ytt – YAML Templating Tool

an alternative to text templating
of YAML configuration in Helm

https://get-ytt.io

Nima Kaviani
nkavian@us.ibm.com
@nimak

Dmitriy Kalinin
dkalinin@pivotal.io
@dmitriykalinin

configuration ecosystem

- text templating tools (e.g. *go's text/template, erb, jinja*)
- overlay tools (e.g. *kustomize*)
- specialized configuration languages (*e.g. dhall, jsonnet*)
- SDKs (*e.g. pulumi*)
- native language (*e.g. python, ruby, go*)

comparison of configuration tools is subjective...

text templating tools

- e.g. go's text/template, jinja
- works with text instead of structure, hence challenges with escaping and indentation
- hard to build reusable libraries
- templating language diverges syntactically from the templated content

```
metadata:
{{- with .Values.Labels  }}
  labels:
{{ toYaml . | indent 4  }}
{{- end  }}
```

overlay tools

- e.g. kustomize, bosh ops files
- good for updating configuration values
- lacks language constructs (e.g. conditionals) necessary to describe more complex configuration scenarios

```
namePrefix: demo-

bases:
- wordpress
- mysql

patchesStrategicMerge:
- patch.yaml

vars:
- name: WORDPRESS_SERVICE
  objref:
    kind: Service
    name: wordpress
    apiVersion: v1
- name: MYSQL_SERVICE
  objref:
    kind: Service
    name: mysql
    apiVersion: v1
```

specialized configuration languages

- e.g. dhall, jsonnet
- typically an entirely new language
- frequently lacks tooling
  (e.g. editors, syntax highlighting)

```
local Mojito(virgin=false, large=false) = {
  local factor = if large then 2 else 1,
  ingredients: [
    {
      kind: 'Mint',
      action: 'muddle',
      qty: 6 * factor,
      unit: 'leaves',
    },
  ] + (
    if virgin then [] else [
      { kind: 'Banks', qty: 1.5 * factor },
    ]
  ),
  garnish: if large then 'Lime wedge',
  served: 'Over crushed ice',
};

{
  Mojito: Mojito(),
  'Virgin Mojito': Mojito(virgin=true),
  'Large Mojito': Mojito(large=true),
}
```

SDKs

- e.g. pulumi
- a lot more than just configuration management
- steep learning curve

```
let redisMasterLabels = {
  app: "redis-master"
};

let redisMasterDeployment = \
  new k8s.apps.v1.Deployment("redis-master", {
    spec: {
      selector: {
        matchLabels: redisMasterLabels
      },
      template: {
        metadata: {
          labels: redisMasterLabels
        },
        spec: {
          containers: [{
            name: "master",
            image: "k8s.gcr.io/redis:e2e",
            resources: {
              requests: {
                cpu: "100m",
                memory: "100Mi"
              }
            },
            ports: [{
              containerPort: 6379
            }]
          }]
        }
      }
    }
  });
```

programming languages

- e.g. python, ruby, go
- no sandboxing
- not designed to conveniently describe deeply nested data structures

```
require 'yaml'

puts YAML.dump({
  "apiVersion" => "v1",
  "kind" => "Deployment",
  # ...
})
```

what is ytt?

- Structure-aware templating (not text templating)
- Uses a Pythonic language for templating
- Supports both declarative and imperative constructs (conditionals, loops)
- Allows modularization of configuration structures (via functions)
- Supports data injection
- Supports structure merging (via overlays)
- Allows data validation (via assertions)

ytt basics

YAML annotations

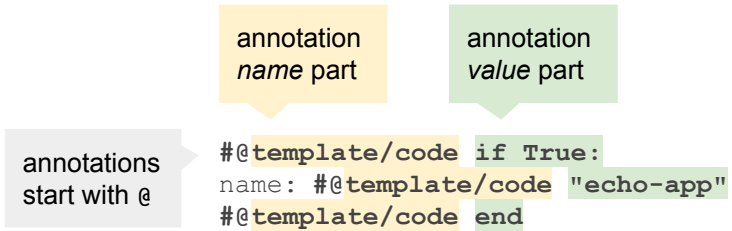
textual tpl. markers

```
{{ before }}
name: {{ inline }}
{{ after }}
```

plain YAML comments

```
# before
name: # inline
# after
```

YAML annotations (cont.)



YAML annotations (cont.)

