GitOps, DSL and App Model: Getting Started Building **Developer Centric Kubernetes**

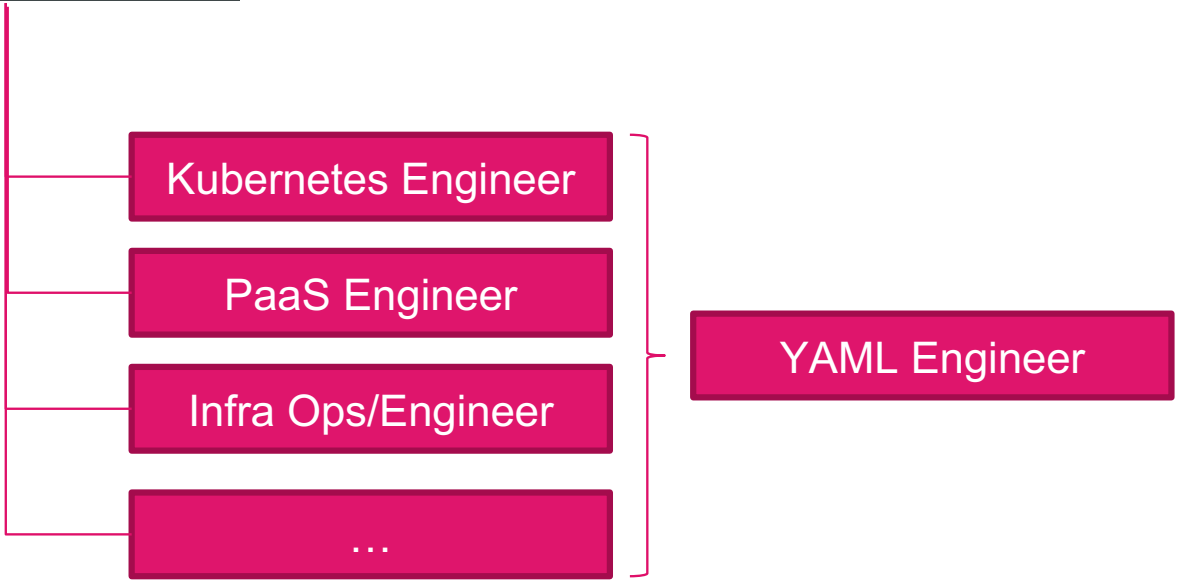
Lei Zhang (Harry) @Alibaba, CNCF Ambassador

CLOUD NATIVE COMPUTING FOUNDATION

Who am I?

I am a …

- Platform builder @Alibaba



What do I build?

Well … lots of platforms on top of k8s

Users (*developers, operators*)

PaaS | Serverless | Operator Platform | E-business PaaS

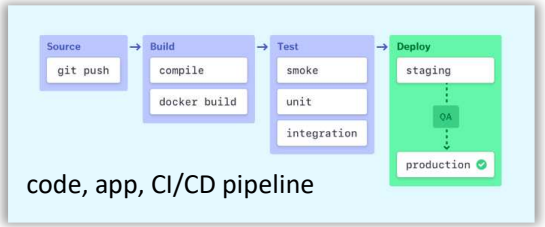
Kubernetes

Me & team

Why do we build platforms on top of K8s?

