Tutorial: Manage Apps and Cloud Resources in Unified Approach with Kubernetes

Jianbo Sun, Andy Shi @Alibaba & Jared Watts @Upbound

Speakers

KubeCon CloudNativeCon North America 2020 Virtual

Jianbo Sun

Senior Engineer @Alibaba, Cloud Native App Platform team, working on Alibaba's web scale app delivery system and OAM/KubeVela

Andy Shi

Staff Engineer,
Alibaba Cloud

Jared Watts

Co-creator of @crossplane_io
Founding Engineer @upbound_io

Checkpoints

1. Prerequisites
2. Introducing KubeVela
3. Ship the first cloud native application with KubeVela
4. Introducing Crossplane
5. Prepare cloud resources with Crossplane
6. Configure the application to consume cloud resources
7. Ship the application to another cloud w/o modification
8. Wrap up

Prerequisites

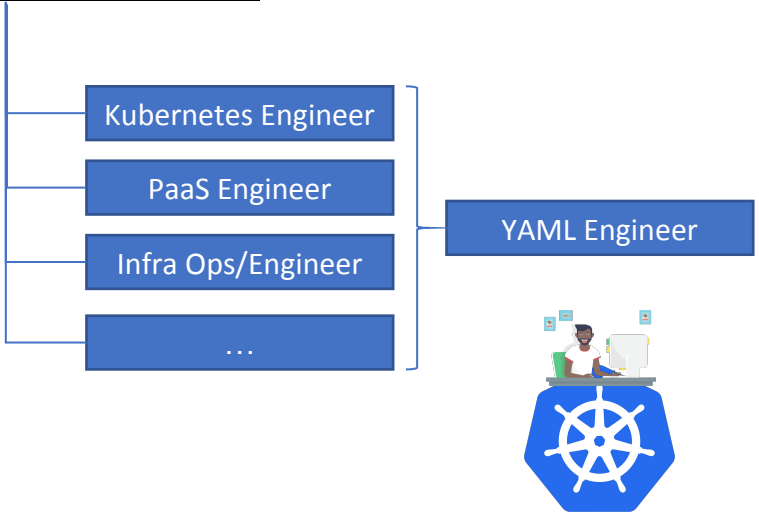
Instruction and files at
https://github.com/oam-dev/kubevela/tree/master/documentation/kubecondemo

1. A Kubernetes cluster >1.16
2. A cloud provider api key and secret
3. Download vela following https://github.com/oam-dev/kubevela/blob/master/README.md
4. Install crossplane (later)
5. Install Kubevela
6. sudo cp bin/vela /usr/local/bin/vela
7. vela install

Who Are We?

We are:

- **Platform Builders** @Alibaba



How Does Alibaba Ensure the Performance of System Components in a 10,000-node Kubernetes Cluster?

Alibaba Developer October 24, 2019 23,653 0

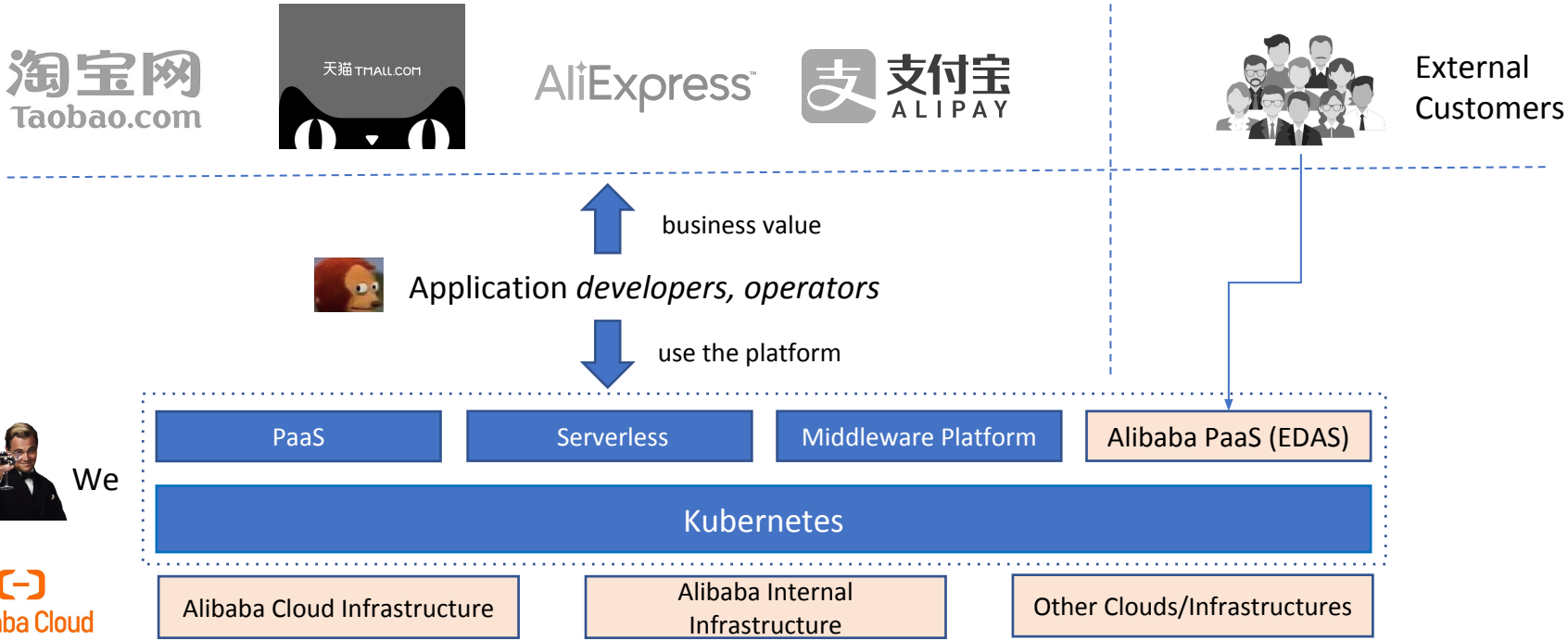
This article looks the problems and challenges that Alibaba Cloud overcame for Kubernetes to function at an ultra-large scale and the specific solutions proposed.

