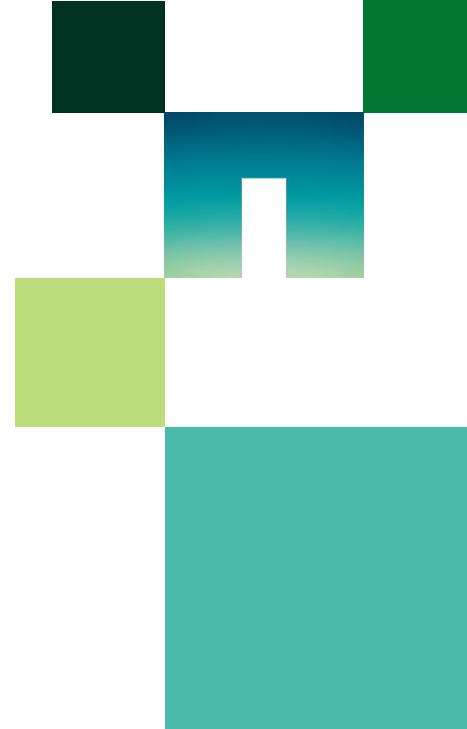




YAML is Optional

Exploring an App Developer's
Kubernetes Options

A link to this deck is bit.ly/2NPZTWE



Outline

1. Why containers? +/-
2. The tragedy of YAML
3. Summarizing dev issues
4. Exploring solutions
5. Summary



V12.13.0

Part 1

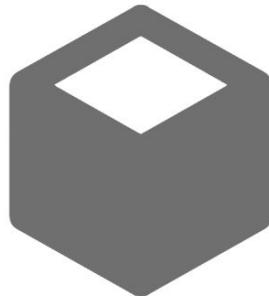
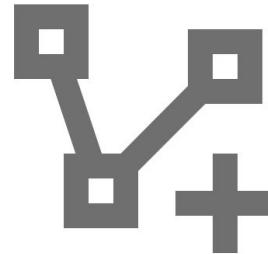
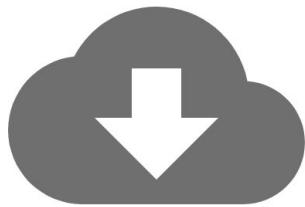


It works on
my local

Hell is other people

Hell is other people's

Dev Environment



So... no more `nodemon` ?

```
FROM node:6.11.5
```

```
WORKDIR /usr/src/app
```

```
COPY package.json .
```

```
RUN npm install
```

```
COPY . .
```

```
CMD [ "npm", "start" ]
```



Part 2

A close-up photograph of a weathered stone or concrete sculpture of a horse's head and neck. The horse has a decorative bridle with a small red jewel in the center. A speech bubble originates from the horse's mouth, containing the text "Oh, hey there Kubernetes...".

Oh, hey there
Kubernetes...



Learning
curve

YAML



Jessie Frazelle ✅
@jessfraz



Replying to [@beajammingh](#) and [@benjammingh](#)

SPACES MATTER M'KAY BEN (I literally hate yaml)



Joe Beda ✅
@jbeda



I want to go on record: the amount of yaml required to do anything in k8s is a tragedy. Something we need to solve. (Subtweeting HN comment)



Bryan Liles
@bryanl



Kelsey is right. The YAML in Kubernetes doesn't need to go anywhere. It's low level and we should be thinking in higher constructs. IMHO those higher constructs should not be templating.



Kelsey Hightower @kelseyhightower · Sep 12

The way to think about this is: **YAML is Kubernetes' assembly code.** How you generate it is up to you. That's where tools like Helm and Pulumi come in.