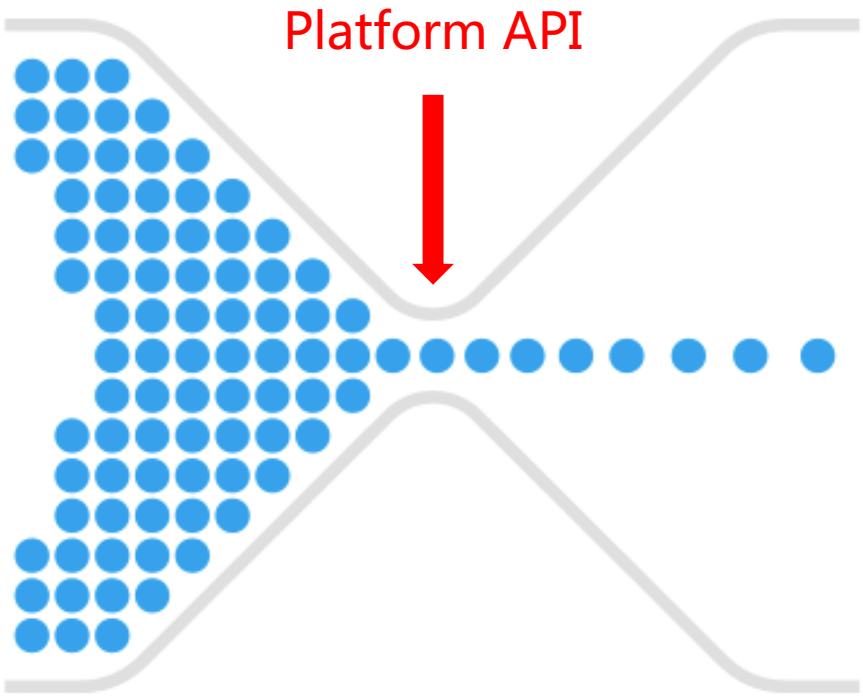
CLOUD NATIVE
COMPUTING FOUNDATION

Serve Our Users better!

	API & Primitives	Levels of Abstraction	User Interfaces
users' expectation		**scaling** • auto scale +100 instances when latency > 10% **rollout** • promote the canary instance with step of 10%	 GUI  CLI  IaC
what k8s provides	 Deployment, Pod, HPA, Controller, Sidecar, Node, NetworkPolicy, CR/CRD	 HorizontalPodAutoscaler, Deployment, Service, Ingress, Prometheus Service Monitor, CustomMetricsServer, Istio Virtual Service	 

But, Platform Is Not Silver Bullet

The **unlimited** capabilities in k8s ecosystem



Users ' rapid growing requirements

~~Build platforms on top of K8s~~

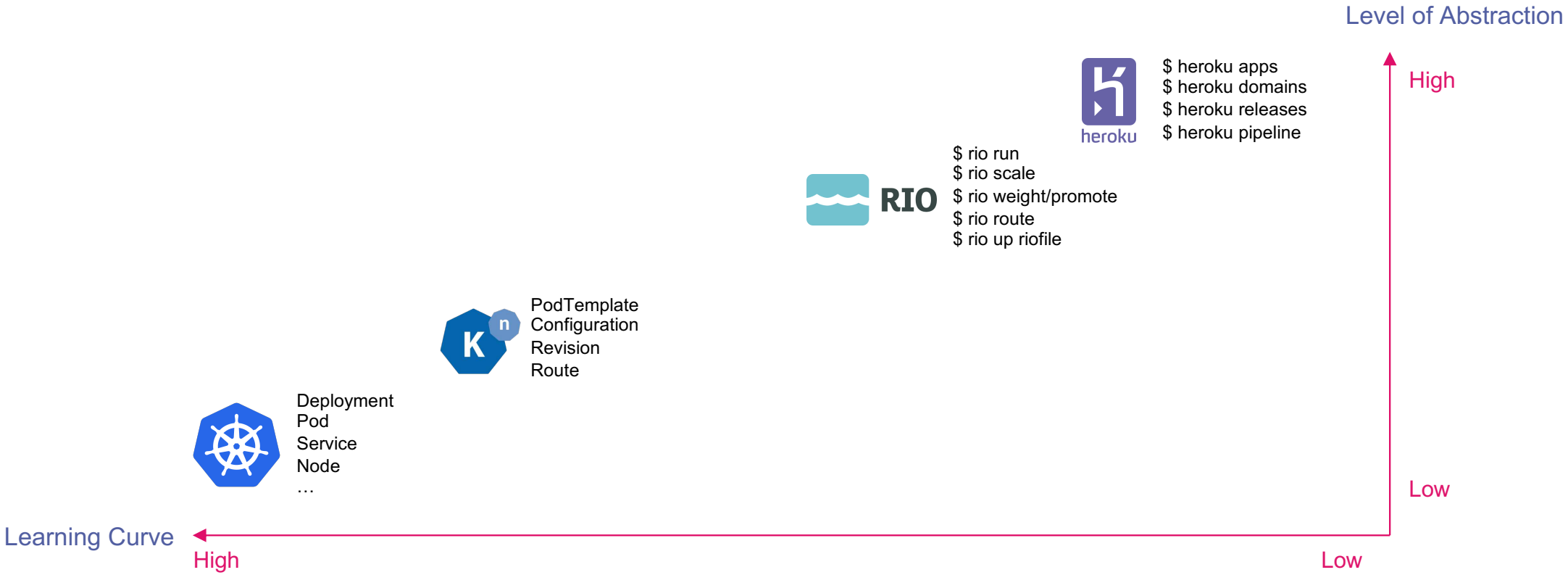
Build an developer centric k8s?