By Zeng Fansong, senior technical expert for the Alibaba Cloud Container Platform, and Chen Jun, systems technology expert at Ant Financial.

This article will take a look at some of the problems and challenges that Alibaba and its ecosystem partner Ant Financial had to overcome for Kubernetes to function properly at mass scale, and will cover the solutions proposed to the various problems the Alibaba engineers encountered. Some of these solutions include improvements to the underlying architecture of the Kubernetes deployment, such as enhancements to the performance and stability of etcd, the kube-apiserver, and kube-controller. These were all crucial for Alibaba to ensure the support needed for the 2019 Tmall 618 Shopping Festival to take full advantage of the 10,000-node Kubernetes cluster deployment. They are also important lessons for any enterprise interested in following Alibaba's footsteps.

What We Build?

- Well … lots of platforms on top of k8s, in hybrid environments



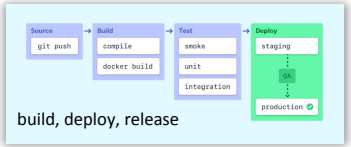
Why Build App Platforms?

Bring **application** context back to k8s!

Application *developers, operators*

what the platforms provide

App-Centric API



App-Centric Abstractions

scaling
- auto scale +100 instances when latency > 10%

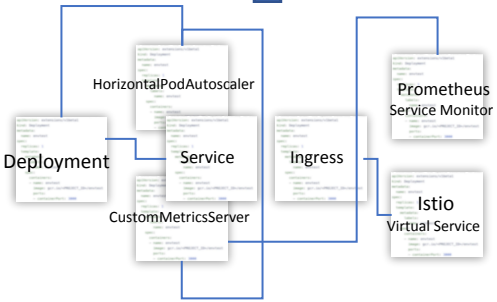
rollout
- promote the canary instance with step of 10%

App-Centric User Interfaces

GUI CLI IaC

what k8s provides

Deployment	Pod
HPA	Controller
Sidecar	Node
NetworkPolicy	CR/CRD



Why Build Unified App Platforms?

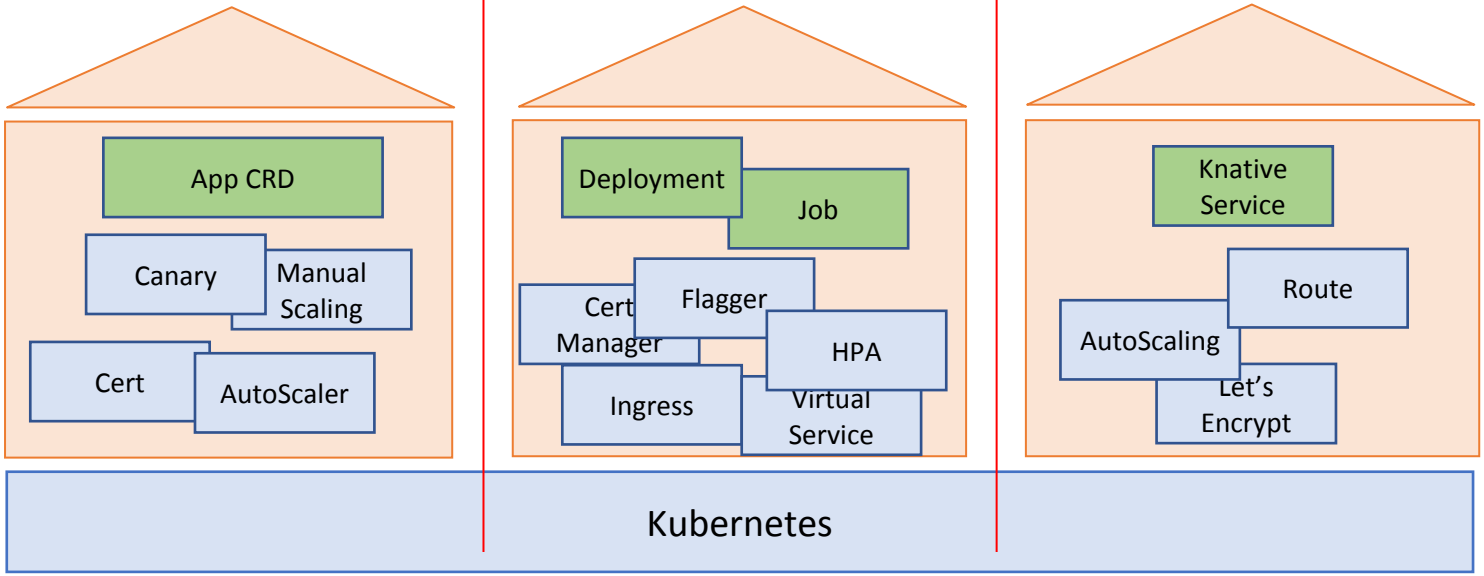
Building app platforms is hard ...

