



Exercise 10.1: Working with Helm and Charts

Overview

helm allows for easy deployment of complex configurations. This could be handy for a vendor to deploy a multi-part application in a single step. Through the use of a Chart, or template file, the required components and their relationships are declared. Local agents like **Tiller** use the API to create objects on your behalf. Effectively its orchestration for orchestration.

There are a few ways to install **Helm**. The newest version may require building from source code. We will download a recent, stable version. Once installed we will deploy a Chart, which will configure **MariaDB** on our cluster.

Install Helm

1. On the cp node use **wget** to download the compressed tar file. Various versions can be found here: <https://github.com/helm/helm/releases/>

```
student@cp:~$ wget https://get.helm.sh/helm-v3.6.0-linux-amd64.tar.gz
```

```
1 <output_omitted>
2 helm-v3.6.0-linux-a 100%[=====] 13.51M --.-KB/s in 0.1s
3
4 2021-06-11 03:18:50 (70.0 MB/s) - 'helm-v3.6.0-linux-amd64.tar.gz' saved [14168950/14168950]
```

2. Uncompress and expand the file.

```
student@cp:~$ tar -xvf helm-v3.6.0-linux-amd64.tar.gz
```

```
1 linux-amd64/
2 linux-amd64/helm
3 linux-amd64/README.md
4 linux-amd64/LICENSE
```

3. Copy the **helm** binary to the `/usr/local/bin/` directory, so it is usable via the shell search path.

```
student@cp:~$ sudo cp linux-amd64/helm /usr/local/bin/helm
```

4. A Chart is a collection of files to deploy an application. There is a good starting repo available on <https://github.com/kubernetes/charts/tree/master/stable>, provided by vendors, or you can make your own. Search the current Charts in the Helm Hub or an instance of Monocular for available stable databases. Repos change often, so the following output may be different from what you see.

```
student@cp:~$ helm search hub database
```

URL	APP VERSION	DESCRIPTION	CHART VERSION
https://artifacthub.io/packages/helm/drycc/data...			1.0.2
		A PostgreSQL database used by Drycc Workflow.	
https://artifacthub.io/packages/helm/drycc-cana...			1.0.0
		A PostgreSQL database used by Drycc Workflow.	
https://artifacthub.io/packages/helm/camptocamp...			0.0.6
	1.0	Expose services and secret to access postgres d...	
https://artifacthub.io/packages/helm/cnieg/h2-d...			1.0.3
	1.4.199	A helm chart to deploy h2-database	

```
11 <output_omitted>
```

5. You can also add repositories from various vendors, often found by searching artifacthub.io such as [ealenn](https://ealenn.github.io/charts), who has an echo program.

```
student@cp:~$ helm repo add ealenn https://ealenn.github.io/charts
```

```
1 "ealenn" has been added to your repositories
```

```
student@cp:~$ helm repo update
```

```
1 Hang tight while we grab the latest from your chart repositories...
2 ...Successfully got an update from the "ealenn" chart repository
3 Update Complete. Happy Helming!
```

6. We will install the **tester** tool. The **-debug** option will create a lot of output. The output will typically suggest ways to access the software.

```
student@cp:~$ helm upgrade -i tester ealenn/echo-server --debug
```

```
1 history.go:56: [debug] getting history for release tester
2 Release "tester" does not exist. Installing it now.
3 install.go:173: [debug] Original chart version: ""
4 install.go:190: [debug] CHART PATH: /home/student/.cache/helm/repository/echo-server-0.3.0.tgz
5
6 client.go:122: [debug] creating 4 resource(s)
7 NAME: tester
8 <output_omitted>
```

7. Ensure the newly created `tester-echo-server` pod is running. Fix any issues, if not.

8. Look for the newly created service. Send a **curl** to the ClusterIP. You should get a lot of information returned.

```
student@cp:~$ kubectl get svc
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	26h
tester-echo-server	ClusterIP	10.98.252.11	<none>	80/TCP	11m

