Tanzu®
by Broadcom

Helm Library Charts

F. Sauerborn

21/2/2024

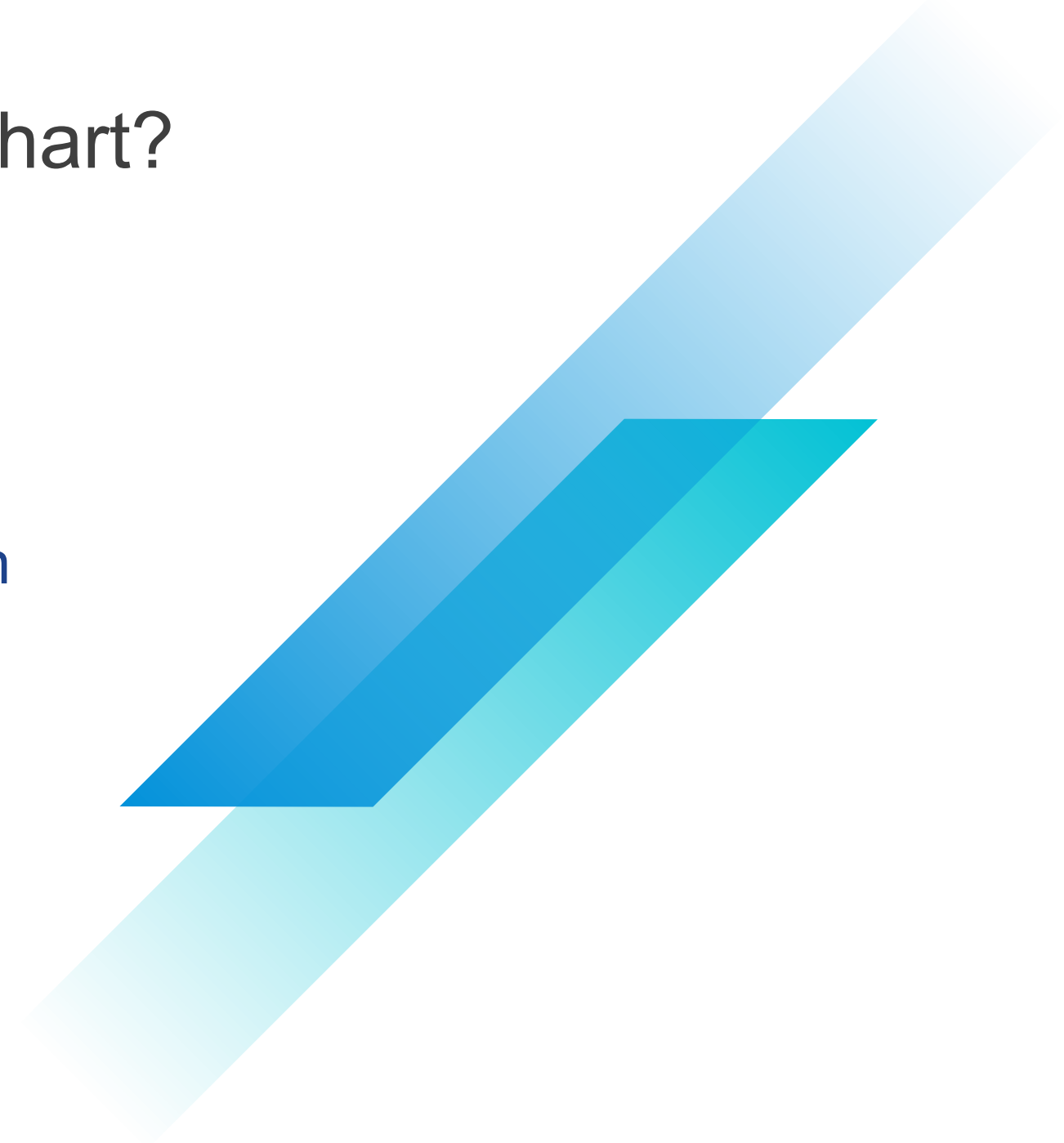
Agenda

1. What is a Helm Library Chart?
2. Why is it important?
3. How to create it
4. Examples
    - Extracting a named template into a library chart
    - Templating whole Chart
    - Overwriting parts of the library
5. Q&A

What is a Helm Library Chart?

