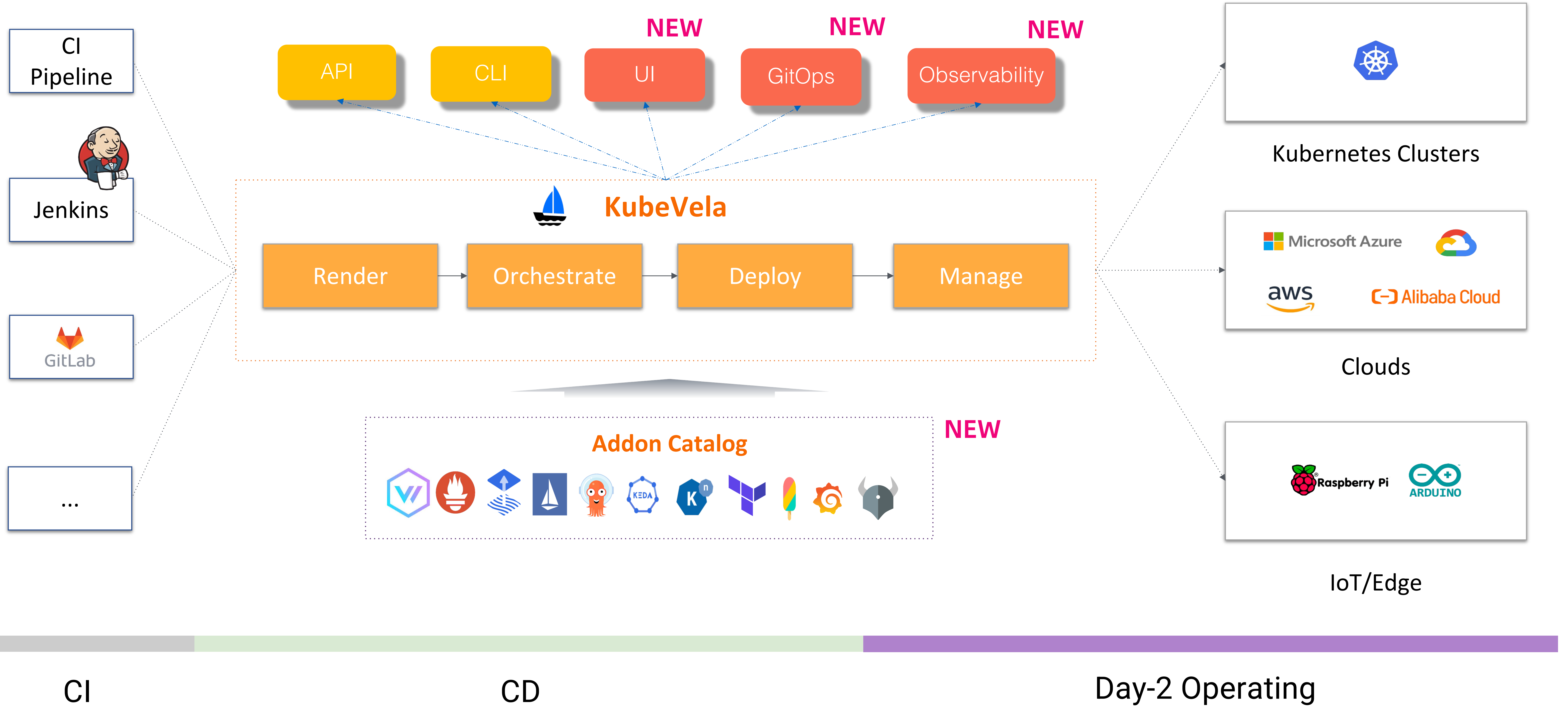
KubeVela: Make shipping applications more enjoyable

KubeVela team

What is KubeVela

KubeVela is a modern software platform that makes delivering and operating applications across today's hybrid, multi-cloud environments easier, faster and more reliable.

- ❑ Infrastructure agnostic
- ❑ Programmable
- ❑ Application-centric



1 Day 1 - Application Delivery

Application Model

Alibaba Cloud

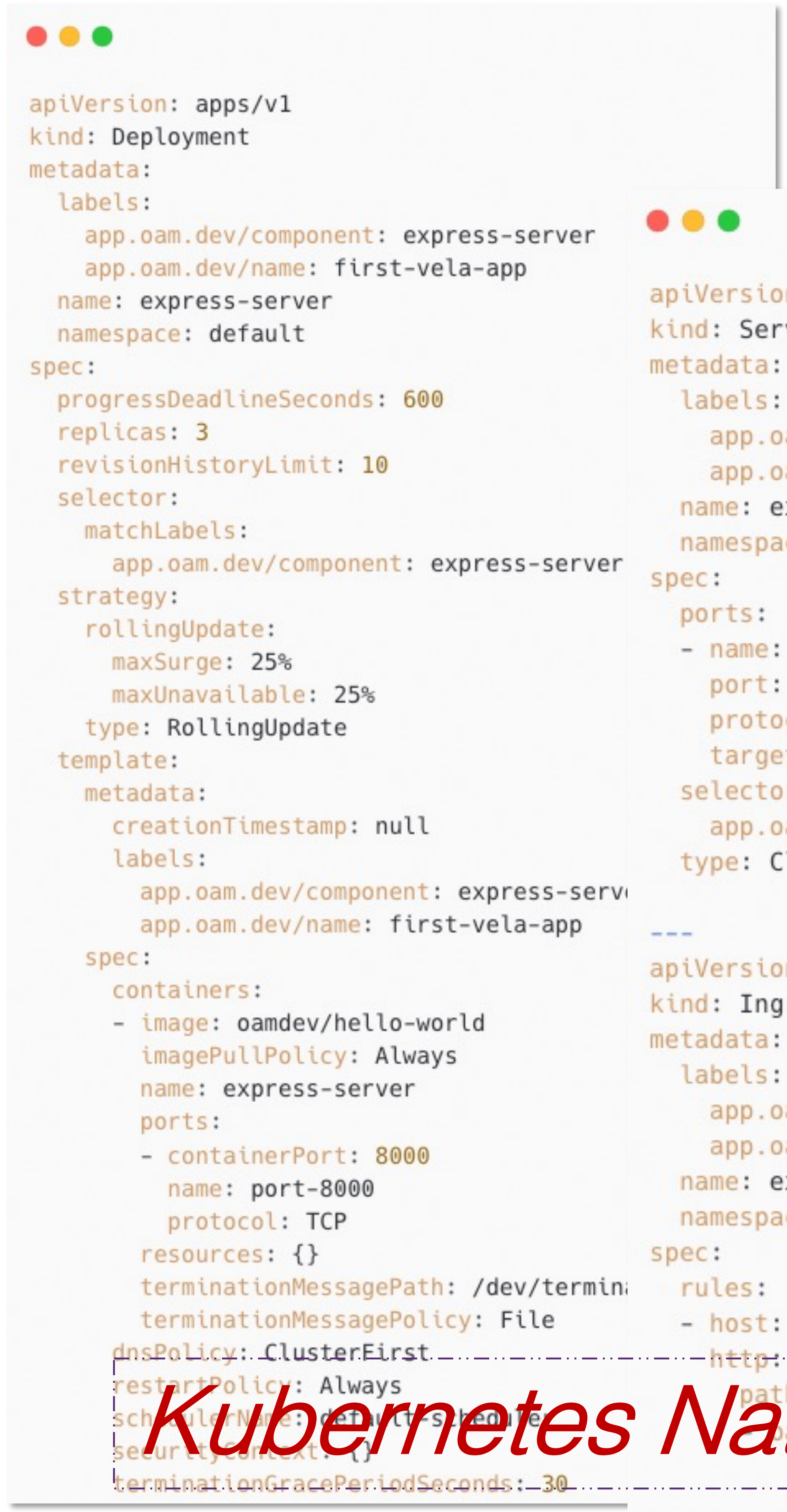
```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: podinfo
spec:
  components:
    - type: webservice
      name: podinfo
      properties:
        image: stefanprodan/podinfo
      traits:
        - type: scaler
          properties:
            replicas: 3
        - type: gateway
          properties:
            class: traefik
            http:
              /: 9898
```

Application
The higher level abstraction to model a full functional microservice unit.

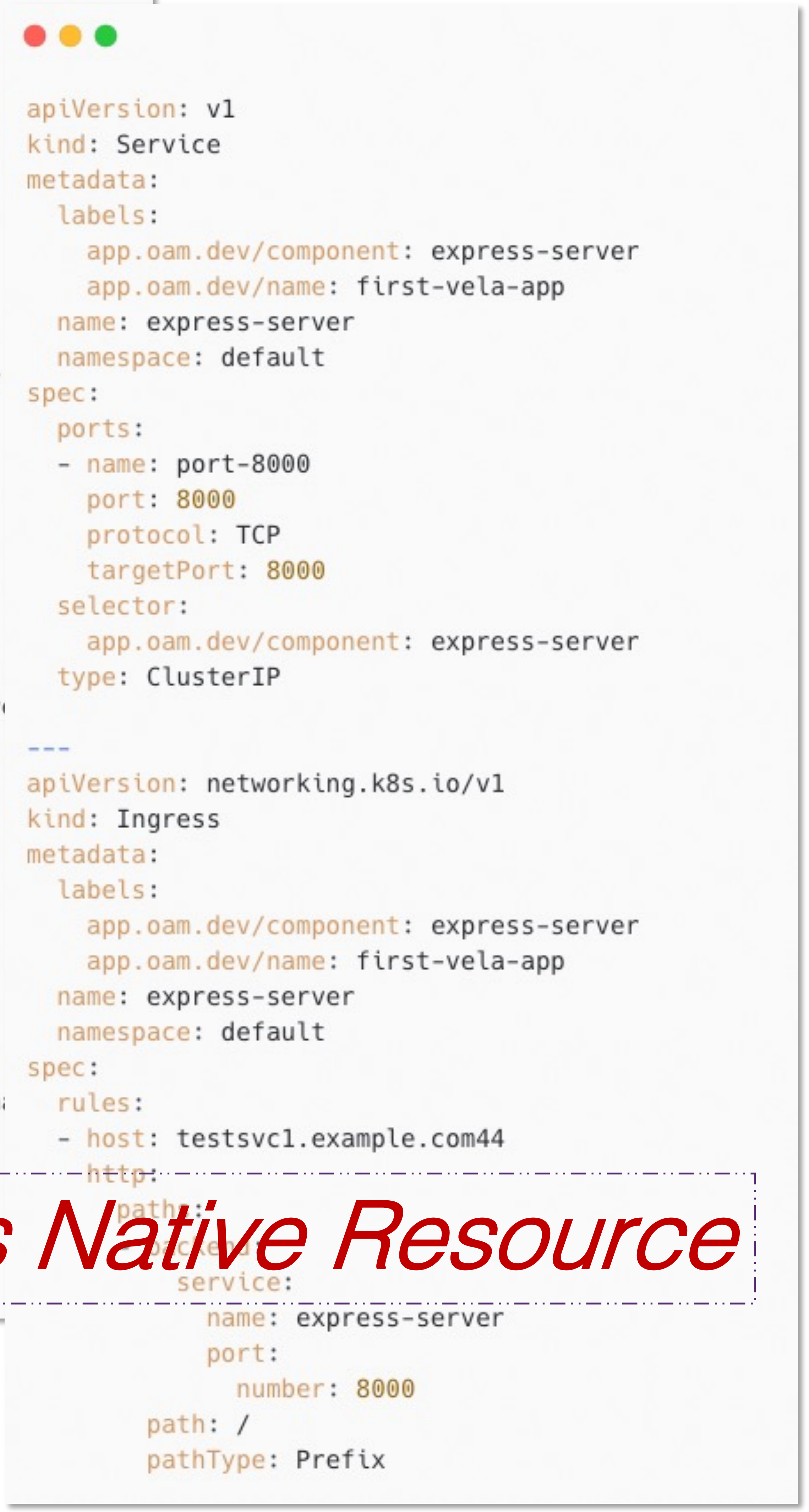
Component
The main workload to run such as web services, jobs, databases.

Trait
Operational auxiliaries that help the component to work, like scaling, storage, gateway.

VS



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.oam.dev/component: express-server
    app.oam.dev/name: first-vela-app
  name: express-server
  namespace: default
spec:
  progressDeadlineSeconds: 600
  replicas: 3
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      app.oam.dev/component: express-server
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app.oam.dev/component: express-serv
        app.oam.dev/name: first-vela-app
    spec:
      containers:
      - image: oamdev/hello-world
        imagePullPolicy: Always
        name: express-server
        ports:
        - containerPort: 8000
          name: port-8000
          protocol: TCP
        resources: {}
        terminationMessagePath: /dev/termina
        terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
```



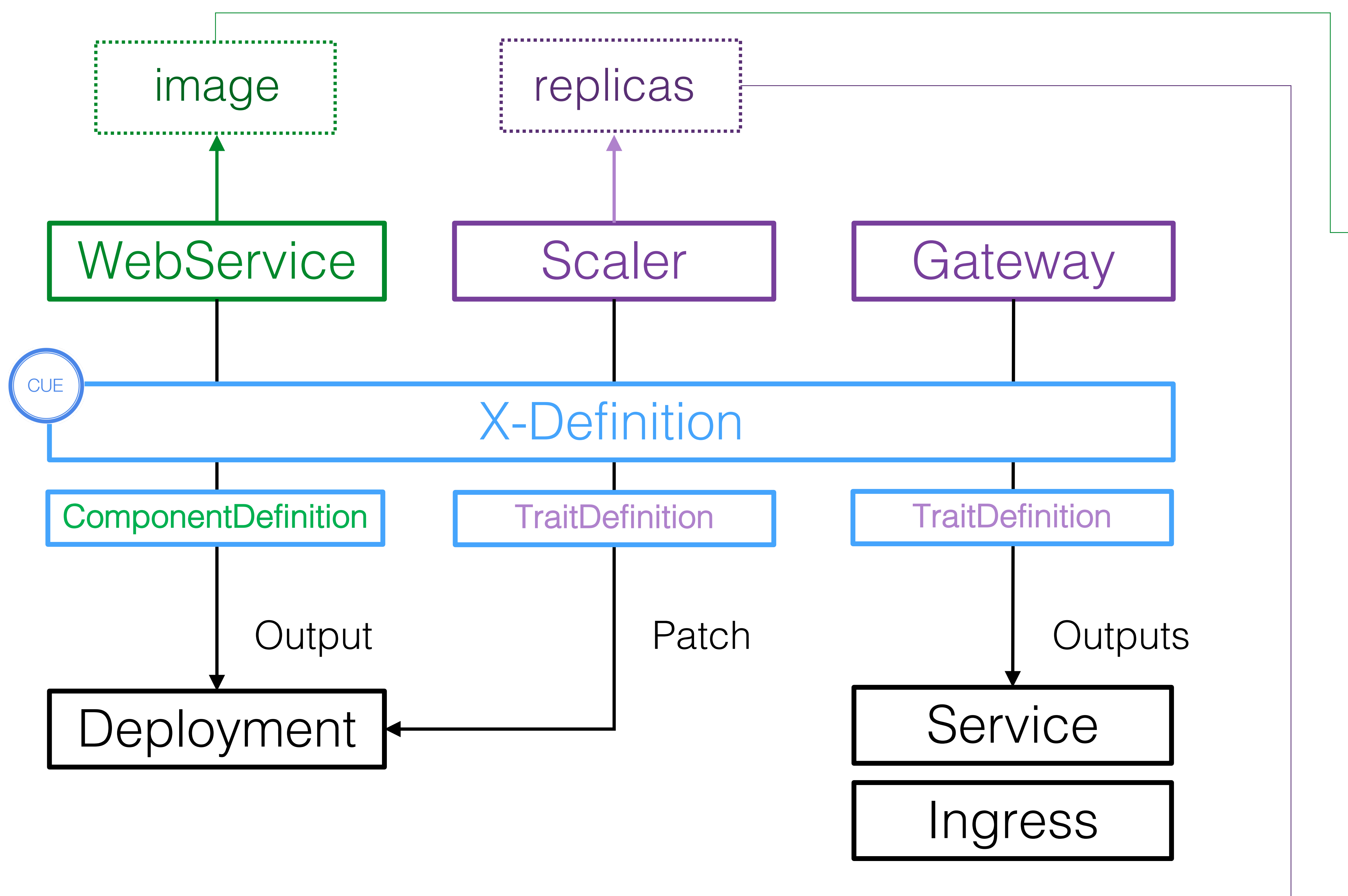
```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app.oam.dev/component: express-server
    app.oam.dev/name: first-vela-app
  name: express-server
  namespace: default
spec:
  ports:
  - name: port-8000
    port: 8000
    protocol: TCP
    targetPort: 8000
  selector:
    app.oam.dev/component: express-server
  type: ClusterIP