```
student@cp:~$ curl 10.98.252.11
```

```
1 {"host":{"hostname":"10.98.252.11","ip":"","ffff:192.168.74.128","ips":
2 []},"http":{"method":"GET","baseUrl":"","originalUrl":"/","protocol":
3 "http"},"request":{"params":{"0":"/"},"query":{"},"cookies":{"},"body":
4 {}},"headers":{"host":"10.98.252.11","user-agent":"curl/7.58.0","accept":
5 "*/"},"environment":{"PATH":"/usr/local/sbin:/usr/local/bin:/usr/sbin:
6 /usr/bin:/sbin:/bin","TERM":"xterm","HOSTNAME":"tester-echo-server-
7 786768d9f4-4zsz9","ENABLE__HOST":"true","ENABLE__HTTP":"true","ENABLE__
8 <output_omitted>
```

9. View the Chart history on the system. The use of the **-a** option will show all Charts including deleted and failed attempts.

```
student@cp:~$ helm list
```

NAME	NAMESPACE	REVISION	UPDATED
STATUS	CHART	APP VERSION	
tester	default	1	2021-06-11 07:31:56.151628311 +0000 UTC
deployed	echo-server-0.3.0	0.4.0	

10. Delete the **tester** Chart. No releases of tester should be found.

```
student@cp:~$ helm uninstall tester
```

```
1 release "tester" uninstalled
```

```
student@cp:~$ helm list
```

```
1 NAME          NAMESPACE      REVISION      UPDATED             STATUS      CHART          APP VERSION
```

11. Find the downloaded chart. It should be a compressed tarball under the user's home directory. Your **echo** version may be slightly different.

```
student@cp:~$ find $HOME -name *echo*
```

```
1 /home/student/.cache/helm/repository/echo-server-0.3.0.tgz
```

12. Move to the archive directory and extract the tarball. Take a look at the files within.

```
student@cp:~$ cd $HOME/.cache/helm/repository ; tar -xvf echo-server-*
```

```
1 echo-server/Chart.yaml
2 echo-server/values.yaml
3 echo-server/templates/_helpers.tpl
4 echo-server/templates/configmap.yaml
5 echo-server/templates/deployment.yaml
6 <output_omitted>
```

13. Examine the `values.yaml` file to see some of the values that could have been set.

```
student@cp:~/.cache/helm/repository$ cat echo-server/values.yaml
```

```
1 <output_omitted>
```

14. You can also download and examine or edit the values file before installation. Add another repo and download the Bitnami Apache chart.

```
student@cp:~$ helm repo add bitnami https://charts.bitnami.com/bitnami
```

```
student@cp:~$ helm fetch bitnami/apache --untar
```

```
student@cp:~$ cd apache/
```

15. Take a look at the chart. You'll not it looks similar to the previous. Read through the `:values.yaml`:

```
student@cp:~$ ls
```

```
1 Chart.lock Chart.yaml README.md charts ci files templates
2 values.schema.json values.yaml
```

```
student@cp:~$ less values.yaml
```

```
1 ## Global Docker image parameters
2 ## Please, note that this will override the image parameters, including dependencies, configured....
3 ## Current available global Docker image parameters: imageRegistry and imagepullSecrets
4 ##
5 # global:
6 #   imageRegistry: myRegistryName
7 #   imagePullSecrets:
8 #     - myRegistryKeySecretName
9 <output_omitted>
```

16. Use the `values.yaml` file to install the chart. Take a look at the output and ensure the pod is running.

```
student@cp:~$ helm install anotherweb .
```

```
1 NAME: anotherweb
2 LAST DEPLOYED: Fri Jun 11 08:11:10 2021
3 NAMESPACE: default
4 STATUS: deployed
5 REVISION: 1
6 TEST SUITE: None
7 <output_omitted>
```

17. Test the newly created service. You should get an HTML response saying It works!
18. Remove anything you have installed using **helm**. Reference earlier in the chapter if you don't remember the command. We will use **helm** again in another lab.