Helm CheatSheet

Get Started

- https://deis.com/blog/2016/getting-started-authoring-helm-charts/
- https://docs.bitnami.com/kubernetes/how-to/
- https://github.com/kubernetes/helm/blob/master/docs/charts.md
- https://docs.helm.sh/chart-template-guide/
- http://helm.readthedocs.io/en/latest/architecture/

Struture

```
.
├── Chart.yaml --> metadata info
├── README.md
├── requirements.yaml --> define dependencies
├── templates
│   ├── spark-master-deployment.yaml --> configuration with template supported
│   ├── spark-worker-deployment.yaml
│   └── spark-zeppelin-deployment.yaml
│   └── NOTES.txt --> display when run "helm chart"
│   └── _helpers.tpl --> template handler
```

```
└── values.yaml --> variable list, will be interpolated on templates file during deployment
│
└── charts
    ├── apache/
        ├── Chart.yaml
```

- Chart.yaml

```
name: The name of the chart (required)
version: A SemVer 2 version (required)
description: A single-sentence description of this project (optional)
keywords:
  - A list of keywords about this project (optional)
home: The URL of this project's home page (optional)
sources:
  - A list of URLs to source code for this project (optional)
maintainers: # (optional)
  - name: The maintainer's name (required for each maintainer)
    email: The maintainer's email (optional for each maintainer)
engine: gotpl # The name of the template engine (optional, defaults to gotpl)
icon: A URL to an SVG or PNG image to be used as an icon (optional).
appVersion: The version of the app that this contains (optional). This needn't be SemVer.
deprecated: Whether or not this chart is deprecated (optional, boolean)
tillerVersion: The version of Tiller that this chart requires. This should be expressed as
```

- **requirements.yaml**

Adding an `alias` for a dependency chart would put a chart in dependencies using alias as name of new dependency. `Condition` - The condition field holds one or more YAML paths (delimited by commas). If this path exists in the top parent's values and resolves to a boolean value, the chart will be enabled or disabled based on that boolean value. Only the first valid path found in the list is evaluated and if no paths exist then the condition has no effect. `Tags` - The tags field is a YAML list of labels to associate with this chart. In the top parent's values, all charts with tags can be enabled or disabled by specifying the tag and a boolean value. `Conditions` (when set in values) always override `tags`

```
dependencies:
- name: apache
  version: 1.2.3
  repository: http://example.com/charts
  alias: new-subchart-1
  condition: subchart1.enabled, global.subchart1.enabled
      tags:
        - front-end
        - subchart1

- name: mysql
  version: 3.2.1
```

```
  repository: http://another.example.com/charts
  alias: new-subchart-2
  condition: subchart2.enabled,global.subchart2.enabled
      tags:
        - back-end
        - subchart1
```

General Usage

```
helm list --all
helm repo (list|add|update)
helm search
helm inspect <chart-name>
hem install --set a=b -f config.yaml <chart-name> -n <release-name> # --set take precedent
helm status <deployment-name>
helm delete <deployment-name>
helm inspect values <chart-name>
helm upgrade -f config.yaml <deployment-name> <chart-name>
helm rollback <deployment-name> <version>

helm create <chart-name>
helm package <chart-name>
helm lint <chart-name>

helm dep up <chart-name> # update dependency
helm get manifest <deployment-name> # prints out all of the Kubernetes resources that were
helm install --debug --dry-run <deployment-name> # it will return the rendered template to
```

- --set outer.inner=value is translated into this:

```
outer:
inner: value
```

- --set servers[0].port=80,servers[0].host=example:

```
servers:
- port: 80
  host: example
```

- --set name={a, b, c} translates to:

```
name:
- a
```

```
- b
- c
```

- --set name=value1,value2:

```
name: "value1,value2"
```

- --set nodeSelector."kubernetes.io/role"=master

```
nodeSelector:
kubernetes.io/role: master
```

