KubeCon | CloudNativeCon

Europe 2023

KubeCon | CloudNativeCon

Europe 2023

Deploying Cloud-Native Applications Using Kubevela and OAM

Daniel Higuero

Deploying Cloud-Native Applications Using Kubevela and OAM

KubeCon | CloudNativeCon

Europe 2023

18-21 April

Daniel Higuero

CTO

About me

- CTO @ NAPPTIVE
- OAM and KubeVela maintainer
- Work in the past on
  - Big Data
  - Streaming ML models
  - Edge computing

Questions on OAM/KubeVela?
➪ **KubeVela CNCF kiosk #27**

Agenda

- Open Application Model
- KubeVela
- Installing KubeVela
- Basic operations
- VelaUX
- Application workflows
- Multi-cluster deployment
- GitOps
- Extra content

Associated content

https://github.com/napptive/kubecon-23-oam-kubevela-tutorial

Open Application Model

Open Application Model (OAM)

Open Application Model (OAM) is a set of standard yet higher level abstractions for modeling cloud native applications on top of today's hybrid and multi-cloud environments.

https://oam.dev/

Open Application Model

- Application as the top level entity
  - Born on 2019
  - Latest revision is v0.3.1
- Infrastructure agnostic
- Provider agnostic
- Focus on
  - Reusability
  - Understandability
- Can easily be extended