- **Fragmentation:** ~11 PaaS/Serverless in Alibaba
- **Silos:** no interoperability, reusability, or portability
- **Close:** many in-house wheels due to in-house app crd

I run stateful workloads!

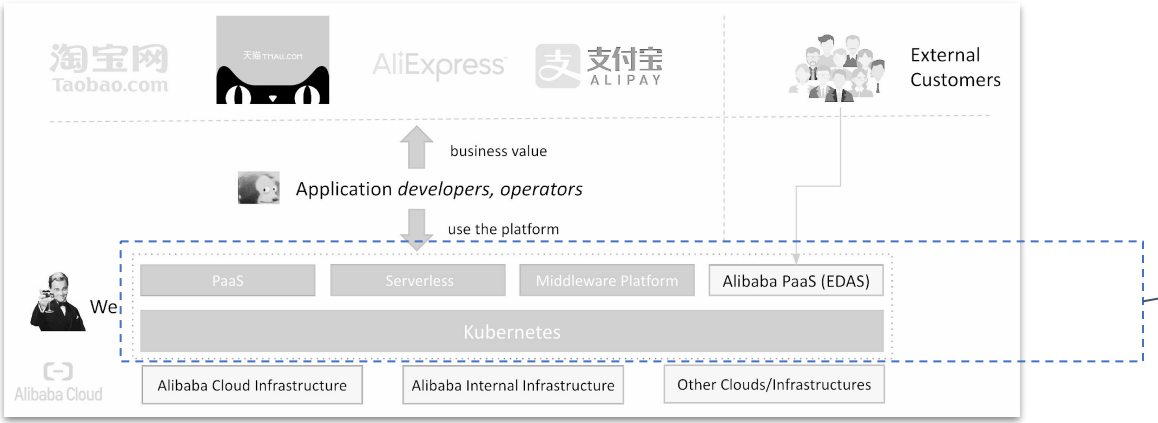
I run stateless apps!

I run stateless serverless containers!



Introducing KubeVela

- Come from real-world practices:
  - KubeVela's design originated from Alibaba's **unified application platform engine**
- Community owned project:
  - Initialized by **bootstrapping contributors** from 8+ different organizations since day 0
  - Incubated by OAM community and intend to be donated to neutral foundation at early stage



KubeVela

https://github.com/oam-dev/kubevela

The Goal of KubeVela

KubeVela aims at **both** *developers* and *platform builders*

developers

ship it!

Application

code → build → deploy → release → target k8s cluster

- Dockerfile
- Buildpack
- …

- workload
- storage
- ingress
- autoscaling
- …

- traffic shifting
- domain
- canary
- blue-green
- …

capabilities

platform builders

- ***Developers***:
  - here's my code, please ship it!
- ***Platform Builders***:
  - how can I assemble capabilities of k8s into a developer friendly platform at ease?

Design Principles of KubeVela

- **Application-Centric**
  - We believe “application” should be the main (maybe the only?) API our platform exposes to users.
  - e.g. KubeVela adopts *Open Application Model (OAM)* as the app-centric API and introduces *Appfile* as the last mile developer tool
- **Capability Oriented Architecture (COA)**
  - **Every** feature in KubeVela is a independent plugin (either a k8s built-in resource or your own CRD controller).
  - e.g. Alibaba use KubeVela adopts Flagger as *rollout* trait, KEDA as *autoscaling* trait
- **Highly extensible, even for its user interface**
  - When a new capability is installed, it should immediately consumable by end users without re-compiling or re-installing KubeVela.
    - e.g. KubeVela’s *Appfile*

Exercise 1

Ship the first cloud native application with KubeVela

1. vela system update
2. vela workloads
3. vela traits
4. vela comp deploy mycomp -t webservice --image crccheck/hello-world --port 8000 --app myapp
5. vela app status

Appfile


```yaml
services:
  express-server:
    build:
      image: oamdev/testapp:v1
      docker:
        file: Dockerfile
        context: .

    cmd: ["node", "server.js"]
    route:
      domain: example.com
      http: # match the longest prefix
        "/": 8080

    env:
      - FOO=bar
      - FOO2=sec:my-secret # map the key sam
      - FOO3=sec:my-secret:key # map specifi
      - sec:my-secret # map all KV pairs fro

    files: # Mount secret as a file
      - /mnt/path=sec:my-secret

    scale:
      replica: 2
      auto: # automatic scale up and down ba
        range: "1-10"
        cpu: 80 # if cpu utilization is abov
        qps: 1000 # if qps is higher than 1k

    canary: # Auto-create canary deployment.
      replica: 1 # canary deployment size
      headers:
        - "foo:bar.*"
```

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: WorkloadDefinition
metadata:
  name: webservice
spec:
  definitionRef:
    name: deployments.apps
  extension:
    template: |
      parameter: #webservice
      #webservice: {
        // +vela:cli:enabled=true
        // +vela:cli:usage=specify commands to run in container
        // +vela:cli:short=c
        cmd: [...string]

        env: [...string]