If you treat the generated YAML like data you can interchange the tooling; the serialized files can also be version controlled. [twitter.com/yassimn_tw/sta...](https://twitter.com/yassimn_tw/status/13034444444444444)

[Show this thread](#)

KUBERNETES / SECURITY

Kubernetes ‘Billion Laughs’ Vulnerability Is No Laughing Matter

9 Oct 2019 8:11am, by [Jack Wallen](#)

A new vulnerability has been discovered within the Kubernetes API. This flaw is centered around the parsing of YAML manifests by the Kubernetes API server. During this process, the API server is open to potential Denial of Service (DoS) attacks. The issue ([CVE-2019-11253](#) — which has yet to have any details

 Closed

CVE-2019-11253: Kubernetes API Server JSON/YAML parsing vulnerable to resource exhaustion

raesene opened this issue on Sep 27 · 16 comments · Fixed by #83261

Applying this manifest to a cluster causes the client to hang for some time with considerable CPU usage.

```
apiVersion: v1
data:
  a: &a ["web", "web", "web", "web", "web", "web", "web", "web", "web"]
  b: &b [*a, *a, *a, *a, *a, *a, *a, *a, *a]
  c: &c [*b, *b, *b, *b, *b, *b, *b, *b, *b]
  d: &d [*c, *c, *c, *c, *c, *c, *c, *c, *c]
  e: &e [*d, *d, *d, *d, *d, *d, *d, *d, *d]
  f: &f [*e, *e, *e, *e, *e, *e, *e, *e, *e]
  g: &g [*f, *f, *f, *f, *f, *f, *f, *f, *f]
  h: &h [*g, *g, *g, *g, *g, *g, *g, *g, *g]
  i: &i [*h, *h, *h, *h, *h, *h, *h, *h, *h]
kind: ConfigMap
metadata:
  name: yaml-bomb
  namespace: default
```



jbeda commented on Sep 27

Member



...

Just saw this -- we should stop accepting yaml server side. Or have a "simple yaml" variant that gets rid of references.

Any real world usages of users sending yaml to the api server? Can we go JSON/proto only?



9



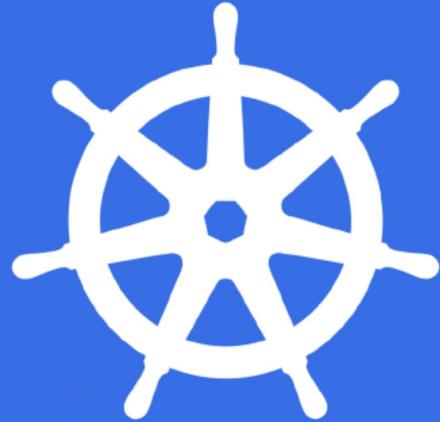
1



I say we take off
and nuke the entire
site from orbit.

