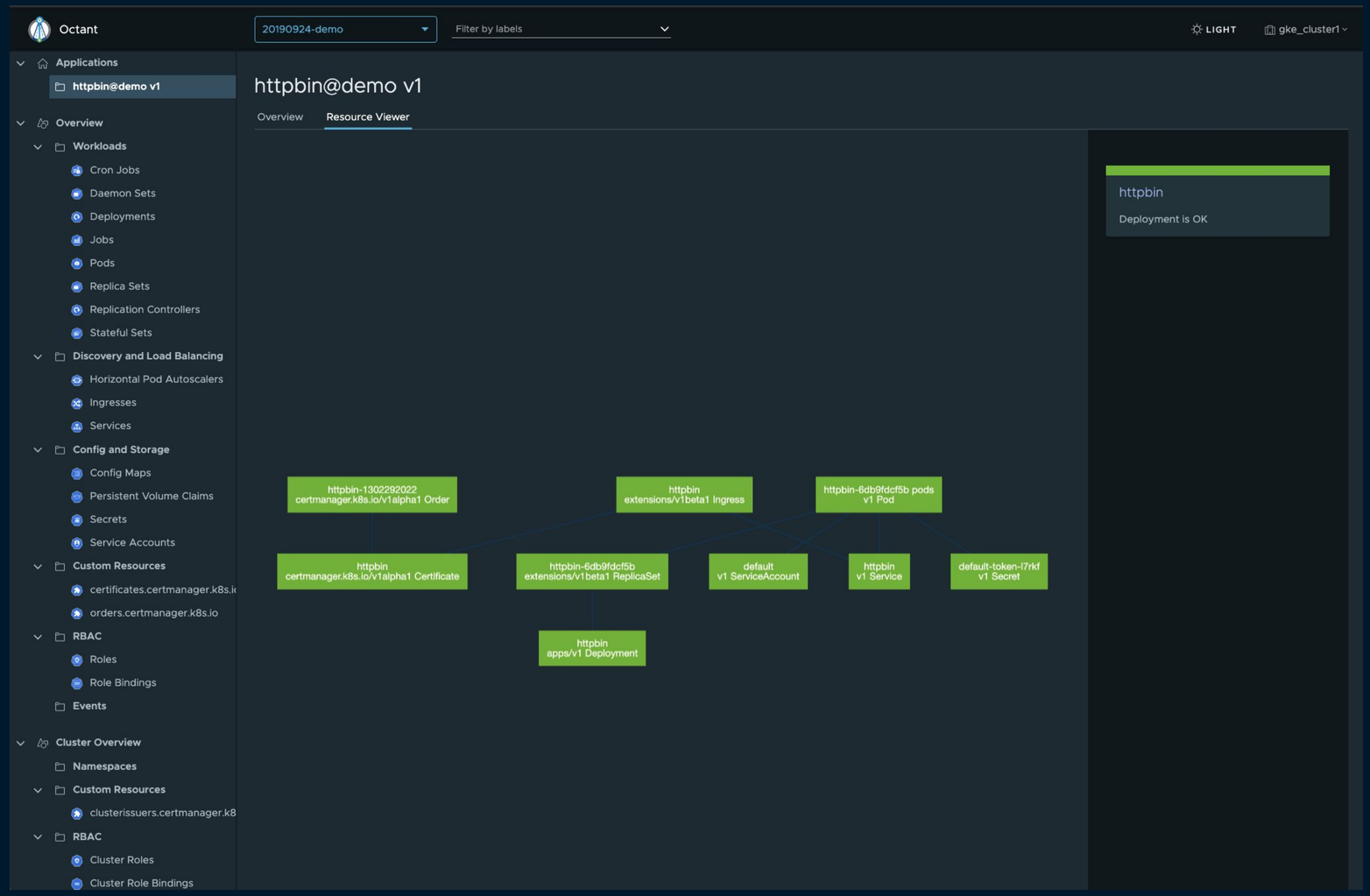
In Search of the Kubernetes *Rails* Moment

Bryan Liles
Senior Staff Engineer, VMware

I like:

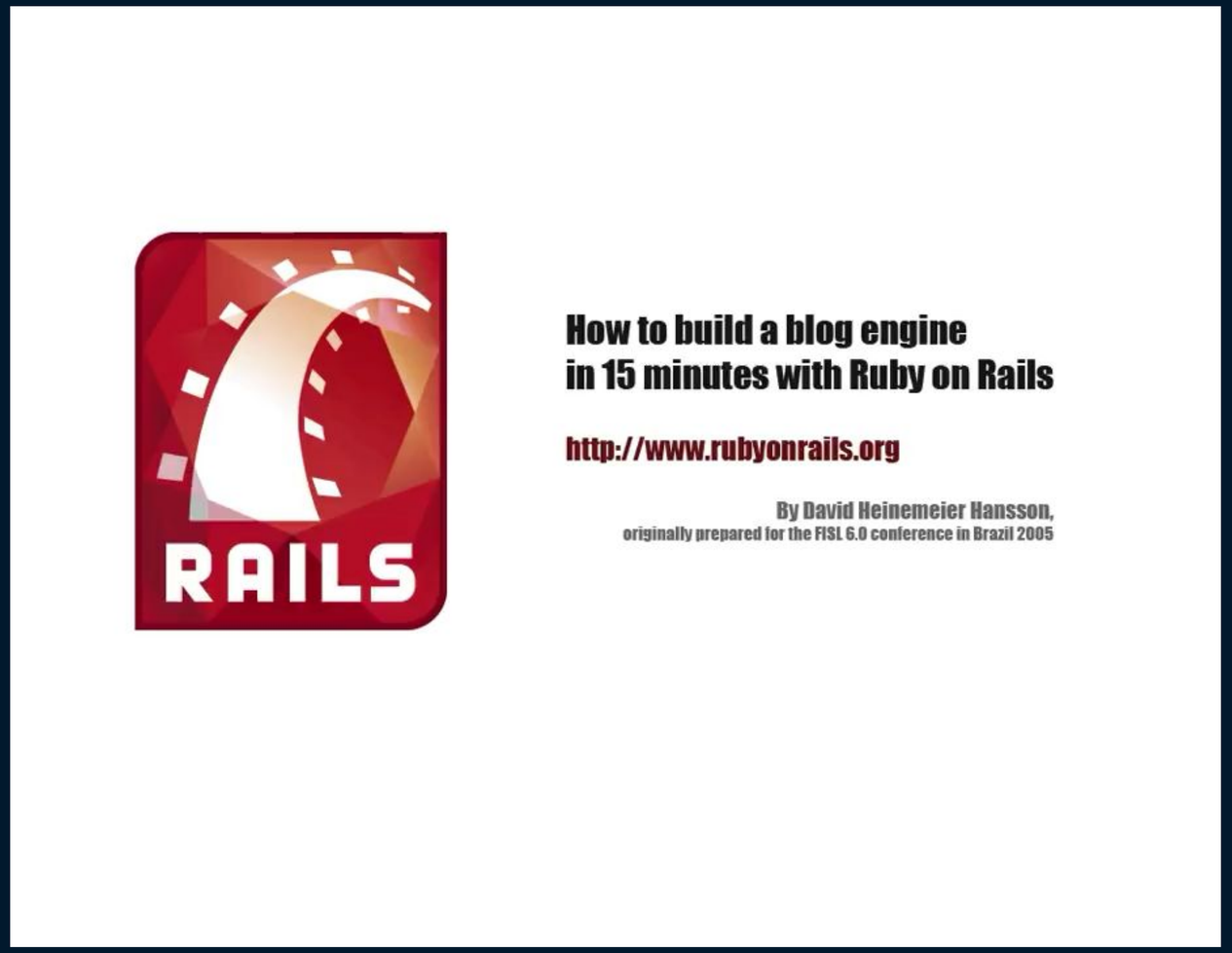
Celebrating the Hip Hop Culture
Cars
Computers
Creating a better world



Octant

understanding what's going on in your Kubernetes clusters shouldn't be hard

https://github.com/vmware-tanzu/octant



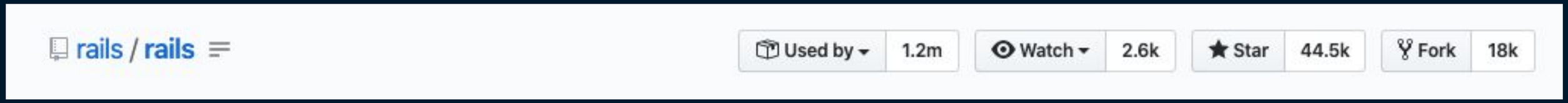
Remember 2005?

