Deleting a Namespace and All Its Objects

In this lesson, we will delete the Namespace and everything associated with it.

WE'LL COVER THE FOLLOWING

The Cascaded Deletion

Another handy feature of the Namespaces is their cascading effect. If, for example, we delete the `testing` Namespace, all the objects and the resources running inside it will be removed as well.

```
kubectl delete ns testing
kubectl -n testing get all
```

We deleted the `testing` Namespace and retrieved all the objects residing in it. If the resources were not deleted very quickly, the **output** is as follows.

```
NAME                                READY STATUS      RESTARTS AGE
po/go-demo-2-api-56dfb69dbd-8w6rf 0/1   Terminating 0        2m
po/go-demo-2-api-56dfb69dbd-hrr4b 0/1   Terminating 0        2m
po/go-demo-2-api-56dfb69dbd-ws855 0/1   Terminating 0        2m
po/go-demo-2-db-5b49cc946b-xdd6v  0/1   Terminating 0        2m
```

Please note that, in your case, the output might show more objects. If that's the case, you were too fast, and Kubernetes did not yet have time to remove them.

After a second or two, the only objects in the `testing` Namespace are the Pods with the status `terminating`. Once the grace period is over, they will be removed as well. The Namespace is gone, and everything we created in it was removed as well.

When it Comes in Handy?

The ability to remove a Namespace and all the objects and the resources it hosts is especially useful when we want to create temporary objects. A good example would be continuous deployment (CDP) processes. We can create a Namespace to build, package, test, and do all the other tasks our pipeline requires. Once we're finished, we can simply remove the Namespace. Otherwise, we would need to keep track of all the objects we created and make sure that they are removed before we terminate the CDP pipeline.

Verification

Now that the Namespace hosting our release 2.0 is gone, we might want to double check that the production release (1.0) is still running.

```
kubectl get all
```

The **output** should show the `go-demo-2` Deployments, ReplicaSets, Pods, and Services since we are still using the `default` context.

To be on the safe side, we'll check that a request coming from the `go-demo-2.com` domain still returns a response from the release 1.0.

```
curl -H "Host: go-demo-2.com" \
    "http://$(minikube ip)/demo/hello"
```

As expected, the response is `hello, release 1.0!`.

Command for CDP

If this were a continuous deployment pipeline, the only thing left would be to execute rolling updates that would change the image of the production release to `vfarcic/go-demo-2:2.0`. The command could be as follows.

```
kubectl set image \
    deployment/go-demo-2-api \
    api=vfarcic/go-demo-2:2.0 \
    --record
```

That was all about the deletion process of Namespaces.

It’s time for a quick tester!