---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  labels:
    app.oam.dev/component: express-server
    app.oam.dev/name: first-vela-app
  name: express-server
  namespace: default
spec:
  rules:
  - host: testsvc1.example.com44
    http:
      paths:
      - backend:
          service:
            name: express-server
            port:
              number: 8000
        path: /
        pathType: Prefix
```

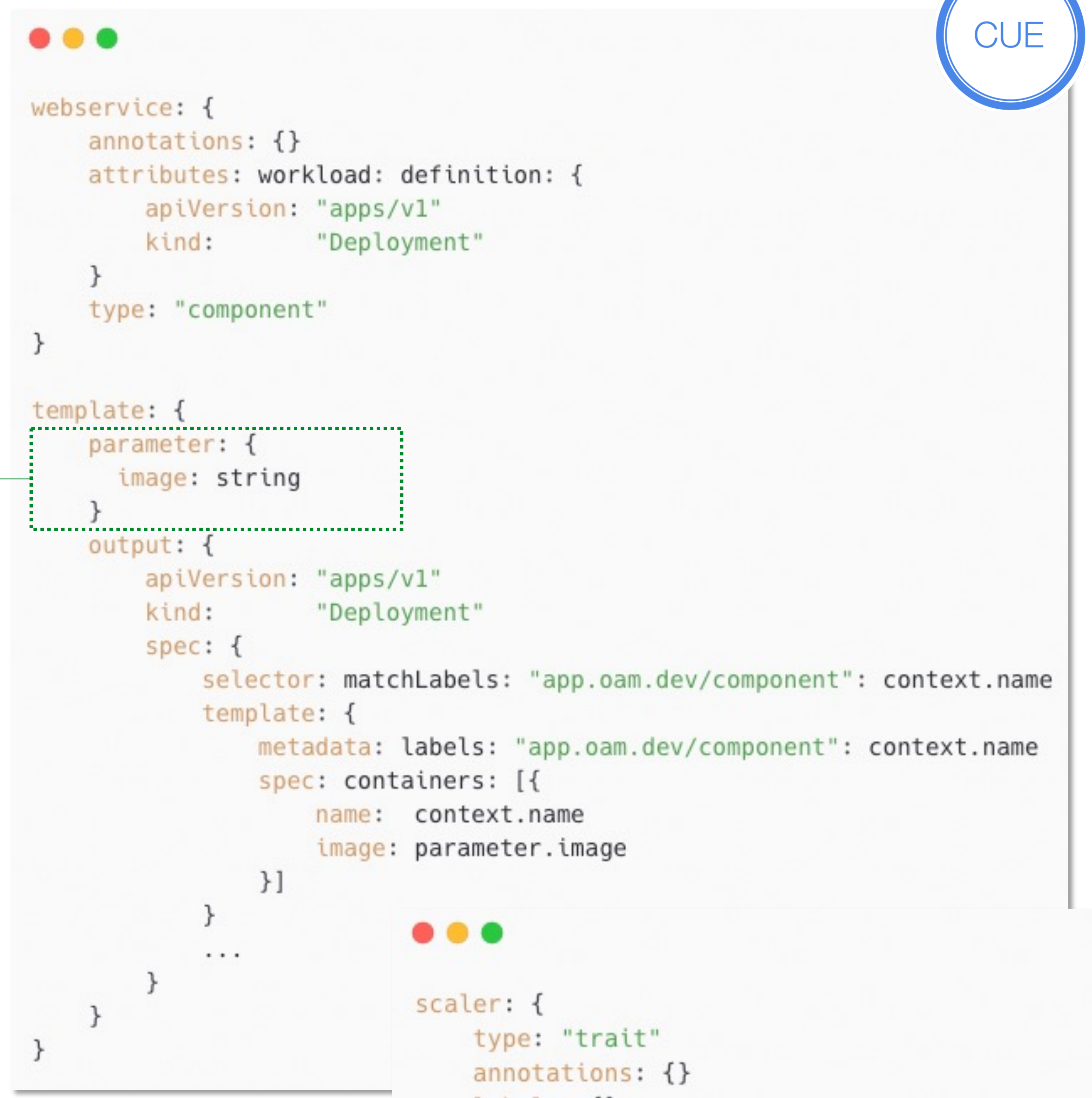
Kubernetes Native Resource

❑ **Consistent Application Model** for Application Delivery.

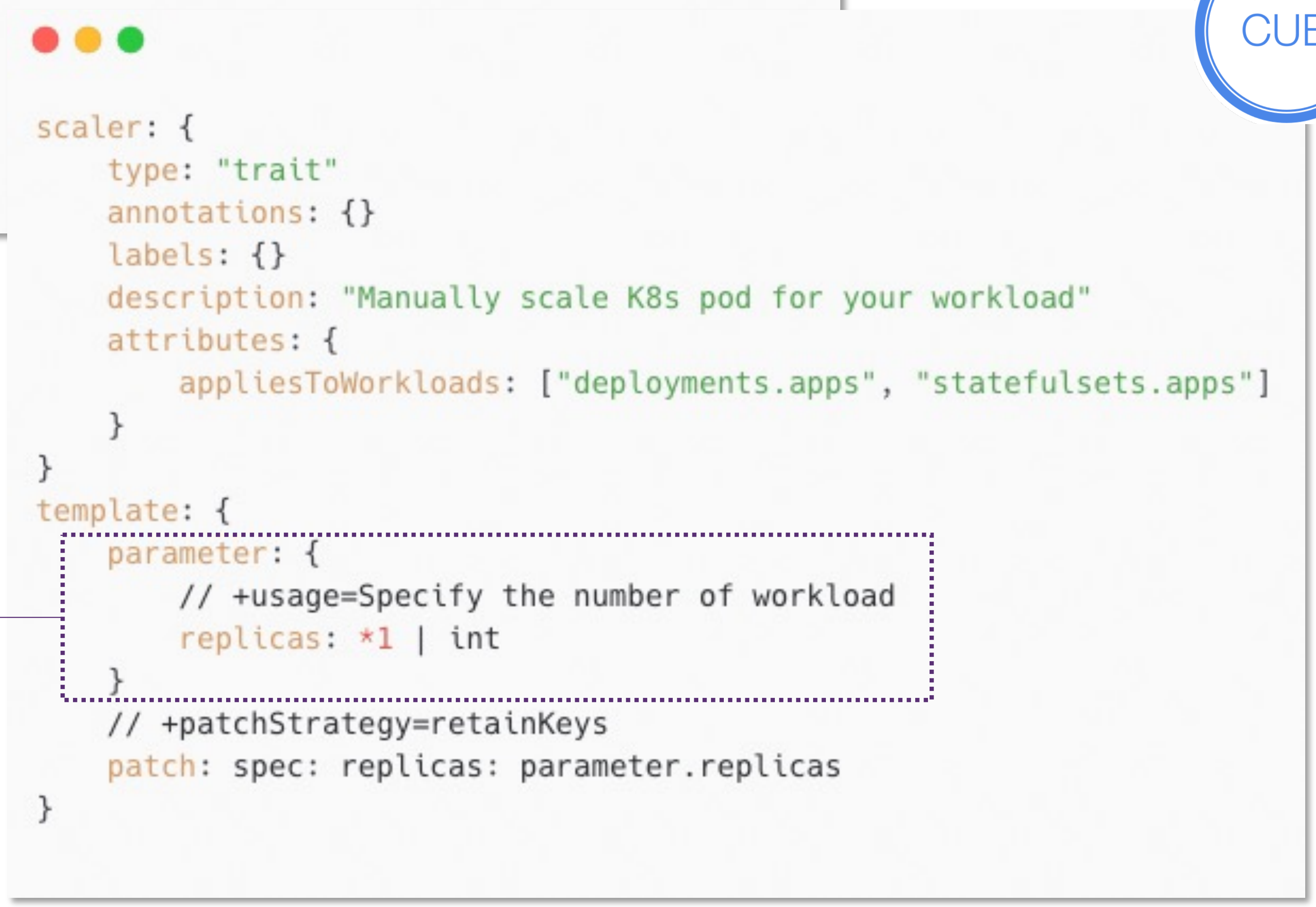
Alibaba Cloud

Application Model





```
webservice: {
    annotations: {}
    attributes: workload: definition: {
        apiVersion: "apps/v1"
        kind:       "Deployment"
    }
    type: "component"
}

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

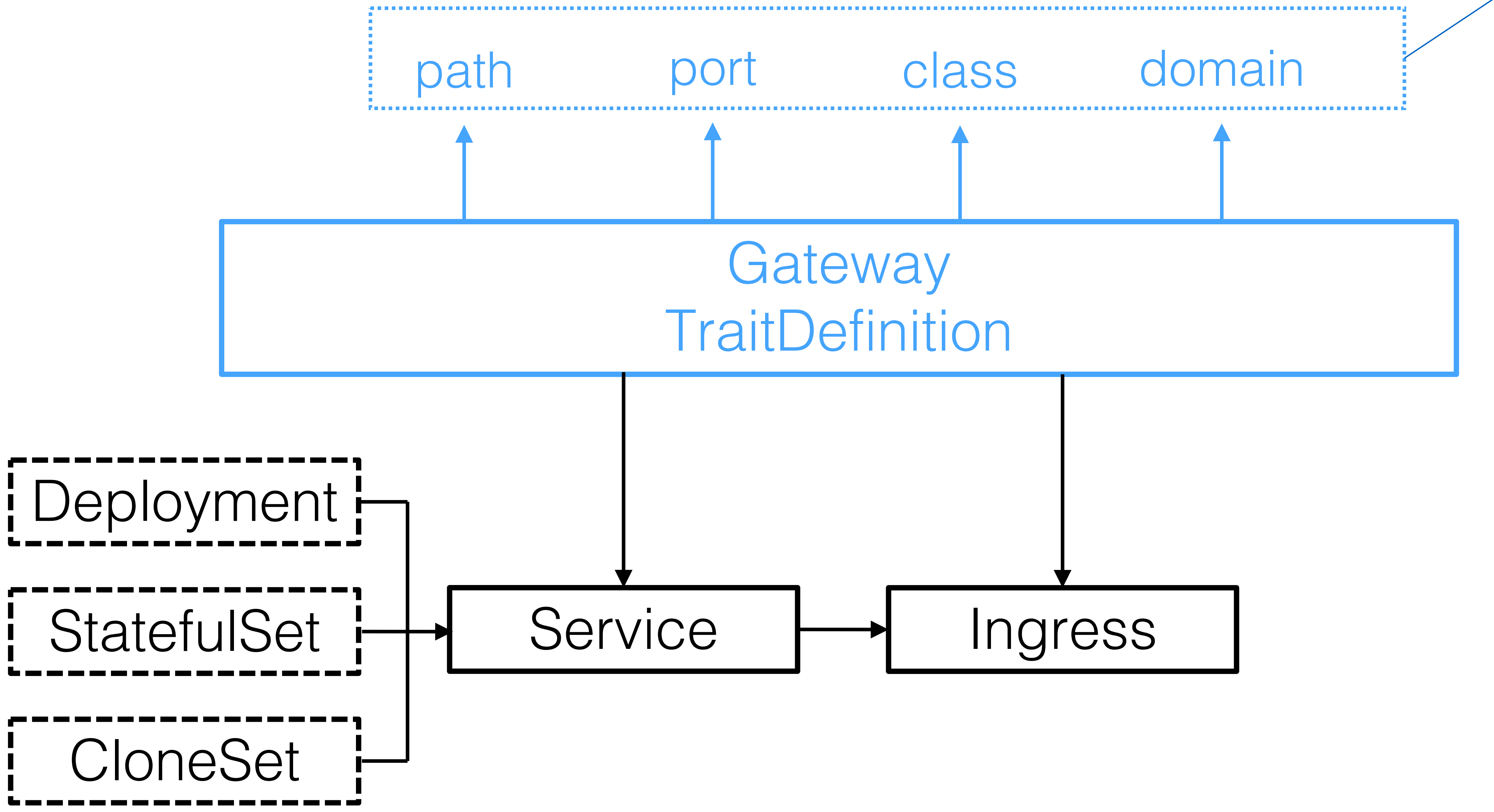


```
scaler: {
    type: "trait"
    annotations: {}
    labels: {}
    description: "Manually scale K8s pod for your workload"
    attributes: {
        appliesToWorkloads: ["deployments.apps", "statefulsets.apps"]
    }
}
template: {
    parameter: {
        // +usage=Specify the number of workload
        replicas: *1 | int
    }
    // +patchStrategy=retainKeys
    patch: spec: replicas: parameter.replicas
}
```

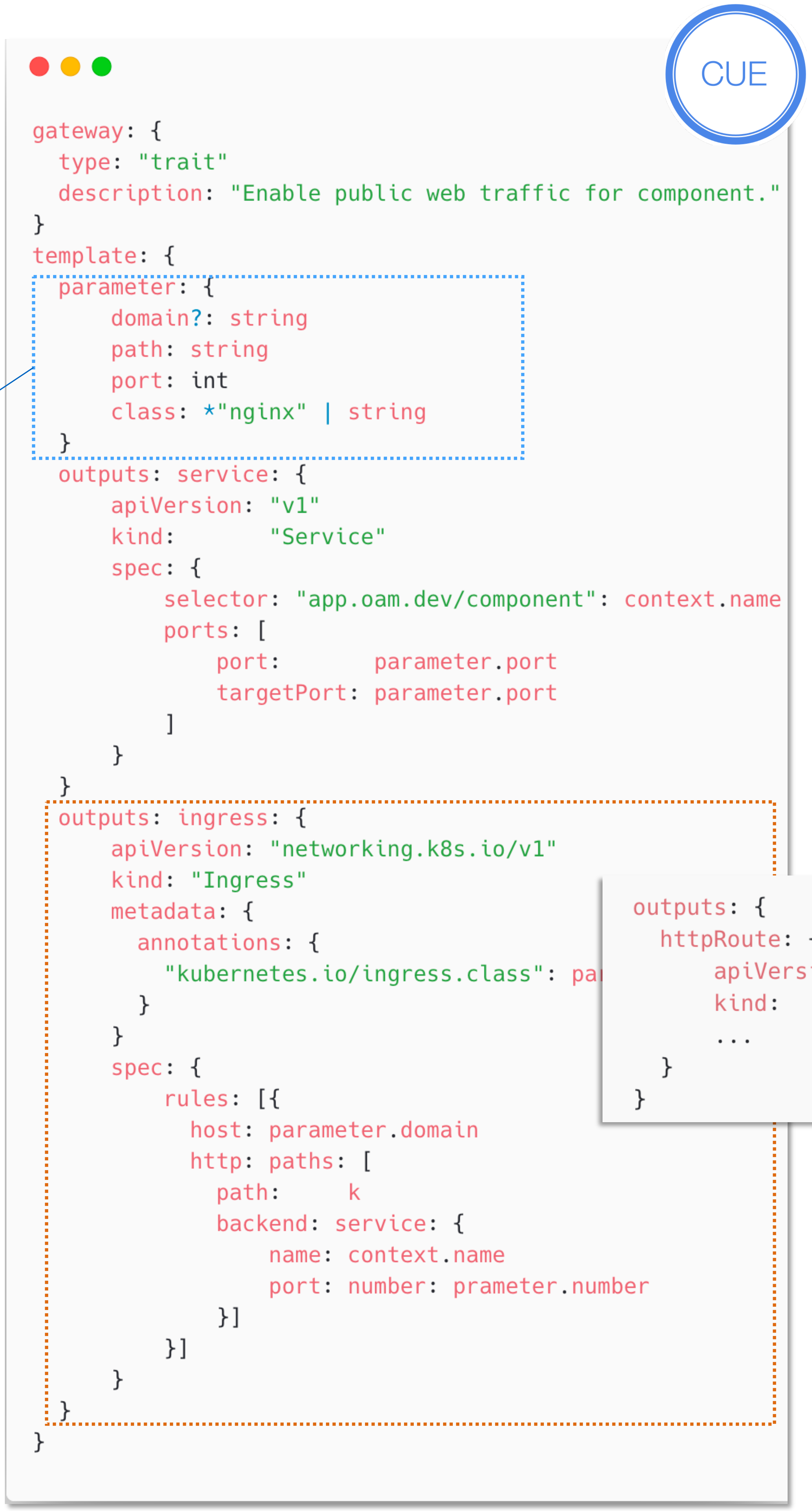
❑ **Programmable Extensibility** with **CUE** Configuration.

Application Model

- ❑ **Abstraction from Best Practices**

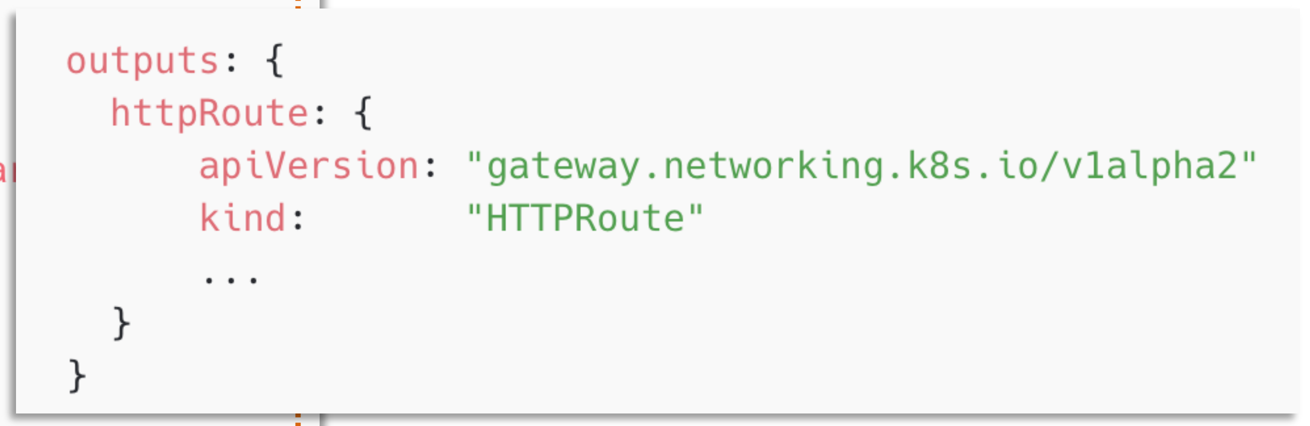


- ❑ **Applies to different workloads.**



```
gateway: {
  type: "trait"
  description: "Enable public web traffic for component."
}
template: {
  parameter: {
      domain?: string
      path: string
      port: int
      class: *"nginx" | string
  }
  outputs: service: {
      apiVersion: "v1"
      kind:       "Service"
      spec: {
          selector: "app.oam.dev/component": context.name
          ports: [
              port:       parameter.port
              targetPort: parameter.port
          ]
      }
  }
  outputs: ingress: {
      apiVersion: "networking.k8s.io/v1"
      kind: "Ingress"
      metadata: {
        annotations: {
          "kubernetes.io/ingress.class": pa
        }
      }
      spec: {
          rules: [{
            host: parameter.domain
            http: paths: [
              path:     k
              backend: service: {
                  name: context.name
                  port: number: prameter.number
              }]
          }]
      }
  }
}
```

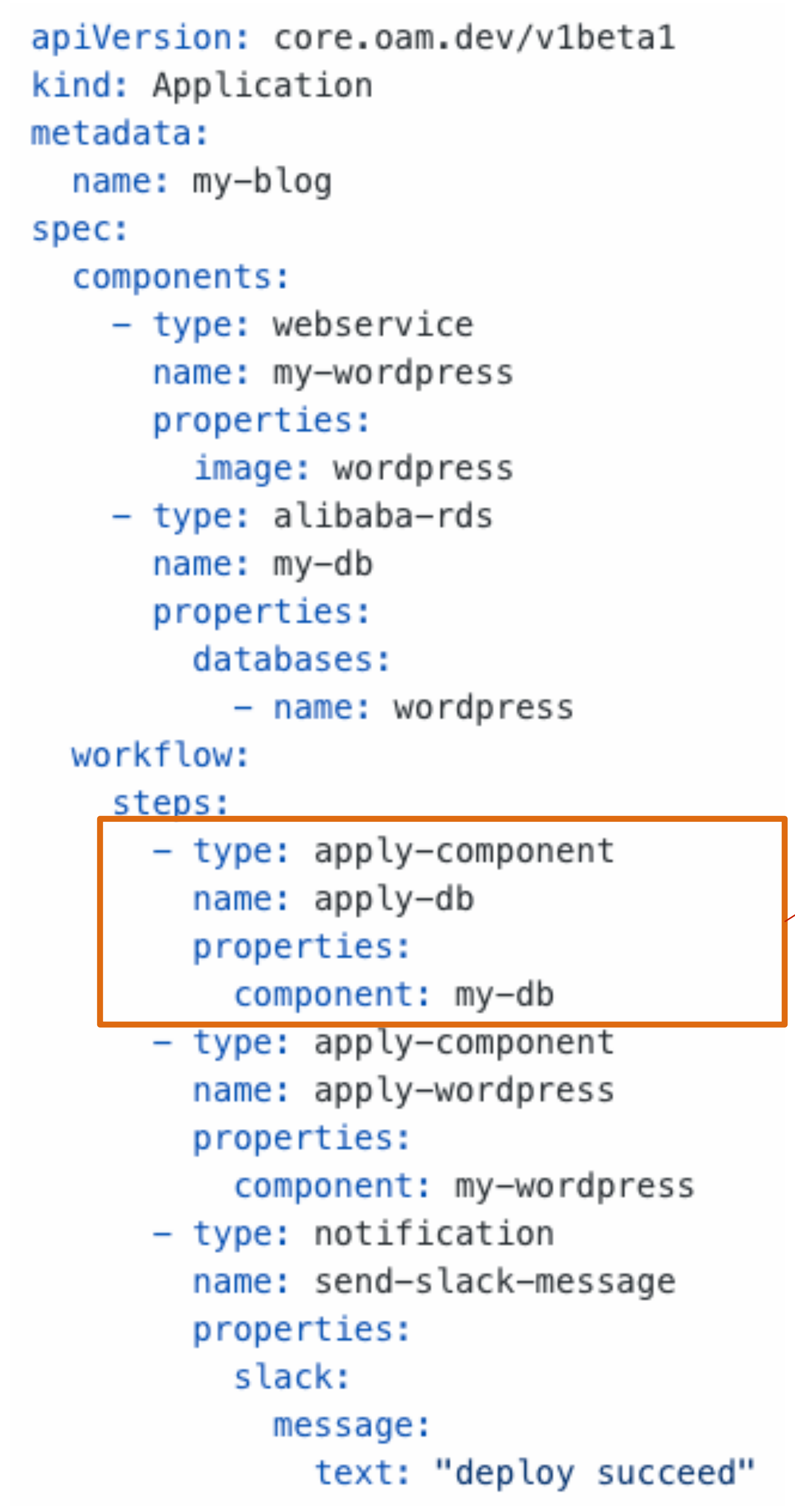
- ❑ **Implementation agnostic.**



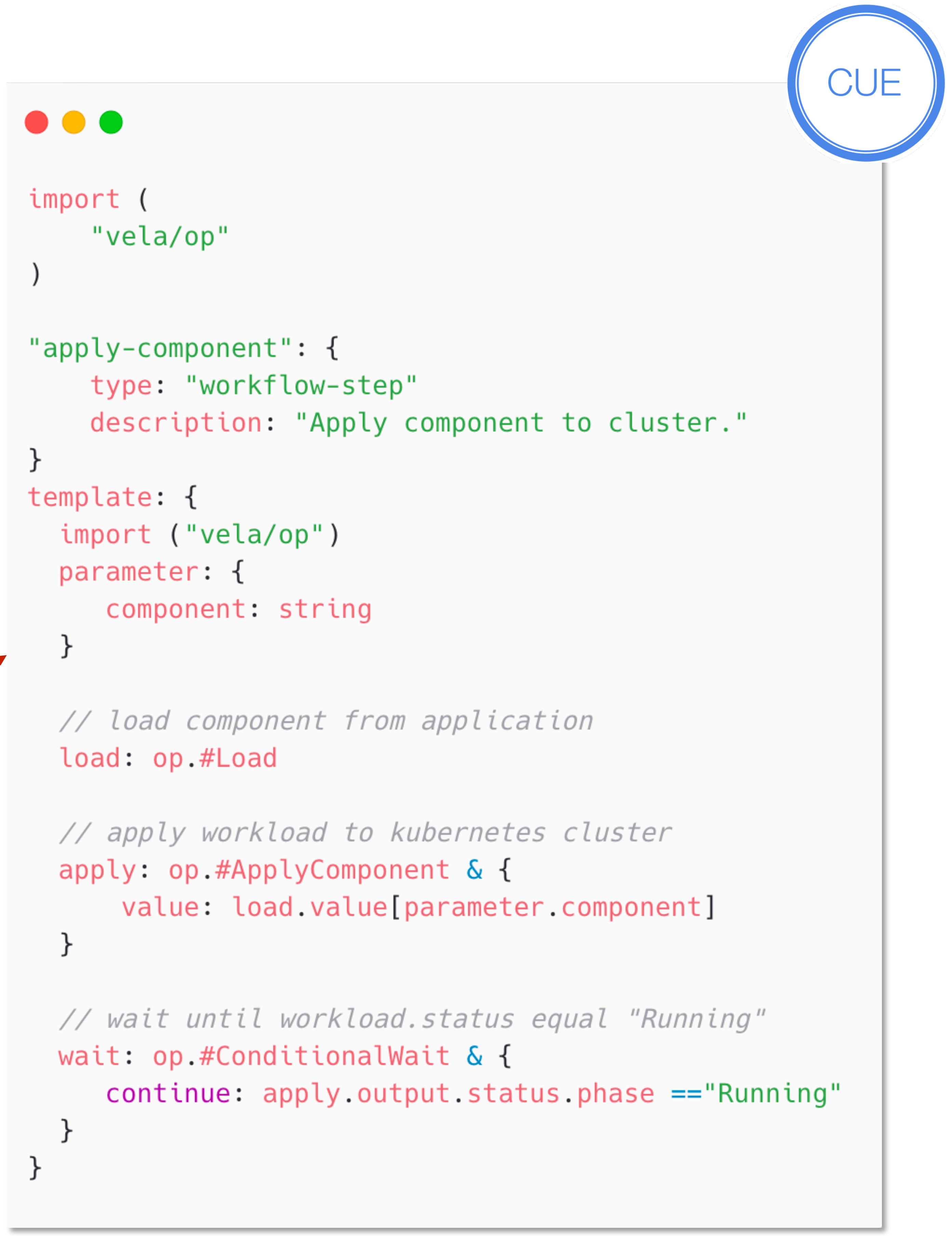
```
outputs: {
  httpRoute: {
      apiVersion: "gateway.networking.k8s.io/v1alpha2"
      kind:       "HTTPRoute"
      ...
  }
}
```