New type of Helm Chart introduced in Helm 3 which can be shared by Helm templates in other charts.

Source: Helm Library Charts



Why is it important?

- avoid duplication and keeping your charts DRY
- clear distinction between library and application charts
- library Charts are none installable charts
- allow to use the importer’s context

How to create it

Create a library Chart

1. Create a new Chart

```
$ helm create examplelibrarychart
```

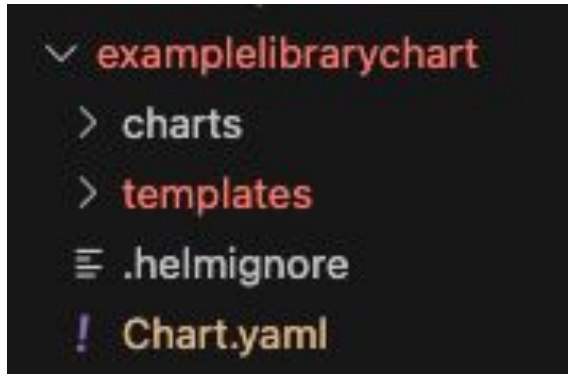
2. Change the type in the *Chart.yaml* to library

```
type: library
```

3. Get rid of unnecessary files

```
$ rm -r examplelibrarychart/templates/*
$ rm examplelibrarychart/values.yaml
```

Finally the folder structure should be like the following:



Where *charts* and *templates* folders are empty.

And the *Chart.yaml* (without comments) looks like:

Examples

Extracting a named template into a library chart

Example: Extracting a named template into a library chart

Create a named template in the library Chart

1. Create a new template file
   In our example `_helpers.tpl`

2. Create the named template `examplelibrarychart.chart` in the template file

```
{{/*
Create chart name and version as used by the chart label.
*/}}
{{- define "examplelibrarychart.chart" -}}
{{- printf "%s-%s" .Chart.Name .Chart.Version | replace "+" "_" | trunc 63 |
trimSuffix "-" }}
{{- end }}
```

Example: Extracting a named template into a library chart

Create a named template in the library Chart

3. Create the named template `examplelibrarychart.labels` in the template file

```
{{/*
Common labels
*/}}
{{- define "examplelibrarychart.labels" -}}
helm.sh/chart: {{ include "examplelibrarychart.chart" . }}
{{ include "examplelibrarychart.selectorLabels" . }}
{{- if .Chart.AppVersion }}
app.kubernetes.io/version: {{ .Chart.AppVersion | quote }}
{{- end }}
app.kubernetes.io/managed-by: {{ .Release.Service }}
{{- end }}
```

Example: Extracting a named template into a library chart

Use the library Chart from an application Chart

1. Create a new Chart

```
$ helm create examplechartusingthelibrary
```

2. Get rid of unnecessary files

```
$ rm -r examplelibrarychart/templates/*
```

3. Add the library Chart as dependency in *Chart.yaml* specify the version which we used during the creation and since the folder is next to our chart we can reference it locally

examplechartusingthelibrary > ! Chart.yaml > ...

```
25
26  # add the library chart as dependency
27  dependencies:
28  - name: examplelibrarychart
29    version: 0.1.0
30    repository: file://../examplelibrarychart
```

4. Finally we add a deployment template which uses our named template from the library:

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: examplechartusingthelibrary-deployment
5    labels:
6      {{- include "examplelibrarychart.labels" . | nindent 4 }}
7  spec:
8    template:
9      metadata:
10       labels:
11         {{- include "examplelibrarychart.labels" . | nindent 8 }}
12     spec:
13       containers:
14         - name: examplechartusingthelibrary
15           image: "nginx:1.16.0"
16           ports:
17             - name: http
18               containerPort: 80
19               protocol: TCP

"examplechartusingthelibrary/templates/deployment.yaml" 19L, 484B
```

Example: Extracting a named template into a library chart

Use the library Chart from an application Chart

5. Retrieve the library