        files: [...string]
      }

      output: {
        apiVersion: "apps/v1"
        kind: "Deployment"
        metadata:
          name: context.name
        spec: {
          selector: {
            matchLabels:
              app: context.name
          }
          template: {
            metadata:
              labels:
                app: context.name
            spec: {
              containers: [{
                name:  context.name
                image: context.image
                command: parameter.cmd
              }]
            }
          }
      }
```

Capability Definition 1

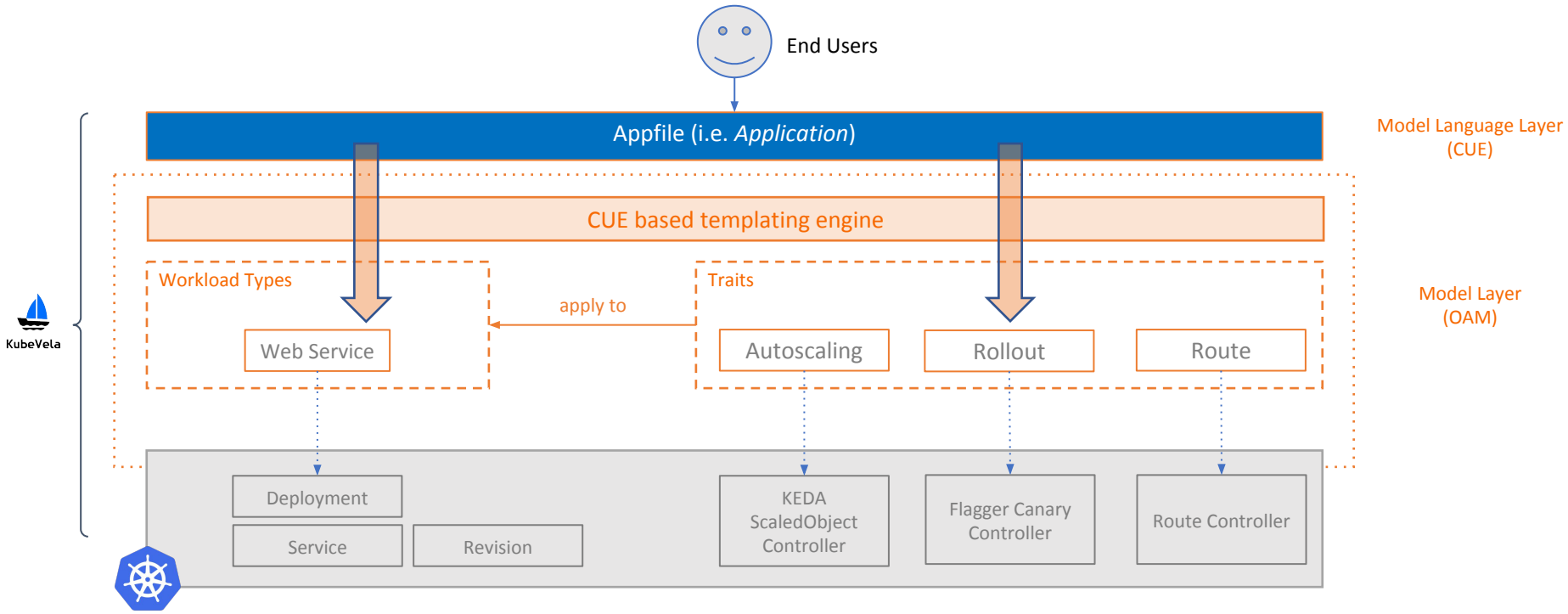
```yaml
apiVersion: core.oam.dev/v1alpha2
kind: TraitDefinition
metadata:
  name: route
spec:
  definitionRef:
    name: routes.standard.oam.dev
  extension:
    template: |
      parameter: #route
      #route: {
        domain: string
        http: [string]: int
      }

      // trait template can have multiple outputs and they are all traits
      outputs: service: {
        apiVersion: "v1"
        kind: "Service"
        metadata:
          name: context.name
        spec: {
          selector:
            app: context.name
          ports: [
            for k, v in parameter.http {
              port: v
              targetPort: v
            },
          ]
        }
```

Capability Definition 2

- **Simple**
  - Think about docker-compose but for Kubernetes.
  - Designed to ship (build -> release) cloud native app by one click.
- **Extensible**
  - Every section in Appfile references a independent capability definition
- **CUE based**
  - The schema of each section is enforced by CUE template defines in capability definition.

How Does Appfile Work?



Extend KubeVela!

Any capability in the open source community, could be shipped as a ***feature*** of KubeVela by a simple *$ kubectl apply -f definition.yaml*, and become usable immediately to users through *Appfile* and other UI tools

- This is also how all built-in features in KubeVela are shipped btw.

Let's say:

- add a new workload type of
  - AWS RDS?
  - stateful workload?
  - Redis Operator?
- add a new capability such as:
  - app metrics?
  - blue-green rollout?
  - custom domain?
  - ... enable Dapr?

$ kubectl apply -f

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: TraitDefinition
metadata:
  name: metric
  namespace: default
  annotations:
    definition.oam.dev/apiVersion: standard.oam.dev/v1alpha1
    definition.oam.dev/kind: MetricsTrait
    definition.oam.dev/description: "Add metric monitoring for workload"
spec:
  appliesToWorkloads:
    - webservice
    - backend
    - task
    - containerizedworkloads.core.oam.dev
    - clonesetworkloads.apps.kruise.io
    - deployments.apps
    - statefulsets.apps
  definitionRef:
    name: metricstraits.standard.oam.dev
  workloadRefPath: spec.workloadRef
  extension:
    template: |-
      #metrics: {
        // +usage=format of the metrics, default as prometheus
        // +short=f
        format:  *"prometheus" | string
        path:    *"/metrics" | string
```