Workflow: orchestrate and glue **ANY** delivery actions

Alibaba Cloud



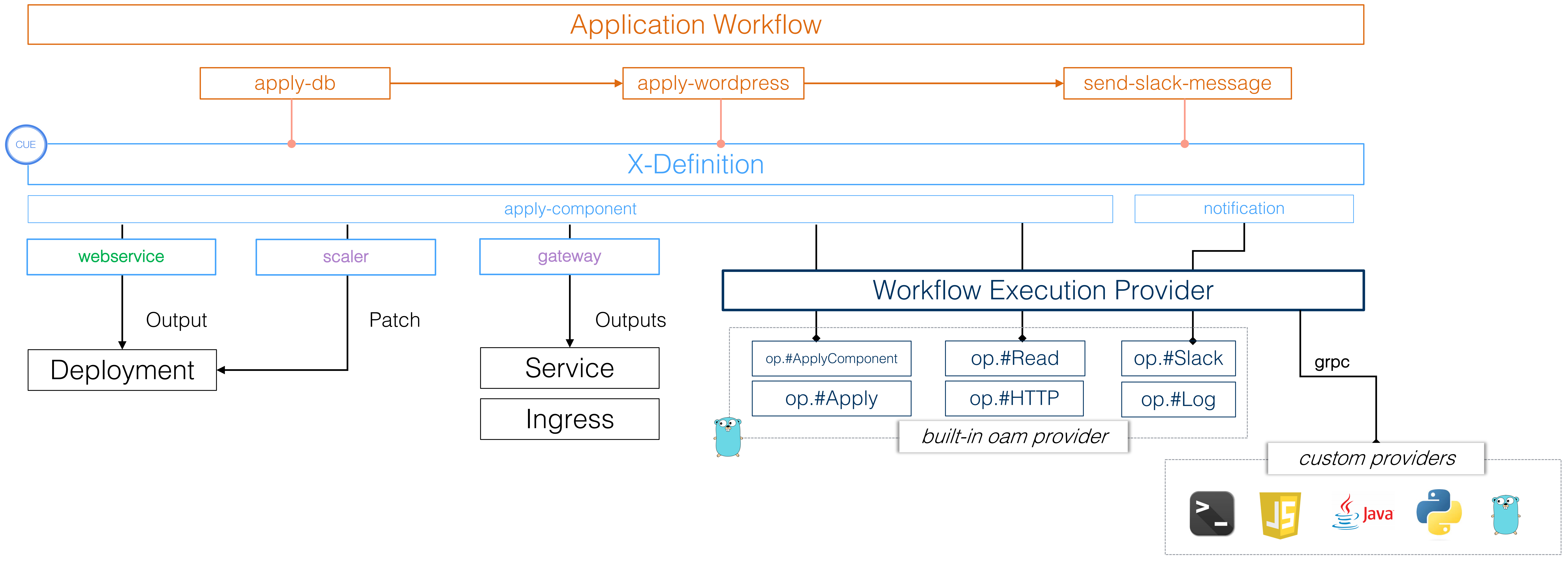
```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: my-blog
spec:
  components:
    - type: webservice
      name: my-wordpress
      properties:
        image: wordpress
    - type: alibaba-rds
      name: my-db
      properties:
        databases:
          - name: wordpress
  workflow:
    steps:
      - type: apply-component
        name: apply-db
        properties:
          component: my-db
      - type: apply-component
        name: apply-wordpress
        properties:
          component: my-wordpress
      - type: notification
        name: send-slack-message
        properties:
          slack:
            message:
              text: "deploy succeed"
```



```
import (
    "vela/op"
)

"apply-component": {
    type: "workflow-step"
    description: "Apply component to cluster."
}
template: {
  import ("vela/op")
  parameter: {
     component: string
  }

  // load component from application
  load: op.#Load

  // apply workload to kubernetes cluster
  apply: op.#ApplyComponent & {
      value: load.value[parameter.component]
  }

  // wait until workload.status equal "Running"
  wait: op.#ConditionalWait & {
     continue: apply.output.status.phase =="Running"
  }
}
```

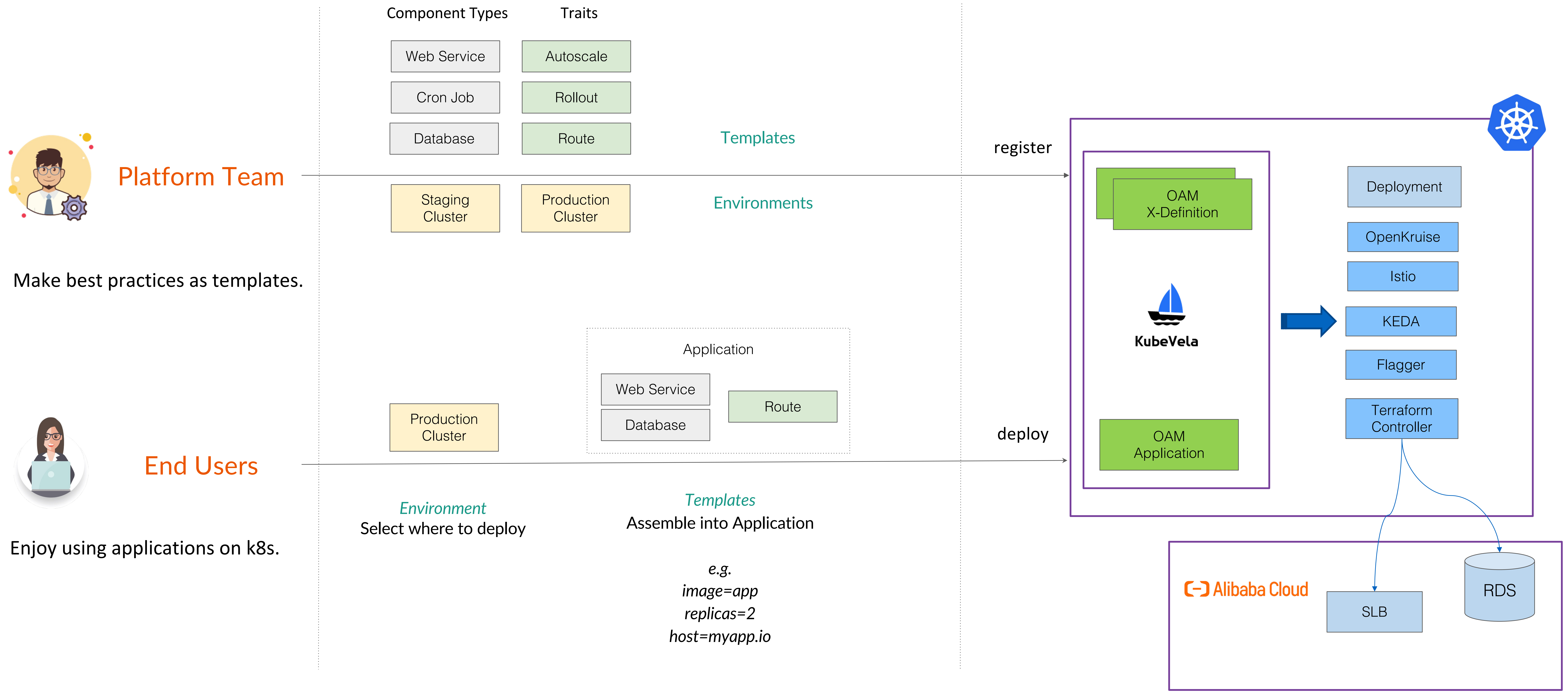
- ❑ Flexible, Extensible and Programmable.
- ❑ Rich built-in capabilities.
- ❑ Lightweight execution engine.
- ❑ Safe execution with schema level validation.

KubeVela Application Delivery, a **consistent, programmable, declarative** workflow!

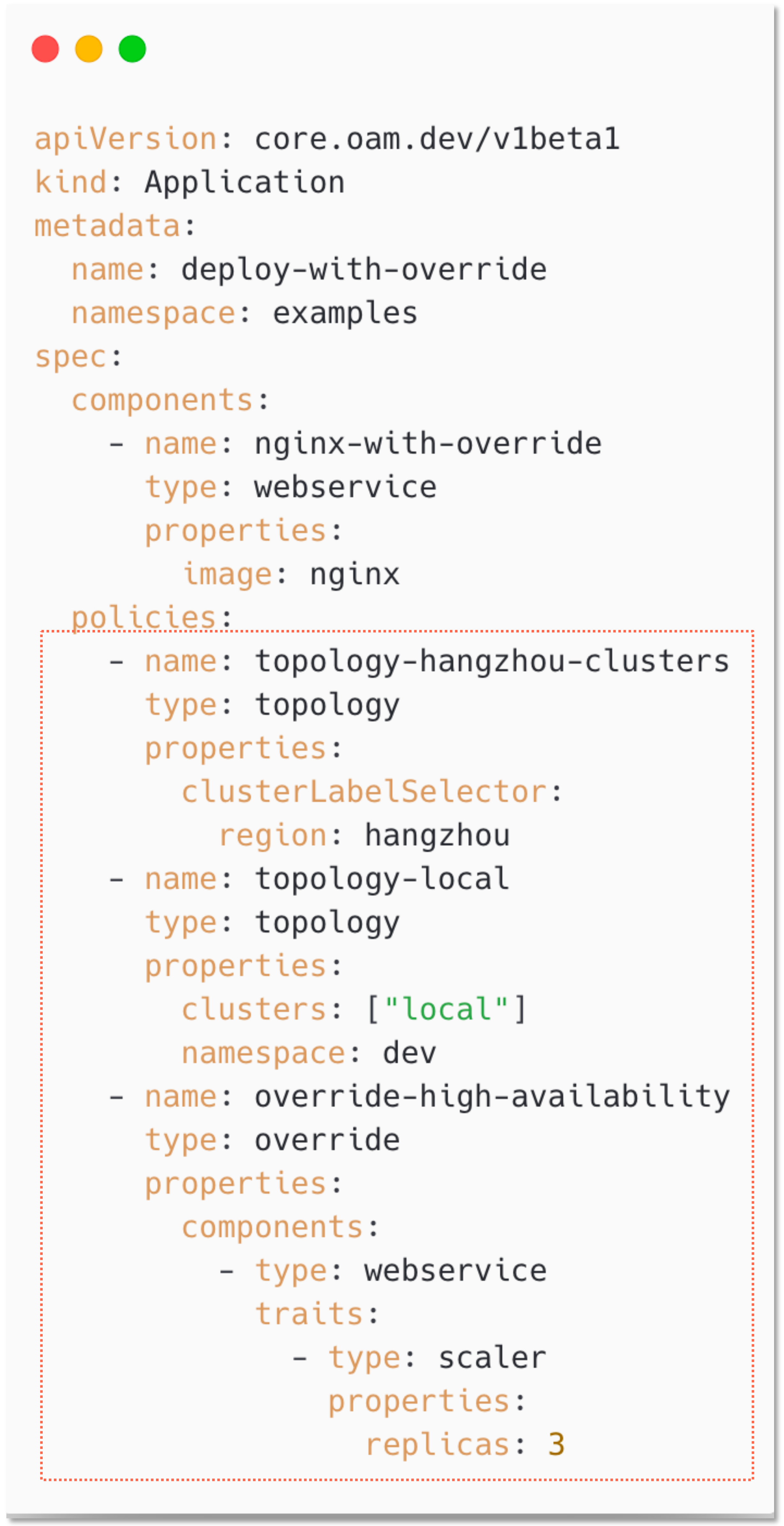


❑ User level multi-language provider supports.

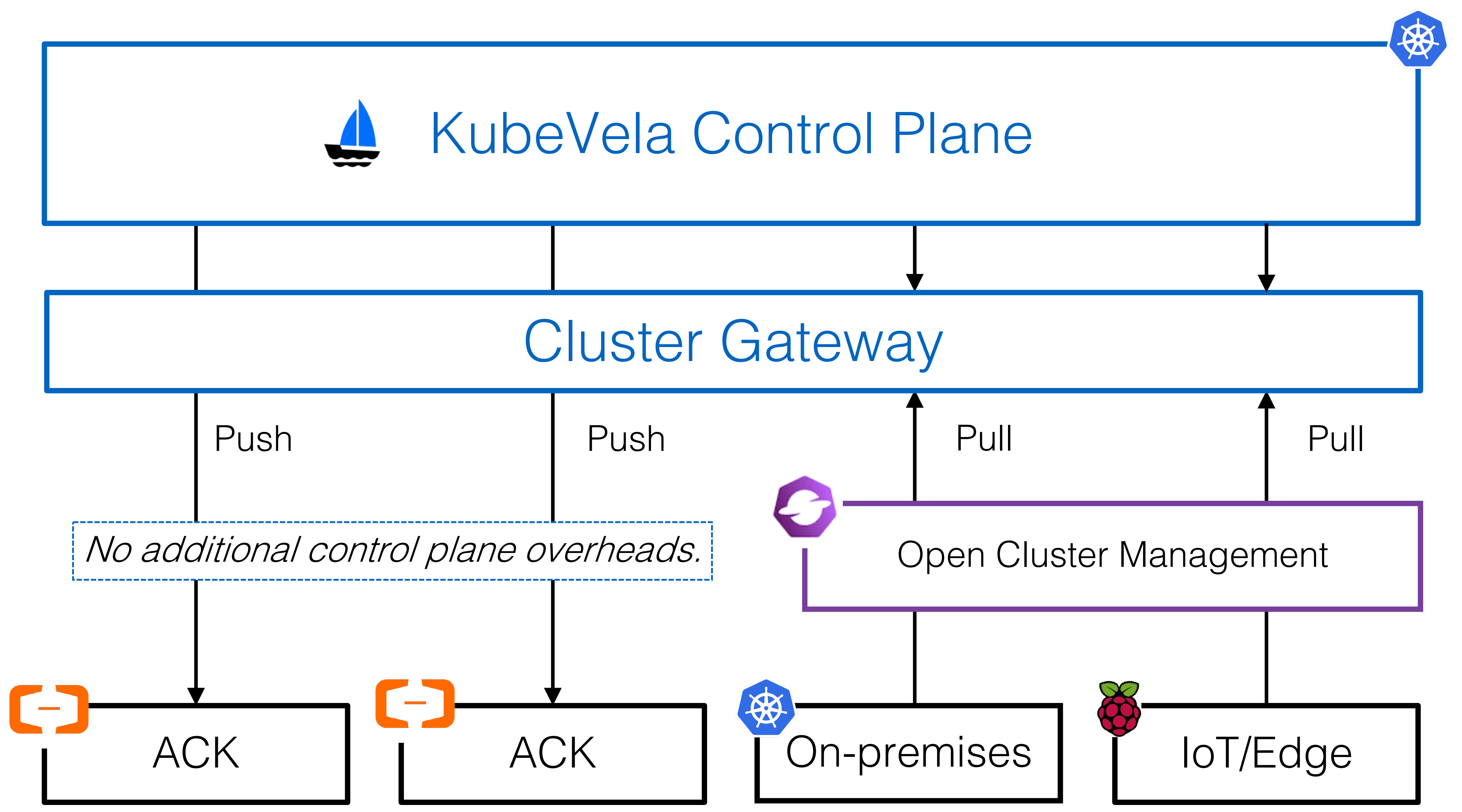
OAM/KubeVela basic work flow



Multi-clusters/hybrid-environments as first-class citizen



```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: deploy-with-override
  namespace: examples
spec:
  components:
    - name: nginx-with-override
      type: webservice
      properties:
        image: nginx
  policies:
    - name: topology-hangzhou-clusters
      type: topology
      properties:
        clusterLabelSelector:
          region: hangzhou
    - name: topology-local
      type: topology
      properties:
        clusters: ["local"]
        namespace: dev
    - name: override-high-availability
      type: override
      properties:
        components:
          - type: webservice
            traits:
              - type: scaler
                properties:
                  replicas: 3
```

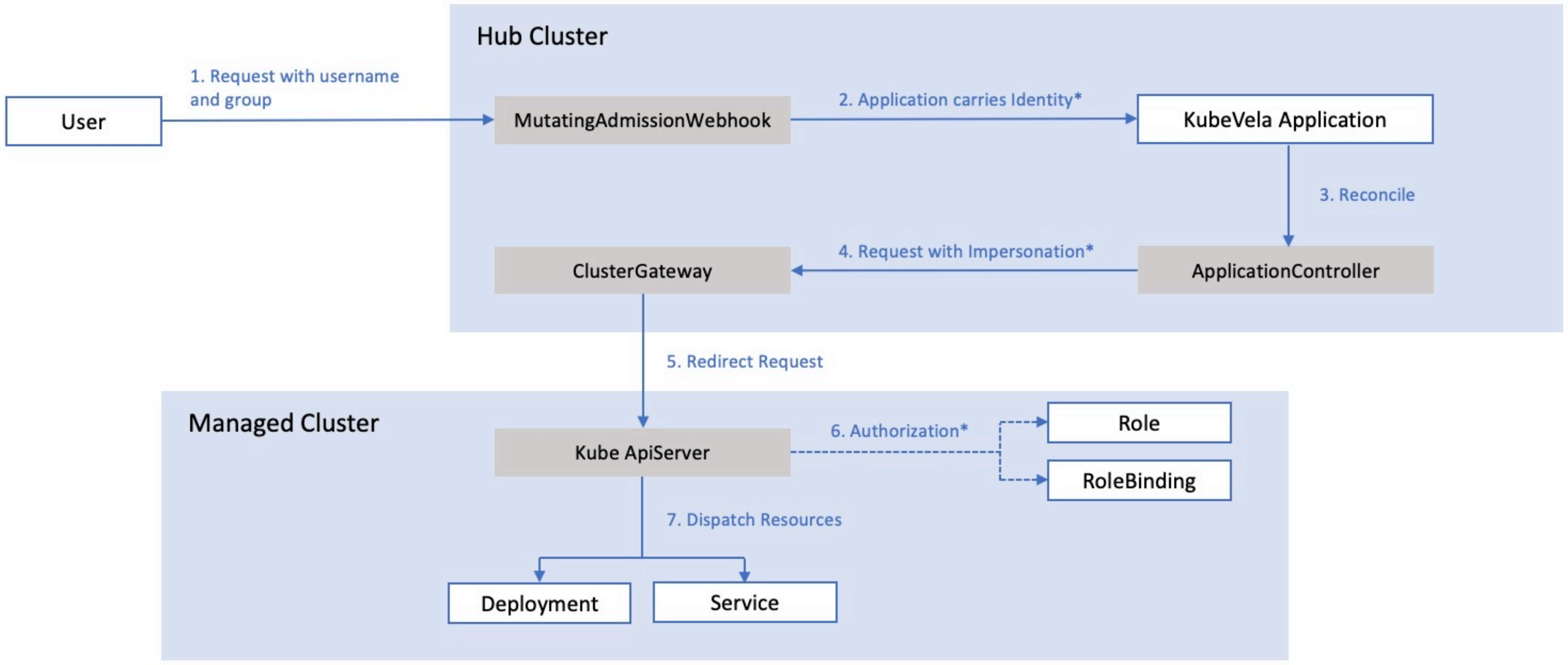


- ❑ Natively supports multi-clusters with rich placement strategy.
- ❑ Support both Push and Pull model for cluster management.
- ❑ Runtime agnostic, adopts any plugins and manage them only in the control plane.

