Presented By Raman

Kubernetes

Assigning Pods to Namespace

Production-Grade Container Orchestration

Kubernetes-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.
- Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces. Namespaces can not be nested inside one another and each Kubernetes resource can only be in one namespace.
- Namespaces are a way to divide cluster resources between multiple users.
- It is not necessary to use multiple namespaces just to separate slightly different resources, such as different versions of the same software: use **labels** to distinguish resources within the same namespace.

Working with Namespaces

Kubectl get namespace

```
NAME                   STATUS   AGE
default                Active   11d
kube-node-lease        Active   11d
kube-public            Active   11d
kube-system            Active   11d
kubernetes-dashboard   Active   11d
```

Default	The default namespace for objects with no other namespace
kube-system	The namespace for objects created by the Kubernetes system
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-lease	Each Node has an associated Lease object in the kube-node-lease namespace. Lease is a lightweight resource, which improves the performance of the node heartbeats as the cluster scales.

Create a namespace

- kubectl create namespace demo1
- kubectl get namespace

```
NAME                   STATUS   AGE
default                Active   11d
demo1                  Active   46s
kube-node-lease        Active   11d
kube-public            Active   11d
kube-system            Active   11d
kubernetes-dashboard   Active   11d
```

- kubectl run nginx-pod --image=nginx --namespace=demo1
- kubectl get pods –namespace demo1
- kubectl delete pod nginx-pod --namespace demo1

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  namespace: demo1
spec:
  containers:
    - name: nginx-container
      image: nginx
```

```
root@kmaster:/home/ubuntu# kubectl run nginx-pod --image=nginx --namespace=demo1
pod/nginx-pod created
root@kmaster:/home/ubuntu# kubectl get pods --namespace demo1
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          39s
```

Kubeconfig files

- Use kubeconfig files to organize information about clusters, users, namespaces, and authentication mechanisms. The kubectl command-line tool uses kubeconfig files to find the information it needs to choose a cluster and communicate with the API server of a cluster.

Context

- A context element in a kubeconfig file is used to group access parameters under a convenient name. Each context has three parameters: cluster, namespace, and user. By default, the kubectl command-line tool uses parameters from the current context to communicate with the cluster.

kubectl config view	**To display current kube configuration**
kubectl config get-contexts	Display all the contexts
kubectl config set-context kubesys --namespace=kubesystem --user=kubernetes-admin --cluster=kubernetes	Create a new context
kubectl config use-context kubesys	Switched to context "kubesys"
kubectl config delete-context kubesys	Delete the context kubesys
kubectl config current-context	Displays the current context