Writing Less YAML ...

Using jsonnet and kubecfg to manage Kubernetes resources

https://bit.do/jjo-cncf-kubecfg-18

CNCF webinar - 2018-03-20

JuanJo Ciarlante
Sr SRE at Bitnami
@xjjo

Bitnami

Packaged Applications for Any platform

DEVELOPER TOOLS

APPLICATIONS

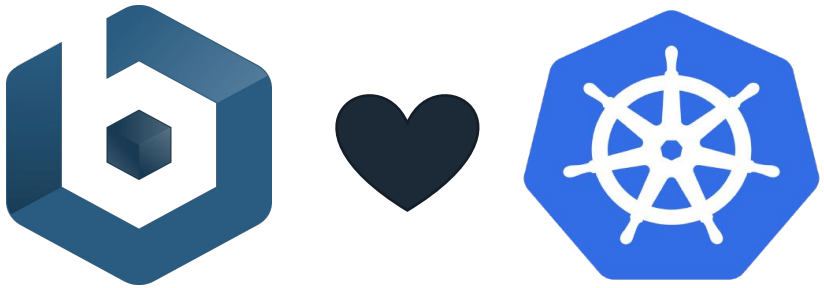
INFRASTRUCTURE

bitnami

Bitnami's Kubernetes Projects

Leverage Application delivery on Next Generation Platforms

- **Kubeapps**
  Application Delivery Environment
- **Kubeless**
  Leading Kubernetes Native Serverless Platform
- **Helm**
  The package manager for Kubernetes
- **Kubecfg**
  Manage Kubernetes resources as code
- **SealedSecrets**
  Securely manage kubernetes secrets in the clear

Charts 

Agenda

- scope of this talk
- from YAML to jsonnet
- and more jsonnet digging ...
- that small staging vs prod divergence
- kubecfg PoWeR
- demo

Scope of this talk

where humans and CIs live together ...

- for Kubernetes
  - manifests
    - naturally *declarative* by design
    - typically use YAML manifests to describe our workloads
  - re: *IaC* Infrastructure as Code
    - for Kubernetes, it'd be *Infrastructure as Software* continuously reconciling manifests with actual resources
    - *what-if* IaC on the source manifests *themselves* ?
      - code re-use - think *staging* vs *prod*
      - testing

Some k8s application deployment projects

The nice thing about standards is that you have so many to choose from. -- A. Tanenbaum

- DIY templating `sed` and friends ;)
- CNCF HELM (w/Microsoft, Google, Bitnami)
  - a note about Helm incluster and multi-tenant [in]security
- Kubernetes' incubator Kcompose
- Heptio's ksonnet.io
- Bitnami'i kubecfg using jsonnet
- and **so many** others ...
  - see @garethr Kubernetes app mgmt tools spreadsheet (60+ entries)

Deployment example: plain YAML

my-nginx.yaml

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: my-nginx
  labels:
     app: my-nginx
spec:
                          replicas:                2
  selector:
    matchLabels:
      app: my-nginx
  template:
    metadata:
      labels:
        app: my-nginx
    spec:
      containers:
        - name: my-nginx
          image: 'nginx:1.12'
          ports:
            - containerPort: 80
```

Deployment example: helm templated YAML

nginx/templates/deployment.yaml

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: {{ template "nginx.fullname" . }}
  labels:
     app: {{ template "nginx.name" . }}
spec:
       replicas:    {{    .Values.replicaCount    }}
  selector:
    matchLabels:
      app: {{ template "nginx.name" . }}
  template:
    metadata:
      labels:
        app: {{ template "nginx.name" . }}
    spec:
      containers:
        - name: {{ template "nginx.name" . }}
          image: {{ Values.Image }}
          ports:
            - containerPort: 80
```

bitnami

Deployment example: jsonnet

nginx-base.jsonnet

```
local deploy = import "kube-deployment.libsonnet";

deploy {
  name: "my-nginx",
  container: {
    image: "nginx:1.12",
  },
}
```

kube-deployment.libsonnet

```
{
  name:: error "name is required",
  container:: error "container is required",

  apiVersion: "apps/v1beta1",
  kind: "Deployment",
  metadata: {
    name: $.name,
    labels: { app: $.name },
  },
  spec: {
    selector: { matchLabels: $.metadata.labels },
    template: {
      metadata: { labels: $.metadata.labels },
      spec: {
        containers: [
          $.container { name: $.name },
        ],
      },
    },
  },
}
```

bitnami

Deployment example: jsonnet

nginx-base.jsonnet

```
local deploy = import "kube-deployment.libsonnet";

deploy {
  name: "my-nginx",
  container: {
    image: "nginx:1.12",
  },
}
```

kube-deployment.libsonnet

```
{
  name:: error "name is required",
  container:: error "container is required",

  apiVersion: "apps/v1beta1",
  kind: "Deployment",
  metadata: {
    name: $.name,
    labels: { app: $.name },
  },
  spec: {
    selector: { matchLabels: $.metadata.labels },
    template: {
      metadata: { labels: $.metadata.labels },
      spec: {
        containers: [
          $.container { name: $.name },
        ],
      },
    },
  },
}
```