Multi-clusters/hybrid-environments as first-class citizen

❑ Enhanced multi-cluster authentication and authorization.

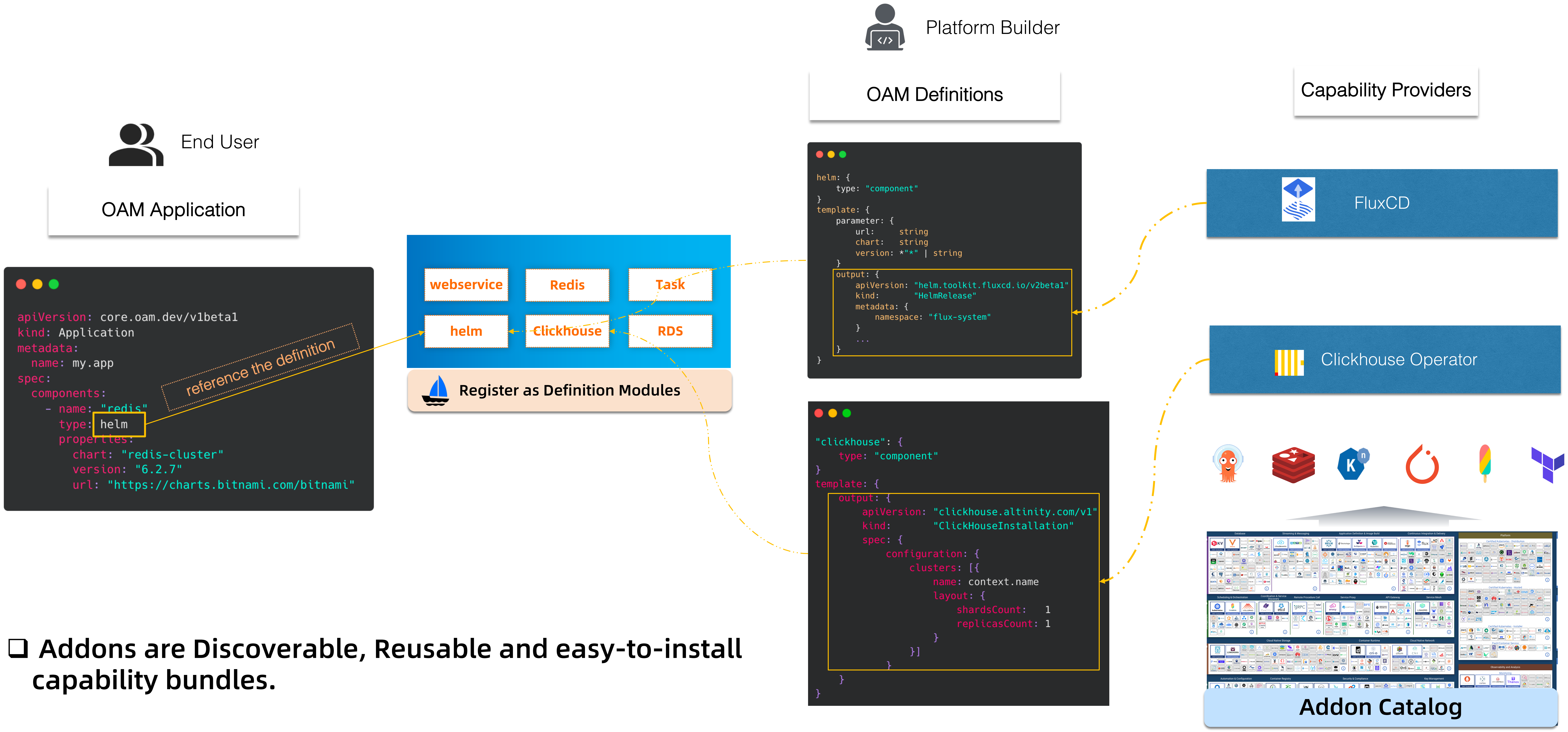
```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  annotations:
    app.oam.dev/username: alice
    app.oam.dev/group: dev-team
  name: nginx
spec:
  components:
    - type: webservice
      name: nginx
      properties:
        image: nginx
  policies:
    - type: topology
      name: europe-clusters
      properties:
        clusterLabelSelector:
          region: europe
    - type: topology
      name: china-clusters
      properties:
        clusterLabelSelector:
          region: china
```



The deploy process will use the identity of the user who created the application.
Leveraging the authentication of Kubernetes, this ensures unprivileged actions to be rejected across multi-clusters.

Need more? Use addon to build your own extension!

Alibaba Cloud



```
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: my.app
spec:
  components:
    - name: "redis"
      type: helm
      properties:
        chart: "redis-cluster"
        version: "6.2.7"
        url: "https://charts.bitnami.com/bitnami"
```

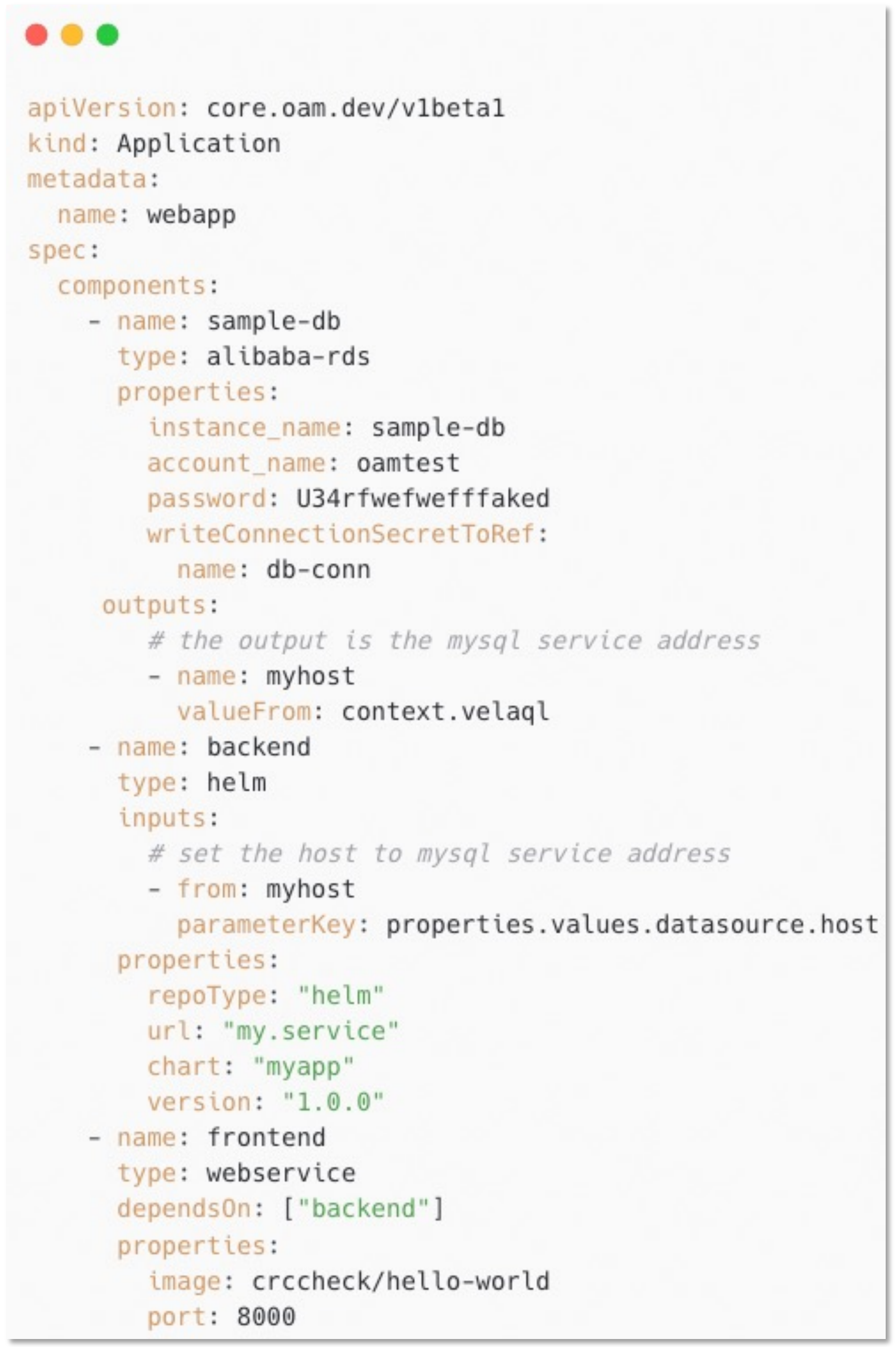
```
helm: {
    type: "component"
}
template: {
    parameter: {
        url:     string
        chart:   string
        version: *"*" | string
    }
    output: {
        apiVersion: "helm.toolkit.fluxcd.io/v2beta1"
        kind:       "HelmRelease"
        metadata: {
            namespace: "flux-system"
        }
        ...
    }
}
```

```
"clickhouse": {
    type: "component"
}
template: {
    output: {
        apiVersion: "clickhouse.altinity.com/v1"
        kind:       "ClickHouseInstallation"
        spec: {
            configuration: {
                clusters: [{
                    name: context.name
                    layout: {
                        shardsCount:   1
                        replicasCount: 1
                    }
                }]
            }
        }
    }
}
```

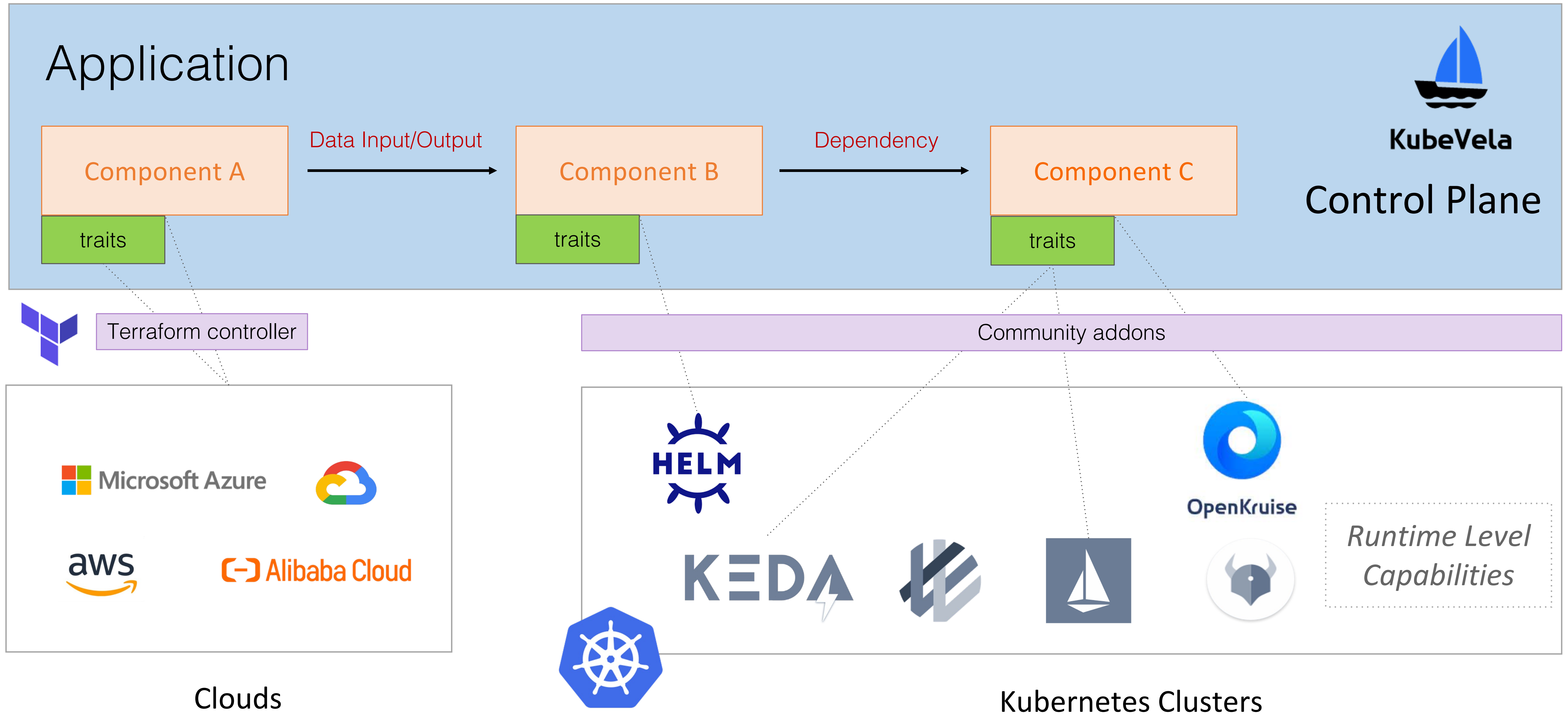
- ❑ **Addons are Discoverable, Reusable and easy-to-install capability bundles.**
- ❑ **The community currently have 50+ addons already.**

https://github.com/kubevela/catalog

Example: addon for unified application delivery



```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: webapp
spec:
  components:
    - name: sample-db
      type: alibaba-rds
      properties:
        instance_name: sample-db
        account_name: oamtest
        password: U34rfwefwefffaked
        writeConnectionSecretToRef:
          name: db-conn
     outputs:
        # the output is the mysql service address
        - name: myhost
          valueFrom: context.velaql
    - name: backend
      type: helm
      inputs:
        # set the host to mysql service address
        - from: myhost
          parameterKey: properties.values.datasource.host
      properties:
        repoType: "helm"
        url: "my.service"
        chart: "myapp"
        version: "1.0.0"
    - name: frontend
      type: webservice
      dependsOn: ["backend"]
      properties:
        image: crccheck/hello-world
        port: 8000
```

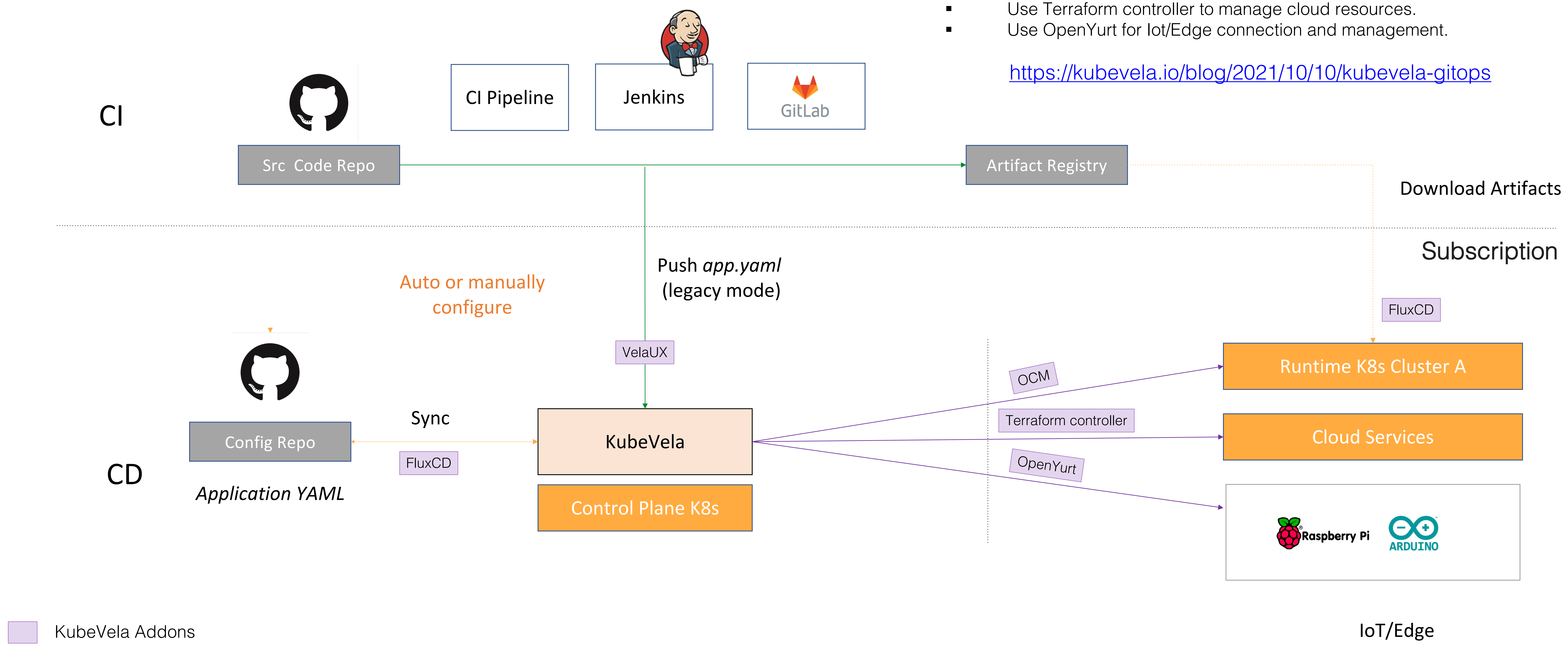


▢ KubeVela Addons

Example: addons for GitOps solution

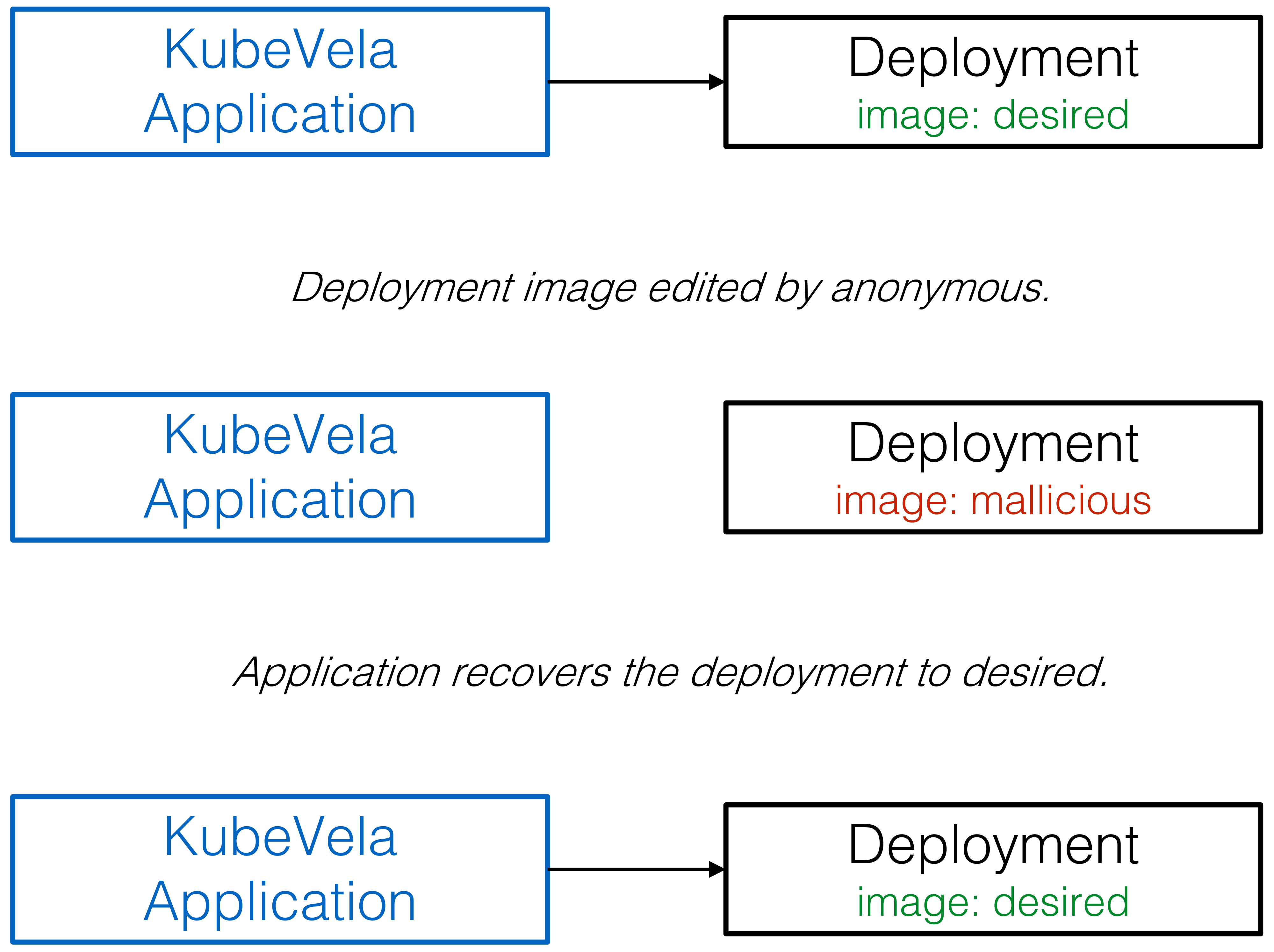
- Use fluxcd to sync YAML Artifacts from Repo to clusters.
- Use VelaUX to integrate with legacy push mode from CI systems.
- Use OCM to manage clusters in PULL mode.
- Use Terraform controller to manage cloud resources.
- Use OpenYurt for Iot/Edge connection and management.