- --set livenessProbe.exec.command=[cat,docroot/CHANGELOG.txt] --set livenessProbe.httpGet=null

```
livenessProbe:
-  httpGet:
-    path: /user/login
-    port: http
  initialDelaySeconds: 120
+  exec:
+    command:
+    - cat
+    - docroot/CHANGELOG.txt
```

- --timeout
- --wait
- --no-hooks
- --recreate-pods

Template

Values that are supplied via a `values.yaml` file (or via the --set flag) are accessible from the `.Values` object in a template

```
Release.Name:
Release.Time:
Release.Namespace: The namespace the chart was released to.
Release.Service: The service that conducted the release. Usually this is Tiller.
Release.IsUpgrade: This is set to true if the current operation is an upgrade or rollback.
Release.IsInstall: This is set to true if the current operation is an install.
Release.Revision: The revision number. It begins at 1, and increments with each helm upgrade
Chart: The contents of the Chart.yaml. Thus, the chart version is obtainable as "Chart.Versi
Files: Files can be accessed using {{index .Files "file.name"}} or using the "{{.Files.Get n
Capabilities: "({{.Capabilities.KubeVersion}}", Tiller "({{.Capabilities.TillerVersion}}", a
```

```
{{.Files.Get config.ini}}
{{.Files.GetBytes}} useful for things like images

{{.Template.Name}}
{{.Template.BasePath}}
```

- default value

```
{{default "minio" .Values.storage}}

//same
{{ .Values.storage | default "minio" }}
```

- put a quote outside

```
heritage: {{.Release.Service | quote }}

# same result
heritage: {{ quote .Release.Service }}
```

- global variable

```
global:
  app: MyWordPress

// could be access as "{{.Values.global.app}}"
```

- Includes a template called `mytpl.tpl`, then `lowercases` the result, then wraps that in `double quotes`

```
value: {{include "mytpl.tpl" . | lower | quote}}
```

- `required` function declares an entry for `.Values.who` is required, and will print an `error message` when that entry is missing

```
value: {{required "A valid .Values.who entry required!" .Values.who }}
```

- The `sha256sum` function can be used together with the `include` function to ensure a deployments template section is updated if another spec changes

```
kind: Deployment
spec:
  template:
```

```
      metadata:
        annotations:
          checksum/config: {{ include (print $.Template.BasePath "/secret.yaml") . | sha256sum
[...]
```

- The annotation `"helm.sh/resource-policy": keep` instructs `Tiller` to skip this resource during a helm `delete` operation
- In the `templates/` directory, any file that begins with an `underscore(_)` is `not` expected to output a Kubernetes manifest file. So by convention, `helper templates` and `partials` are placed in a `_helpers.tpl` file.

Hooks

Read more

- include these annotation inside hook yaml file, for e.g `templates/post-install-job.yaml`

```
apiVersion: batch/v1
kind: Job
metadata:
  annotations:
    # This is what defines this resource as a hook. Without this line, the
    # job is considered part of the release.
    "helm.sh/hook": post-install, post-upgrade
    "helm.sh/hook-weight": "-5"
```

Chart Repository

Read more

Signing

Read more

Test

Read more

Flow Control

If/Else

```
{{ if PIPELINE }}
  # Do something
{{ else if OTHER PIPELINE }}
  # Do something else
{{ else }}
  # Default case
{{ end }}

data:
  myvalue: "Hello World"
  drink: {{ .Values.favorite.drink | default "tea" | quote }}
  food: {{ .Values.favorite.food | upper | quote }}
  {{- if eq .Values.favorite.drink "lemonade" }}
  mug: true
  {{- end }} # notice the "-" in the left, if will help eliminate newline before variable
```

With

`with` can allow you to set the current `scope` (.) to a particular object

```
data:
  myvalue: "Hello World"
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  {{- end }} # instead of writing ".Values.favorite.drink"
```

`Inside` of the `restricted scope`, you will `not` be able to access the other objects from the `parent scope`