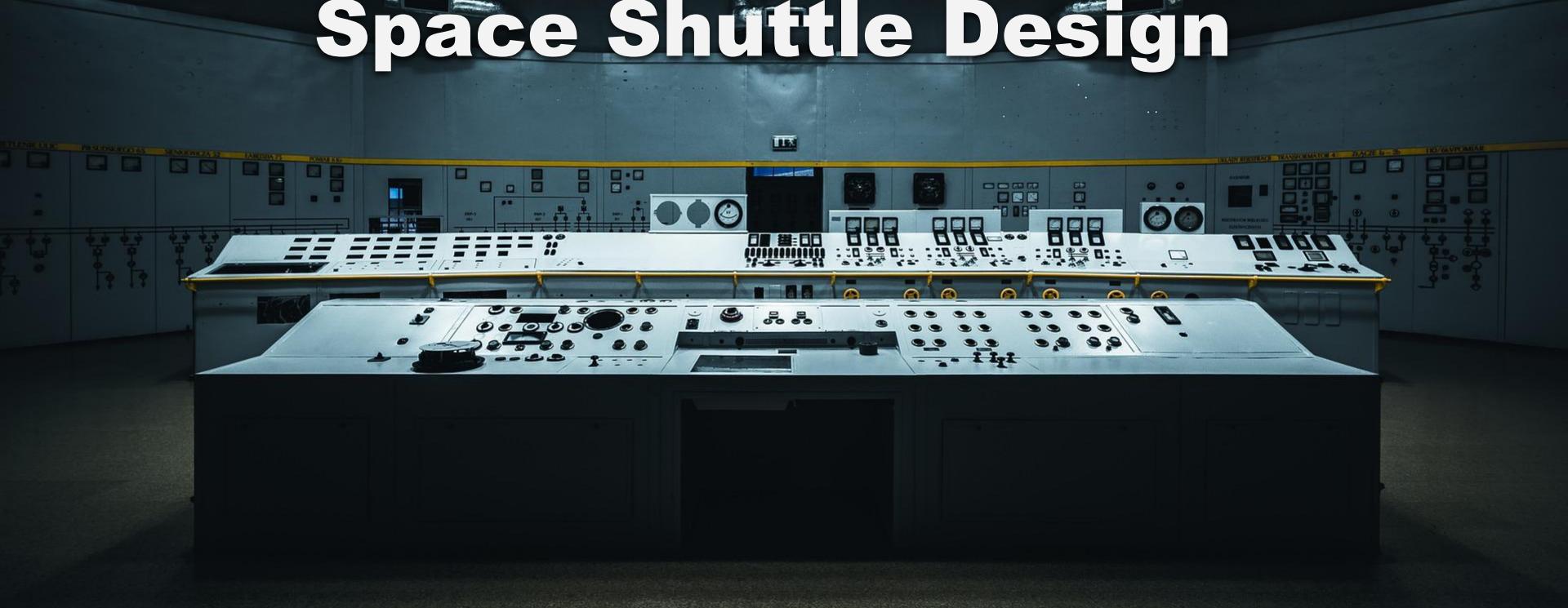


It's the only way to be sure.



kubelet	Job	StatefulSet
kube-proxy	Service	PersistentVolume
containerD	Ingress	PersistentVolumeClaim
kubectl	ConfigMap	NetworkPolicy
CoreDNS	Namespace	AdmissionController
metrics-server	Secret	CustomResourceDefinition
Pod	ServiceAccount	Taints
Deployment	Label	Tolerations
Replica Set	Annotation	RuntimeClass

Space Shuttle Design



Human Centered Design



Part 3



Developers,
Developers,
Developers

Dockerfiles

Dev env setup

Iterative dev loop

CI workflow

Debugging tools

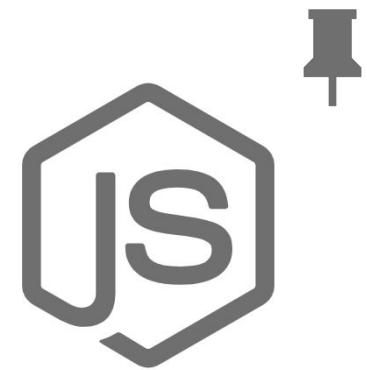
Container patterns

K8s learning curve

YAML avoidance

Dockerfiles
Dev env setup
Iterative dev loop
CI workflow
Debugging tools
Container patterns
K8s learning curve
YAML avoidance

vs



V12.13.0

YAML sucks less



YAML is optional

Part 4



Ksonnet

Kustomize

Whatever,
just use
sed

Brigade

Simple, powerful pipes

Each project gets a `brigade.js` config file, which is where you can write dynamic, interwoven pipelines and tasks for your Kubernetes cluster. Leave your YAML at home!

```
// Run unit tests for a Github push
const { events, Job , Group} = require("brigadier");
const dest = "$GOPATH/src/github.com/technosophos/ulid";

events.on("push", (e, p) => {
  console.log(e.payload)
  var gh = JSON.parse(e.payload)
  var test = new Job("test", "golang:1.9")
  test.tasks = [
    "mkdir -p " + dest,
    "cp -a /src/* " + dest,
    "cd " + dest,
    "go get -u github.com/golang/dep/cmd/dep",
    "dep ensure",
    "make test"
  ];
  test.run()
});

// Updating a cosmosDB database
```

```
const { events, Job } = require("brigadier")

events.on("exec", (e, p) => {
  var mongo = new Job("update-db", "mongo:3.2")
```

Example use cases:

 Unit tests for a Github push

 Updating a cosmosDB database

 Sending a Slack message

 Sending a Twitter DM

Check out [our docs](#) for more. →

Brigade

Solves: integrating CI deeper with
Kubernetes, and opens the door to `git push`
workflows

Metaparticle

```
...
@containerize(
    'docker.io/your-docker-user-goes-here',
    options={
        'replicas': 4,
        'executor': 'metaparticle',
        'ports': [8080],
        'name': 'my-image',
        'publish': True
    })
...
```


Metaparticle

Solves: the need to learn Dockerfile and k8s YAML formats, lowering the learning curve.