```
$ helm dependency update
```

6. Verify it gets rendered as expected

```
$ helm install mydemo \
  examplechartusingthelibrary \
  --dry-run --debug
```

Should return the template filled as shown on the right

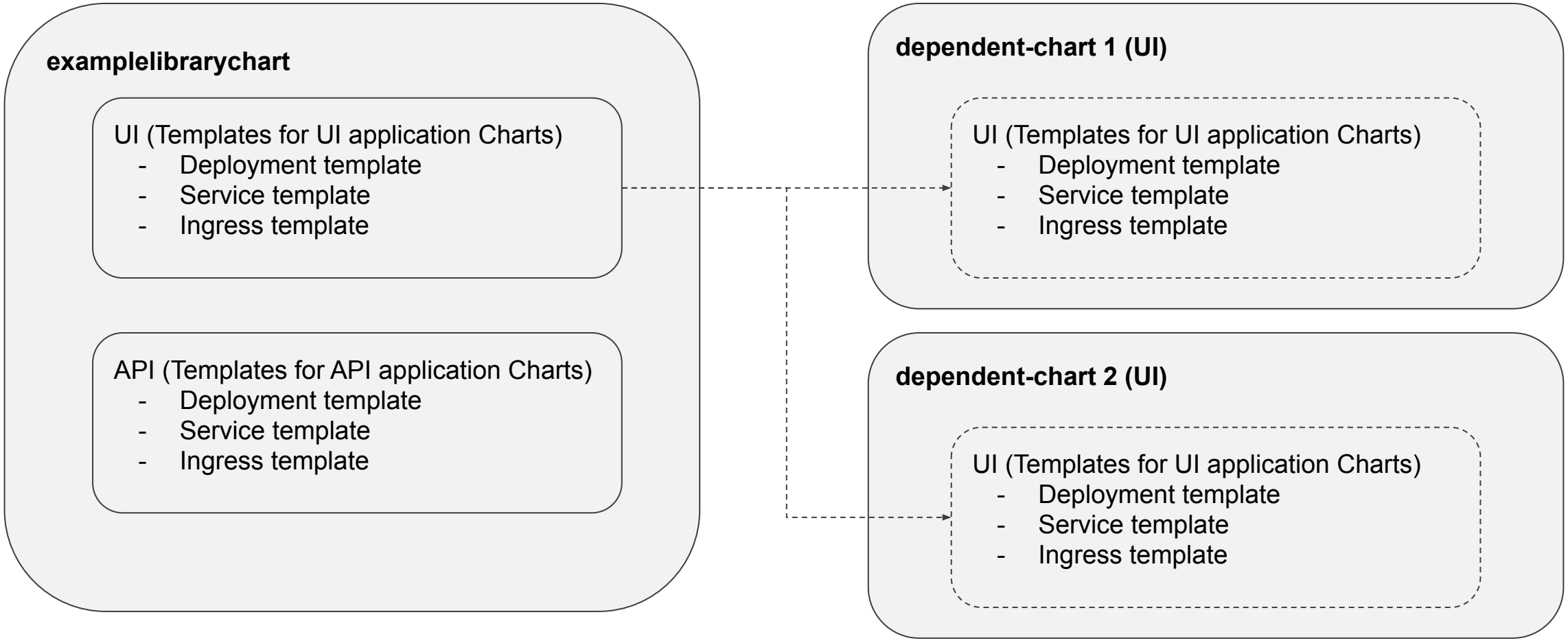
```
# Source: examplechartusingthelibrary/templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: examplechartusingthelibrary-deployment
  labels:
    helm.sh/chart: examplechartusingthelibrary-0.1.0
    app.kubernetes.io/name: examplechartusingthelibrary
    app.kubernetes.io/instance: mydemo
    app.kubernetes.io/version: "1.16.0"
    app.kubernetes.io/managed-by: Helm
spec:
  template:
    metadata:
      labels:
        helm.sh/chart: examplechartusingthelibrary-0.1.0
        app.kubernetes.io/name: examplechartusingthelibrary
        app.kubernetes.io/instance: mydemo
        app.kubernetes.io/version: "1.16.0"
        app.kubernetes.io/managed-by: Helm
    spec:
      containers:
        - name: examplechartusingthelibrary
          image: "nginx:1.16.0"
          ports:
            - name: http
              containerPort: 80
              protocol: TCP
```

Examples

Templating whole Chart

Example: Templating whole Chart

Managing multiple charts (fully depend on the library, not having any own templates)



Example: Templating whole Chart

Create named templates as wrapper for each template you want to provide through the library.

For example for all UI related templates this could be

a. Deployment