https://kubevela.io/blog/2021/10/10/kubevela-gitops



2 Day 2 - Application Operating

Resource Management

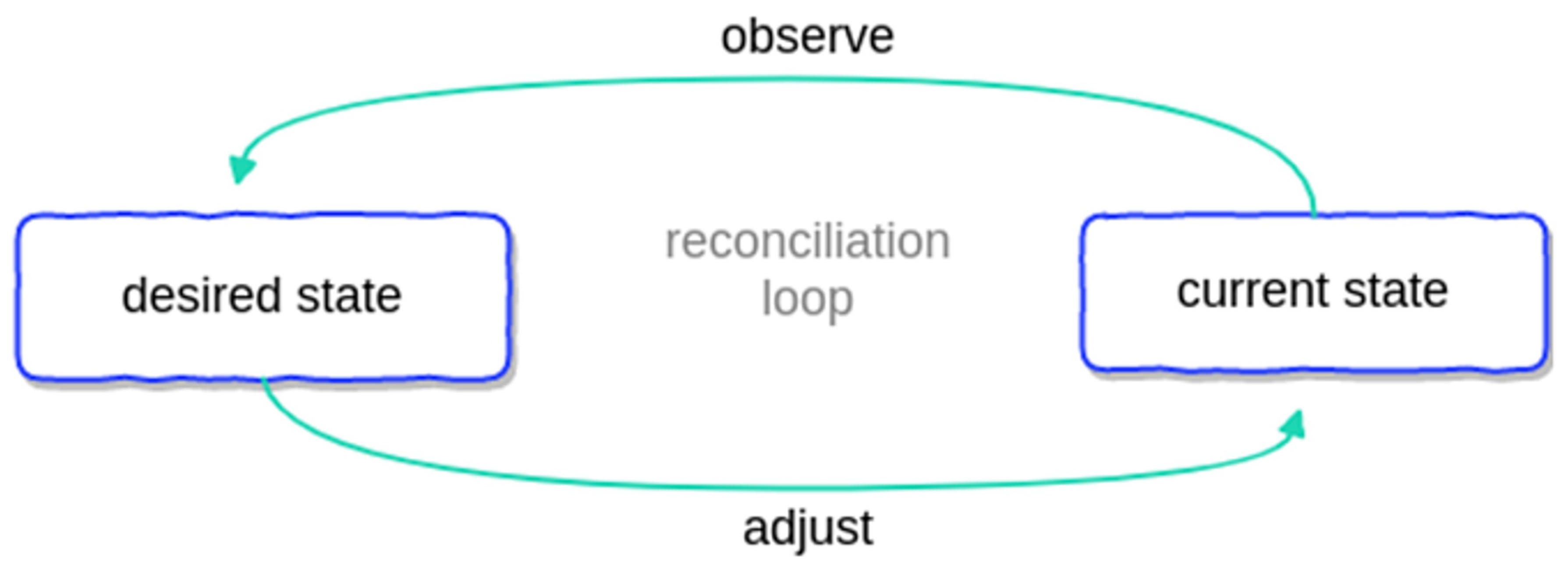


Deployment image edited by anonymous.

Application recovers the deployment to desired.

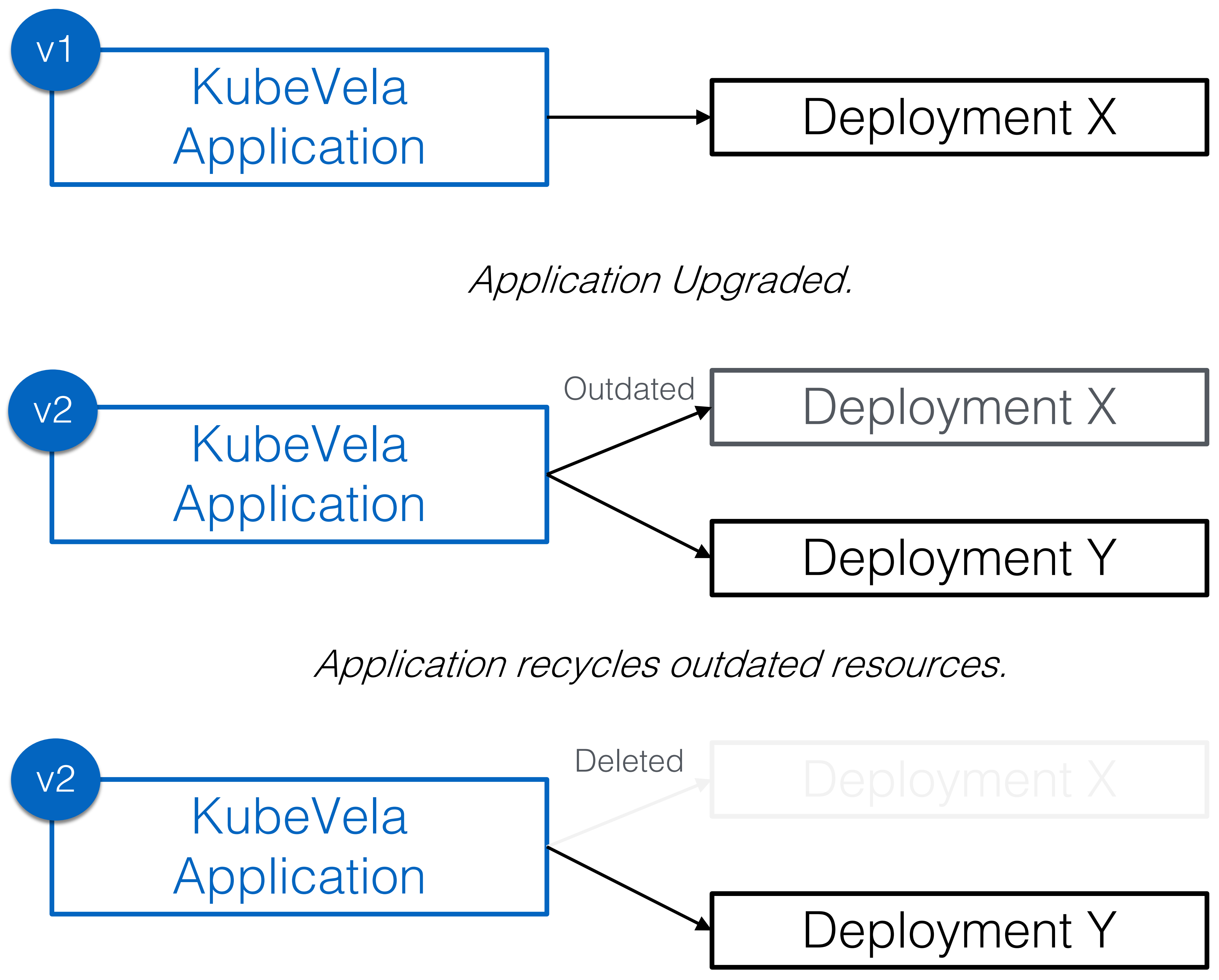
❑ No configuration drift.

The KubeVela Application repeatedly checks if managed resources are always in accord with the spec declared during the delivery process. It can effectively prevent configuration drift.



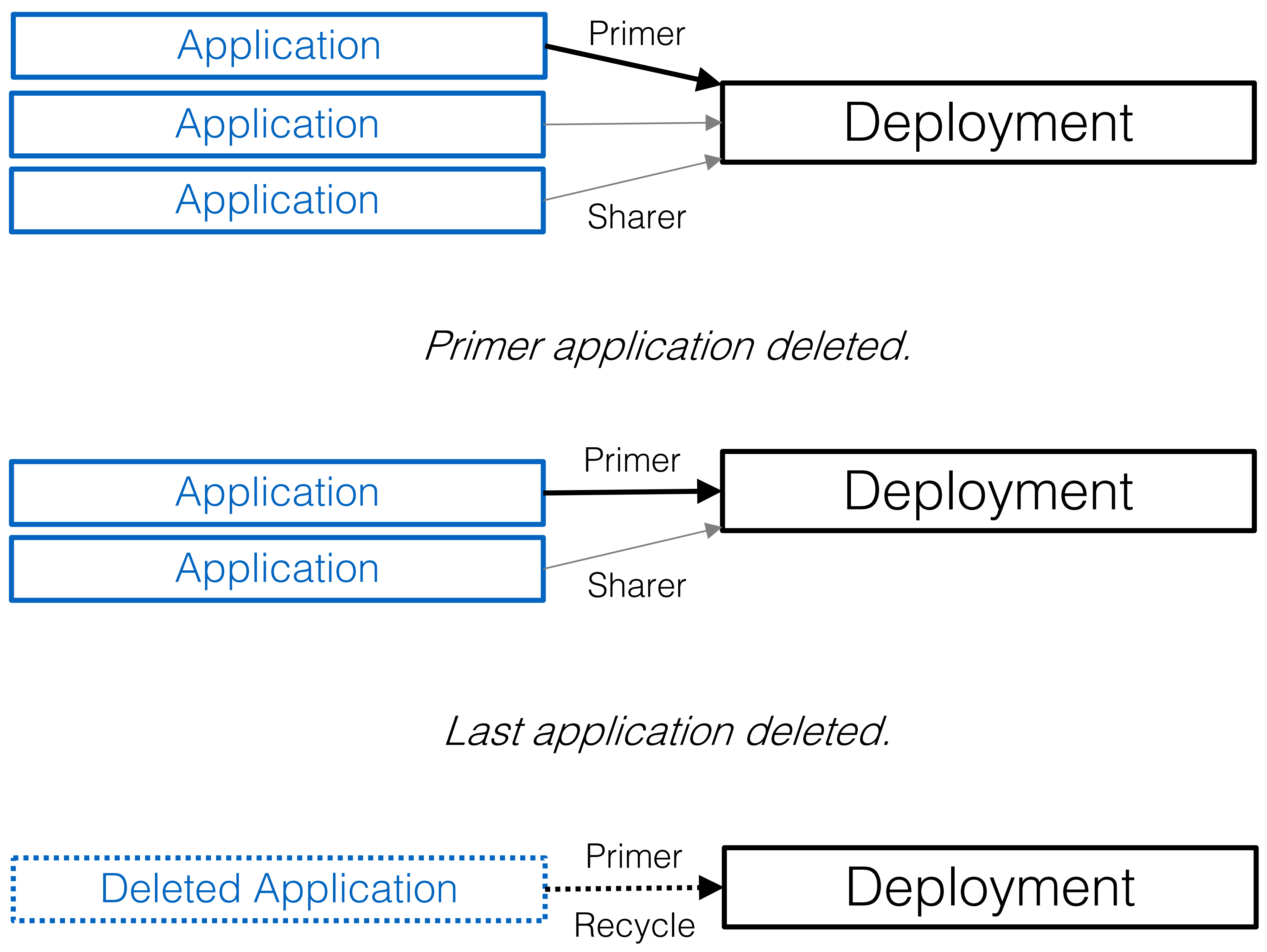
Resource Management

- ❑ Automated garbage collection.



The KubeVela Application recycles resources when the application itself is updated or deleted. Users can configure various strategy for outdated resources, such as keeping them or removing them.

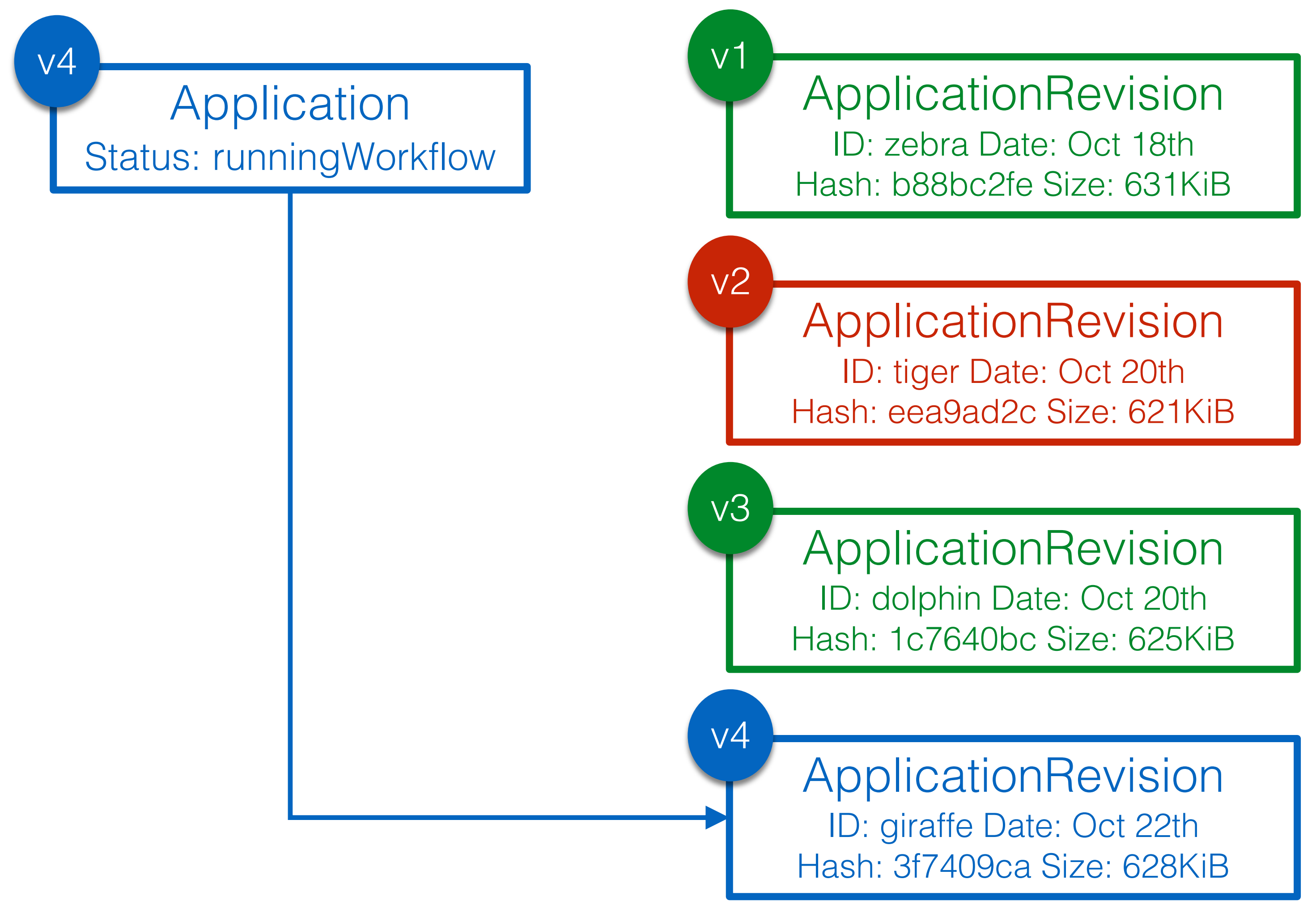
- ❑ Resources sharing across applications.



Resources can be shared by multiple applications. Shared resources are editable by the primer application and readable by all sharers. The last exit application is responsible for recycling them.

Application Version Control

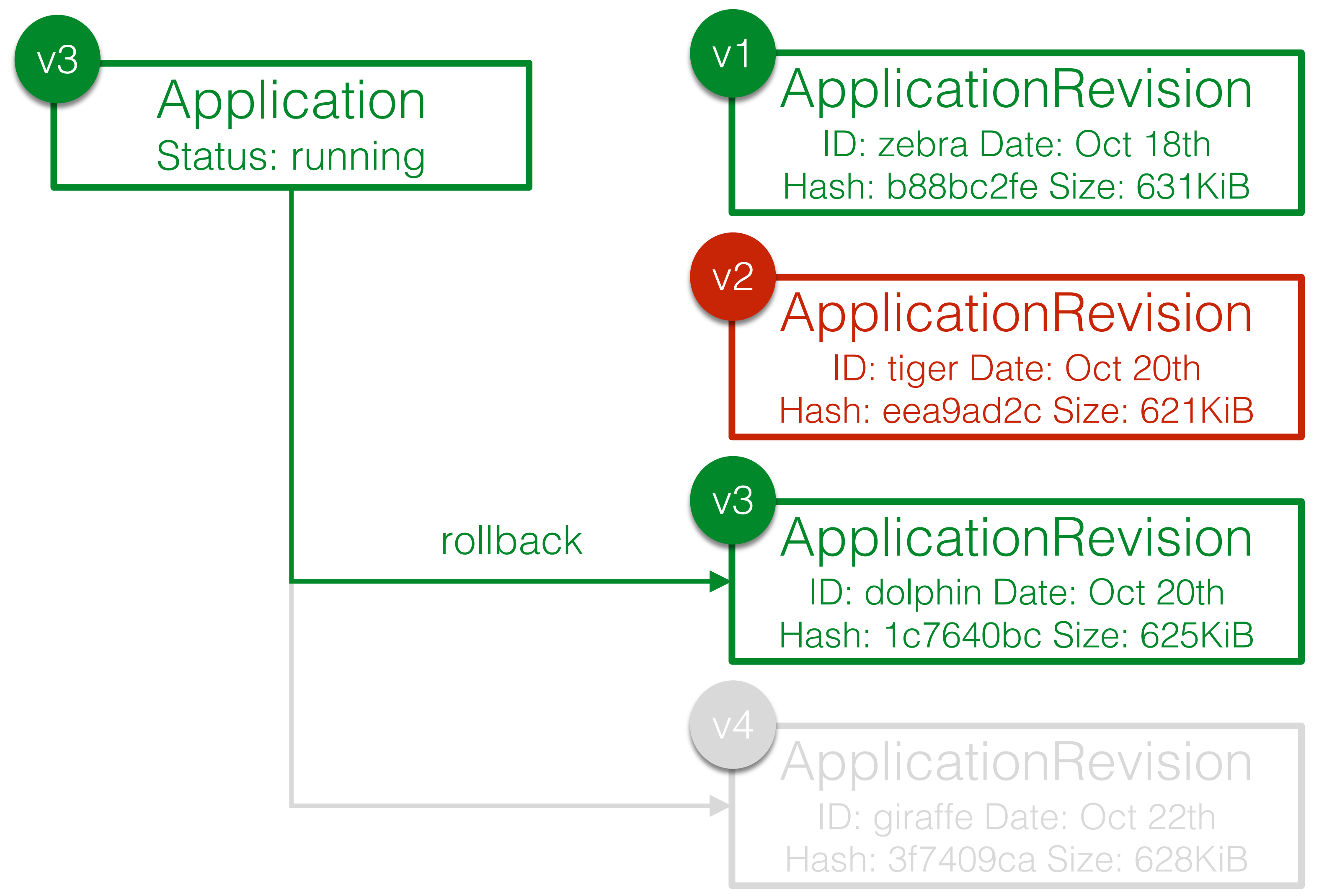
❑ History version recorded.



```
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: podinfo
  annotations:
-   app.oam.dev/publishVersion: v3
+   app.oam.dev/publishVersion: v4
  ...
```

Each KubeVela Application keeps limited history versions. Each version is a snapshot for the past delivery. Both the application and related definitions are recorded.