A k8s that ***speaks higher level API***, is ***end user friendly,*** **and still** ***highly extensible?***

CLOUD NATIVE
COMPUTING FOUNDATION

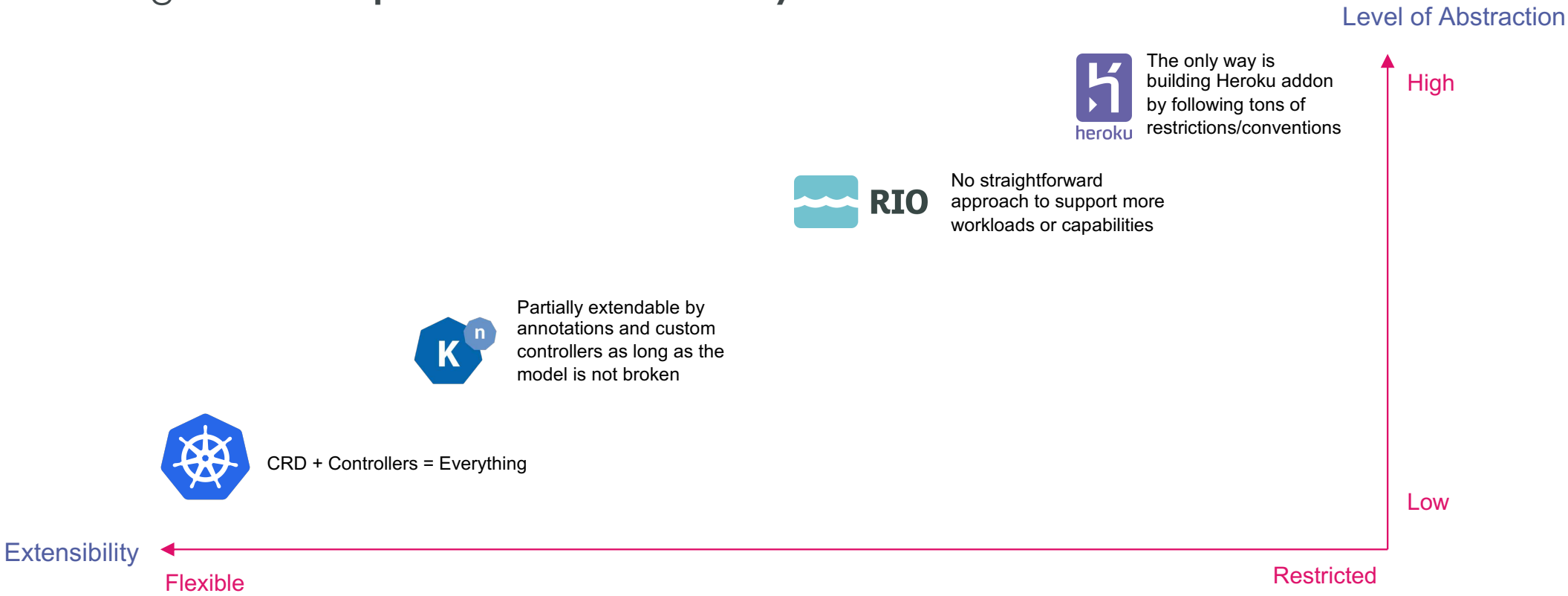
Higher Level API

