Providing a Rich Interface to Prometheus

David Flanagan
rawkode

David Flanagan

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

- https://rawkode.live
- https://rawkode.chat
-

I’m here to remove YAML from your lives

What’s the standard operating procedure look like?

Deploy Your Application

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: example-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: example-app
  template:
    metadata:
      labels:
        app: example-app
    spec:
      containers:
      - name: example-app
        image: fabxc/instrumented_app
        ports:
        - name: web
          containerPort: 8080
```

Expose it over a Service

```yaml
kind: Service
apiVersion: v1
metadata:
  name: example-app
  labels:
    app: example-app
spec:
  selector:
    app: example-app
  ports:
  - name: web
    port: 8080
```

Create a Service or Pod Monitor

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: example-app
  labels:
    team: frontend
spec:
  selector:
    matchLabels:
      app: example-app
  endpoints:
  - port: web
```

Connect ServiceMonitor to Prometheus

```yaml
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: prometheus
spec:
  serviceAccountName: prometheus
  serviceMonitorSelector:
    matchLabels:
      team: frontend
  resources:
    requests:
      memory: 400Mi
  enableAdminAPI: false
  alerting:
    alertmanagers:
    - namespace: default
      name: alertmanager-example
      port: web
  ruleSelector:
    matchLabels:
      role: alert-rules
      prometheus: example
  ruleNamespaceSelector:
```

Deploy AlertManager

```yaml
apiVersion: monitoring.coreos.com/v1
kind: Alertmanager
metadata:
  name: example
spec:
  replicas: 3
  alertmanagerConfigSelector:
    matchLabels:
      alertmanagerConfig: example
---
apiVersion: monitoring.coreos.com/v1alpha1
kind: AlertmanagerConfig
metadata:
  name: config-example
  labels:
    alertmanagerConfig: example
spec:
  route:
    groupBy: ['job']
    groupWait: 30s
    groupInterval: 5m
    repeatInterval: 12h
    receiver: 'webhook'
  receivers:
```

Add a PrometheusRule

```yaml
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  creationTimestamp: null
  labels:
    prometheus: example
    role: alert-rules
  name: prometheus-example-rules
spec:
  groups:
  - name: ./example.rules
    rules:
    - alert: ExampleAlert
      expr: vector(1)
```


I am not fond of this.


Wtf?


How do we make this better?
The um...