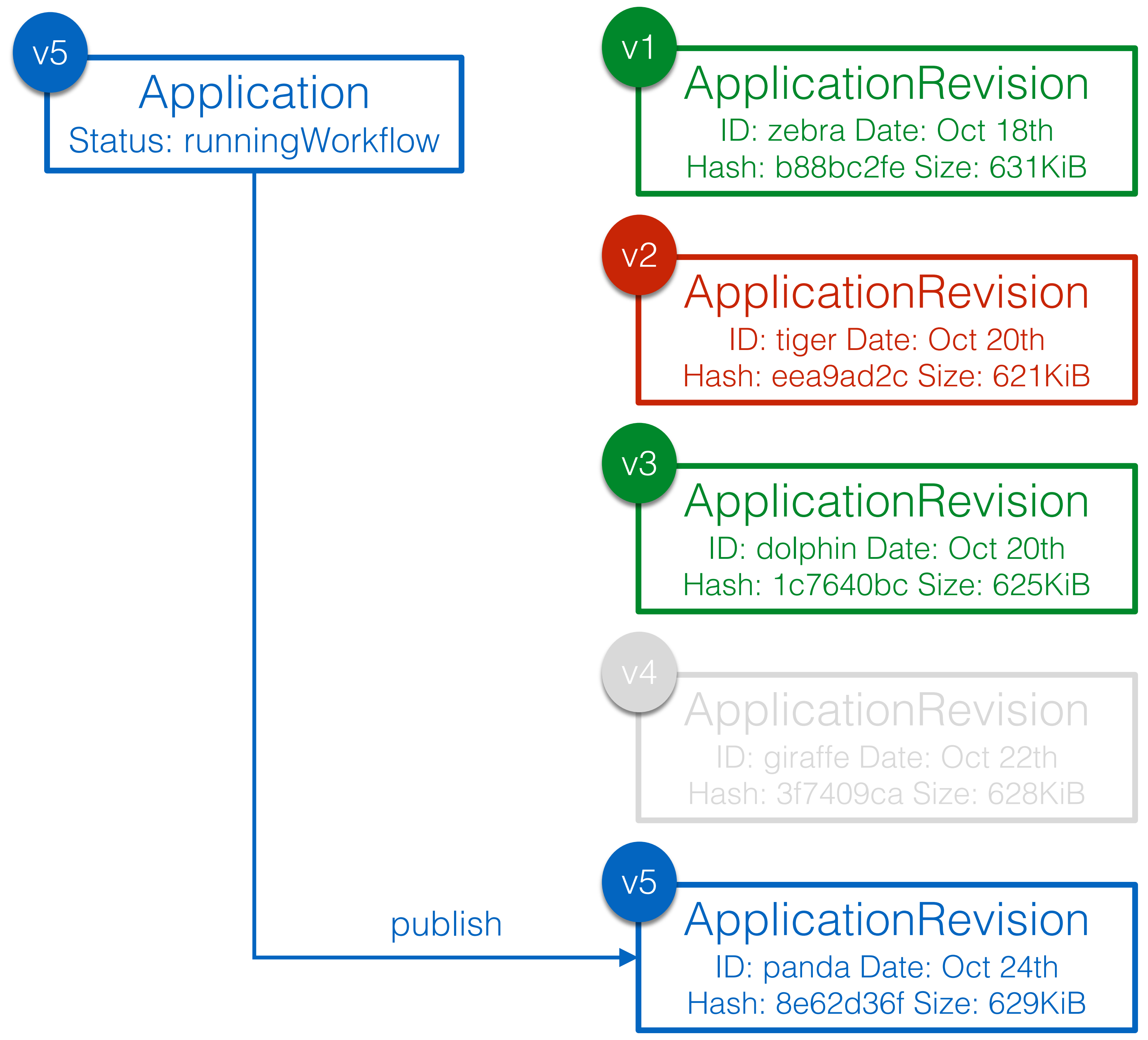
❑ Rollback to succeeded version.



```
$ vela live-diff my-app
    - type: webservice
      name: webapp
      properties:
-       image: webapp:dolphin
+       image: webapp:panda
```

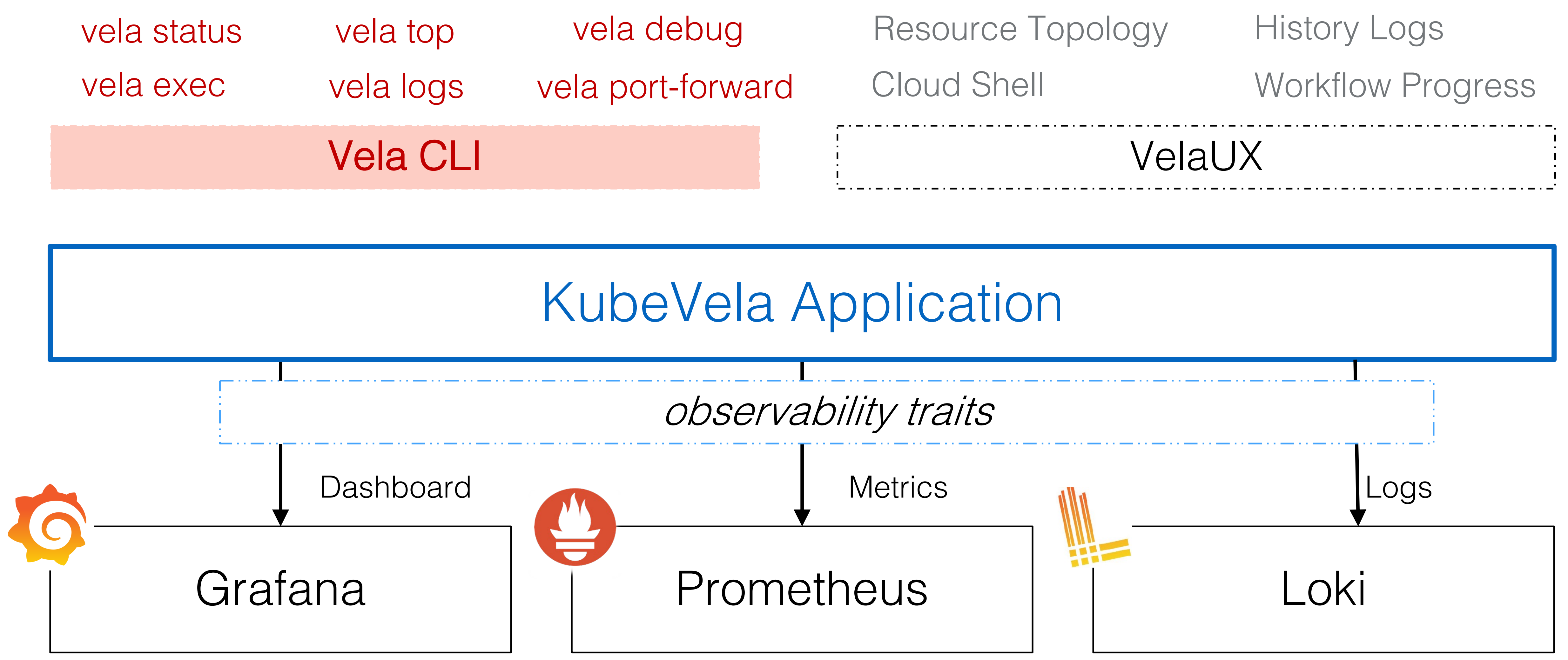
The KubeVela Application supports rolling back to history succeeded versions when new publish failed. Inspecting differences across versions is available as well.

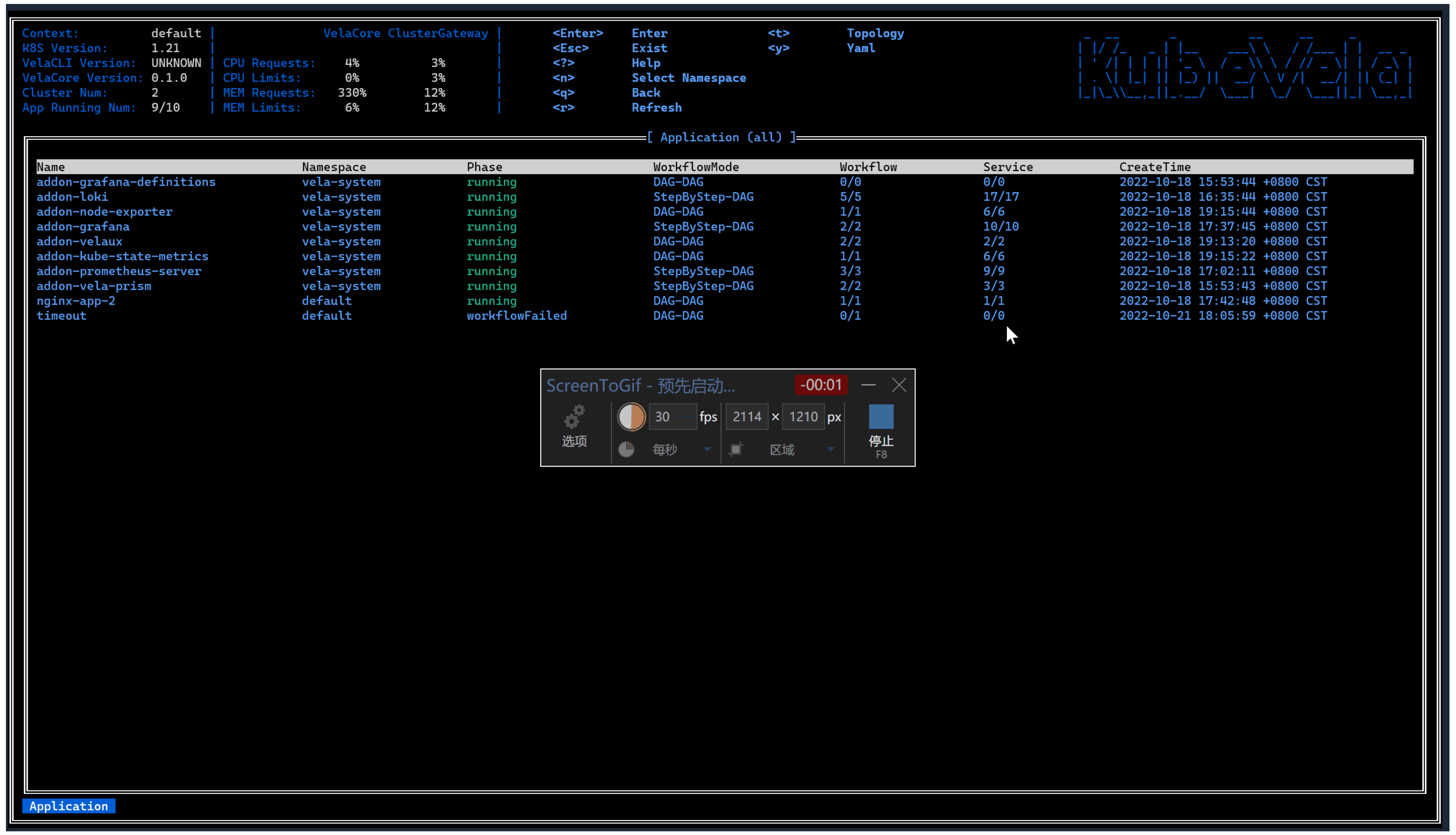
❑ Inspect changes across versions.



While KubeVela application usually automatically publishes new versions on spec changes, it is also possible to manually control the version publish, which allows users to edit application first and commit changes later.

Observability as first class citizen

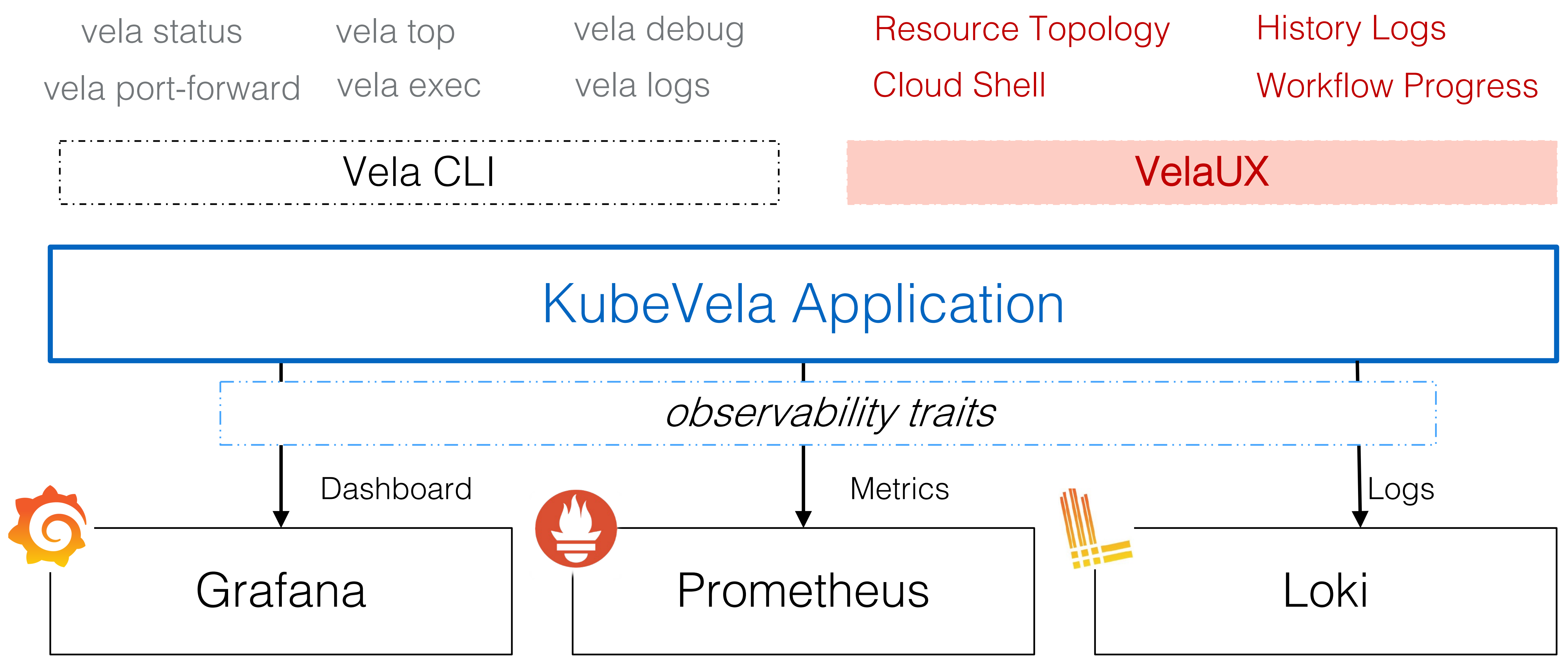



```
Context:           default |          VelaCore ClusterGateway |     <Enter>    Enter              <t>     Topology
K8S Version:       1.21    |                                  |     <Esc>      Exist              <y>     Yaml
VelaCLI Version:   UNKNOWN | CPU Requests:    4%        3%    |     <?>        Help
VelaCore Version:  0.1.0   | CPU Limits:      0%        3%    |     <n>        Select Namespace
Cluster Num:       2       | MEM Requests:  330%       12%    |     <q>        Back
App Running Num:   9/10    | MEM Limits:      6%       12%    |     <r>        Refresh

                                            [ Application (all) ]
Name                        Namespace     Phase             WorkflowMode     Workflow   Service   CreateTime
addon-grafana-definitions   vela-system   running           DAG-DAG          0/0        0/0       2022-10-18 15:53:44 +0800 CST
addon-loki                  vela-system   running           StepByStep-DAG   5/5        17/17     2022-10-18 16:35:44 +0800 CST
addon-node-exporter         vela-system   running           DAG-DAG          1/1        6/6       2022-10-18 19:15:44 +0800 CST
addon-grafana               vela-system   running           StepByStep-DAG   2/2        10/10     2022-10-18 17:37:45 +0800 CST
addon-velaux                vela-system   running           DAG-DAG          2/2        2/2       2022-10-18 19:13:20 +0800 CST
addon-kube-state-metrics    vela-system   running           DAG-DAG          1/1        6/6       2022-10-18 19:15:22 +0800 CST
addon-prometheus-server     vela-system   running           StepByStep-DAG   3/3        9/9       2022-10-18 17:02:11 +0800 CST
addon-vela-prism            vela-system   running           StepByStep-DAG   2/2        3/3       2022-10-18 15:53:43 +0800 CST
nginx-app-2                 default       running           DAG-DAG          1/1        1/1       2022-10-18 17:42:48 +0800 CST
timeout                     default       workflowFailed    DAG-DAG          0/1        0/0       2022-10-21 18:05:59 +0800 CST

Application
```

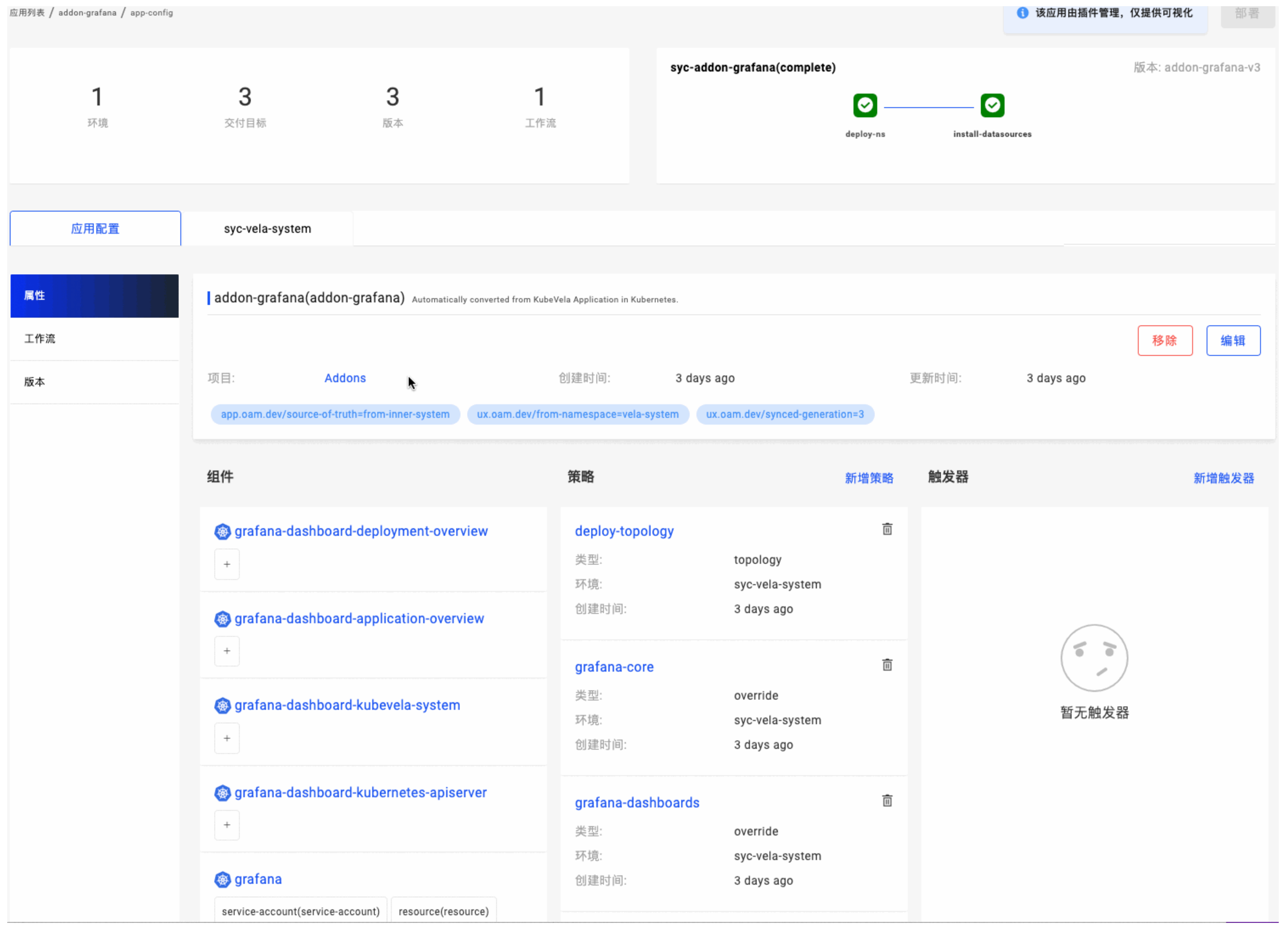
- ❑ Observability works for all extended resources across multi-clusters.
- ❑ Operating multi-cluster resources in a consistent way.