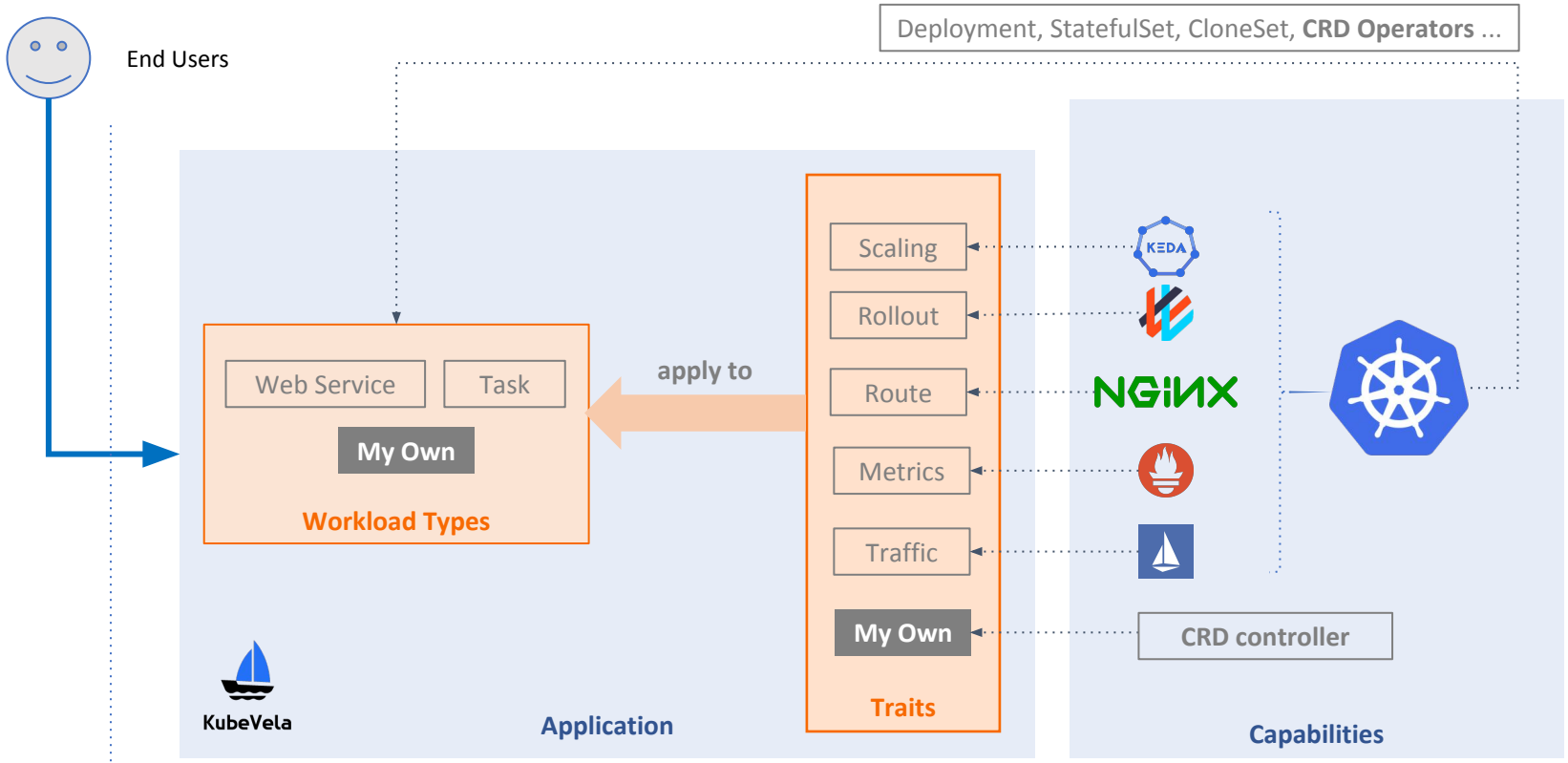
definition.yaml

Exercise 2

Add a new capability to KubeVela

1. Create a slack bot https://api.slack.com/apps?new_app=1
2. vela cap center config mycap https://github.com/oam-dev/catalog/tree/master/registry
3. vela cap ls
4. vela cap add mycap/kubewatch
5. vela comp deploy mycomp -t webservice --image crccheck/hello-world --port 8000 --app myapp
6. vela kubewatch mycomp --app myapp --webhook https://hooks.slack.com/<*yourid*>