Rails didn't create the idea of web frameworks, but it did popularize the notion of

convention over configuration

What lessons could the Kubernetes community *learn* from Rails?

Vanity Metrics?



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
 replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

By default replicas are set to 1, so setting it might be superfluous

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
   matchLabels:
     app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

By default replicas are set to 1, so setting it might be superfluous

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Selectors and pod template labels usually match, so combine them

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Selectors and pod template labels usually match, so combine them

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ~~ports:~~
        ~~- containerPort: 80~~
```

Assume everything is at port 80?

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
```

Assume everything is at port 80?

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
```

Introducing conventions into object manifests is easy?

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        livenessProbe:
          httpGet:
            path: /healthz
            httpHeaders:
            - name: You-Cool
              Value: Yup
```

I want a liveness check

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        livenessProbe:
          httpGet:
            path: /healthz
            httpHeaders:
            - name: You-Cool
              Value: Yup
        readinessProbe:
          exec:
            command:
            - cat
            - /swag
```

I want a readiness check

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        volumeMounts:
        - name: config
          mountPath: /config
        livenessProbe:
          httpGet:
            path: /healthz
            httpHeaders:
            - name: You-Cool
              Value: Yup
        readinessProbe:
          exec:
            command:
            - cat
            - /swag
      volumes:
        name: config
        hostPath:
          path: /config
```

I want to access a volume

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        env:
        - name: APP_USERNAME
          valueFrom:
            secretKeyRef:
              name: snitches
              key: username
        volumeMounts:
        - name: config
          mountPath: /config
        livenessProbe:
          httpGet:
            path: /healthz
            httpHeaders:
            - name: You-Cool
              value: Yup
        readinessProbe:
          exec:
            command:
            - cat
            - /swag
      volumes:
        name: config
        hostPath:
          path: /config
```

I want to access secrets

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  template:
    metadata:
      labels:
        app: nginx
    spec:
      podAffinity:
        requiredDuringSchedulingIngnoredDuringExecution:
        - labelSelector:
          matchExpressions:
          - key: hood
            operator: In
            values:
            - park-hills
          topologyKey: failure-domain.beta.kubernetes.io/zone
      podAntiAffinity:
        preferredDuringSchedulingIngnoredDuringExecution:
        - weight: 100
          podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: security
              operator: In
              values:
              - stapleton
          topologyKey: failure-domain.beta.kubernetes.io/zone
      containers:
      - name: nginx
        image: nginx:1.7.9
        env:
        - name: APP_USERNAME
          valueFrom:
            secretKeyRef:
              name: snitches
              key: username
        volumeMounts:
        - name: config
          mountPath: /config
        livenessProbe:
          httpGet:
            path: /healthz
            httpHeaders:
            - name: You-Cool
              value: Yup
        readinessProbe:
          exec:
            command:
            - cat
            - /swag
      volumes:
        name: config
        hostPath:
          path: /config
```

I want to set up affinities

Convention over configuration is not simply eliding functionality

Kubernetes complexity is *necessary* complexity

How can we capture the essence of Rails without diluting the power of the Kubernetes platform?

Before we continue, let's agree on one thing...

YAML is for computers...

Before we continue, let's agree on one thing...

YAML is for computers, therefore
we can consider YAML as
assembler for Kubernetes.

mov ax, 13h

```
mov ax, 13h
int 10h
```

```
mov ax, 13h
int 10h
mov ax, 0A000h
```

```
mov ax, 13h
int 10h
mov ax, 0A000h
mov es, ax
```

```
mov ax, 13h
int 10h
mov ax, 0A000h
mov es, ax
mov ax, 0
```

```
mov ax, 13h
int 10h
mov ax, 0A000h
mov es, ax
mov ax, 0
mov di, ax
```

```
mov ax, 13h
int 10h
mov ax, 0A000h
mov es, ax
mov ax, 0
mov di, ax
mov dl, 7
```

```
mov ax, 13h
int 10h
mov ax, 0A000h
mov es, ax
mov ax, 0
mov di, ax
mov dl, 7
mov [es:di], dl
```