https://oam.dev/

Alibaba Cloud

NAPPTIVE

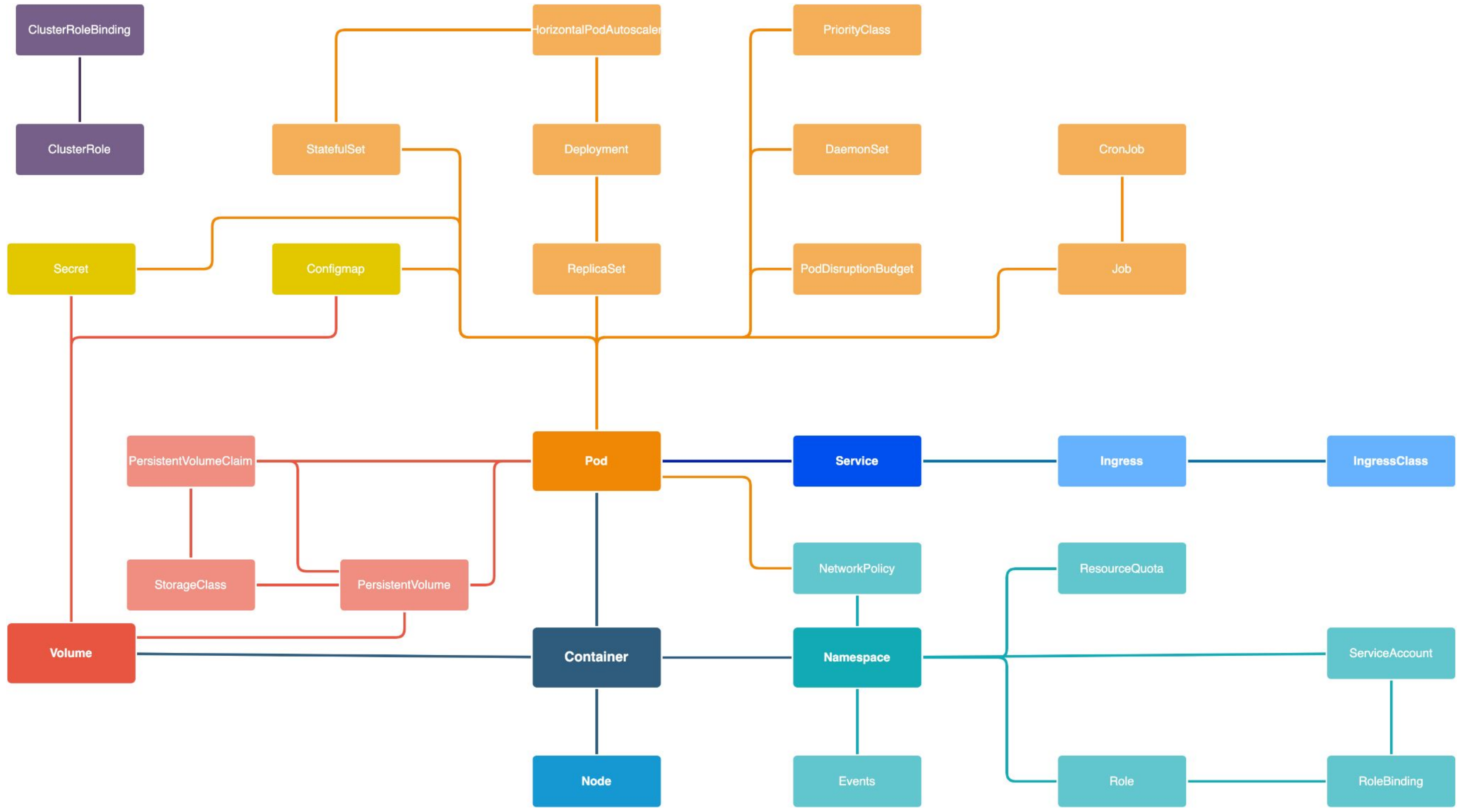
Microsoft

Which problem are we addressing?

- Kubernetes is a high-performance, battle-tested framework to run our applications
  - Can be used to run any application
  - Can be adapted to any use case
- But
  - It is **difficult to learn**
  - It works with low-level entities
  - It requires making several entities work coherently together

Kubernetes entity map

KubeCon | CloudNativeCon Europe 2023

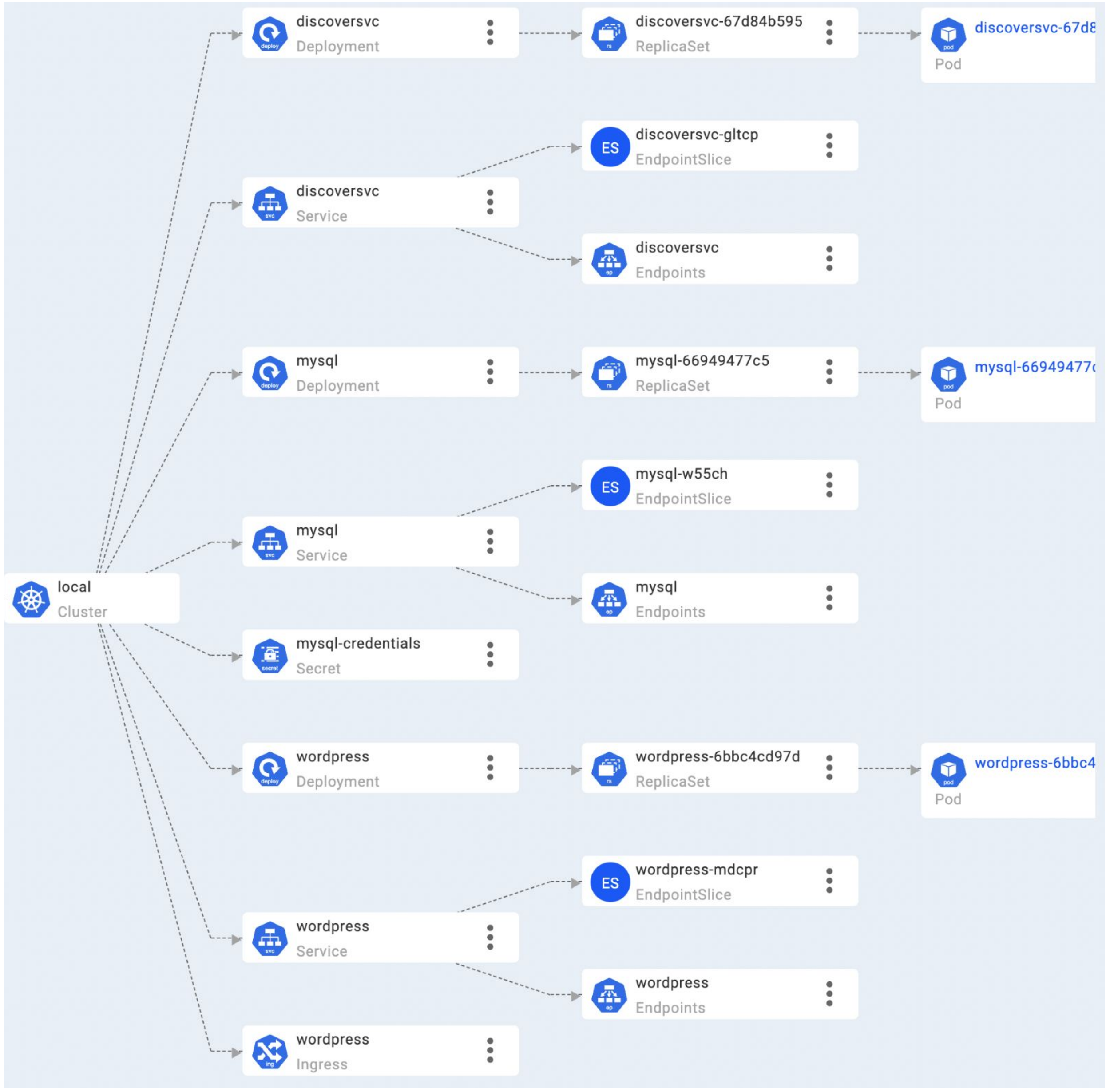


Is my app running?