KubeVela in Nutshell

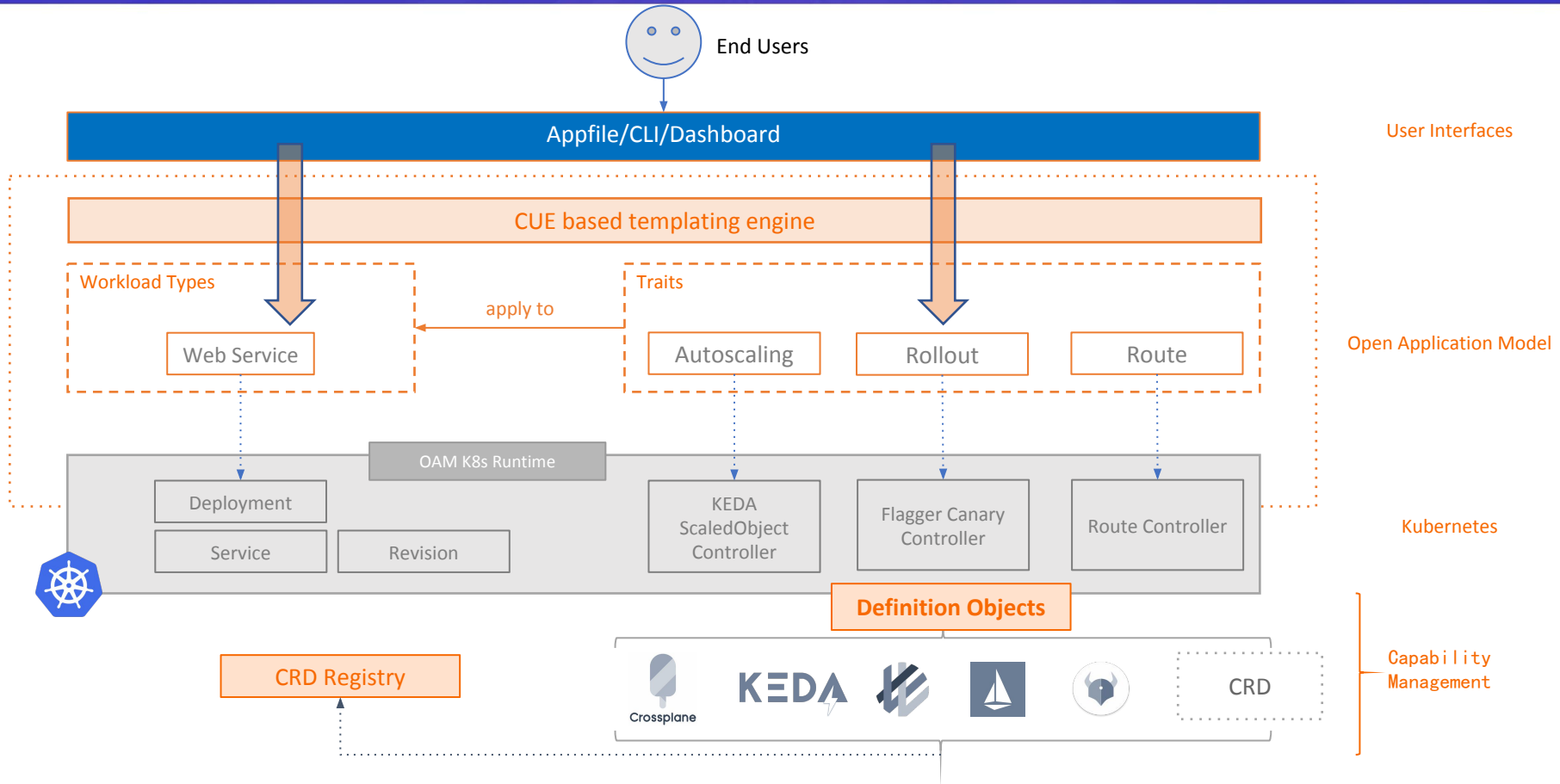


Extensible UI

"Application" as the main API

Capability discovery and management (COA)

Overall Architecture



What is Crossplane?

- Handles the **infrastructure** for your applications
- Based on Kubernetes control plane (CRDs and controllers)
- CNCF Sandbox project in June 2020, from the creators of Rook (CNCF graduated)
- 3 main feature areas
  - Provision infrastructure declaratively using the Kubernetes API
  - Build your own declarative infrastructure API without code
  - Run and deploy applications alongside infrastructure

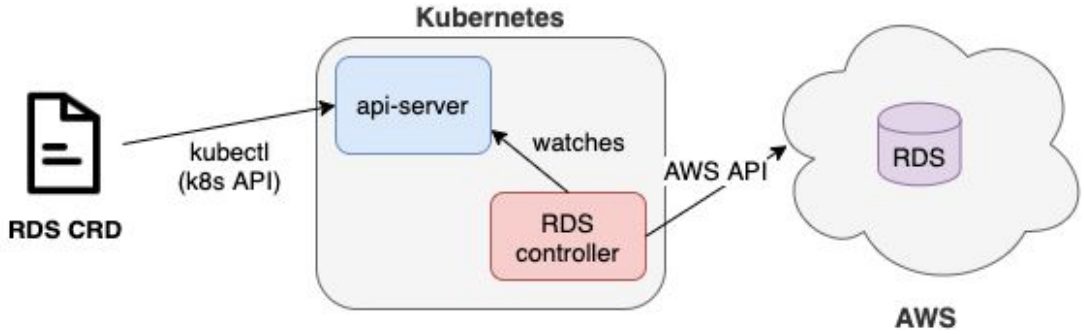
Exercise 3

1. Prepare cloud resources with Crossplane

   https://crossplane.io/docs/v0.13/getting-started/install-configure.htm
2. Configure the application to consume cloud resources
   a. Create RDS workload definition
   b. Deploy comps running with these definitions

1) Provision Infrastructure with K8s API

- Cloud and on-prem infrastructure is represented as a CRD
- Declaratively configured (e.g. YAML)
- Controllers reconcile infrastructure CRDs with cloud provider or on-prem APIs (e.g., GCP, AWS, Azure, Alibaba, Packet)
- Provision infrastructure with `kubectl` or any tool that talks with the Kubernetes API



Ex: Cloud Primitives for AWS - https://doc.crds.dev/

Kubernetes Clusters

Databases

Networking

Certificates

Caches

Message Queues

....

