Lab 3.3 - Running Your First Application

Creating a New Chart

Helm has a command **`helm create`** which you can use to create a new Helm chart which follows the best practices.

Step 1

Run the following command to create a sample chart:

```
$ helm create myapp
Creating myapp
```

This will create a directory called **`myapp`** in the current directory containing several Helm-specific files. We will not dive into the contents of the chart in this chapter, but feel free to take a look inside.

The example application in the boilerplate chart is a simple Nginx deployment (Nginx is a popular HTTP web server https://nginx.org/).

Installing the Chart

Step 2

Installing the chart will create a new Helm release in your Kubernetes cluster. So first come up with some unique release name, such as **demo**, and install the chart with the following command:


```
$ helm install demo myapp
NAME: demo
LAST DEPLOYED: Thu Apr 16 19:05:05 2020
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/name=myapp,app.kubernetes.io/instance=demo" -o
jsonpath="{.items[0].metadata.name}")
  echo "Visit http://127.0.0.1:8080 to use your application"
  kubectl --namespace default port-forward $POD_NAME 8080:80
```

That's it! You've just installed your first application in Kubernetes using Helm. If you run into any issues at this point, double check that you have access to a running Kubernetes cluster (see earlier in this chapter).

Step 3

Run a simple **`kubectl`** command to see a newly-created pod:

```
$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
demo-myapp-d96454b47-2q856     1/1     Running   0          24s
```

Accessing the Application

Many charts, upon install, will share some output regarding how to access the application (see previous section).

Step 4

In order to access the Nginx web server in this example, run the following commands which will forward local traffic on port 8080 to the pod:

```
$ export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/name=myapp,app.kubernetes.io/instance=demo" -o
jsonpath="{.items[0].metadata.name}")
$ kubectl --namespace default port-forward $POD_NAME 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
```


Note that this command will hang in the terminal (this is expected).

Step 5

With the command still running, open another terminal window on the same machine. Run the following command to access the running Nginx application:

```
$ curl http://127.0.0.1:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully
installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

In the original terminal, you'll notice some output:

```
Handling connection for 8080
```

This indicates that the request from the **`curl`** command was successfully forwarded to the running Nginx pod. Click "Ctrl+C" in the original terminal window to stop the port forwarding. You

can close the other terminal window.

Deleting a Release

Step 6

To delete the release from this example, run the following command:

```
$ helm delete demo
release "demo" uninstalled
```

Step 7

This will cause all Kubernetes resources associated with the release to be removed. For example, lets see if that pod is still there:

```
$ kubectl get pods
No resources found.
```

Gone! You've successfully deleted the release.