Isopod

```
CLUSTERS = [
    onprem(env="dev", cluster="minikube"),
    gke(
        env="prod",
        cluster="paas-prod",
        location="us-west1",
        project="cruise-paas-prod",
    ),
]

def clusters(ctx):
    if ctx.cluster != None:
        return [c for c in CLUSTERS if c.cluster == ctx.cluster]
    elif ctx.env != None:
        return [c for c in CLUSTERS if c.env == ctx.env]
    return CLUSTERS

def addons(ctx)
    return [
        addon("ingress", "configs/ingress.ipd", ctx),
    ]
```

Isopod

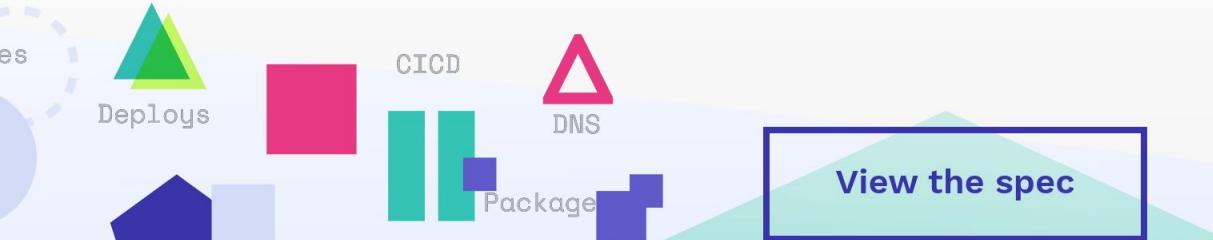
Solves: Configs are an important part of code, and need testing. A single language used for Dockerfiles, k8s resources, and pushing code.

CNAB



A spec for packaging distributed apps.

CNABs facilitate the bundling, installing and managing of container-native apps — and their coupled services.

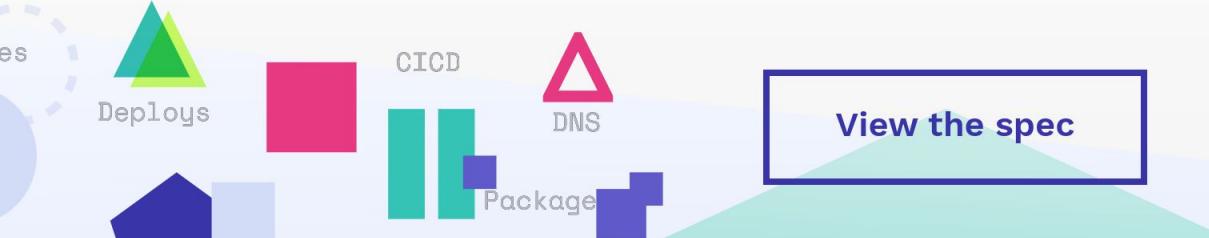




A spec for packaging distributed apps.

It's `docker compose` ... but, like more neutral.

[View the spec](#)



OCI Artifacts

Push it all to the registry!



Jimmy Zelinskie [Follow](#)

Aug 22 · 3 min read



This kind of open container won't get you arrested.

In 2016, CoreOS hired Antoine Legrand ([@ant31](#)).

Antoine is an incredible engineer and a huge asset to the Kubernetes community; you may recognize him from many contributions, but most people probably would be familiar with a little project he started called [Kubespray](#).