- User facing abstractions for **workloads** and **operational capabilities**

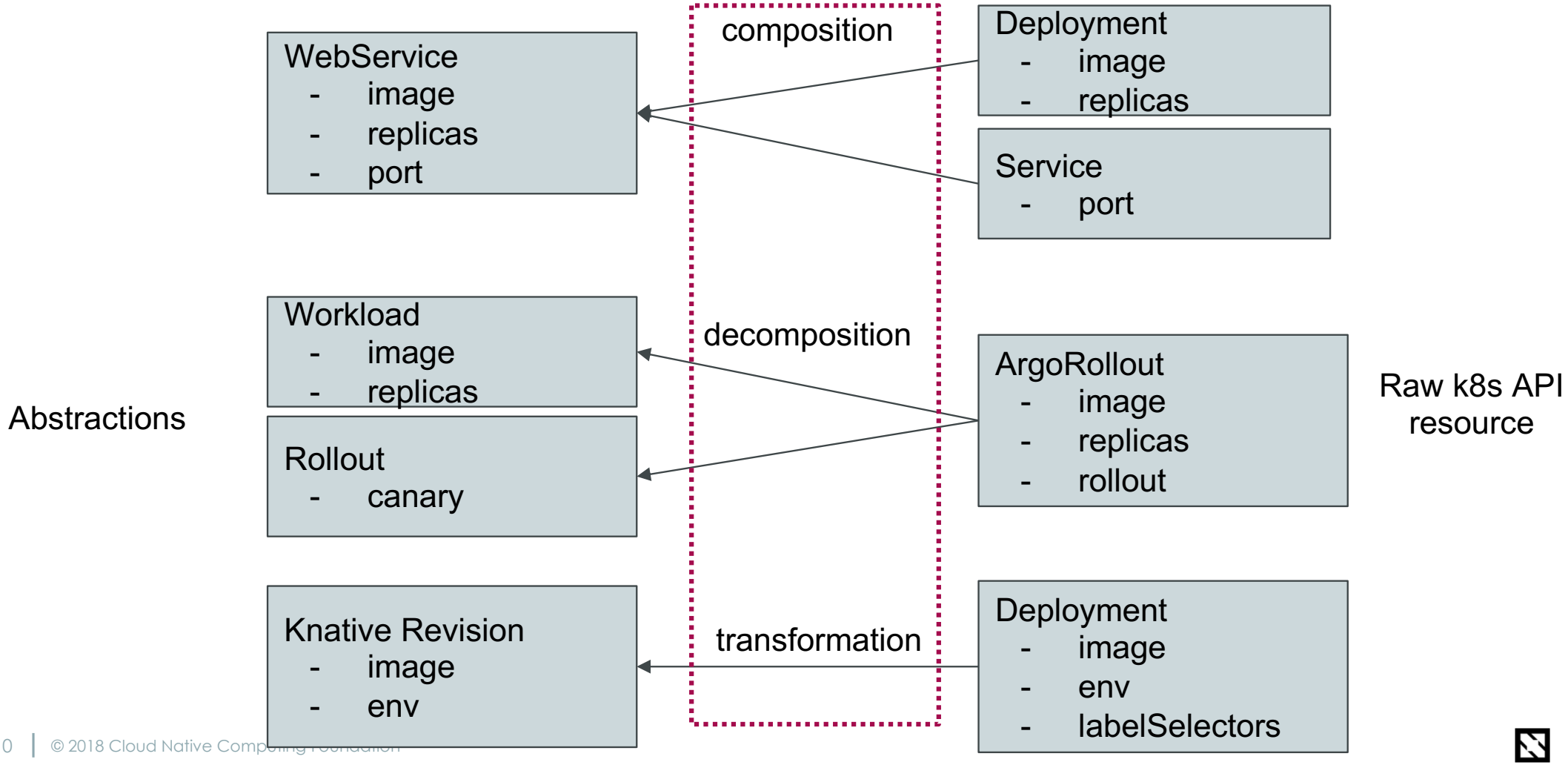


Abstractions vs Extensibility

- Higher level abstraction can significantly **lower the learning curve**, though with **compromise on extensibility**