`template/code` is default

```
#@ if True:
name: #@ "echo-app"
#@ end
```

YAML annotations (cont.)

various other annotations

```
#@ if True:
name: #@ "echo-app"
#@ end

#@yaml/map-key-override
name: echo-app2

ownerReferences:
  #@overlay/match by="uid",missing_ok=True
  - uid: ...
```

annotation arguments

templating

templating - setting value

```
#@ load("@ytt:data", "data")

#@ def labels():
app: echo
org: test
#@ end

kind: Pod
apiVersion: v1
metadata:
  name: #@ "echo-app"
  labels: #@ labels()
spec:
  containers:
  #@ for/end svc in data.values.services:
  - name: #@ name(svc)
    image: hashicorp/http-echo
    #@ if/end not svc.restart:
    restartPolicy: Never
    args:
    - #@ "-listen=:" + str(svc.port)
    - #@ "-text=" + svc.text
```

set value of map item

set value of array item

```
kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels:
    app: echo
    org: test
spec:
  containers:
  - name: echo-first
    image: hashicorp/http-echo
    args:
    - -listen=:8080
    - '-text=Hello #ytt World on 8080!'
  - name: echo-second
    image: hashicorp/http-echo
    restartPolicy: Never
    args:
    - -listen=:8081
    - -text=non-restarting
```

automatic quoting

templating - conditional

```
#@ load("@ytt:data", "data")

#@ def labels():
app: echo
org: test
#@ end

kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels: #@ labels()
spec:
  containers:
  #@ for/end svc in data.values.services:
  - name: #@ name(svc)
    image: hashicorp/http-echo
    #@ if/end not svc.restart:
    restartPolicy: Never
    args:
    - #@ "-listen=:" + str(svc.port)
    - #@ "-text=" + svc.text
```

single node conditional

```
kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels:
    app: echo
    org: test
spec:
  containers:
  - name: echo-first
    image: hashicorp/http-echo
    args:
    - -listen=:8080
    - '-text=Hello #ytt World on 8080!'
  - name: echo-second
    image: hashicorp/http-echo
    restartPolicy: Never
    args:
    - -listen=:8081
    - -text=non-restarting
```

templating - iteration

```
#@ load("@ytt:data", "data")

#@ def labels():
app: echo
org: test
#@ end

kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels: #@ labels()
spec:
  containers:
  #@ for/end svc in data.values.services:
  - name: #@ name(svc)
    image: hashicorp/http-echo
    #@ if/end not svc.restart:
    restartPolicy: Never
    args:
    - #@ "-listen=:" + str(svc.port)
    - #@ "-text=" + svc.text
```

loop over content

```
kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels:
    app: echo
    org: test
spec:
  containers:
  - name: echo-first
    image: hashicorp/http-echo
    args:
    - -listen=:8080
    - '-text=Hello #ytt World on 8080!'
  - name: echo-second
    image: hashicorp/http-echo
    restartPolicy: Never
    args:
    - -listen=:8081
    - -text=non-restarting
```

templating - functions

```
#@ load("@ytt:data", "data")

#@ def labels():
app: echo
org: test
#@ end

kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels: #@ labels()
spec:
  containers:
  #@ for/end svc in data.values.services:
  - name: #@ name(svc)
    image: hashicorp/http-echo
    #@ if/end not svc.restart:
    restartPolicy: Never
    args:
    - #@ "-listen=:" + str(svc.port)
    - #@ "-text=" + svc.text
```

modularize structures

```
kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels:
    app: echo
    org: test
spec:
  containers:
  - name: echo-first
    image: hashicorp/http-echo
    args:
    - -listen=:8080
    - '-text=Hello #ytt World on 8080!'
  - name: echo-second
    image: hashicorp/http-echo
    restartPolicy: Never
    args:
    - -listen=:8081
    - -text=non-restarting
```

templating - modules

```
#@ def labels():
app: echo
org: test
#@ end
```

separate module:
misc.lib.yml

```
#@ load("@ytt:data", "data")
#@ load("misc.lib.yml", "labels")

kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels: #@ labels()
spec:
  containers:
  #@ for/end svc in data.values.services:
  - name: #@ name(svc)
    image: hashicorp/http-echo
    #@ if/end not svc.restart:
    restartPolicy: Never
    args: #! ...
```

```
kind: Pod
apiVersion: v1
metadata:
  name: echo-app
  labels:
    app: echo
    org: test
spec:
  containers:
  - name: echo-first
    image: hashicorp/http-echo
    args:
    - -listen=:8080
    - '-text=Hello #ytt World on 8080!'
  - name: echo-second
    image: hashicorp/http-echo
    restartPolicy: Never
    args:
    - -listen=:8081
    - -text=non-restarting
```

overlays

overlays - updating

```
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/limit-rps: 1000
```

```
#@ load("@ytt:overlay", "overlay")

#@overlay/match by=overlay.subset({"kind": "Ingress"})
---
metadata:
  annotations:
    nginx.ingress.kubernetes.io/limit-rps: 2000
    #@overlay/match missing_ok=True
    nginx.ingress.kubernetes.io/enable-access-log: "true"
    #@overlay/remove
    ingress.kubernetes.io/rewrite-target:
```

find document with content

update node

```
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/limit-rps: 2000
    nginx.ingress.kubernetes.io/enable-access-log: "true"
```

overlays - inserting

```
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/limit-rps: 1000
```