crossplane/provider-aws

github.com/crossplane/provider-aws/tree/master

CRDs discovered: 39

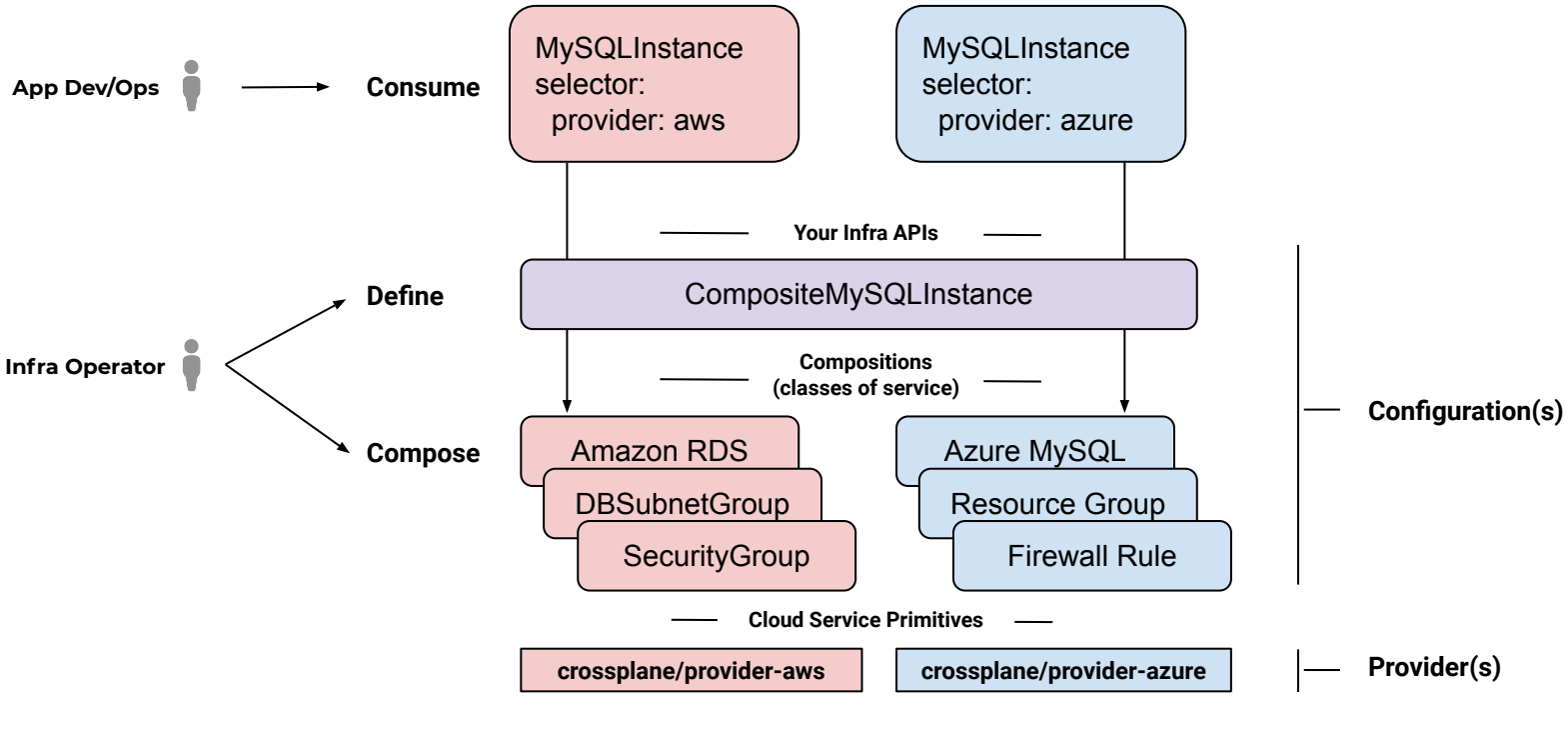
Last parsed: Sun, 25 Oct 2020 16:00:33 +0000

Kind	Group	Version
Certificate	acm.aws.crossplane.io	v1alpha1
CertificateAuthority	acmpca.aws.crossplane.io	v1alpha1
CertificateAuthorityPermission	acmpca.aws.crossplane.io	v1alpha1
Provider	aws.crossplane.io	v1alpha3
ProviderConfig	aws.crossplane.io	v1beta1
ProviderConfigUsage	aws.crossplane.io	v1beta1
CacheCluster	cache.aws.crossplane.io	v1alpha1
CacheSubnetGroup	cache.aws.crossplane.io	v1alpha1
ReplicationGroup	cache.aws.crossplane.io	v1beta1
DBSubnetGroup	database.aws.crossplane.io	v1beta1
DynamoTable	database.aws.crossplane.io	v1alpha1
RDSInstance	database.aws.crossplane.io	v1beta1

2) Offer declarative Infra APIs for teams to use

- Define, compose, and offer your own infrastructure API abstractions on top of the cloud service primitives included
- Example: **Define** a MySQL resource that is **composed** of Azure MySQL, resource group, and firewall rule. **Offer** it for applications to consume on demand.
- Hide infrastructure complexity and include policy guardrails
- **No code** required, it's all **declarative**