Building Abstractions on Server Side is Not Easy



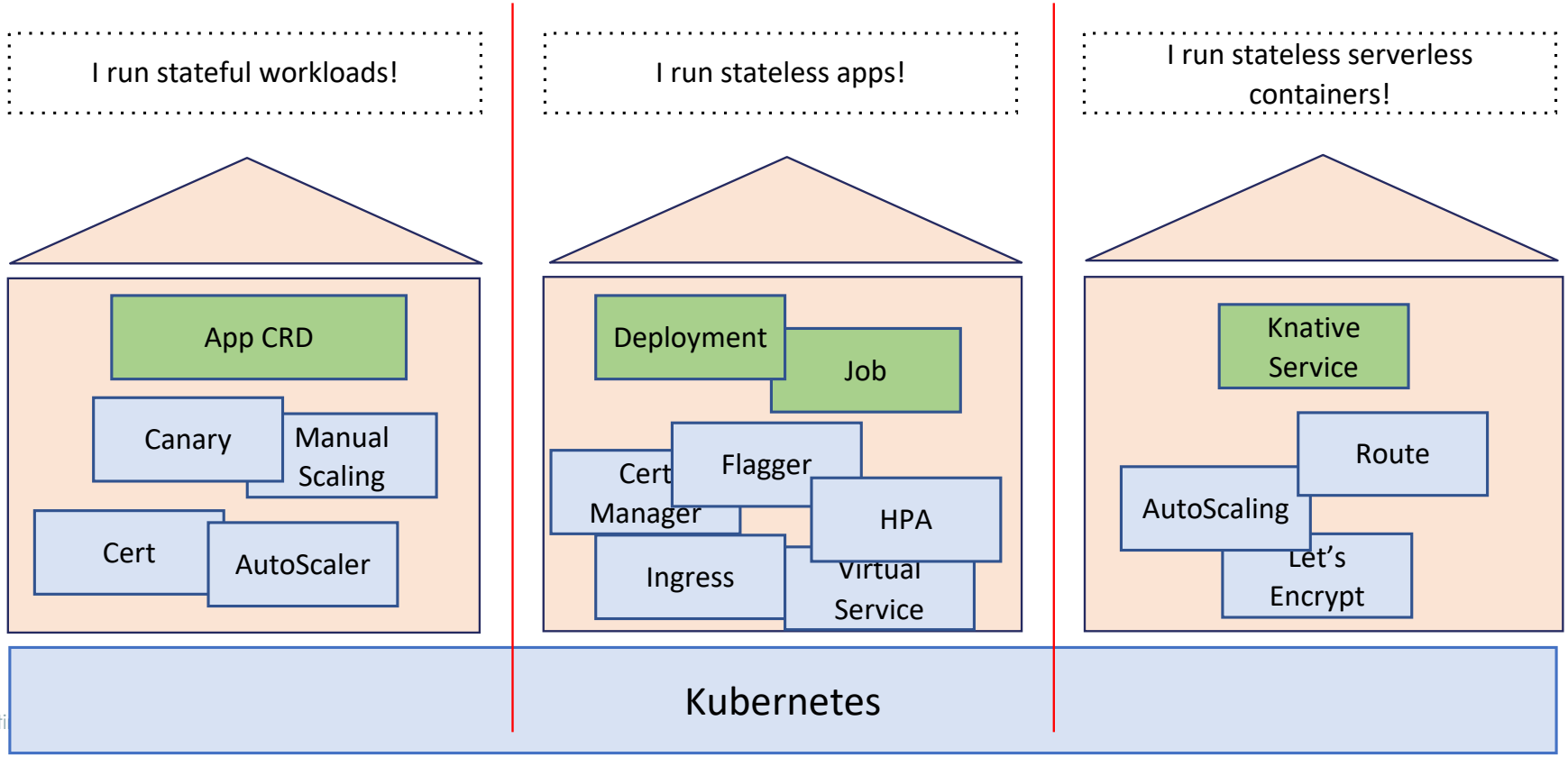
 |

Abstractions Create Silos

- Because we can't use **single abstraction** to serve all

Users

No interoperability, reusability, or portability, reinventing wheels due to different APIs are spoken



Build an developer centric k8s? Yes!

But How?