The first time I met Antoine, he had hacked together a demo of

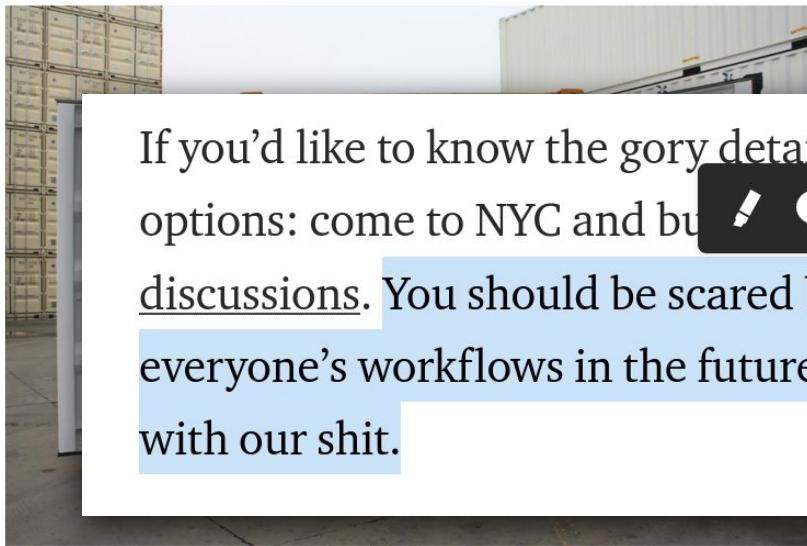
OCI Artifacts

Push it all to the registry!



Jimmy Zelinskie [Follow](#)

Aug 22 · 3 min read



In 2016, CoreOS hired
Antoine Legrand ([@ant31](#)).

If you'd like to know the gory details about what's going on you have two options: come to NYC and bu  or join the upstream OCI discussions. You should be scared because this is going to affect absolutely everyone's workflows in the future and if we get no feedback, you'll be stuck with our shit.

This kind of open container won't get you arrested.

The first time I met Antoine, he had hacked together a demo of

CNAB

Solves: how to organize containers into a logical app in a platform and vendor neutral way.

OAM implemented as Rudr



Lei Zhang (Harry)

@resouer

Replying to @KarlKFI and @jbeda

that's why we came out OAM
openappmodel.io , developers don't have
to consume system calls directly to
write program on Linux

1:41 PM · Nov 5, 2019 · Twitter for iPhone

2 Likes



Karl Isenberg @KarlKFI · 1m

Replying to @resouer and @jbeda

Rudr, huh? Interesting.
Any good intro videos?
What companies back the project?



Concepts

Using Rudr

Learn the basics of using Rudr.

Application Configuration

As an *application operator*, define how your overall application will be instantiated and configured.

Traits

As an *application operator*, attach operational features to component workloads of your application.

Component Schematic

As a *developer*, define the operational characteristics of your component of code.

Workloads

As a *developer*, designate the appropriate workload type to execute your component on the Rudr runtime.

How-To's

Create Component from Scratch

Build a component from source code to use for testing.

Using Helm/Kustomize to manage OAM yaml

```
apiVersion: core.oam.dev/v1alpha1
kind: ComponentSchematic
metadata:
  name: nginx-replicated
spec:
  workloadType: core.oam.dev/v1alpha1.Server
  osType: linux
  arch: amd64
  containers:
    - name: server
      image: nginx:latest
      config:
        - name: "/etc/access/default_user.txt"
        - value: "admin"
      ports:
        - name: http
          containerPort: 80
          protocol: TCP
  parameters:
    - name: poet
      type: string
      default: Yeats
```

OAM implemented as Rudr

Solves: Conway's Law. The communication structure of your org can be reflected in YAML to improve collaboration between dev and ops roles.

Buildpacks

```
{  
  "name": "Start on Heroku: Node.js",  
  "description": "A barebones Node.js app using Express 4",  
  "repository": "https://github.com/heroku/node-js-getting-started",  
  "logo":  
    "https://cdn.rawgit.com/heroku/node-js-getting-started/master/public/node.s  
vg",  
  "keywords": ["node", "express", "heroku"],  
  "image": "heroku/nodejs"  
}
```

Buildpacks

Solves: the need to learn about containers, or kubernetes resources. Git as the source of truth for your platform.

