HELM

Creating Helm charts

IBM Cloud

Creating a chart

Generates a directory with sample files:

```
$ helm create my-chart
$ tree my-chart
```

```
my-chart/               # The content of this directory is the chart
    |- chart.yaml        # Information about the chart
    |- values.yaml       # The default configuration values for the chart
    |- charts/           # Charts that this chart depends on
    |- templates/        # The chart's template files
       |- NOTES.txt      # OPTIONAL: A plain text file containing short usage notes
       |- _helpers.tpl       # OPTIONAL: The default location for template partials
       |- deployment.yaml    # Sample template for a deployment resource
       |- service.yaml       # Sample template for a service resource
```

By default, a chart starts with sample templates for a Kubernetes deployment and service. In the simplest case, edit `values.yaml` file.

How Install Uses Charts

The main step of installing a chart is rendering its templates.

How Helm installs a chart:

1. User runs an install in the Helm CLI

   ```
   $ helm install myapp
   ```

2. Helm CLI loads the chart into Tiller
3. Tiller renders the chart templates
4. Tiller loads the resulting resources into Kubernetes
5. Tiller returns the release data to the client
6. The client exits

Rendering the templates:

1. Each template generates a Kubernetes resource manifest file (yaml)
2. Tiller runs each of the template files generating the resource files
3. Tiller then loads the resources—as described by the manifests—into the Kubernetes cluster

Chart lifecycle hooks

Hook	Description
`pre-install`	• Executes after templates are rendered • Before any resources are created in Kubernetes
`post-install`	• Executes after all resources are loaded into Kubernetes
`pre-delete`	• Executes before any resources are deleted from Kubernetes
`post-delete`	• Executes after all of the release's resources have been deleted
`pre-upgrade`	• Executes after templates are rendered • Before any resources are loaded into Kubernetes
`post-upgrade`	• Executes after all resources have been upgraded
`pre-rollback`	• Executes after templates are rendered • Before any resources have been rolled back
`post-rollback`	• Executes after all resources have been modified

Hooks

Can be any Kubernetes resource
Often a Kubernetes job
Resides in the `templates` directory

Chart lifecycle hooks (continued)

Hooks in the Helm install lifecycle:

1. User runs an install in the Helm CLI
2. Helm CLI loads the chart into Tiller
3. Tiller renders the chart templates
4. **Tiller executes the *pre-install hooks***
5. Tiller loads the resulting resources into Kubernetes
6. **Tiller executes the *post-install hook***
7. Tiller returns the release data to the client
8. The client exits

Sharing charts

A chart is a directory:

- Easy for a Helm client to use the chart directories on the same computer
- Difficult to share with other users on other computers

Packaging a chart:

- Bundle `chart.yaml` and related files into a tar file

```
$ helm package <chart-path>.    # Bundles chart directory into a tar file
$ helm install <chart-name>.tgz # Installs the chart in the chart file
```

Chart repository:

- HTTP server that houses an `index.yaml` and optionally some packaged charts
- Server can be any HTTP server that can serve YAML and tar files and can answer GET requests
  Ex: Google Cloud Storage (GCS) bucket, Amazon S3 bucket, Github Pages, or even create your own web server
- To add a chart to the repository, copy it to the directory and regenerate the index

```
$ helm repo index <charts-path>  # Generates index of the charts in the repo
```

Creating templates

The main aspect of implementing a chart is implementing its templates.

A related task: Create and populate the settings files used by the templates.

- These files, specifically `values.yaml`, define the chart's API
- The settings files list the variables the templates can use, therefore the only values worth changing

Examples of chart templates can be found in https://github.com/kubernetes/charts/.

- Each file is a Golang template
- Includes functions from the Sprig template library
- A template can create the manifest for any type of Kubernetes resource

Each file in a chart's `templates` directory is expected to be a template.

- Expected to generate a Kubernetes resource manifest
- Filename can be anything, should describe the resource it defines
- Exception: The notes file (`NOTES.txt`) provides instructions to the chart's users
- Exception: Files whose names begin with an underscore (`_helpers.tpl`) are expected to contain partials

Chart template for deployment manifest

Kubernetes deployment manifest:

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

Helm deployment template:

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: {{ template "fullname" . }}
  labels:
    app: {{ template "name" . }}
    chart: {{ .Chart.Name }}-{{ .Chart.Version }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
spec:
  replicas: {{ .Values.replicaCount }}
  template:
    metadata:
{{- if .Values.podAnnotations }}
      annotations:
{{ toYaml .Values.podAnnotations | indent 8 }}
{{- end }}
      labels:
        app: {{ template "name" . }}
        release: {{ .Release.Name }}
    spec:
      containers:
        - name: {{ template "name" . }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
          imagePullPolicy: {{ .Values.image.pullPolicy }}
           ports:
          - name: http
            containerPort: 80
            protocol: TCP
. . .
```

Chart template for service manifest

Kubernetes service manifest:

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Helm service template:

```
apiVersion: v1
kind: Service
metadata:
{{- if .Values.service.annotations }}
  annotations:
{{ toYaml .Values.service.annotations | indent 4 }}
{{- end }}
  name: {{ template "fullname" . }}
  labels:
    app: {{ template "name" . }}
    chart: {{ .Chart.Name }}-{{ .Chart.Version }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
spec:
  selector:
    app: {{ template "name" . }}
    release: {{ .Release.Name }}
  ports:
    - name: http
      protocol: TCP
      port: {{ .Values.service.port }}
      targetPort: http
      {{- if (and (eq .Values.service.type "NodePort") ...) }}
      nodePort: {{ .Values.service.nodePort }}
      {{- end }}
. . .
```

values.yaml – A chart's API

Values (values.yaml):

```
replicaCount: 1
restartPolicy: Never
# Evaluated by the post-install hook
sleepyTime: "10"
index: >-
  <h1>Hello</h1>
  <p>This is a test</p>
image:
  repository: nginx
  tag: 1.11.0
  pullPolicy: IfNotPresent
service:
  annotations: {}
  clusterIP: ""
  externalIPs: []
  loadBalancerIP: ""
  loadBalancerSourceRanges: []
  type: ClusterIP
  port: 8888
  nodePort: ""
podAnnotations: {}
resources: {}
nodeSelector: {}
```

Helm deployment template:

```
. . .
spec:
  replicas: {{ .Values.replicaCount }}
  template:
    metadata:
{{- if .Values.podAnnotations }}
      annotations:
{{ toYaml .Values.podAnnotations | indent 8 }}
{{- end }}
. . .
```

Helm service template:

```
. . .
spec:
  ports:
    - name: http
      protocol: TCP
      port: {{ .Values.service.port }}
      targetPort: http
      {{- if (and (eq .Values.service.type
"NodePort") ...) }}
      nodePort: {{ .Values.service.nodePort }}
      {{- end }}
. . .
```

chart.yaml – A chart's meta information

Chart (chart.yaml):

```
name: nginx
description: A basic NGINX HTTP server
version: 0.1.0
keywords:
  - http
  - nginx
  - www
  - web
home: https://github.com/kubernetes/helm
sources:
  - https://hub.docker.com/_/nginx/
maintainers:
  - name: technosophos
    email: mbutcher@deis.com
```

Helm template:

```
. . .
metadata:
{{- if .Values.service.annotations }}
  annotations:
{{ toYaml .Values.service.annotations | indent 4 }}
{{- end }}
  name: {{ template "fullname" . }}
  labels:
    app: {{ template "name" . }}
    chart: {{ .Chart.Name }}-{{ .Chart.Version }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
. . .
```

Chart template helpers – More default settings

Helpers (`templates/_helpers.tpl`):

```
{{/* vim: set filetype=mustache: */}}
{{/* Expand the name of the chart. */}}
{{- define "name" -}}
{{- default .Chart.Name .Values.nameOverride | trunc 63 | trimSuffix "-" -}}
{{- end -}}
{{/* Create a default fully qualified app name. We truncate at 63 chars because . . . */}}
{{- define "fullname" -}}
{{- $name := default .Chart.Name .Values.nameOverride -}}
{{- printf "%s-%s" .Release.Name $name | trunc 63 | trimSuffix "-" -}}
{{- end -}}
```

Helm template:

```
. . .
metadata:
  name: {{ template "fullname" . }}
  labels:
    app: {{ template "name" . }}
    chart: {{ .Chart.Name }}-{{ .Chart.Version }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
. . .
```

Chart predefined values – More default settings

Predefined values:

Release – Information about the release being created
- Release.Name – The name of the release (not the chart)
- Release.Service – The service that conducted the release, normally Tiller
- Release.Revision – The revision number. Begins at 1, and increments with each helm upgrade

Chart – The contents of the chart.yaml
- Chart.Name
- Chart.Version
- Chart.Maintainers

Files – Map of all non-special files in the chart

Capabilities – Map of info about Kubernetes and Helm
- Capabilities.KubeVersion
- Capabilities.TillerVersion
- Capabilities.APIVersions

Template – Information about the current template

Helm Template:

```
. . .
metadata:
{{- if .Values.service.annotations }}
  annotations:
{{ toYaml .Values.service.annotations | indent 4 }}
{{- end }}
  name: {{ template "fullname" . }}
  labels:
    app: {{ template "name" . }}
    chart: {{ .Chart.Name }}-{{ .Chart.Version }}
    heritage: {{ .Release.Service }}
    release: {{ .Release.Name }}
. . .
```

Resources – Developing charts

Helm examples
https://github.com/kubernetes/helm/tree/master/docs/examples

Stable Helm charts
https://github.com/kubernetes/charts/tree/master/stable

Golang templates
https://golang.org/pkg/text/template

Sprig template library
https://godoc.org/github.com/Masterminds/sprig

Getting Started Authoring Helm Charts
https://deis.com/blog/2016/getting-started-authoring-helm-charts

How to Create Your First Helm Chart
https://docs.bitnami.com/kubernetes/how-to/create-your-first-helm-chart

Packaged Kubernetes Deployments – Writing a Helm Chart
https://www.influxdata.com/packaged-kubernetes-deployments-writing-helm-chart