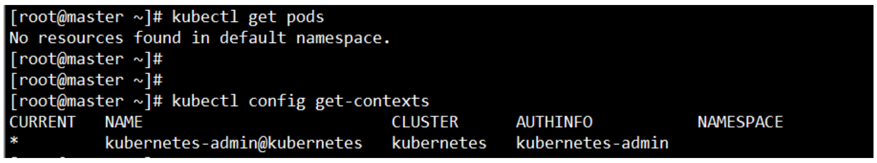
# kubectl config set-context kubernetes-admin@kubernetes --namespace=



```
[root@master ~]# kubectl config set-context kubernetes-admin@kubernetes --namespace=
Context "kubernetes-admin@kubernetes" modified.
```

9) Verify get context and status and make sure we switch to default namespace

   # kubectl get pods
   # kubectl config get-contexts



```
[root@master ~]# kubectl get pods
No resources found in default namespace.
[root@master ~]#
[root@master ~]#
[root@master ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin
```

4. Add new context

1) assign default NS

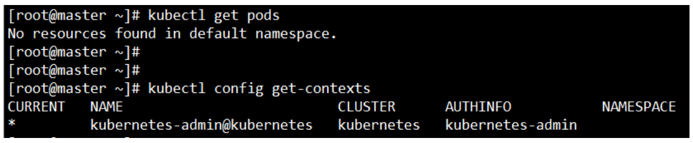
Scenario

Create new context and associate namespace with it.

- Ideal practice to create new context and associate namespace with it
  - Namespaces works on single cluster level
  - Context works on Multi cluster level

1) Check status of Pod . There will be no pod as we are in default namespace.
   Also check how many contexts are available. There is only one context available

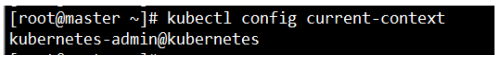
   # kubectl get pods



```
[root@master ~]# kubectl get pods
No resources found in default namespace.
[root@master ~]#
[root@master ~]#
[root@master ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin
```

2) Check current context. As there is only one context, so it will show only one

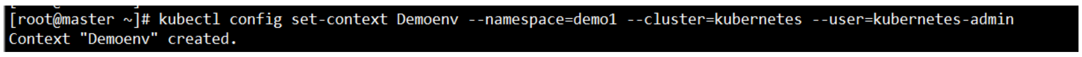
   # kubectl config current-context



```
[root@master ~]# kubectl config current-context
kubernetes-admin@kubernetes
```

3) Create new context called demoenv and associate demo1 namespace with it.

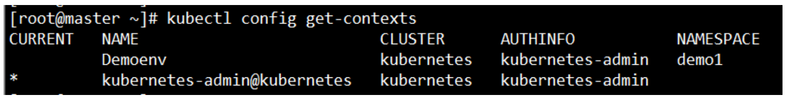
   # kubectl config set-context Demoenv --namespace=demo1 --cluster=kubernetes --user=kubernetes-admin



```
[root@master ~]# kubectl config set-context Demoenv --namespace=demo1 --cluster=kubernetes --user=kubernetes-admin
Context "Demoenv" created.
```

4) View status of contexts with get-contexts commands. Now it will show 2 contexts but "*" is in front of default context. So still it will show all things as per Default context and namespace.

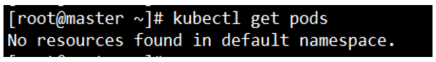
   # kubectl config get-contexts



```
[root@master ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
          Demoenv                       kubernetes   kubernetes-admin   demo1
*         kubernetes-admin@kubernetes   kubernetes   kubernetes-admin
```

5) Verify with check pod status. No pods will be shows as it still uses default context and namespace.

   # kubectl get pods

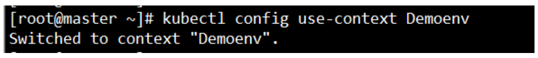


```
[root@master ~]# kubectl get pods
No resources found in default namespace.
```

Switch/Use new context

1) Switch to new context Demoenv

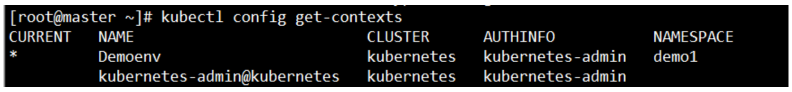
   # kubectl config use-context Demoenv



```
[root@master ~]# kubectl config use-context Demoenv
Switched to context "Demoenv".
```

2) Check get contexts and notice "*" . It will be in front of Demoenv context. And this context has namespace demo1.
   So it will read all information from demo1 namespace

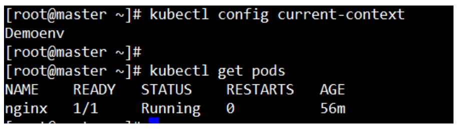
   # kubectl config get-contexts



```
[root@master ~]# kubectl config get-contexts
CURRENT   NAME                          CLUSTER      AUTHINFO           NAMESPACE
*         Demoenv                       kubernetes   kubernetes-admin   demo1
          kubernetes-admin@kubernetes   kubernetes   kubernetes-admin
```

3) Check current context . It will show Demoenv and also check pods. It will show pod from demo1 namespace

# kubectl config current-context
# kubectl get pods



```
[root@master ~]# kubectl config current-context
Demoenv
[root@master ~]#
[root@master ~]# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          56m
```