insert node

find document with content

```
#@ load("@ytt:overlay", "overlay")

#@overlay/match by=overlay.subset({"kind": "Ingress"})
---
metadata:
  annotations:
    nginx.ingress.kubernetes.io/limit-rps: 2000
    #@overlay/match missing_ok=True
    nginx.ingress.kubernetes.io/enable-access-log: "true"
    #@overlay/remove
    ingress.kubernetes.io/rewrite-target:
```

```
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/limit-rps: 2000
    nginx.ingress.kubernetes.io/enable-access-log: "true"
```

overlays - deleting

```
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/limit-rps: 1000
```

```
#@ load("@ytt:overlay", "overlay")

#@overlay/match by=overlay.subset({"kind": "Ingress"})
---
metadata:
  annotations:
    nginx.ingress.kubernetes.io/limit-rps: 2000
    #@overlay/match missing_ok=True
    nginx.ingress.kubernetes.io/enable-access-log: "true"
    #@overlay/remove
    ingress.kubernetes.io/rewrite-target:
```

find document with content

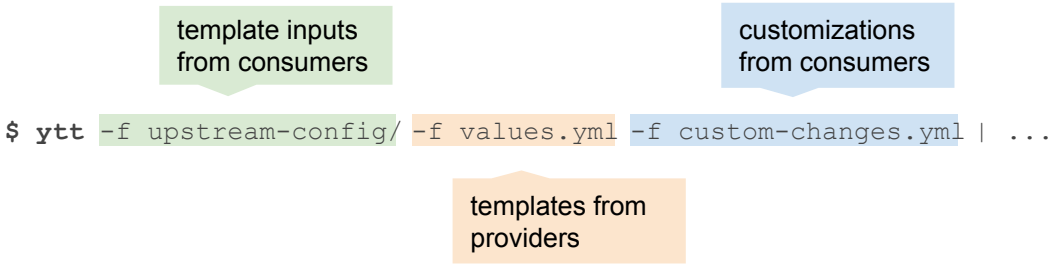
remove node by key

```
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/limit-rps: 2000
    nginx.ingress.kubernetes.io/enable-access-log: "true"
```

templating vs overlays

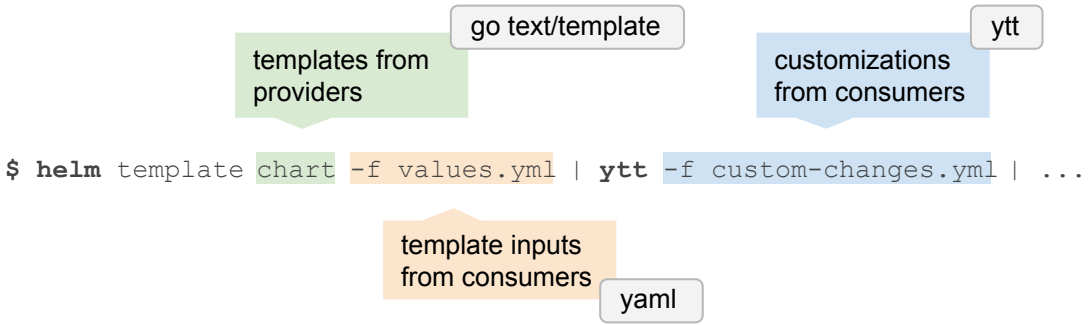
- personas: configuration providers vs configuration consumers
- providers
  - typically need language constructs when describing a set of configuration constraints, and **templating** provides enough power to do so
- consumers
  - need to customize result of templates in ways that providers did not foresee, via **overlays**

templating and overlays



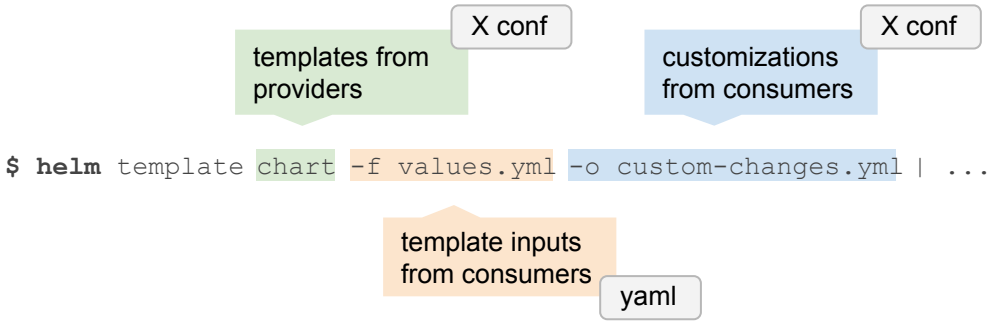
templating and overlays

- useful today to augment upstream charts



pluggable helm templating engines

- https://github.com/helm/helm/issues/6184



... where X conf could be ytt, jsonnet, etc.

demos

demos

- YAML data in ConfigMaps
- Relative rolling update configuration
- Docker image pull secret
- vault-helm chart configuration
  - https://gist.github.com/cppforlife/90fd3ae8817e715d18483f4d8e746cb3
- k8s-lib app

Questions & Thank You!

https://get-ytt.io	interactive playground
#k14s	k8s slack channel
@k14s_io	twitter