Helm Workshop

Prerequisites

Run through the prereqs from Ben's kubenernetes workshop, specifically make sure you have kubectl installed and the kube config and the docker config.
https://github.com/benmathews/KubernetesWorkshop

Helm Installation

Mac Os

Install via brew:

```
brew install helm
```

Linux

Install via snap:

```
sudo snap install helm --classic
```

Alternative

Download a binary from https://github.com/helm/helm/releases

helm concepts

chart

Helm package that includes everything to run an app, tool, or service in a k8s cluster. (deployment, service, ingress, etc.) Think of it like a homebrew formula or an apt dpkg.

repository

A place where charts can be collected and shared.

release

An instance of a chart running in a kubernetes cluster. One chart could be installed multiple times and each time a release is created. If you want two copies of mongo running in your cluster, you can install the mongo chart twice and each will have its own release with its own release name.

find and install a mongo chart in your namespace and then uninstall it

Add the bitnami repo

```
helm repo add "bitnami" "https://charts.bitnami.com/bitnami"
```

See what is available in the repo

```
helm search repo mongo
```

Install mongo

```
helm install mongo bitnami/mongodb
```

Check that stuff is running

```
kubectl get po
kubectl get deploy
kubectl get service
kubectl port-forward --namespace t svc/mongo-mongodb 27017:27017 &
mongo localhost
```

Check our our list of releases

`helm list`

Uninstall our mongo release using the release name

`helm uninstall mongo`

build a familiar demowebapp and push the image

```
cd demowebapp
docker build . -t 10.1.31.199:32000/<your name>/demowebapp:v1.0.0
docker push 10.1.31.199:32000/<your name>/demowebapp:v1.0.0
```

create our chart

`helm create demowebapp`

Take a look at what's been generated by helm

```
├── Chart.yaml    # Information about your chart
├── values.yaml   # The default values for your templates
├── charts/       # Charts that this chart depends on
└── templates/    # The template files
    └── tests/    # The test files
```

Edit `values.yaml` so that our chart deploys our demowebapp instead of nginx

```
repository: localhost:32000/<your-name>/demowebapp
tag: v1.0.0
```

Try out these commands before installing your release

`helm lint .` lints your chart for errors. this can be very helpful for diagnosing whitespace errors -- helm is very specific about whitespace (2 spaces, not tabs!)

`helm template .` shows your what your entire chart looks like after templating

`helm install --dry-run .` do a dry run of the chart installation

`helm install <your-release-name> .` install this chart!

Check that things are working:

```
kubectl port-forward <pod name> 8090:8080
```

- Navigate to our service in your browser.

Exercises:

- Add the environment variable FRIENDS to the deployment and make that configurable in values.yaml, then redeploy and confirm that the configured value gets returned by our service. You can also set variables at install time like this:
  `helm install <my-release> . --set <variable>=<value>`

values.yaml

```
app:
  friends: "ben and thomas"
```

deployment.yaml (in the containers section)

```
env:
- name: "FRIENDS"
  value: {{ .Values.app.friends | quote }}
```

result

```
Hello World from 10.1.106.125 and your friends at ben and thomas"
```

Templating example:

Let's make it so we can specify a list of friends in values.yaml like so

values.yaml

```
app:
  friends:
  - "ben"
  - "thomas"
```

Let's make it so that if it's not specified, there is a default of "all your friends at vivint".
Also because we like shouting let's make it turn all of our friends uppercase.
You can find the pipelines needed here:
https://helm.sh/docs/chart_template_guide/functions_and_pipelines/

deployment.yaml

```
{{- define "friends" -}}
{{- join "," .Values.app.friends | default "all your friends at vivint" | upper | quote }}
{{- end -}}

# new value for our env var
value: {{ template "friends" . }}
```

Using helm at vivint:

https://source.vivint.com/projects/PL/repos/servicegen/browse generates services with a basic chart for you.

`helm lint`, `helm template`, `helm install --dry-run` are very useful for debugging and testing your chart without needed to go through mrmeseeks.

Additional templating exercise

Change our friends template to have it set several environment variables, friend1, friend2, etc instead of a single list

Hints: https://helm.sh/docs/chart_template_guide/control_structures/ <-- check out the section about the `range keyword`

Useful links

More template/pipeline examples:
https://helm.sh/docs/chart_template_guide/functions_and_pipelines/
Useful helm tips/tricks: https://helm.sh/docs/howto/charts_tips_and_tricks/
More on helm templating: https://helm.sh/docs/chart_template_guide/