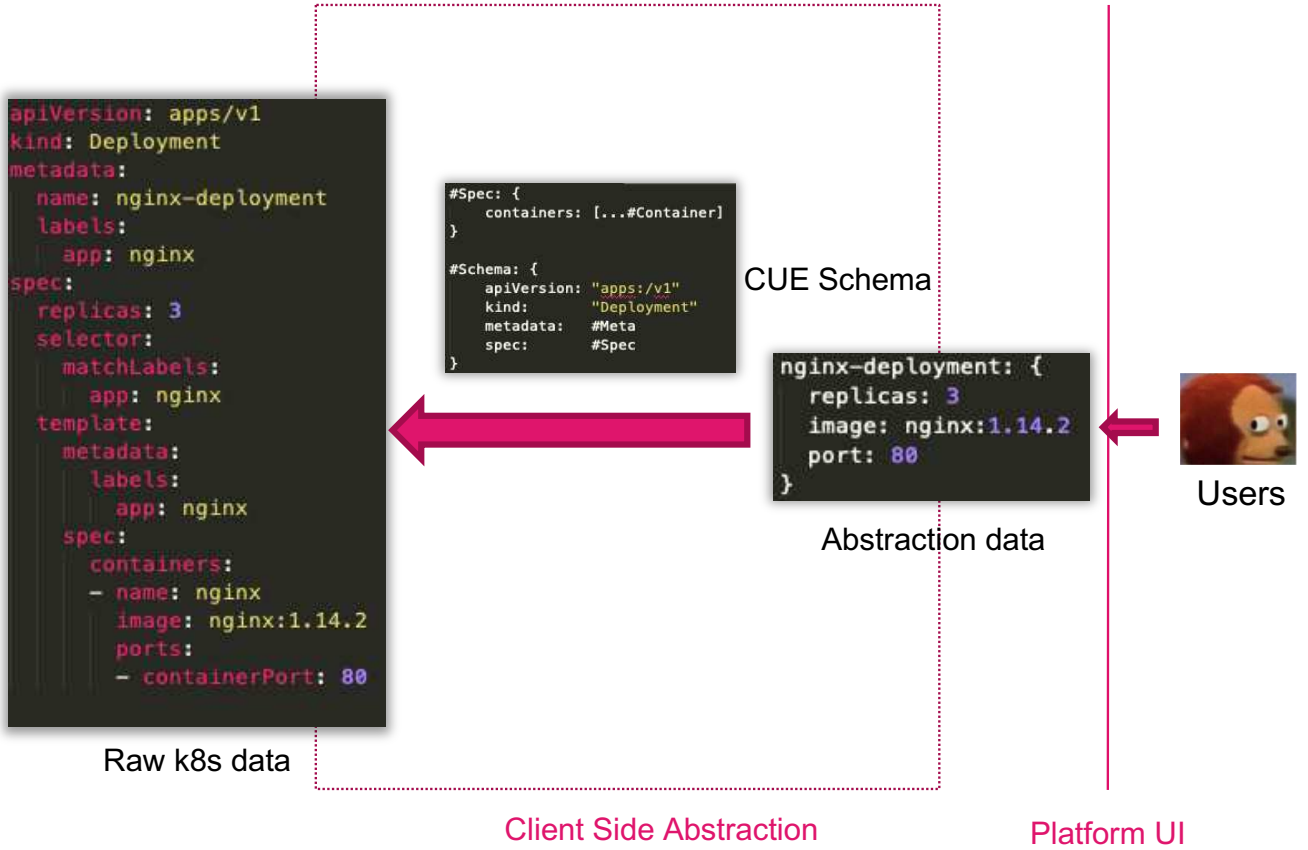
CLOUD NATIVE
COMPUTING FOUNDATION

Easier Abstraction – DCL (Data Configuration Language)

Abstraction is about **data manipulation** (data = k8s API resource), this can be handled by DCL way easier than CRD + controller

CUE

- Focus only on manipulating configuration data, instead of "writing code"
  - the main difference between cdk8s
- **Superset** of JSON
- Define **schema** and **value** in consistent gramma



 |

Standardized Higher Level API - App Model

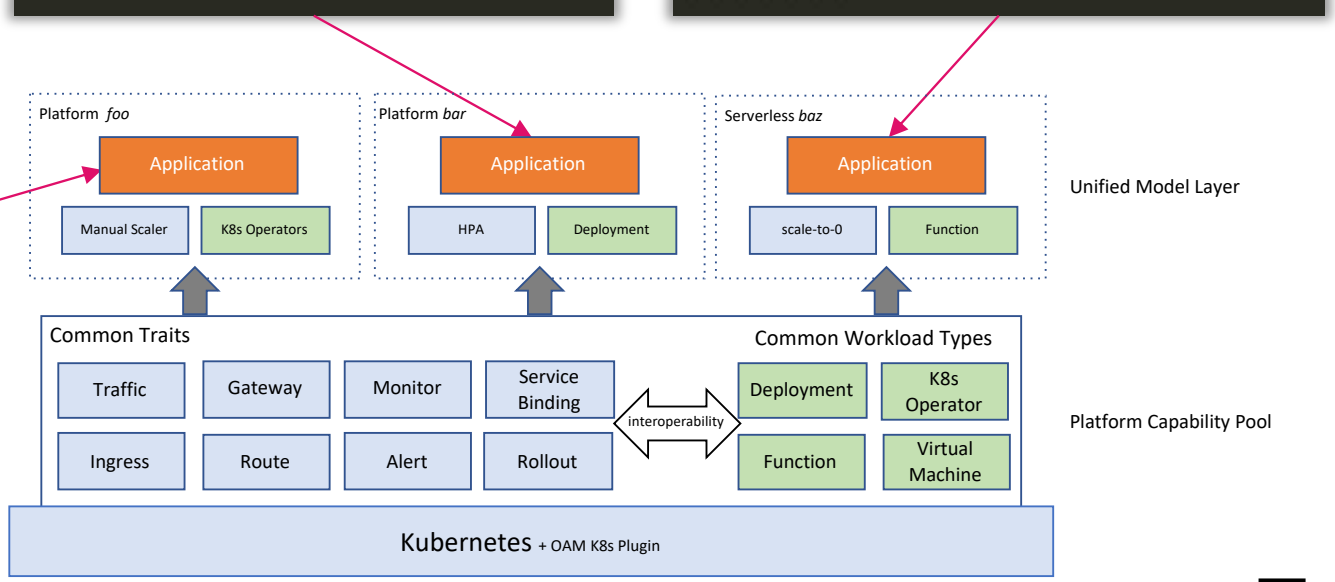
Open Application Model (OAM)