- The low level entities allow us to
  - Represent the majority of use-cases
  - Deploy any application
  - Extract the required performance from the application
  - Adapt the cluster resources to the deployed workload
- But follows a bottom-up approach
  - Determining if an application is running means identifying and checking all the base K8s entities
  - Some entities are provider dependent

Is my app running?

- Typical mitigation approach
  - Labeling all resources
  - But we may depend on existing labeling approaches for existing applications



Why OAM?

- Offer a top-down approach to reason about applications
- An application becomes the focus of the system
  - All resources related to an application are automatically identified
  - Status information can be easily aggregated

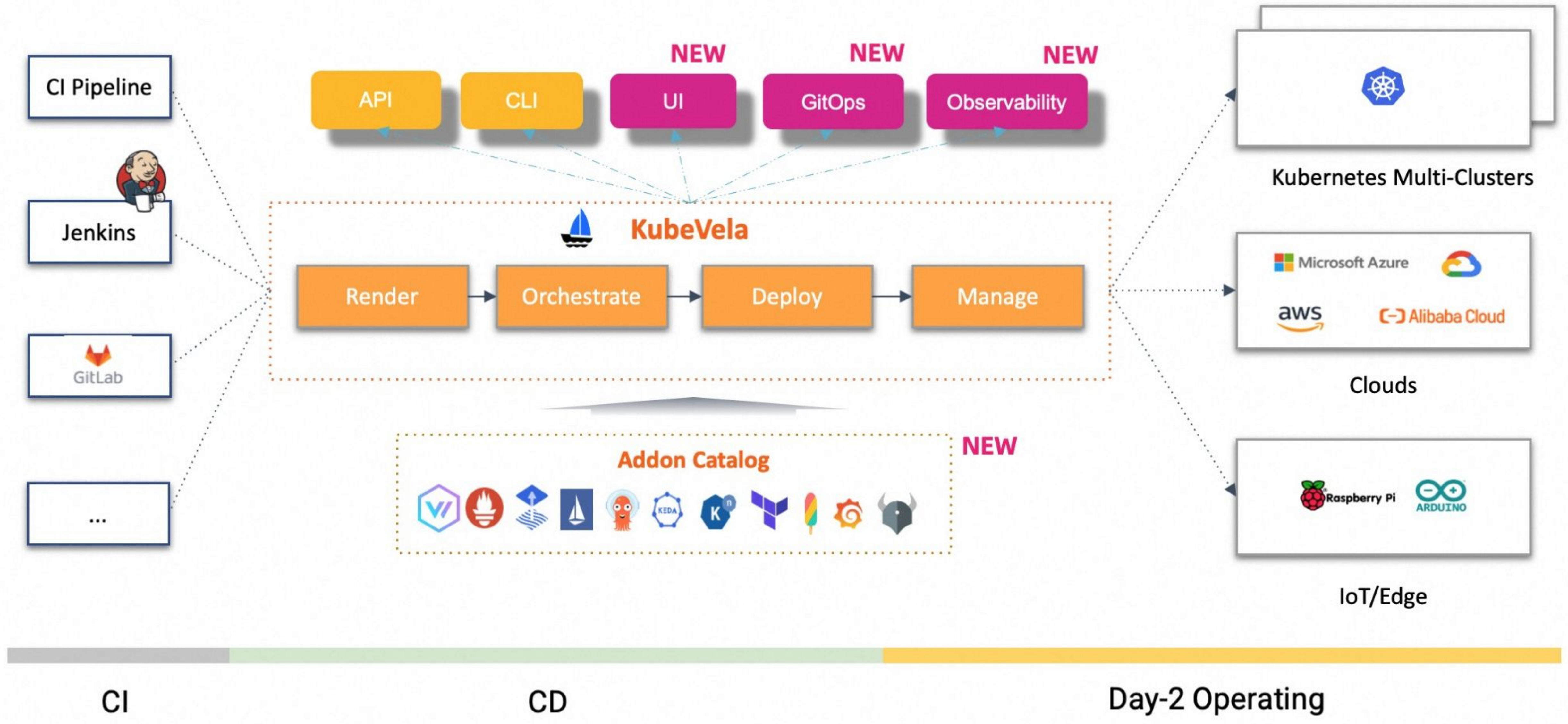
KubeVela

KubeVela

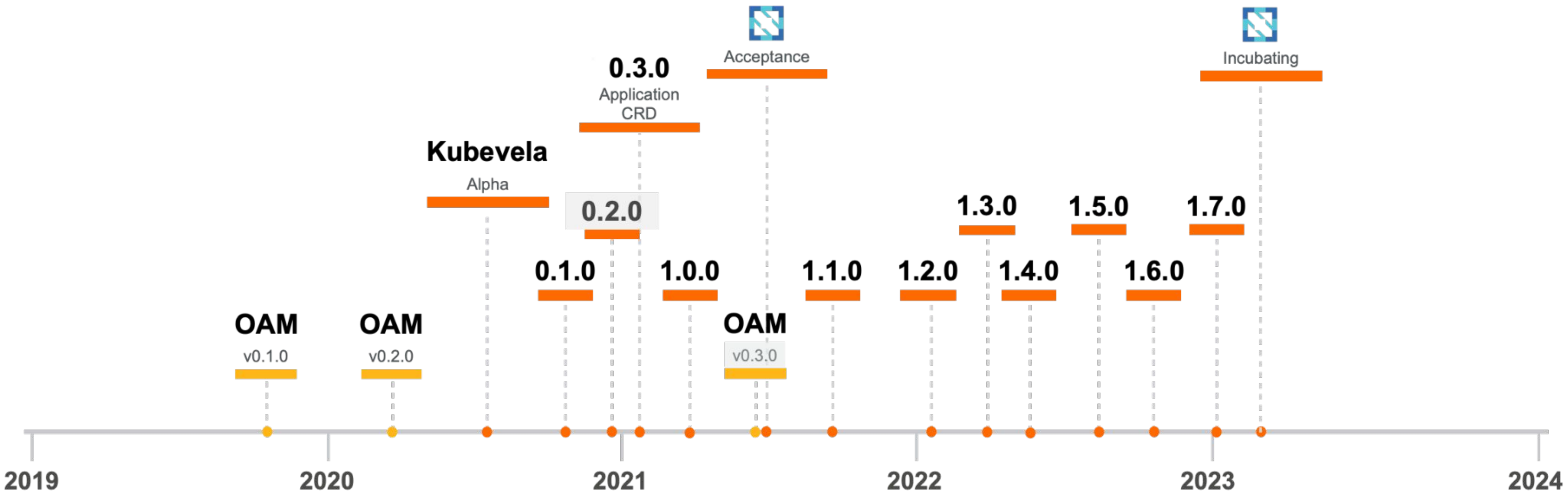
- K8s OAM runtime
- CNCF Incubating project
- Multi-tenant
- Multi-cluster
- Extensible
  - Custom definitions
  - Addons

https://kubevela.io

KubeVela


CI Pipeline
Jenkins
GitLab
...
NEW
NEW
NEW
API
CLI
UI
GitOps
Observability
KubeVela
Render
Orchestrate
Deploy
Manage
Addon Catalog
NEW
Kubernetes Multi-Clusters
Microsoft Azure
aws
Alibaba Cloud
Clouds
Raspberry Pi
ARDUINO
IoT/Edge
CI
CD
Day-2 Operating

Project timeline



Anatomy of an application

- Application is the top-level entity
- Application components are associated with the different microservices
- Traits modify the behaviour of the components

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: nginx-app
spec:
  components:
    - name: nginx
      type: webservice
      properties:
        image: nginx:1.20.0
        ports:
        - port: 80
          expose: true
      traits:
        - type: gateway
          properties:
            http:
              "/": 80
```

Application components

- Typically associated with a microservice
  - But could be: configuration, resources, or even infra…
- Default types
  - webservice
  - worker
  - daemon
  - cron-task
  - task
- Each type defines their own parameters

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: nginx-app
spec:
  components:
    - name: nginx
      type: webservice
      properties:
        image: nginx:1.20.0
        ports:
        - port: 80
          expose: true
```

Traits

- Modifies or augment the component functionality
  - Adding extra elements linked with the deployment
    - Ingresses, secrets, etc.
  - Patching the underlying deployment
    - Request quotas, default labels, extra annotations, etc.
  - Add sidecars
    - Export logs