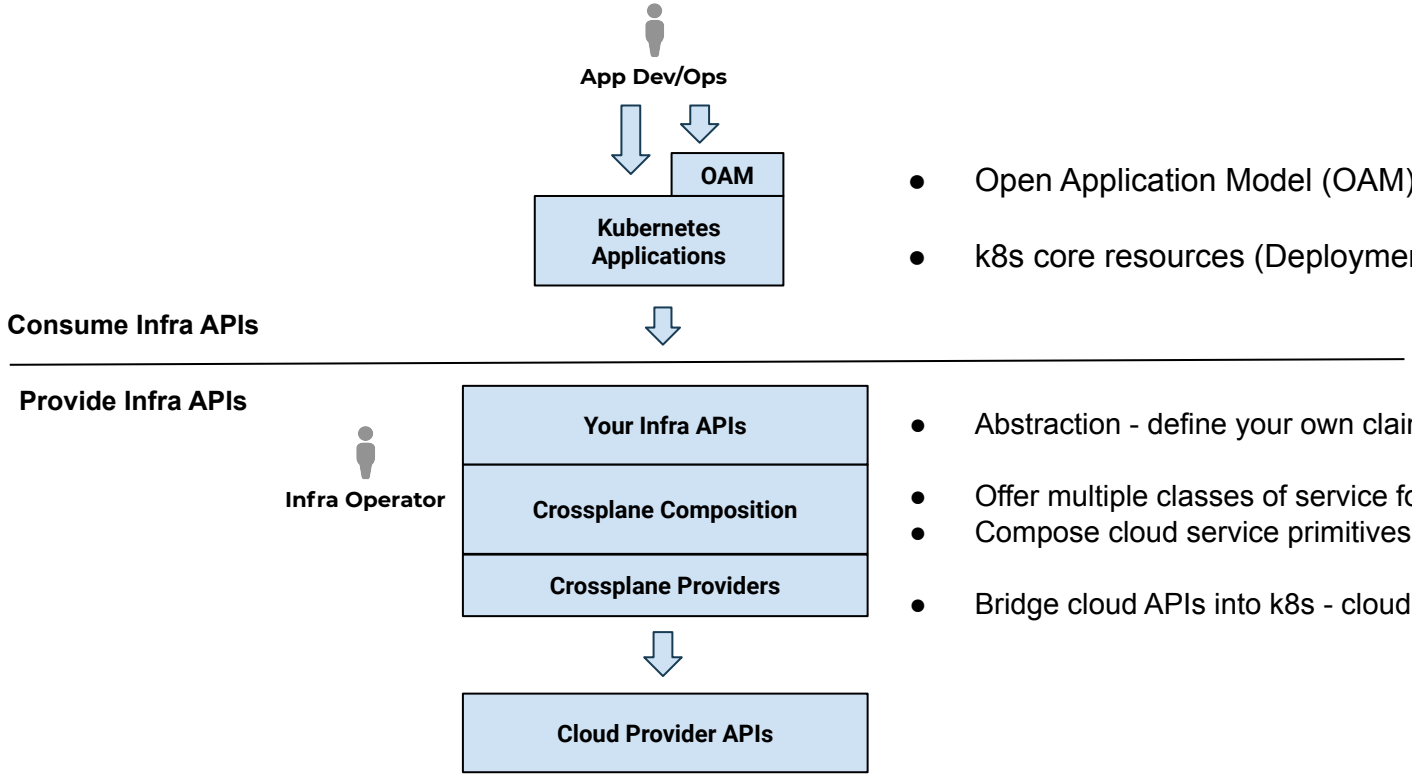
Composition - define & compose your own Infra APIs



3) Run and deploy applications

- Support for Open Application Model (OAM)
- Deploy applications alongside infrastructure
  - Standardize on a single workflow for both
- Strong “separation of concerns”
  - Infrastructure operators - provide infrastructure and services
  - Application developers - build application components independent of infrastructure
  - Application operators - compose, deploy, and run application configurations

Crossplane Arch Summary



- Open Application Model (OAM)
- k8s core resources (Deployments, Services, …)
- Abstraction - define your own claim kinds and schema
- Offer multiple classes of service for an Infra API
- Compose cloud service primitives
- Bridge cloud APIs into k8s - cloud service primitives

Exercise 4

Build and offer an infrastructure API to your team

1. **Define** your platform's composite resources and claims
   a. https://github.com/upbound/platform-ref-aws/
   b. Network, Cluster, PostgreSQL
2. **Push** this configuration to a registry
3. **Install** the configuration into a Crossplane instance
4. **Offer** the desired claims to your team(s)
5. Teams **provision** their infrastructure self-service, with the policy/config you declared

Community - OAM/KubeVela

- **OAM Specification**
  - Now [v0.2.2 working draft](), moving to beta release with full backward compatibility
- **KubeVela**
  - *Developer Preview* stage with features still WIP, **NOT** ready for production.
  - Roadmap: https://github.com/oam-dev/kubevela/projects/1
    - v1.0.0 release targets at Dec. 2020
  - Current feature set:
    - *Appfile*, *CLI*, *dashboard* (*preview*)
    - ***Web Service*** & ***Task*** workload types, ***Route***, ***Rollout*** (*Flagger*) & ***Autoscaling*** (*KEDA*)
- **Community**
  - Gitter: https://gitter.im/oam-dev/
  - Slack: https://cloud-native.slack.com/messages/kubevela/

Community - Crossplane

Get involved with this CNCF Sandbox project!

- **Website**: https://crossplane.io/
- **Slack**: https://slack.crossplane.io/
- **Docs**: https://crossplane.io/docs
- **GitHub**: https://github.com/crossplane/crossplane
- **Blog**: https://blog.crossplane.io/
- **Twitter**: https://twitter.com/crossplane_io
- **Youtube**: Crossplane Youtube

Roadmap - Crossplane

Working towards (and beyond) a v1.0 release for end of year!

- **Crossplane Core**
  - Composition - composition revisions, validation, custom compositions
  - Package Manager - dependency resolution, multiple versions
  - Hardening, metrics, APIs to v1beta1+
- **Providers for AWS, GCP, Azure, Alibaba**
  - Rapidly expanding cloud services to 90% coverage
  - AWS ACK & Azure ASO code-gen of Crossplane providers
  - Wrapping stateless Terraform providers for clouds w/o code-gen
- **Open Application Model (OAM)**
  - oam-kubernetes-runtime APIs to v1beta1 & subchart install
  - health scopes & additional trait support


KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual
HELM
KV