- Define **application centric primitives** enforced by OAM spec
- **Break the silos!**

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: ApplicationConfiguration
metadata:
  name: stateful-app
spec:
  components:
    - componentName: nginx-deployment
    - componentName: postgres-operator
      traits:
        - trait:
            apiVersion: core.oam.dev/v1alpha2
            kind: ManualScaler
            spec:
              replicaCount: 2
```

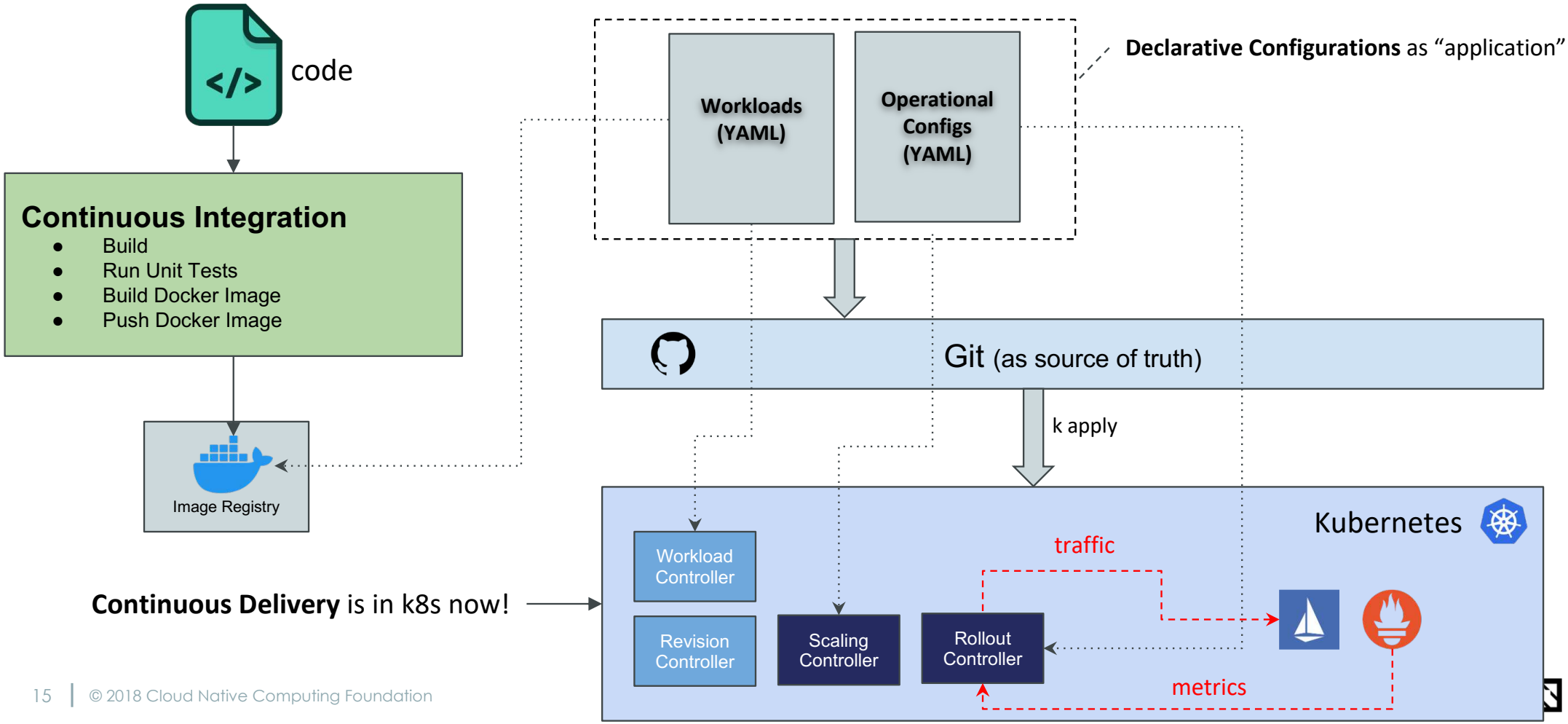
```yaml
apiVersion: core.oam.dev/v1alpha2
kind: ApplicationConfiguration
metadata:
  name: stateless-app
spec:
  components:
    - componentName: nginx-deployment
      traits:
        - trait:
            apiVersion: autoscaling/v2beta2
            kind: HorizontalPodAutoscaler
            spec:
              minReplicas: 1
              maxReplicas: 10
```

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: ApplicationConfiguration
metadata:
  name: serverless-app
spec:
  components:
    - componentName: knative-svc
      traits:
        - trait:
            apiVersion: autoscaling.knative.dev/v1
            kind: Autoscaler
            spec:
              minScale: 0
              maxScale: 100
```



 |

The Modularized CD System - GitOps



 |

Let's Put Them Together?