Observability as first class citizen

- ❑ Observability works for all extended resources across multi-clusters.



- ❑ Consistent experience for all extension.
- ❑ Topology graph from application to underlying resources.


应用列表 / addon-grafana / app-config

该应用由插件管理，仅提供可视化　部署

1	3	3	1
环境	交付目标	版本	工作流

syc-addon-grafana(complete)　版本: addon-grafana-v3

deploy-ns → install-datasources

应用配置　syc-vela-system

属性
工作流
版本

addon-grafana(addon-grafana) Automatically converted from KubeVela Application in Kubernetes.

移除　编辑

项目: Addons　创建时间: 3 days ago　更新时间: 3 days ago

app.oam.dev/source-of-truth=from-inner-system　ux.oam.dev/from-namespace=vela-system　ux.oam.dev/synced-generation=3

组件
- grafana-dashboard-deployment-overview
- grafana-dashboard-application-overview
- grafana-dashboard-kubevela-system
- grafana-dashboard-kubernetes-apiserver
- grafana: service-account(service-account)　resource(resource)

策略　新增策略

deploy-topology
类型:	topology
环境:	syc-vela-system
创建时间:	3 days ago

grafana-core
类型:	override
环境:	syc-vela-system
创建时间:	3 days ago

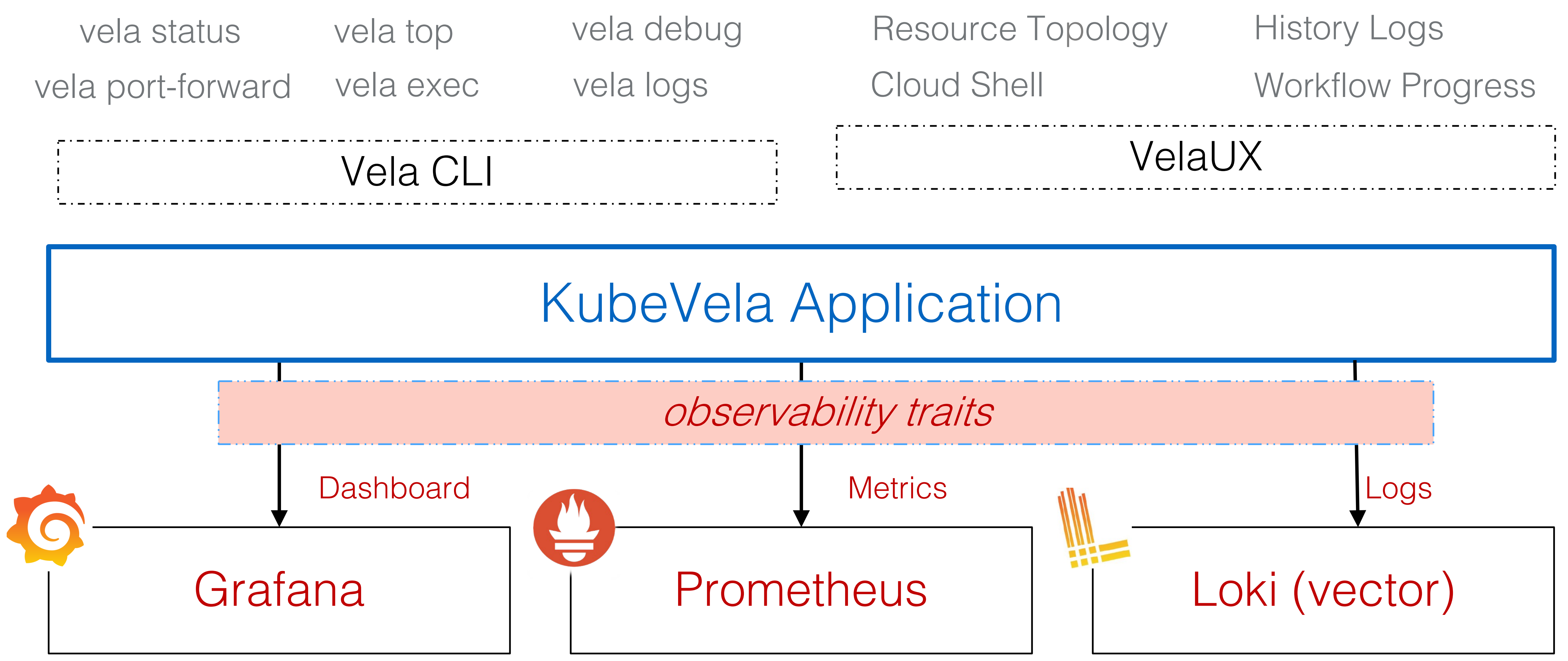
grafana-dashboards
类型:	override
环境:	syc-vela-system
创建时间:	3 days ago

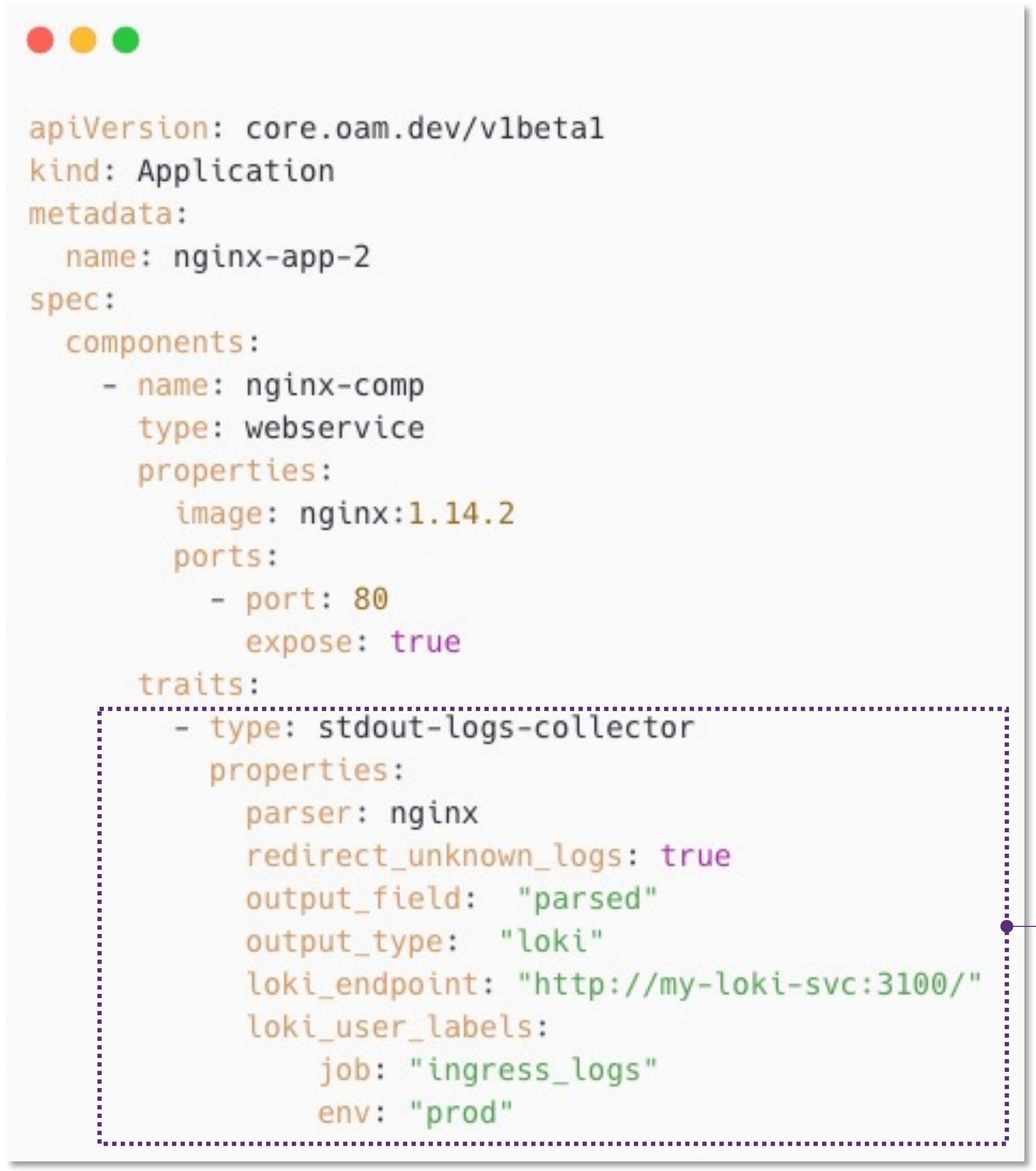
触发器　新增触发器

暂无触发器

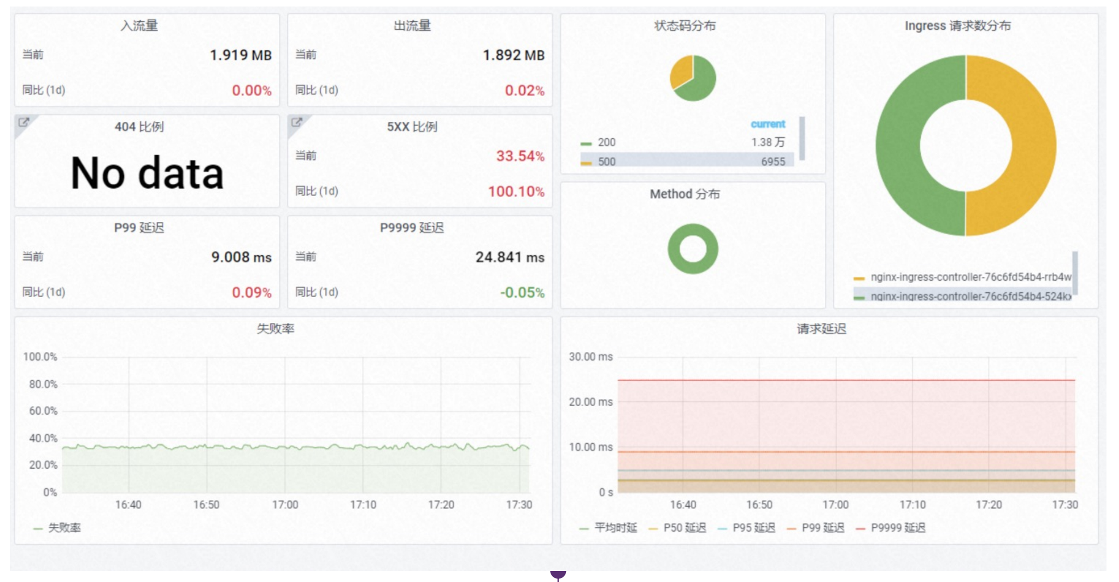
Observability as first class citizen

❑ Observability for all extended resources.





```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: nginx-app-2
spec:
  components:
    - name: nginx-comp
      type: webservice
      properties:
        image: nginx:1.14.2
        ports:
          - port: 80
            expose: true
      traits:
        - type: stdout-logs-collector
          properties:
            parser: nginx
            redirect_unknown_logs: true
            output_field:  "parsed"
            output_type:  "loki"
            loki_endpoint: "http://my-loki-svc:3100/"
            loki_user_labels:
                job: "ingress_logs"
                env: "prod"
```



❑ Application Centric Observability.

With the use of customized Components and Traits, users can define how to monitor applications, for example, the way logs are collected and the dashboards metrics are plotted on.

❑ Observability as Code.

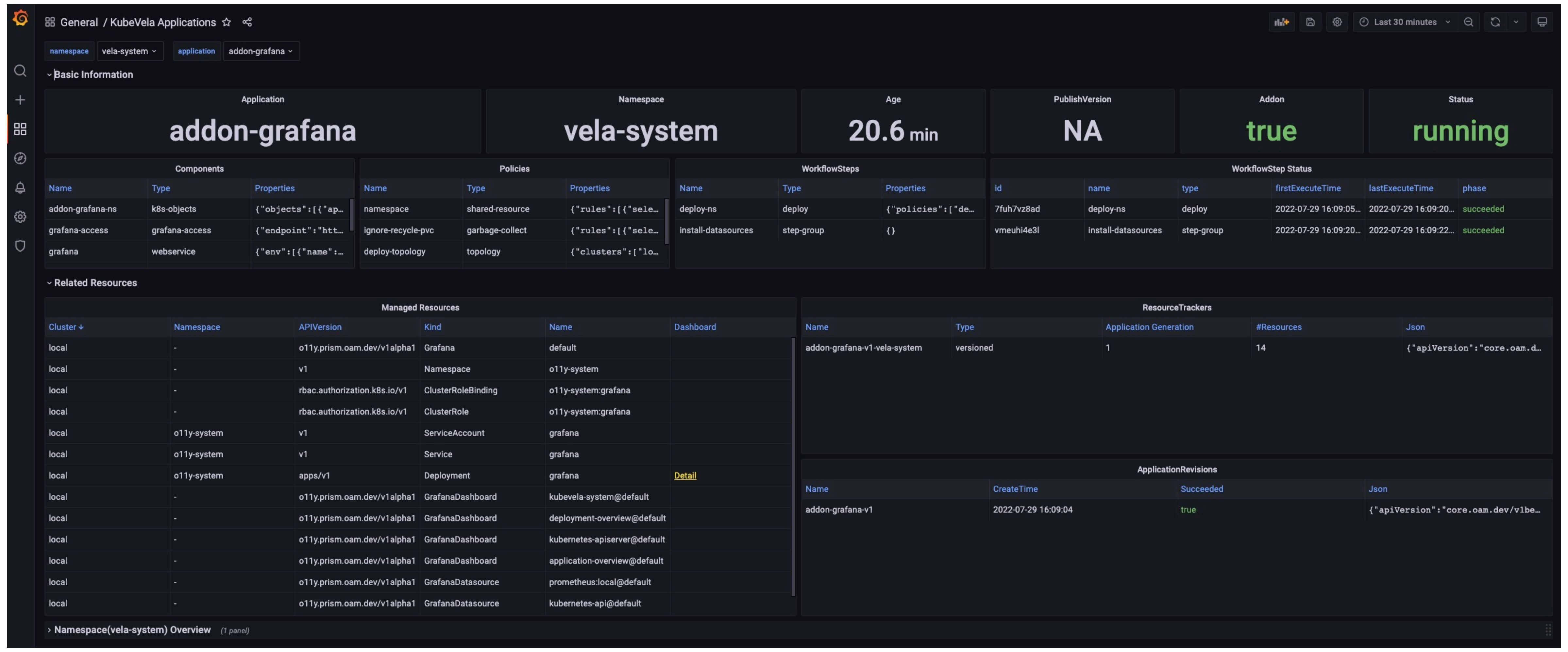
The Observability rules for applications are managed in declarative ways. It makes updates and migrations more convenient and controllable. Developers can leverage the power of underlying monitoring infrastructures without the need of learning varying complex syntax.

Observability as first class citizen

Automated System Observability

Alibaba Cloud

❑ Application Dashboard.



General / KubeVela Applications

namespace: vela-system — application: addon-grafana — Last 30 minutes

Basic Information

Application	Namespace	Age	PublishVersion	Addon	Status
addon-grafana	vela-system	20.6 min	NA	true	running

Components

Name	Type	Properties
addon-grafana-ns	k8s-objects	{"objects":[{"ap…
grafana-access	grafana-access	{"endpoint":"htt…
grafana	webservice	{"env":[{"name":…

Policies

Name	Type	Properties
namespace	shared-resource	{"rules":[{"sele…
ignore-recycle-pvc	garbage-collect	{"rules":[{"sele…
deploy-topology	topology	{"clusters":["lo…

WorkflowSteps

Name	Type	Properties
deploy-ns	deploy	{"policies":["de…
install-datasources	step-group	{}

WorkflowStep Status

id	name	type	firstExecuteTime	lastExecuteTime	phase
7fuh7vz8ad	deploy-ns	deploy	2022-07-29 16:09:05…	2022-07-29 16:09:20…	succeeded
vmeuhi4e3l	install-datasources	step-group	2022-07-29 16:09:20…	2022-07-29 16:09:22…	succeeded

Related Resources

Managed Resources

Cluster	Namespace	APIVersion	Kind	Name	Dashboard
local	-	o11y.prism.oam.dev/v1alpha1	Grafana	default	
local	-	v1	Namespace	o11y-system	
local	-	rbac.authorization.k8s.io/v1	ClusterRoleBinding	o11y-system:grafana	
local	-	rbac.authorization.k8s.io/v1	ClusterRole	o11y-system:grafana	
local	o11y-system	v1	ServiceAccount	grafana	
local	o11y-system	v1	Service	grafana	
local	o11y-system	apps/v1	Deployment	grafana	Detail
local	-	o11y.prism.oam.dev/v1alpha1	GrafanaDashboard	kubevela-system@default	
local	-	o11y.prism.oam.dev/v1alpha1	GrafanaDashboard	deployment-overview@default	
local	-	o11y.prism.oam.dev/v1alpha1	GrafanaDashboard	kubernetes-apiserver@default	
local	-	o11y.prism.oam.dev/v1alpha1	GrafanaDashboard	application-overview@default	
local	-	o11y.prism.oam.dev/v1alpha1	GrafanaDatasource	prometheus:local@default	
local	-	o11y.prism.oam.dev/v1alpha1	GrafanaDatasource	kubernetes-api@default	

ResourceTrackers

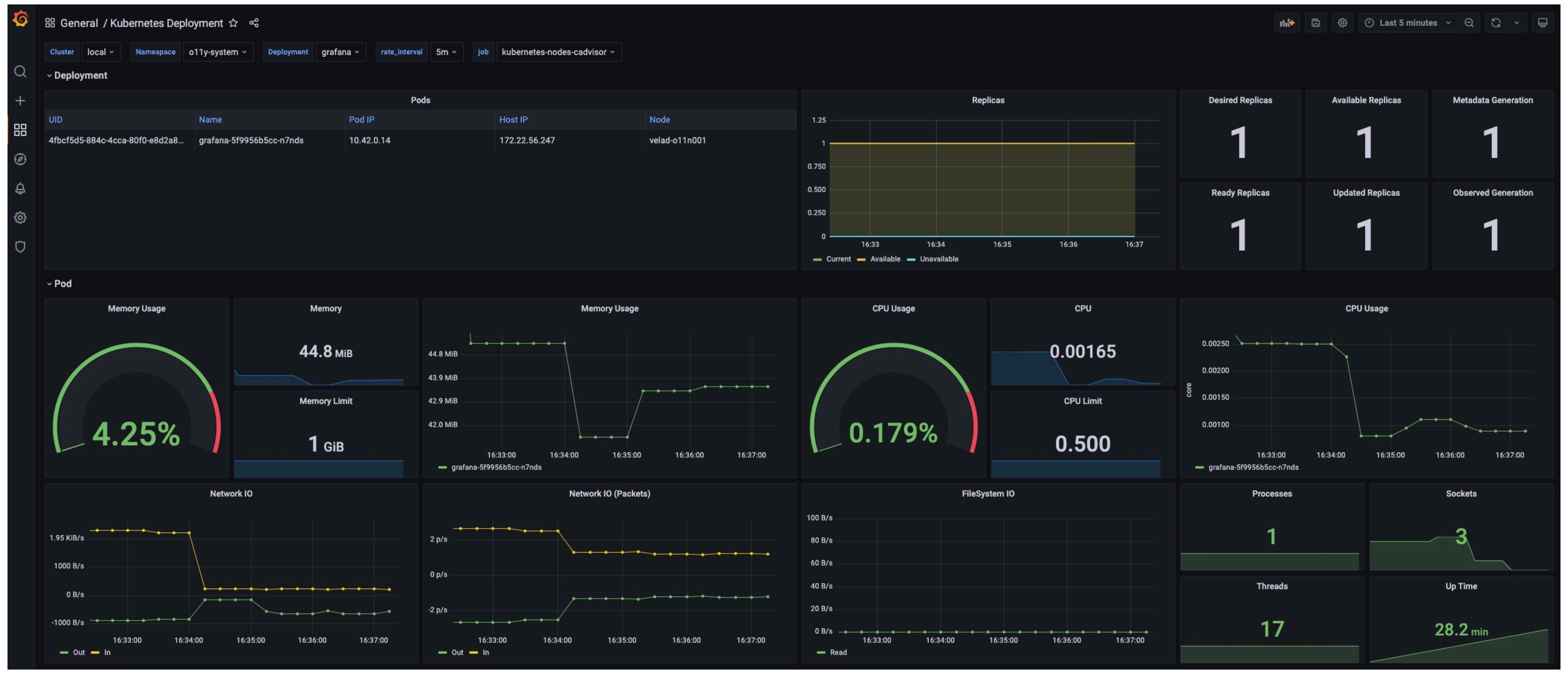
Name	Type	Application Generation	#Resources	Json
addon-grafana-v1-vela-system	versioned	1	14	{"apiVersion":"core.oam.d…

ApplicationRevisions

Name	CreateTime	Succeeded	Json
addon-grafana-v1	2022-07-29 16:09:04	true	{"apiVersion":"core.oam.dev/v1be…

Namespace(vela-system) Overview (1 panel)

❑ Kubernetes native resources Dashboard.