Range

```
# predefined variable
pizzaToppings:
  - mushrooms
  - cheese
  - peppers
  - onions

toppings: |-
    {{- range $i, $val := .Values.pizzaTopping }}
    - {{ . | title | quote }}  # upper first character, then quote
    {{- end }}

sizes: |-
    {{- range tuple "small" "medium" "large" }}
```

```
    - {{ . }}
    {{- end }} # make a quick list
```

Variables

It follows the form `$name`. Variables are assigned with a special assignment operator: `:=`

```
data:
  myvalue: "Hello World"
  {{- $relname := .Release.Name -}}
  {{- with .Values.favorite }}
  drink: {{ .drink | default "tea" | quote }}
  food: {{ .food | upper | quote }}
  release: {{ $relname }}
  {{- end }}

# use variable in range
 toppings: |-
    {{- range $index, $topping := .Values.pizzaToppings }}
      {{ $index }}: {{ $topping }}
    {{- end }}

#toppings: |-
#      0: mushrooms
#      1: cheese
#      2: peppers
#      3: onions

{{- range $key,$value := .Values.favorite }}
  {{ $key }}: {{ $value }}
  {{- end }} # instead of specify the key, we can actually loop through the values.yaml file
```

There is one variable that is always global - $ - this variable will always point to the root context

```
...
labels:
    # Many helm templates would use `.` below, but that will not work,
    # however `$` will work here
    app: {{ template "fullname" $ }}
    # I cannot reference .Chart.Name, but I can do $.Chart.Name
    chart: "{{ $.Chart.Name }}-{{ $.Chart.Version }}"
    release: "{{ $.Release.Name }}"
    heritage: "{{ $.Release.Service }}"
...
```

Named Templates

template names are **`global`**

```
# _helpers.tpl
{{/* Generate basic labels */}}
{{- define "my_labels" }}
  labels:
    generator: helm
    date: {{ now | htmlDate }}
    version: {{ .Chart.Version }}
    name: {{ .Chart.Name }}
{{- end }}
```

When a named template (created with define) is rendered, it will receive the scope passed in by the template call.

```
# configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
  {{- template "my_labels" . }} # Notice the final dot, it will pass the global scope inside
  {{- include "my_labels" . | indent 2 }} # similar to "template" directive, have the abilit
```

referable to use `include` over `template`. Because `template` is an `action`, and not a `function`, there is no way to pass the output of a template call to other functions; the data is simply inserted `inline`.

Files inside Templates

```
# file located at parent folder
# config1.toml: |-
#   message = config 1 here
# config2.toml: |-
#   message = config 2 here
# config3.toml: |-
#   message = config 3 here

data:
  {{- $file := .Files }} # set variable
  {{- range tuple "config1.toml" "config2.toml" "config3.toml" }} # create list
  {{ . }}: |- # config file name
    {{ $file.Get . }} # get file's content
  {{- end }}
```

Glob-patterns & encoding

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: conf
data:
+{{ (.Files.Glob "foo/*").AsConfig | indent 2 }}
---
apiVersion: v1
kind: Secret
metadata:
  name: very-secret
type: Opaque
data:
+{{ (.Files.Glob "bar/*").AsSecrets | indent 2 }}

+token: |-
+  {{ .Files.Get "config1.toml" | b64enc }}
```

YAML reference

```
# Force type
age: !!str 21
port: !!int "80"

# Fake first line to preserve integrity
coffee: | # ⏎no strip
  # Commented first line
         Latte
  Cappuccino
  Espresso

coffee: |- # strip off trailing newline
  Latte
  Cappuccino
  Espresso

coffee: |+ # preserve trailing newline
  Latte
  Cappuccino
  Espresso


another: value

myfile: | # insert static file
{{ .Files.Get "myfile.txt" | indent 2 }}
```

```
    coffee: > ⍰# treat as one long line
      Latte
      Cappuccino
      Espresso
```