Tilt

```
# tiltdemo1
k8s_yaml('deployments/demoserver1.yaml')
dm1_img_name = 'gcr.io/windmill-test-containers/tiltdemo/demoserver1'
docker_build(dm1_img_name, '.', dockerfile='Dockerfile.server1',
    live_update=[
        sync('cmd/demoserver1',
        '/go/src/github.com/windmilleng/tiltdemo/cmd/demoserver1'),
        run('go install github.com/windmilleng/tiltdemo/cmd/demoserver1'),
        restart_container(),
    ]
)
# tiltdemo2
k8s_yaml('deployments/demoserver2.yaml')
dm1_img_name = 'gcr.io/windmill-test-containers/tiltdemo/demoserver2'
docker_build(dm1_img_name, '.', dockerfile='Dockerfile.server2',
    live_update=[
        sync('cmd/demoserver2',
        '/go/src/github.com/windmilleng/tiltdemo/cmd/demoserver2'),
        run('go install github.com/windmilleng/tiltdemo/cmd/demoserver2'),
        restart_container(),
    ]
)
```

Tilt

Solves: the need for fast feedback loops as devs are writing new code, or debugging existing code.

Admission Controller



LimitRanger

This admission controller will observe the incoming request and ensure that it does not violate any of the constraints enumerated in the `LimitRange` object in a `Namespace`. If you are using `LimitRange` objects in your Kubernetes deployment, you MUST use this admission controller to enforce those constraints. LimitRanger can also be used to apply default resource requests to Pods that don't specify any; currently, the default LimitRanger applies a 0.1 CPU requirement to all Pods in the `default` namespace.

See the [limitRange design doc](#) and the [example of Limit Range](#) for more details.

Admission Controllers

Solves: reduces the number of fields devs have to remember to fill in on their YAML files.

Helm

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: deis-database
  namespace: deis
  labels:
    app.kubernetes.io/managed-by: deis
spec:
  replicas: 1
  selector:
    app.kubernetes.io/name: deis-database
  template:
    metadata:
      labels:
        app.kubernetes.io/name: deis-database
    spec:
      serviceAccount: deis-database
      containers:
        - name: deis-database
          image: {{.Values.imageRegistry}}/postgres:{{.Values.dockerTag}}
          imagePullPolicy: {{.Values.pullPolicy}}
          ports:
            - containerPort: 5432
      env:
```

Helm

Solves: providing a menu of options for devs to choose from, per org. Basic lifecycle.

Ksonnet



We'll need to do a little of extra package management first, the `redis-stateless` prototype is not available by default.

1. Start by seeing what prototypes we have available out of the box:

```
ks prototype list
```

2. See what packages are currently available for us to download:

```
ks pkg list
```

(Where do these packages come from?) [\[+\]](#)

3. Download a specific version of the ksonnet Redis library (which contains definitions for various Redis prototypes):

```
ks pkg install incubator/redis@master
```

4. Check the updated list of packages and prototypes (you should see `redis` and `stateless-redis`):

```
ks pkg list  
ks prototype list
```

5. Figure out the parameters we need for this prototype:

```
ks prototype describe redis-stateless
```

6. At this point, we're ready to generate the manifest for our Redis component:

```
ks generate redis-stateless redis
```

Overview

o. Prerequisites

1. Initialize your app

2. Generate and deploy an app component

3. Understand how prototypes build components

- Define "prototype"
- Commands (Datastore component)
- Takeaways

4. Set up another environment for your app

5. Customize an environment with parameters

6. Tie it together

Or, use functions:

example2.jsonnet

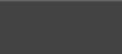
```
1 // A function that returns an object.  
2 local Person(name='Alice') = {  
3     name: name,  
4     welcome: 'Hello ' + name + '!',  
5 };  
6 {  
7     person1: Person(),  
8     person2: Person('Bob'),  
9 }
```



output.json

