

Providing a Rich Interface to Prometheus






David Flanagan

 rawkode

David Flanagan



- Scottish
- Founded Rawcode Academy
- Previously Pulumi, Equinix Metal, and InfluxDB

-
-  <https://rawcode.live>
 -  <https://rawcode.chat>
 -   

I'm here to remove YAML from
your lives

What's the standard operating
procedure look like?

Deploy Your Application

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: example-app
5  spec:
6    replicas: 3
7    selector:
8      matchLabels:
9        app: example-app
10   template:
11     metadata:
12       labels:
13         app: example-app
14     spec:
15       containers:
16       - name: example-app
17         image: fabxc/instrumented_app
18         ports:
19         - name: web
20           containerPort: 8080
```

Expose it over a Service

```
1  kind: Service
2  apiVersion: v1
3  metadata:
4    name: example-app
5    labels:
6      app: example-app
7  spec:
8    selector:
9      app: example-app
10   ports:
11     - name: web
12       port: 8080
```

Create a Service or Pod Monitor

```
1  apiVersion: monitoring.coreos.com/v1
2  kind: ServiceMonitor
3  metadata:
4    name: example-app
5    labels:
6      team: frontend
7  spec:
8    selector:
9      matchLabels:
10       app: example-app
11    endpoints:
12     - port: web
```

Connect ServiceMonitor to Prometheus

```
1  apiVersion: monitoring.coreos.com/v1
2  kind: Prometheus
3  metadata:
4    name: prometheus
5  spec:
6    serviceAccountName: prometheus
7    serviceMonitorSelector:
8      matchLabels:
9        team: frontend
10   resources:
11     requests:
12       memory: 400Mi
13   enableAdminAPI: false
14   alerting:
15     alertmanagers:
16     - namespace: default
17       name: alertmanager-example
18       port: web
19   ruleSelector:
20     matchLabels:
21       role: alert-rules
22       prometheus: example
23   ruleNamespaceSelector:
```


Deploy AlertManager

```
1  apiVersion: monitoring.coreos.com/v1
2  kind: Alertmanager
3  metadata:
4    name: example
5  spec:
6    replicas: 3
7    alertmanagerConfigSelector:
8      matchLabels:
9        alertmanagerConfig: example
10 ---
11 apiVersion: monitoring.coreos.com/v1alpha1
12 kind: AlertmanagerConfig
13 metadata:
14   name: config-example
15   labels:
16     alertmanagerConfig: example
17 spec:
18   route:
19     groupBy: ['job']
20     groupWait: 30s
21     groupInterval: 5m
22     repeatInterval: 12h
23     receiver: 'webhook'
24   receivers:
```

Add a PrometheusRule

