Deploying to a New Namespace

In this lesson, we will deploy a release to our newly created Namespace.

WE'LL COVER THE FOLLOWING

Altering the Deployment Definition

As we explained in the previous lesson, the main objective of the deployment is to provide a means to test the release. It should remain hidden from our users.

The users should be oblivious to the existence of the new Deployment and continue using the release 1.0 until we are confident that 2.0 works as expected.

```
TAG=2.0

DOM=go-demo-2.com

cat ns/go-demo-2.yml \
    | sed -e \
    "s@image: $IMG@image: $IMG:$TAG@g" \
    | sed -e \
    "s@host: $DOM@host: $TAG\.$DOM@g" \
    | kubectl create -f -
```

Just as before, we used `sed` to alter the image definition. This time, we're deploying the tag `2.0`.

Apart from changing the image tag, we also modified the host. This time, the Ingress resource will be configured with the host `2.0.go-demo-2.com`. That will allow us to test the new release using that domain while our users will

continue seeing the production release 1.0 through the domain `go-demo-2.com`.

Verification

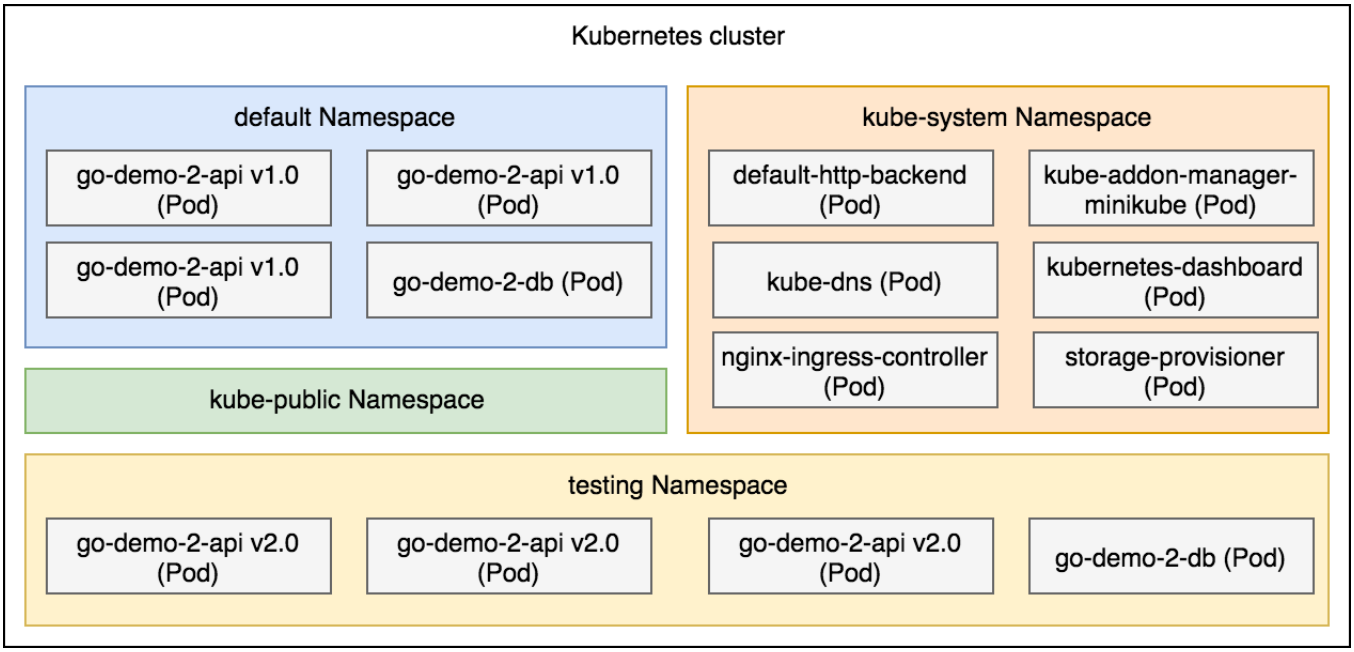
Let's confirm that the rollout finished.

```
kubectl rollout status \
    deploy go-demo-2-api
```

The **output** is as follows.

```
deployment "go-demo-2-api" successfully rolled out
```

As you can see, we rolled out the Deployment `go-demo-2-api`, along with some other resources. That means that we have two sets of the same objects with the same name. One is running in the `default` Namespace, while the other (release 2.0) is running in the `testing` Namespace.



The cluster with the new Namespace testing

Before we celebrate the successful deployment of the new release without affecting production, we should verify that both are indeed working as expected.

If we send a request to `go-demo-2.com`, we should receive a response from the release 1.0 running in the `default` Namespace.

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

The **output** is as follows.

```
hello, release 1.0!
```

If, on the other hand, we send a request to `2.0.go-demo-2.com`, we should get a response from the release 2.0 running in the `testing` Namespace.

```
curl -H "Host: 2.0.go-demo-2.com" \
    "http://$(minikube ip)/demo/hello"
```

The **output** is as follows.

```
hello, release 2.0!
```

Concluding Remarks

The result we accomplished through different Namespaces is very similar to what we'd expect by using separate clusters. The main difference is that we did not need to complicate things by creating a new cluster. We saved time and resources by using a new Namespace instead.

If this would be a "real world" situation, we'd run functional and other types of tests using the newly deployed release. Hopefully, those tests would be automated, and they would last for only a few minutes. We'll skip the testing part since it's not within the scope of this chapter. Instead, we'll imagine that the tests were executed and that they were successful.

Communication is an important subject when working with Namespaces, so in the next lesson, we'll spend a few moments exploring it.