```
{{- define "examplelibrarychart.ui.deployment" -}}
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "examplelibrarychart.fullname" . }}
  labels:
    {{- include "examplelibrarychart.labels" . | nindent 4 }}
spec:
  <...>
{{- end -}}
~
"examplelibrarychart/templates/ui/_deployment.yaml" [noeol] 10L, 250B
```

b. Service

```
{{- define "examplelibrarychart.ui.service" -}}
apiVersion: v1
kind: Service
metadata:
  name: {{ include "examplelibrarychart.fullname" . }}
  labels:
    {{- include "examplelibrarychart.labels" . | nindent 4 }}
spec:
  <...>
{{- end -}}
"examplelibrarychart/templates/ui/_service.yaml" 10L, 240B
```

Example: Templating whole Chart

Use the named template in the dependent chart

In our example the `dependenchart` is for a UI. In the following you see their Deployment and Service

a. deployment.yaml

```
{{- include "examplelibrarychart.ui.deployment" . -}}
```

b. service.yaml

```
{{- include "examplelibrarychart.ui.service" . -}}
```

Examples

Overwriting parts of the library

Example: Overwriting parts of the library

Implement an overwritable configmap on the library side

1. Create a regular template containing your configmap

```
{{- define "examplelibrarychart.overwritable.configmap.tpl" -}}
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Chart.Name }}-cm
data:
  index.html: |-
    <!DOCTYPE html>
    <html>
    <head>
    <title>Welcome to Default ConfigMap nginx!</title>
    <...>
    </html>
{{- end -}}
```

Example: Overwriting parts of the library

Implement an overwritable configmap on the library side

2. Create function to merge the specifications

```
{{- define "examplelibrarychart.util.merge" -}}
{{- $top := first . -}}
{{- $overrides := fromYaml (include (index . 1) $top) | default (dict ) -}}
{{- $tpl := fromYaml (include (index . 2) $top) | default (dict ) -}}
{{- toYaml (merge $overrides $tpl) -}}
{{- end -}}
```

3. Create named template to provide overwritable configmap

```
{{- define "examplelibrarychart.overwritable.configmap" -}}
{{- include "examplelibrarychart.util.merge" (append .
"examplelibrarychart.overwritable.configmap.tpl") -}}
{{- end -}}
```

Example: Overwriting parts of the library

Use the configmap of the library and append or overwrite values

Now on the application chart include the configmap and provide your template with values to overwrite:

```
{{- include "examplelibrarychart.overwritable.configmap" (list .
"overwrite-dependent-chart.configmap.overwrite") -}}
{{- define "overwrite-dependent-chart.configmap.overwrite" -}}
metadata:
  annotations:
    overwritten: "true"
{{- end -}}
```

Example: Overwriting parts of the library

Use the configmap of the library and append or overwrite values

Then those values will be merged an in case overwritten:

```
{{- include "examplelibrarychart.overwritable.configmap" (list
. "overwrite-dependent-chart.configmap.overwrite") -}}
{{- define "overwrite-dependent-chart.configmap.overwrite" -}}
metadata:
  annotations:
    overwritten: "true"
{{- end -}}
```

```
{{- define "examplelibrarychart.overwritable.configmap.tpl" -}}
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Chart.Name }}-cm
data:
  index.html: |-
    <!DOCTYPE html>
    <html>
    <head>
    <title>Welcome to Default ConfigMap nginx!</title>
    <...>
    </html>
{{- end -}}
```

```
apiVersion: v1
data:
  index.html: |-
    <!DOCTYPE html>
    <html>
    <head>
    <title><...>
kind: ConfigMap
metadata:
  annotations:
    overwritten: "true"
  name: overwrite-dependent-chart-cm
```

Q&A

Tanzu®
by Broadcom

Thank You