- CUE for abstraction
- OAM for app model
- GitOps for continues delivery

A developer centric k8s is coming up!

Proof-of-Concept

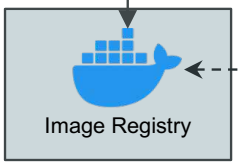
```yaml
name: myapp
components:
  frontend:
    deployment:
      image: inanimate/echo-server
      env:
        PORT: 8080
    traits:
      autoscaling:
        max: 10
        min: 1
      rollout:
        strategy: canary
        step: 5
      expose:
        service:
          type: LoadBalancer
        ports:
          http:
            service_port: 80
            container_port: 8080
```

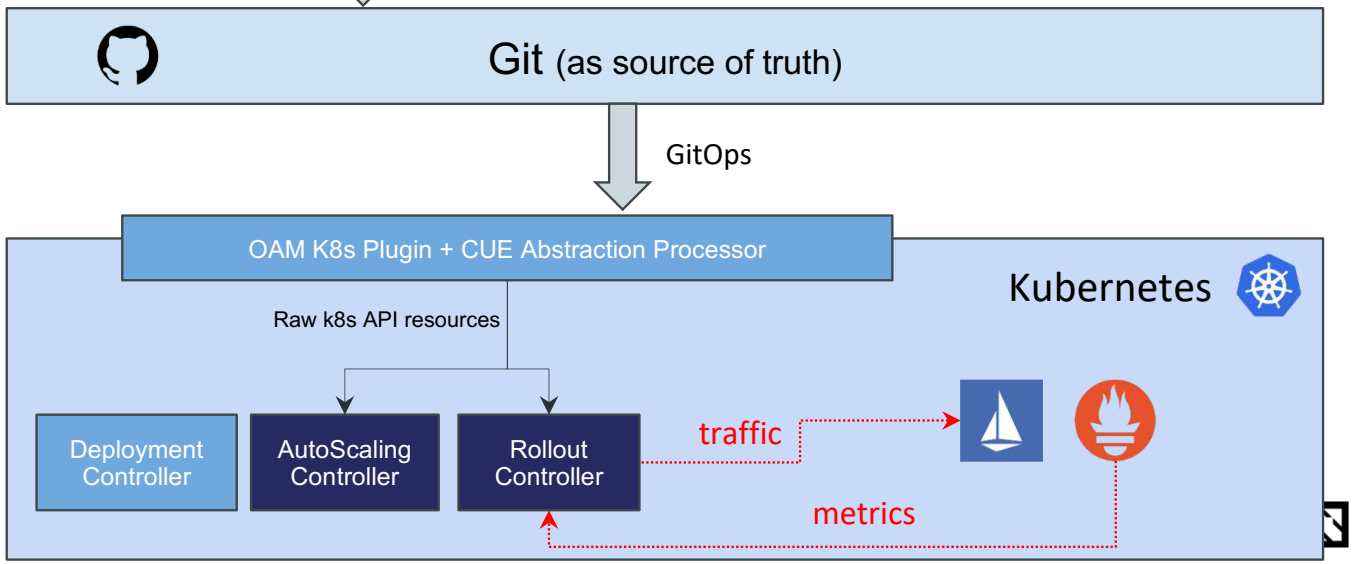
A developer facing app.yaml

- OAM compliant
- CUE based abstraction

Continuous Integration

- Build
- Run Unit Tests
- Build Docker Image
- Push Docker Image





Continuous Delivery

What's Still Missing?

Moving to a real-world solution

- Addon system
  - How to register and discover k8s API/CRD as a OAM workload or trait? And automatically install missing controllers by given CRD?
- Modular CLI/dashboard
  - When I registered a new OAM workload/trait, how my CLI/dashboard immediately show up a new command or tab?
- What will be a developer centric pipeline look like? Will Tekton help us here?

Thank You!

Enjoy the journey of building developer facing Kubernetes!

CLOUD NATIVE
COMPUTING FOUNDATION