bitnami

Deployment example: jsonnet rendering

```
$ jsonnet nginx-base.jsonnet
```

```
{
   "apiVersion": "apps/v1beta1",
   "kind": "Deployment",
   "metadata": {
      "labels": {
         "app": "nginx-base"
      },
      "name": "nginx-base"
   },
   "spec": {
      "selector": {
         "matchLabels": {
            "app": "nginx-base"
         }
      },
      "template": {
         "metadata": {
            "labels": {
               "app": "nginx-base"
            }
         },
         "spec": {
            "containers": [
               {
                  "image": "nginx:1.12",
                  "name": "nginx-base"
               }
            ]
         }
      }
}
```

```
$ kubecfg show nginx-base.jsonnet
```

```
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  labels:
    app: nginx-base
  name: nginx-base
spec:
  selector:
    matchLabels:
      app: nginx-base
  template:
    metadata:
      labels:
        app: nginx-base
    spec:
      containers:
      - image: nginx:1.12
        name: nginx-base
```

bitnami

Deployment example: jsonnet w/ replicas: 2

nginx-base.jsonnet

```
local deploy = import "kube-deployment.libsonnet";

deploy {
  name: "nginx-base",
  container: {
    image: "nginx:1.13",
  },
}
```

my-nginx.jsonnet

```
local nginx_deploy = import "nginx-base.jsonnet";

nginx_deploy {
  name: "my-nginx",
  spec+: {
    replicas: 2
  },
}
```

kube-deployment.libsonnet

```
{
  name:: error "name is required",
  container:: error "container is required",

  apiVersion: "apps/v1beta1",
  kind: "Deployment",
  metadata: {
    name: $.name,
    labels: { app: $.name },
  },
  spec: {
    selector: { matchLabels: $.metadata.labels },
    template: {
      metadata: { labels: $.metadata.labels },
      spec: {
        containers: [
          $.container { name: $.name },
        ],
      },
    },
  },
}
```

bitnami

Deployment example: jsonnet rendering

```
$ jsonnet my-nginx.jsonnet
```

```
{
   "apiVersion": "apps/v1beta1",
   "kind": "Deployment",
   "metadata": {
      "labels": {
         "app": "my-nginx"
      },
      "name": "my-nginx"
   },
   "spec": {
      "replicas": 2,
      "selector": {
         "matchLabels": {
            "app": "my-nginx"
         }
      },
      "template": {
         "metadata": {
            "labels": {
               "app": "my-nginx"
            }
         },
         "spec": {
            "containers": [
               {
                  "image": "nginx:1.12",
                  "name": "my-nginx"
               }
            ]
         }
      }
   }
}
```

```
$ kubecfg show my-nginx.jsonnet
```

```
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  labels:
    app: my-nginx
  name: my-nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: my-nginx
  template:
    metadata:
      labels:
        app: my-nginx
    spec:
      containers:
      - image: nginx:1.12
        name: my-nginx
```

bitnami

wrapup: jsonnet the language, and the tool

there's so much JSON in the world … to let it be written by hand

- jsonnet the language:
  - created by Google https://jsonnet.org
  - JSON-like programming "language", modular, encapsulated, deterministic, **declarative**
  - has its own `stdlib`
  - not restricted to any particular use - anything JSON'able can be templated
- jsonnet CLI:
  - parses jsonnet files to json output, no schema validation
  - C++ implementation: https://github.com/google/jsonnet :
  - Go implementation (beta): https://github.com/google/go-jsonnet:
- kubecfg CLI:
  - parses jsonnet files to yaml (or json) **kubernetes manifests**
  - written in Go: https://github.com/ksonnet/kubecfg:

```
go get github.com/ksonnet/kubecfg
```

Deployment example: staging vs prod on diff. namespaces

nginx-staging.jsonnet

```
local nginx_deploy = import "nginx-base.jsonnet";

deploy {
  name: "my-nginx",
  metadata+: {
    namespace: "app-staging",
  },
  spec+: {
    replicas: 2,
  },
}
```

nginx-production.jsonnet

```
local nginx_deploy = import
"nginx-base.jsonnet";

nginx_deploy {
  name: "my-nginx",
  metadata+: {
    namespace: "app-production",
  },
  spec+: {
    replicas: 3,
    strategy: {
      rollingUpdate: {
        maxSurge: "50%",
        maxUnavailable: "10%",
      },
    },
  },
}
```

kubecfg PoWeR

let's sing some (j)sonnets ♫♫ … directly to that API

- kubecfg doesn't replace kubectl
- rather provides some really cool features re: handling manifests written in jsonnet, yaml or json

update / create resources	`kubecfg update <file>`	`kubectl apply -f <file>` (~)
update with garbage collection	`kubecfg update --gc-tag=<app-id> <file>`	-
diff against existing resources	`kubecfg diff <file>` `kubecfg diff --diff-strategy subset <file>`	-
delete resources	`kubecfg delete <file>`	`kubectl delete -f <file>`
validate manifests	`kubecfg validate <file>`	`kubectl convert --validate -f <file>` (~)

DEMO time ...

References

- These slides:
  - https://bit.do/jjo-cncf-kubecfg-18
- Jsonnet the language
  - https://jsonnet.org/
- Jsonnet the CLI
  - https://github.com/google/jsonnet/
  - https://github.com/google/go-jsonnet/
- Kubecfg CLI
  - https://github.com/ksonnet/kubecfg/
- Bitnami jsonnet manifest libraries
  - https://github.com/bitnami-labs/kube-manifests/
- Ksonnet CLI and libraries
  - https://github.com/ksonnet/ksonnet/

&& thanks :)

JuanJo Ciarlante
@xjjo - https://github.com/jjo

REF -- Terminal Example, 32 pt

Subtitle, Helvetica, 24 pt

```
apiVersion: v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: prydonius/seanmeme:v1.0.0
        ports:
        - containerPort: 80
        livenessProbe:
          httpGet:
            path: /
            port: http
          initialDelaySeconds: 120
          timeoutSeconds: 5
```

bitnami