```
{  
    "person1": {  
        "name": "Alice",  
        "welcome": "Hello Alice!"  
    },  
    "person2": {  
        "name": "Bob",  
        "welcome": "Hello Bob!"  
    }  
}
```

Prior to the acquisition, Heptio had been shifting focus and resources away from ksonnet; with the acquisition, we felt it was the right time to rethink our investment in ksonnet. As a result, work on ksonnet will end and the GitHub repositories will be archived. It's extremely difficult to step back from a project we have worked so hard on, but we're excited about our new ideas and vision for changing how developers experience the Kubernetes and cloud native ecosystems. The problems that ksonnet aimed to solve are still challenges for Kubernetes users and we will be putting our energy into opportunities to contribute to existing or new projects in this space.

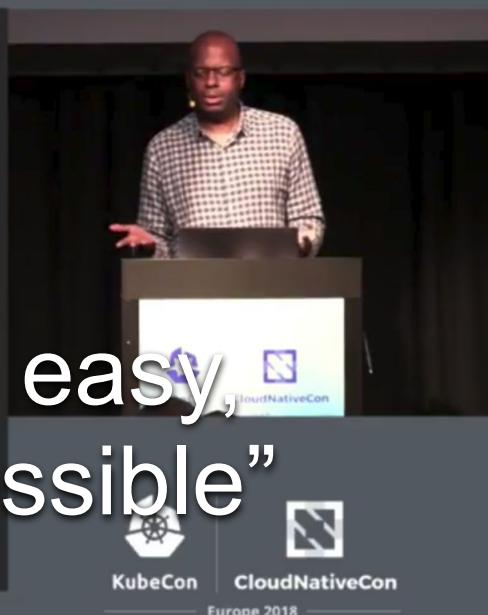


Search



“I want easy things to be easy,
And hard things to be possible”

```
tpbin']"
INFO Writing component at '/Users/bryan/Development/talks/yaml-is-for-computers/app/components/service.jsonnet'
7 app $ # create ingress component
7 app $ echo -e 'local params = std.extVar("__ksonnet/params").components.ingress;
' > components/ingress.jsonnet
7 app $ yaml2json < ../deployment/ingress.yaml | jq -M '.' >> components/ingress.jsonnet
7 app $ ks param set ingress host ks-httpbin.myk8s
7 app $ sed -i '' 's/"httpbin.myk8s"/params.host/' components/ingress.jsonnet
# create deployment cluster
7 app $ k apply -f ./deployment
INFO Updating deployments httpbin
INFO Updating ingresses httpbin
INFO Deployments updated
7 app $ k get pods
yaml-is-for-computers /
```



▶ ▶ 🔍 13:47 / 36:02

Videos Sponsored by Google Cloud



Ksonnet

Solves: how to manage multi-cluster,
multi-env, multiplicatively complex config
scenarios. Keep your configs DRY.

Kustomize



Search



Overlays

```
rcox:overlays$ tree
.
├── base
│   ├── kustomization.yaml
│   └── pod.yaml
└── production
    └── kustomization.yaml
└── staging
    └── kustomization.yaml

3 directories, 4 files
rcox:overlays$ cat base/kustomization.yaml
resources:
- pod.yaml
rcox:overlays$ cat production/kustomization.yaml
bases:
- ./base
namePrefix: prod-
```



KubeCon



CloudNativeCon

North America 2018

▶ ▶ 🔍 7:13 / 35:49

Videos brought to you by: Cockroach DB



Kustomize: Deploy Your App with Template Free YAML - Ryan Cox, Lyft

Up next

AUTOPLAY



```
# Create a directory to hold the base
mkdir base
# Create a base/deployment.yaml
cat <<EOF > base/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
  spec:
    containers:
    - name: my-nginx
      image: nginx
EOF
```

```
# Create a base/service.yaml file
```

```
mkdir dev
cat <<EOF > dev/kustomization.yaml
bases:
- ../base
namePrefix: dev-
EOF
```

```
mkdir prod
cat <<EOF > prod/kustomization.yaml
bases:
- ../base
namePrefix: prod-
EOF
```

Tasks

- ▶ Install Tools
- ▶ Administer a Cluster
- ▶ Configure Pods and Containers
- ▼ Manage Kubernetes Objects

[Declarative Management of Kubernetes Objects Using Configuration Files](#)

[Declarative Management of Kubernetes Objects Using Kustomize](#)

[Managing Kubernetes Objects Using Imperative Commands](#)

[Imperative Management of Kubernetes Objects Using Configuration Files](#)

- ▶ Inject Data Into Applications
- ▶ Run Applications
- ▶ Run Jobs
- ▶ Access Applications in a Cluster
- ▶ Monitoring, Logging, and Debugging
- ▶ Extend Kubernetes
- ▶ TILs

Declarative Management of Kubernetes Objects Using Kustomize



Kustomize is a standalone tool to customize Kubernetes objects through a [kustomization file](#).

Since 1.14, Kubectl also supports the management of Kubernetes objects using a kustomization file. To view Resources found in a directory containing a kustomization file, run the following command:

```
kubectl kustomize <kustomization_directory>
```

To apply those Resources, run `kubectl apply` with `--kustomize` or `-k` flag:

```
kubectl apply -k <kustomization_directory>
```

- [Before you begin](#)
- [Overview of Kustomize](#)
- [Bases and Overlays](#)
- [How to apply/view/delete objects using Kustomize](#)

Customizing Helm Charts, Kube YAML and Knative with Kustomize

Ship exposes the power of Kustomize as an advanced custom configuration management tool for [Helm charts](#), Kubernetes manifests and [Knative](#) applications. The easy-to-use UI of Ship (launched via `ship init`) calculates the minimal patch YAML required to build an overlay and previews the diff that will be the result of applying the drafted overlay.

The screenshot shows the Ship UI interface for managing Kustomize overlays. At the top, a progress bar with six steps is shown, with steps 1 through 4 completed (indicated by checkmarks) and step 5 partially completed (indicated by a blue circle with the number 5). Step 6 is the final step, represented by a grey circle with the number 6.

base section:

- client-clusterrole.yaml
- client-clusterrolebinding.yaml
- client-config-configmap.yaml
- client-daemonset.yaml
- client-serviceaccount.yaml
- connect-inject-clusterrole.yaml
- connect-inject-deployment.yaml
- connect-inject-mutatingwebhook.yaml
- connect-inject-service.yaml
- dns-service.yaml
- server-clusterrole.yaml
- server-clusterrolebinding.yaml

overlays section:

- server-statefulset.yaml

Base YAML section:

Select a file to be used as the base YAML. You can then click the edit icon on the top right to create a patch for that file.

```
4  labels:
5    app: consul
6    release: consul
7    name: consul-server
8    namespace: default
9  spec:
10   podManagementPolicy: Parallel
11   replicas: 3
12   selector:
13     matchLabels:
14       app: consul
15       chart: consul-helm
16       component: server
17       hasDNS: "true"
18       release: consul
19       serviceName: consul-server
20   template:
21     metadata:
```

Patch section:

This file will be applied as a patch to the base manifest. Edit the values that you want patched. The current file you're editing will be automatically saved when you open a new file.

```
1  apiVersion: apps/v1
2  kind: StatefulSet
3  metadata:
4    labels:
5      app: consul
6      release: consul
7      name: consul-server
8      namespace: default
9  spec:
10   template:
11     spec:
12       terminationGracePeriodSeconds: 100
```

Buttons at the bottom:

- Show diff
- Save patch
- Save & continue

Contributed by [Replicated](#)

Additionally, the `unfork` command can [migrate forked manifests](#) and environment versions to Kustomize.

The output of the `init` and `unfork` modes will result in the creation of a directory that includes the finalized overlay YAML.

Kustomize

Solves: how to manage YAML complexity while still remaining Declarative. Keep your configs DRY.

Part 5



Shut up,
Jeff

Why is it all
so
complex?

Deploy and Update Code

Use this command to add a remote to your Git project, and then push to the NKS remote to deploy and update your code with `git push nks master`

```
git remote add nks https://git.admin.netigvzb...
```

Access Your Application

The following ingresses have been configured for this Application.

<https://static-love-htm-master.hello-aws-summit-nyc.admin.netigvzbft...>

**DEPLOY WITH A SIMPLE
git push nks master**

Summary

1. Kubernetes is a “space shuttle” design
2. There are a plethora of dev-focused tools
3. No one tool does it all
4. Some tools reduce the amount of YAML
5. Others obviate YAML altogether
6. To make k8s approachable to devs we need to combine multiple approaches



YAML is Optional

Exploring an App Developer's
Kubernetes Options

A link to this deck is bit.ly/2NPZTWE