Personally, I love operating at this level,
but not when I need to get stuff done

```
import win32gui
import win32api

dc = win32gui.GetDC(0)
c = win32api.RGB(127,127, 127)
win32gui.SetPixel(dc, 0, c)
```

"I don't want to write YAML. I want my app to tell something to deploy it"

Back in 2000, LLVM was created to support dynamic complication techniques

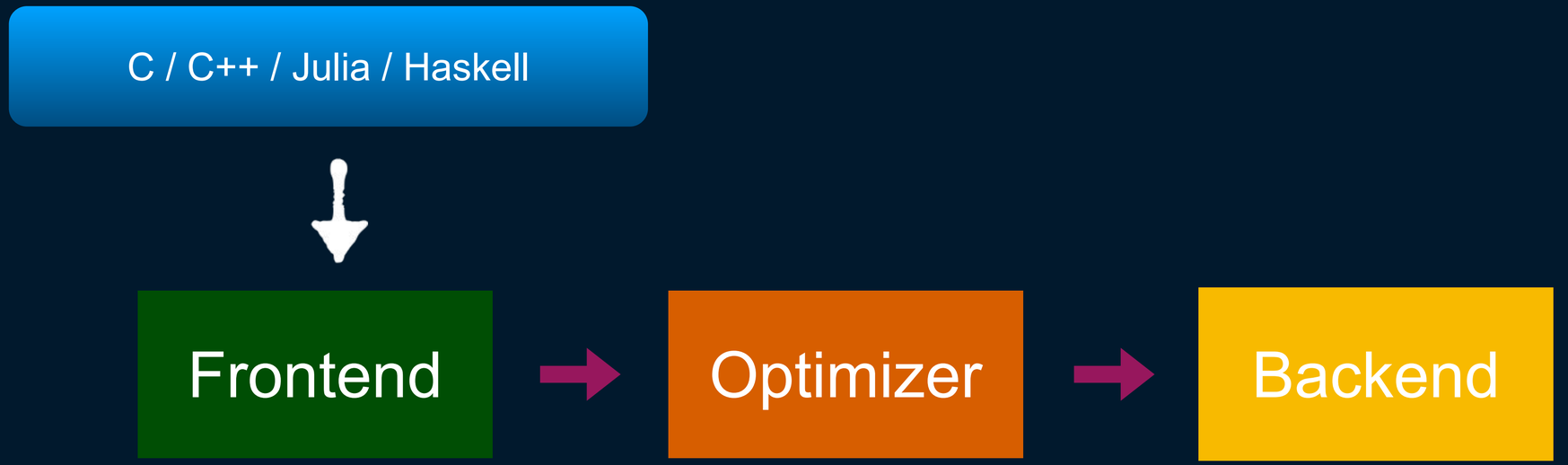
LLVM

Frontend

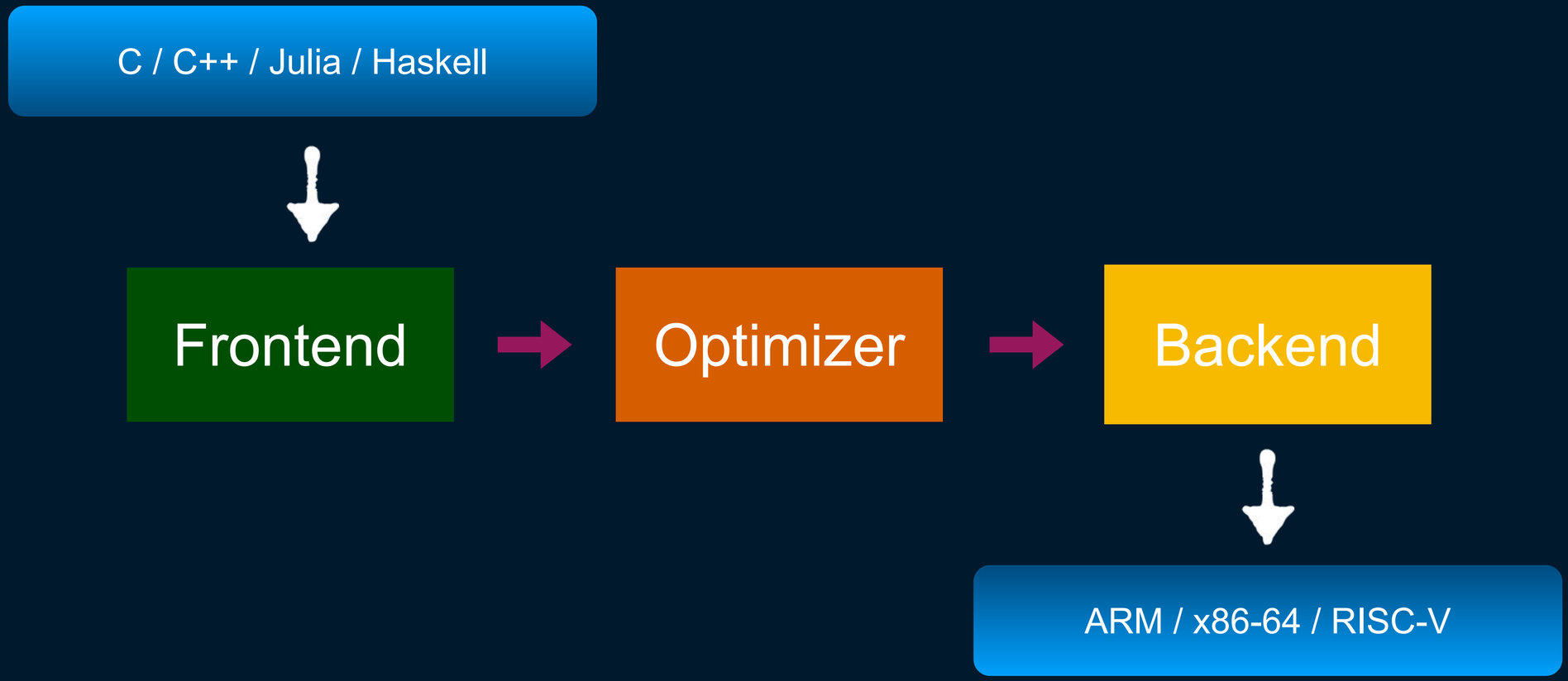
LLVM

LLVM

LLVM

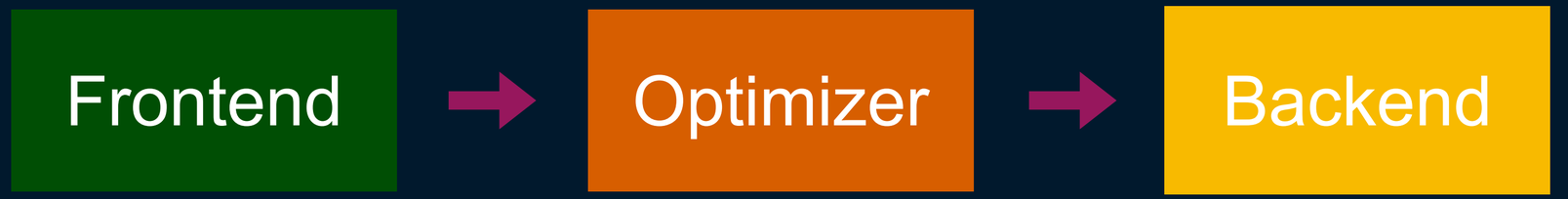


LLVM


C / C++ / Julia / Haskell
Frontend
Optimizer
Backend
ARM / x86-64 / RISC-V

LLVM Why?