```
1  apiVersion: monitoring.coreos.com/v1
2  kind: PrometheusRule
3  metadata:
4    creationTimestamp: null
5    labels:
6      prometheus: example
7      role: alert-rules
8    name: prometheus-example-rules
9  spec:
10   groups:
11     - name: ./example.rules
12       rules:
13         - alert: ExampleAlert
14           expr: vector(1)
```





I am not fond of this.





How do we make this better?

The um...

Spoiler Alert

The answer isn't YAML or templating

- Helm
 - Kustomize
 - ytt
-

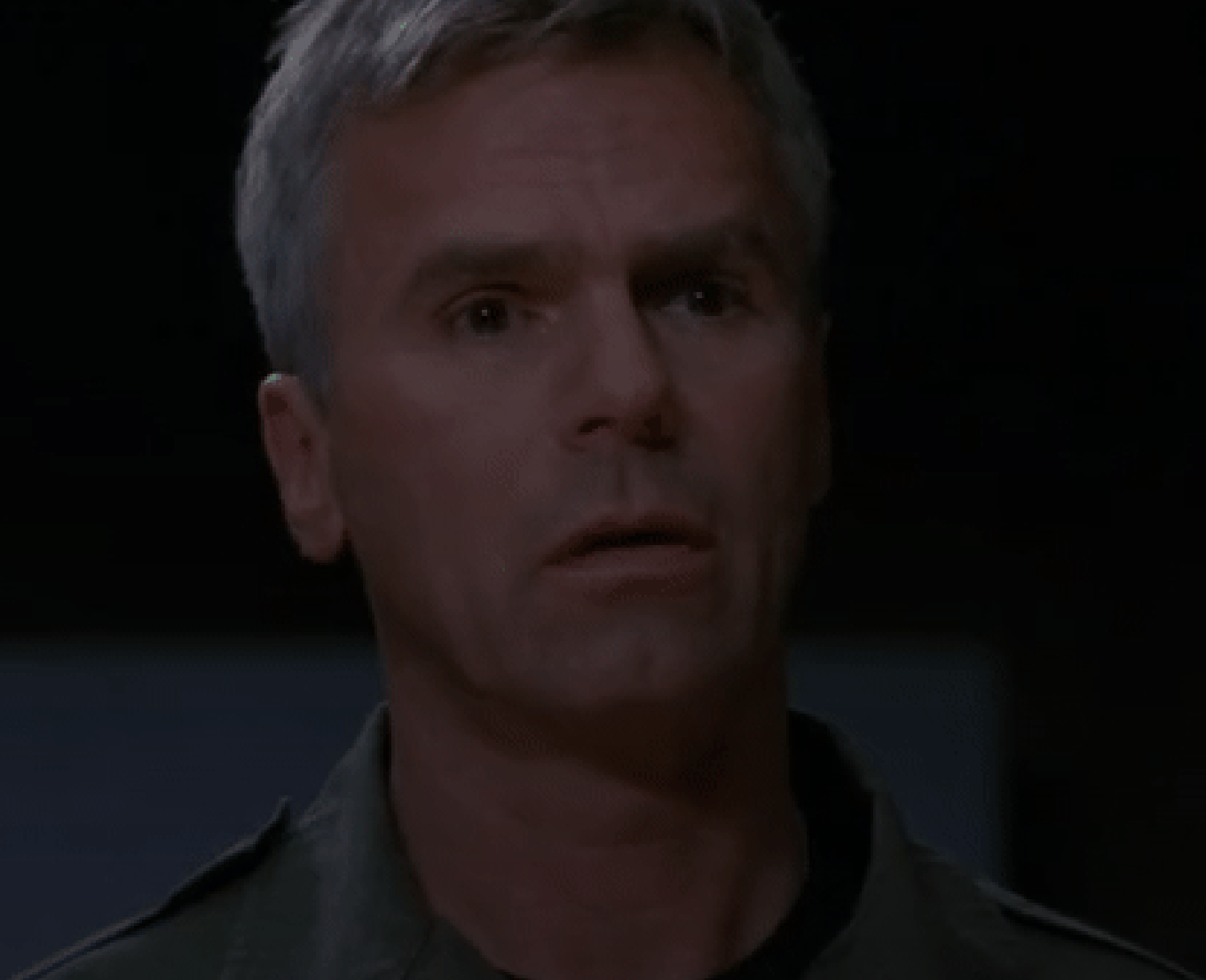
These are great tools, but they solve symptoms: they're palliative

A close-up photograph of a silver spoon holding a small, light-colored, textured food item, possibly a piece of fruit or vegetable. The spoon is positioned in the lower right quadrant of the frame. The background is a solid, dark blue color. The text "ANYWAY," is overlaid on the left side of the image, in a yellow, stylized font with a black outline.

ANYWAY,

A man in a military uniform, likely a pilot, is shown from the chest up. He is wearing a dark uniform with a pilot's wing badge on his left chest and a rank insignia on his right shoulder. The image has a strong blue tint. Overlaid on the image is the text "Why not use a programming language?" in a white, serif font.

Why not use a programming
language?





Infrastructure as Code

Pulumi's open source infrastructure as code SDK enables you to create, deploy, and manage infrastructure on any cloud, using your favorite languages.

Pulumi

- Same stack for infra and applications
- Use familiar programming languages
- Server Side Apply
- CRD support

Pulumi

```
1 crd2pulumi --nodejsPath=pulumi-sdk-nodejs --force crds.yaml
2 crd2pulumi --goPath=pulumi-sdk-go --force https://github.com/raw/thingy/crds.yaml
3 crd2pulumi --dotnetPath=pulumi-sdk-dotnet --force https://doc.crds.dev/package/name
```



Kubernetes as Code

cdk8s is an open-source software development framework for defining Kubernetes applications and reusable abstractions using familiar programming languages

cdk8s

- Use familiar programming languages
- Server Side Apply (kubectl)
- CRD support

cdk8s

```
1  language: typescript
2  app: node main.js
3  imports:
4    - k8s
5    - https://github.com/raw/thingy/crds.yaml
6    - https://doc.crd.dev/package/name
```


Which?

It's personal choice

- Pulumi requires additional work to consume CRDs as code
- cdk8s makes this much easier
- However, Pulumi may be the same language you're building your platform with
- and Pulumi can apply to the cluster

Demo

Fingers crossed!

Summary

- Using programming languages doesn't mean less LOC
 - but our opportunity to abstract and compose is much greater
- Use existing tooling
- Testable
- Distributable / sharable

A low-resolution, pixelated photograph of a man with short brown hair, wearing a dark jacket, holding a small, light-colored dog. The background is blurred, showing green foliage. The text "Thank You" is overlaid in the center in a white serif font, and "Questions?" is overlaid below it in a smaller, lighter grey serif font.

Thank You

Questions?