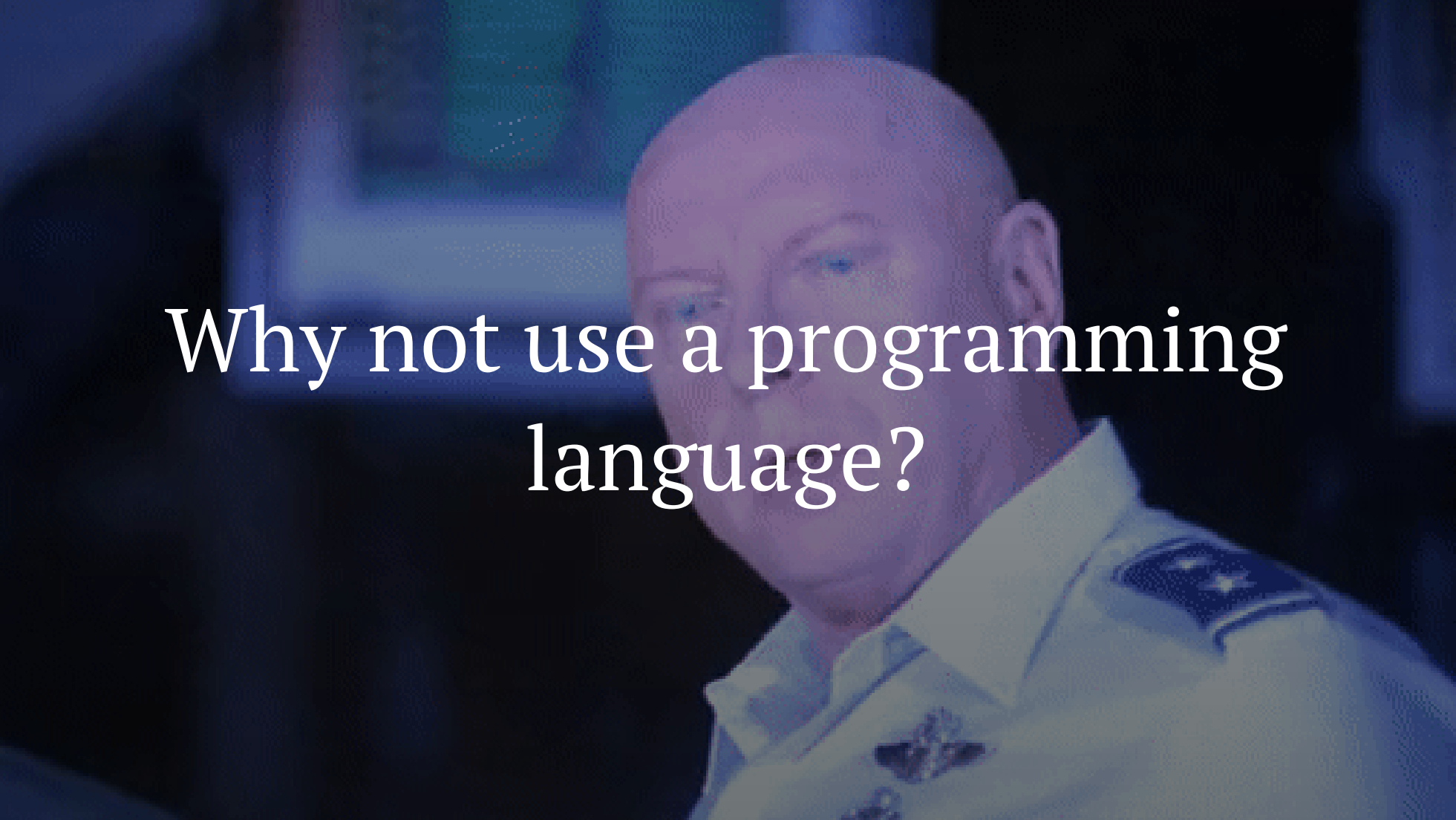
Spoiler Alert

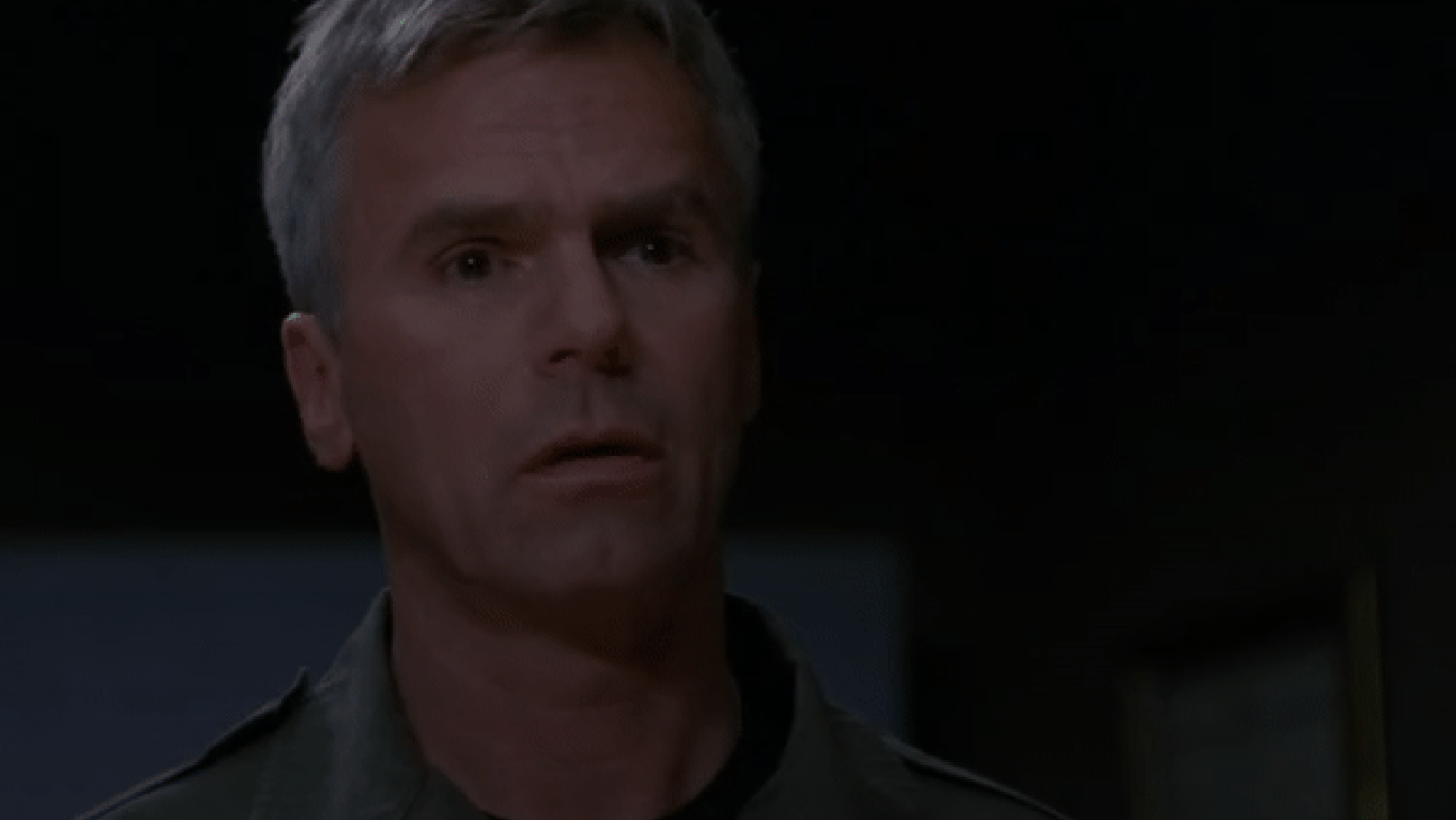
The answer isn't YAML or templating

- Helm
- Kustomize
- ytt

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


ANYWAY,


Why not use a programming language?



Pulumi

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

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

cdk8s

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

```yaml
language: typescript
app: node main.js
imports:
  - k8s
  - https://github.com/raw/thingy/crds.yaml
  - https://doc.crds.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

Thank You

Questions?