General / Kubernetes Deployment

Cluster: local — Namespace: o11y-system — Deployment: grafana — rate_interval: 5m — job: kubernetes-nodes-cadvisor — Last 5 minutes

Deployment

Pods

UID	Name	Pod IP	Host IP	Node
4fbcf5d5-884c-4cca-80f0-e8d2a8…	grafana-5f9956b5cc-n7nds	10.42.0.14	172.22.56.247	vela-o11n001

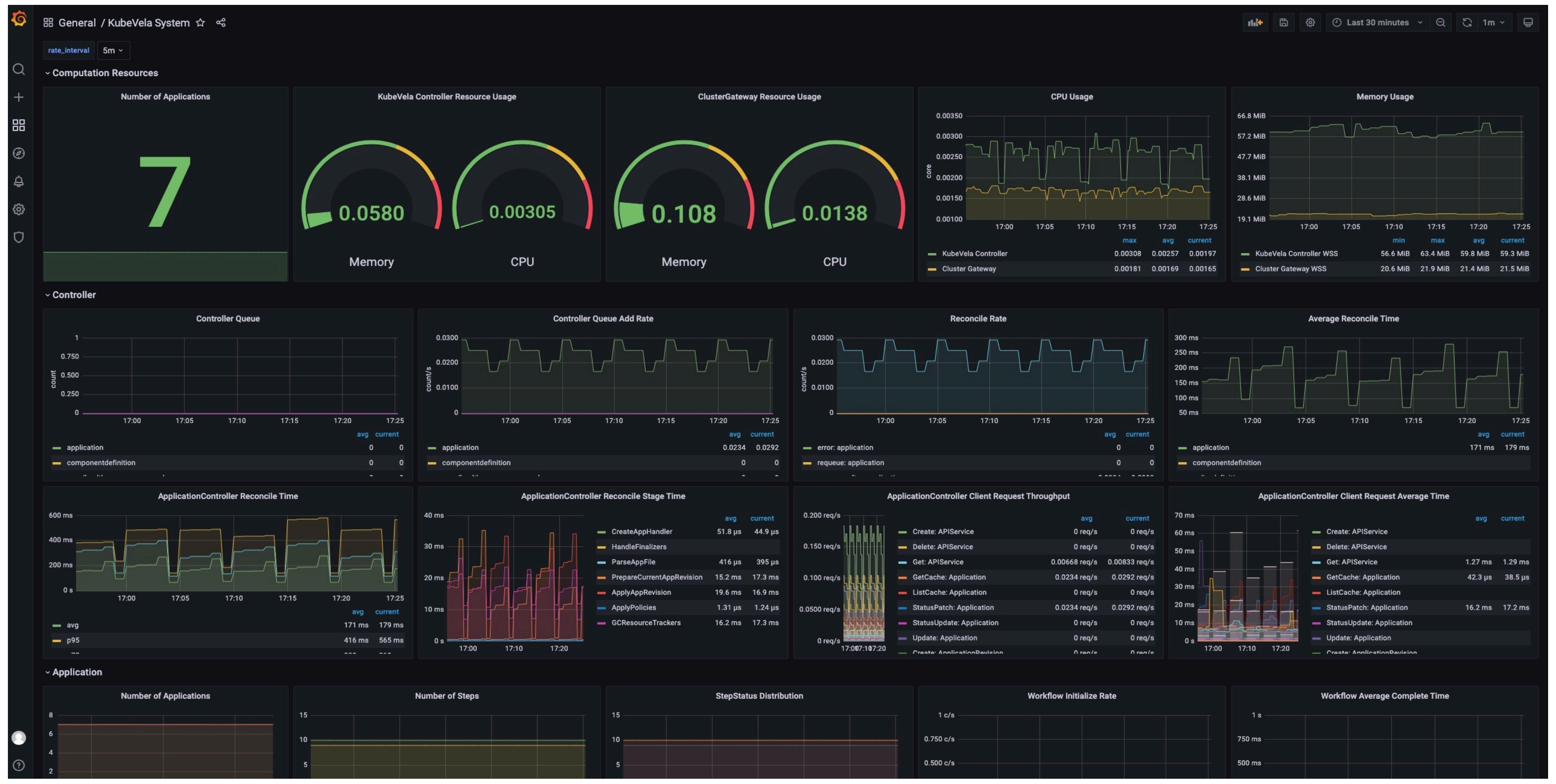
Desired Replicas	Available Replicas	Metadata Generation
1	1	1

Ready Replicas	Updated Replicas	Observed Generation
1	1	1

Pod

Memory Usage	Memory	Memory Limit	CPU Usage	CPU	CPU Limit	Processes	Sockets	Threads	Up Time
4.25%	44.8 MiB	1 GiB	0.179%	0.00165	0.500	1	3	17	28.2 min

❑ KubeVela System Dashboard.



General / KubeVela System

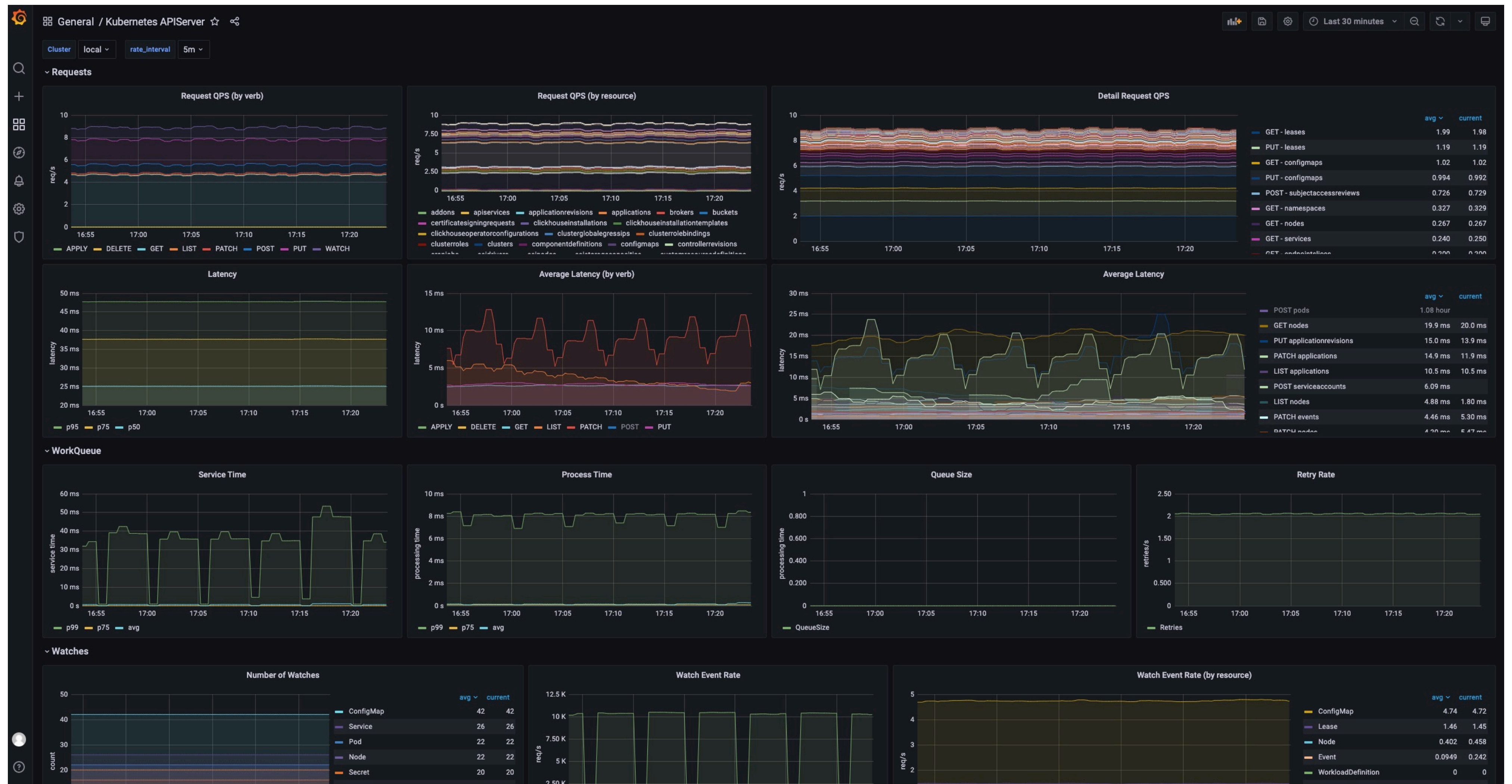
Computation Resources

Number of Applications: 7

KubeVela Controller Resource Usage		ClusterGateway Resource Usage	
Memory	CPU	Memory	CPU
0.0580	0.00305	0.108	0.0138

CPU Usage
Memory Usage
Controller
Controller Queue
Controller Queue Add Rate
Reconcile Rate
Average Reconcile Time
ApplicationController Reconcile Time
ApplicationController Reconcile Stage Time
ApplicationController Client Request Throughput
ApplicationController Client Request Average Time
Application
Number of Applications
Number of Steps
StepStatus Distribution
Workflow Initialize Rate
Workflow Average Complete Time

❑ Kubernetes APIServer Dashboard.



General / Kubernetes APIServer

KubeVela Eco-system: Tools

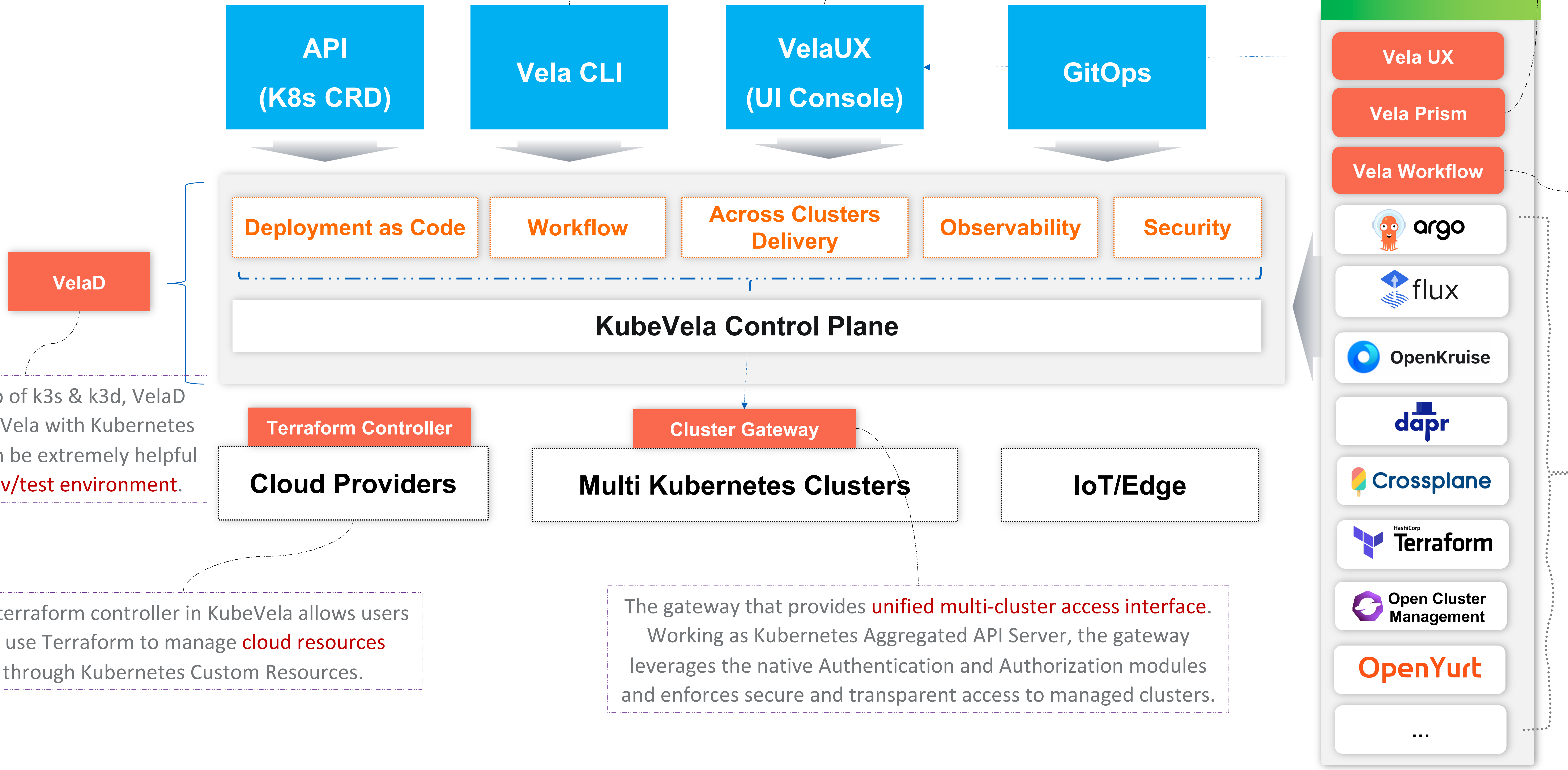
Alibaba Cloud

KubeVela CLI provides various commands that helps you to operate applications, such as managing definitions, viewing resources, restarting workflow, rolling versions.

VelaUX is the Web UI for KubeVela. Besides, it incorporates business logics into fundamental APIs and provides out-of-box user experiences for non-k8s-expert users.

The extension API server for KubeVela built upon the Kubernetes Aggregated API Server. It projects native APIs like creating dashboards on Grafana into Kubernetes resource APIs, so that users can manage 3rd-party resources as Kubernetes native resources.

The workflow engine translates CUE-based steps and executes them. It works as a pure delivery tool and can be used aside by the KubeVela application. Compared to Tekton, it mainly organize the process in CUE style, instead of using Pods and Jobs directly.



More than 50+ Community Addons...

Building on top of k3s & k3d, VelaD integrates KubeVela with Kubernetes cores, which can be extremely helpful for building dev/test environment.

The terraform controller in KubeVela allows users to use Terraform to manage cloud resources through Kubernetes Custom Resources.

The gateway that provides unified multi-cluster access interface. Working as Kubernetes Aggregated API Server, the gateway leverages the native Authentication and Authorization modules and enforces secure and transparent access to managed clusters.

https://github.com/kubevela/

Part 3

Play with KubeVela

KubeVela Stability

Performance and Fine-tuning

Scale	#Nodes	#Apps	#Pods	#Threads	QPS	Burst	CPU	Memory
Small	< 200	< 3,000	< 18,000	2	300	500	0.5	1Gi
Medium	< 500	< 5,000	< 30,000	4	500	800	1	2Gi
Large	< 1,000	< 12,000	< 72,000	4	800	1,000	2	4Gi

NOTE: The above configurations are based on medium size applications (each application contains 2~3 components and 5~6 resources).

System Monitoring

The observability infrastructures also include the necessary tools for monitoring the health status of KubeVela control plane. Exceptions and performance bottlenecks will be exposed by the metrics and dashboards.

Load Testing

Several rounds of load testing of KubeVela system has demonstrated that KubeVela is capable of processing thousands of applications under limited resources. The capacity can be scaled up almost linearly given more resources.

Customized Tunning

As KubeVela can be used under various scenarios, it is possible to crop partial capabilities of KubeVela in return of higher performance.

KubeVela Adopters

Areas uses KubeVela.

Commercial Banks

Car Manufacturers

Cloud Providers

Game Companies

KubeVela is applied across various areas to help manage application systems, especially high-tech industries.

https://github.com/kubevela/community/blob/main/ADOPTERS.md

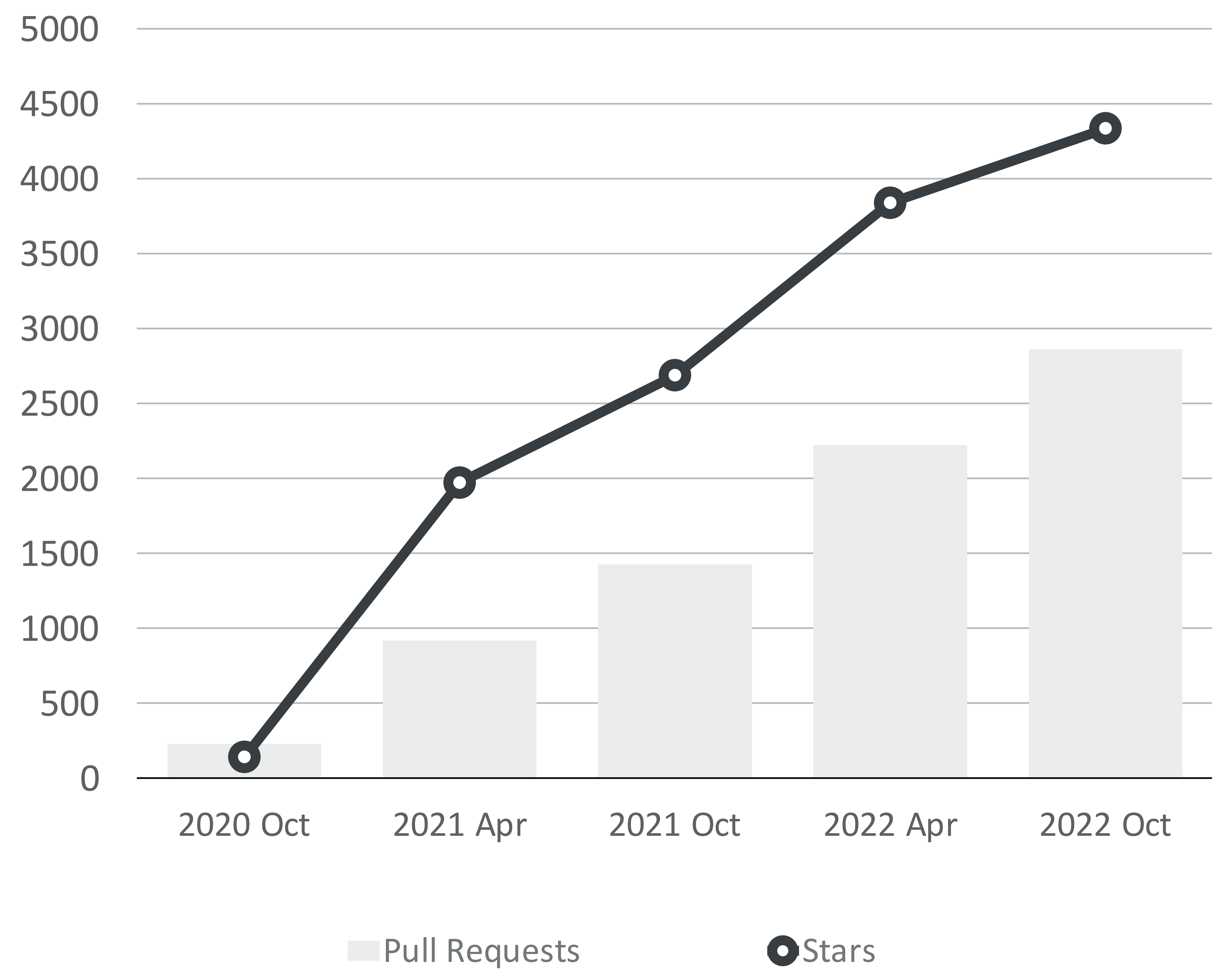
KubeVela Milestones

Version Releases & Key Features

v1.0
Application
v1.1
Workflow
v1.2
VelaUX
v1.3
Multi-cluster
v1.4
Authentication
v1.5
Observability
v1.6
Pipeline
2021 Mar
2021 Sep
2022 Jan
2022 Mar
2022 May
2022 Aug
2022 Oct

KubeVela Community

KubeVela attracts world-wide contributors and continuously evolves.



Contributors

KubeVela has attracted over 200 contributors from various countries, including China, USA, India, Germany, Korea, Spain, etc.

Issues

KubeVela received over 1,400 issues and has solved 85% of them.

Biweekly Community Meetings

KubeVela holds bi-weekly community meetings and has recorded 30+ English meetings on YouTube.

https://github.com/kubevela/community

Alibaba Cloud

Alibaba Cloud | MORE THAN JUST CLOUD