- Optimization
- JIT
- Target multiple architectures
- Target multiple paradigms


Frontend
Optimizer
Backend

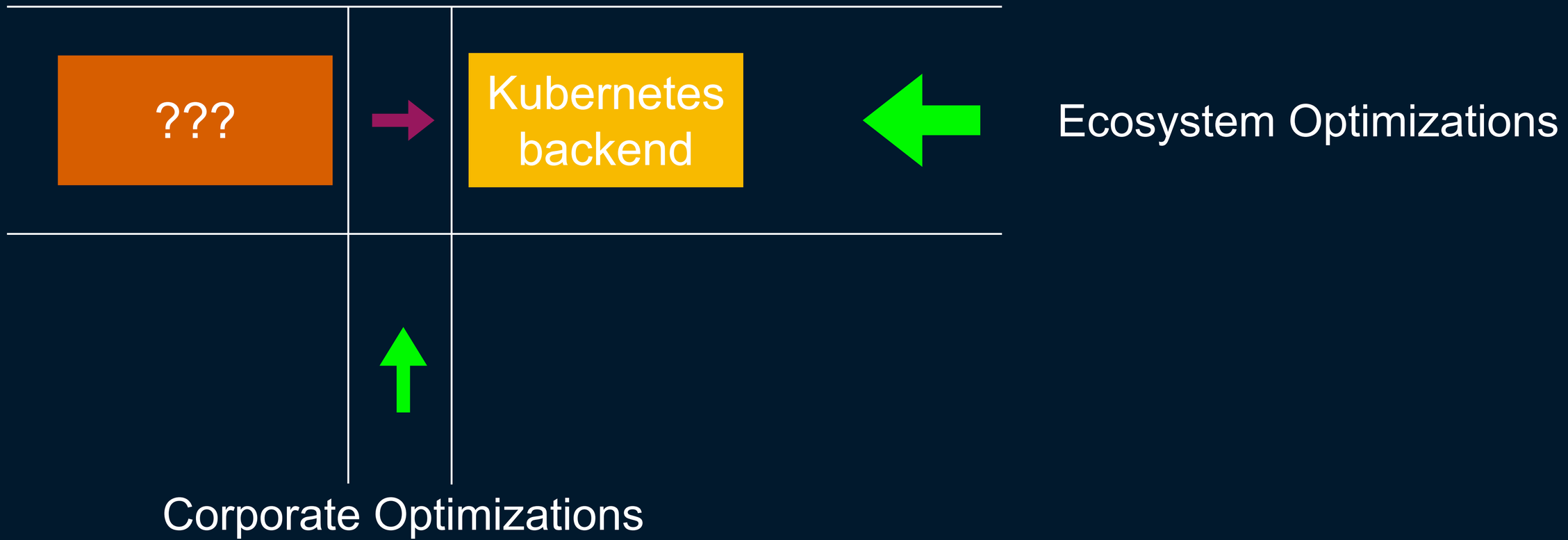

Our app / it's intentions
???
Kubernetes backend

Kubernetes is a vehicle taking us to our destination and not the destination itself...

- Continuous Integration
- Continuous Deployment
- Do security stuff
- Gates
- Auditing

Kubernetes Backend

- Generate YAML
- Apply security policies
- Apply labels / annotations
- Taints and tolerations
- Identify proper workload type
- Wire up observability


???
Kubernetes backend
Ecosystem Optimizations
Corporate Optimizations

What Rails convention over configuration means

I can accept the defaults until I outgrown them or they no longer satisfy my needs

The things we build should provide good defaults...

Our defaults should be easy to override...

OUR PRIMARY JOB IS FIND THOSE GOOD DEFAULTS!

Other things Kubernetes can learn from Rails

Rails freed people from thinking about serving web requests ...

... which led to rethinking how we serve web requests in the Sinatra project ...

... which allowed Heroku to think about PaaS ...

... which gave us the concept of **git push heroku:master** ...

... which inspired Deis Labs to imitate that concept ...

... something, something, something ...

… Helm

DHH developed Rails in TextMate...

... which moved developers to the Mac to use TextMate ...

... which led to the resurgence of the text editor ...

... which ultimately gave us the vibrant text editing community led by Microsoft's VS Code

Rails freed Mac users to think about other problems ...

… which led to creation of Homebrew

Rails has inspired a whole generation of developers to think outside of the box

Kubernetes' *Rails* moment is simply making a better Kubernetes

Kubernetes' *Rails* moment is making the ecosystem better...

... because Kubernetes allows us to think about more important things

We've been here before...

Linux went through the same journey

Linux wasn't the destination.
Linux enabled journey.

Let's work to make Kubernetes enable our journeys. Let's find Kubernetes' *Rails* moment


FX