17.什么是Namespace

我们知道Kubernetes中每个对象都可以附加多个标签，具有同一个标签的所有对象逻辑上可以合成一组。但是这样也会带来一个问题，那就是不同组之间可能存在相同的对象，造成了对象的重叠。如果分别按组来操作组内的对象，就可能会出现重复处理的情况。

什么是命名空间

Kubernetes为我们提供了命名空间（Namespace）的概念，将对象分组到命名空间中。命名空间为对象的名称提供了一个作用域。我们可以将所有资源分组到多个命名空间下，而不是将它们放到一个命名空间下。这样一来，我们就可以跨多个命名空间使用相同的资源名称。
需要注意的是Kubernetes中的命名空间指的是一个作用域，而之前章节提到的用来隔离进程的命名空间是linux内核提供的命名空间（主机名、网络、文件系统、进程等等）。

同一个命名空间下的资源名称必须是唯一的。两个不同的命名空间下可以存在同名的资源。虽然大多数类型的资源是具有命名空间的，但是仍有一些资源例外。比如工作节点（Node）是全局的而且不与单个命名空间绑定。

查看命名空间以及命名空间下的对象

通过如下命令可以显示集群中的所有命名空间：
kubectl get ns

```
[david@dhr-demo root]$ kubectl get ns
NAME                   STATUS   AGE
default                Active   12d
kube-node-lease        Active   12d
kube-public            Active   12d
kube-system            Active   12d
kubernetes-dashboard   Active   7d20h
[david@dhr-demo root]$
```

之前我们在使用kubectl get命令的时候从未指定命名空间，因此系统默认显示default命名空间下的所有对象。

从上图返回的结果中我们可以看到实际上还存在其他命名空间，如kube-public。我们可以通过指定--namespace选项(或者-n)来显示指定命名空间下的对象，如：

kubectl get po --namespace kube-system

```
[david@dhr-demo root]$ kubectl get po --namespace kube-system
NAME                                 READY   STATUS    RESTARTS   AGE
coredns-6c76c8bb89-m9jft             1/1     Running   0          12d
etcd-minikube                        1/1     Running   0          12d
kube-apiserver-minikube              1/1     Running   0          12d
kube-controller-manager-minikube     1/1     Running   0          12d
kube-proxy-bsdlz                     1/1     Running   0          12d
kube-scheduler-minikube              1/1     Running   0          12d
storage-provisioner                  1/1     Running   1          12d
[david@dhr-demo root]$
```

从kube-system命名空间的名字我们也能看出，该命名空间下的对象与Kubernetes系统本身是密切相关的。将具有相关性的一些对象放到同一个命名空间下可以使系统看起来更有组织性。如果所有对象（包含我们自己创建的对象）都在同一个命名空间下就会很难辨别，甚至还有可能导致用户删除系统资源或者其他用户的资源。

命名空间的作用

多个命名空间可以将包含大量组件的复杂系统分拆成多个更小的、不同的组，还可以用于在多租户环境中隔离资源，或者将资源分离成生产、开发、测试环境，或者以其他任何你想要的方式处理都可以。

有了命名空间，我们就能够将不相关的资源分到非重叠的组中。如果有多个用户正在使用同一个Kubernetes集群，而且他们各自管理自己的一组资源，那么他们就应该各自使用自己的命名空间。通过这种方式，就不需要担心修改删除其他用户的资源了，而且也不用关心名称是否冲突，因为命名空间为资源的名称提供了一个作用域。

除了隔离资源，命名空间还可以用于只允许某些用户访问特定的资源，甚至还可以限制单个用户可以使用的计算资源数量。

但是需要说明的是，命名空间是不会隔离运行的对象的。比如，不同命名空间下的Pod之间仍然是可以互相通信的，除非命名空间本身也做了网络隔离

创建命名空间

创建命名空间与创建其他资源对象类似，也可以通过YAML格式的文件来创建：

vim my-namespace.yaml

apiVersion: v1

kind: Namespace

metadata:

  name: my-namespace

```
apiVersion: v1
kind: Namespace
metadata:
  name: my-namespace
~
~
```

使用kubectl将该文件发送到Kubernetes API Sever：

kubectl create –f my–namespace.yaml

```
[david@dhr-demo ~]$ kubectl create -f my-namespace.yaml
namespace/my-namespace created
[david@dhr-demo ~]$ kubectl get ns
NAME                   STATUS   AGE
default                Active   13d
kube-node-lease        Active   13d
kube-public            Active   13d
kube-system            Active   13d
kubernetes-dashboard   Active   8d
my-namespace           Active   3s
[david@dhr-demo ~]$
```

当然我们还可以通过kubectl create命令方式来创建：

kubectl create namespace my–namespace

在指定的命名空间下创建对象

如果我们想在创建资源的时候就为其指定命名空间，可以在YAML文件中的metadata区段下添加namespace: my–namespace，或者在kubectl create命令中指定namespace选项：

kubectl create –f test1–manual.yaml –n my–namespace

在之前的章节中我们使用test1–manual.yaml文件创建了一个pod，由于没有明确指定该pod隶属于哪一个命名空间，所以默认是default命名空间。现在我们又创建了一个同名的pod，该pod位于my–namespace命名空间下。可以指定命名空间查看刚创建的pod：

```
[david@dhr-demo ~]$ kubectl get pod -n my-namespace
NAME          READY   STATUS    RESTARTS   AGE
test1-manual  1/1     Running   0          29s
```

如果不指定命名空间，kubectl命令根据其上下文中默认的命名空间来执行操作。我们可以通过kubectl config命令来修改kubectl当前上下文中的命名空间以及当前上下文本身。

如果想快速地切换到其他命名空间，我们可以设置一个快捷方式：

alias kcd='kubectl config set–context $(kubectl config current–context) ––namespace'

执行kubectl get po默认显示default命名空间下的pod：

kubectl get po

然后切换到my–namespace命名空间：

kcd my–namespace

再次执行kubectl get po，发现显示的是my–namespace下的pod：

```
[david@dhr-demo ~]$ alias kcd='kubectl config set-context $(kubectl config current-context) --namespace'
[david@dhr-demo ~]$ kubectl get po
NAME              READY   STATUS    RESTARTS   AGE
test1             1/1     Running   0          13d
test1-gpu         1/1     Running   0          2d21h
test1-manual      1/1     Running   0          6d9h
test1-manual-v2   1/1     Running   0          4d9h
[david@dhr-demo ~]$ kcd my-namespace
Context "minikube" modified.
[david@dhr-demo ~]$ kubectl get po
NAME           READY   STATUS    RESTARTS   AGE
test1-manual   1/1     Running   0          14m
[david@dhr-demo ~]$
```