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
  name: nginx-app
spec:
  components:
    - type: webservice
      properties:
        ...
      traits:
        - type: gateway
          properties:
            http:
              "/": 80
```

Policies

- Similar in concept to traits, but are applied to the whole application
    - Global configuration
    - Multi-cluster deployment configuration
    - Health checks
    - Integration with third party applications (e.g., monitoring)

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
 name: example-app-policy
spec:
 components:
   - name: hello-world-server
     type: webservice
     properties:
       ...
 policies:
   - name: health-policy-demo
     type: health
     properties:
       probeInterval: 5
       probeTimeout: 10
```

Workflows

- Define how the application should be deployed
- Composed of WorkflowSteps
- Supports basic if-then-else logic
- Use case
    - Component dependency
    - Preload data before processing
    - Interaction with other subsystems upon deployment
    - And much more

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
 name: app-with-workflow
spec:
 components:
 - name: express-server
   type: webservice
   properties:
     ...
   traits:
   ...
 workflow:
   steps:
     - name: express-server
       type: apply-component
       properties:
         component: express-server
```

Custom definitions

- Easy definition of custom definitions of any type
  - Components
  - Traits
  - Policies
  - WorkflowStep
- Defined with CUE
  - Context information
- Can replace simple operators
- Adapt OAM to our use cases

```
apiVersion: core.oam.dev/v1beta1
kind: TraitDefinition
metadata:
 name: myscaler
spec:
 workloadRefPath: spec.workloadRef
 schematic:
   cue:
     template: |
       outputs: scaler: {
         apiVersion: "core.oam.dev/v1alpha2"
         kind:       "ManualScalerTrait"
         spec: {
           replicaCount: parameter.replicas
         }
       }
       parameter: {
         //+short=r
         //+usage=Replicas of the workload
         replicas: *1 | int
       }
```

Installing KubeVela

Installation with kind

- Create a kind cluster

```
kind create cluster --config=installation/basic_cluster/kind_basic_cluster_config.yaml --name=kubevela
kubectl --context kind-kubevela wait --for=condition=Ready nodes --all --timeout=600s
kubectl --context kind-kubevela apply -f
https://raw.githubusercontent.com/kubernetes/ingress-nginx/main/deploy/static/provider/kind/deploy.yaml
```

- Install KubeVela

```
helm repo add kubevela https://charts.kubevela.net/core
helm repo update
helm install --create-namespace -n vela-system kubevela kubevela/vela-core --wait
```

How to get started with OAM

doc/**01.install_kubevela.md**

https://github.com/napptive/kubecon-23-oam-kubevela-tutorial

Vela CLI

Vela CLI

- Tool to facilitate the interaction with OAM entities

```
curl -fsSl https://kubevela.net/script/install.sh | bash
vela comp
```

- You can still use kubectl to retrieve the information

```
kubectl --context kind-kubevela -n vela-system get componentdefinitions.core.oam.dev
```

Basic operations

How to get started with OAM

doc/**02.deploy_basic_app.md**

https://github.com/napptive/kubecon-23-oam-kubevela-tutorial

Deploying the first application

```
vela env init kubecon --namespace kubecon


vela up -f scenarios/basic_app/nginx-app.yml
```

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
 name: nginx-app
spec:
 components:
   - name: nginx
     type: webservice
     properties:
       image: nginx:1.20.0
       ports:
       - port: 80
         expose: true
     traits:
       - type: gateway
         properties:
           http:
             "/": 80
```

Checking application state

- Check the application status

```
vela status nginx-app -n kubecon
kubectl --context kind-kubevela -n kubecon get app
```

- Access the logs

```
vela logs nginx-app -n kubecon
```

- Deleting the app

```
vela delete nginx-app
kubectl --context kind-kubevela -n kubecon delete app nginx-app
```

Deploying multi-component applications

doc/**03.deploy_complex_app.md**

https://github.com/napptive/kubecon-23-oam-kubevela-tutorial

Multi-component applications

```
$ vela up -f scenarios/complex_app/wordpress.yml

$ kubectl -n kubecon get all -l app.oam.dev/name=my-wordpress

NAME                               READY   STATUS    RESTARTS          AGE
pod/mysql-77f7db94b7-j9s4n         1/1     Running   0                 5m
pod/wordpress-7b4b47ff64-97rw9     1/1     Running   1 (3m39s ago)     5m

NAME                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/wordpress    ClusterIP   10.43.239.80   <none>        8080/TCP   5m
service/mysql        ClusterIP   10.43.61.129   <none>        3306/TCP   5m

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mysql          1/1     1            1           5m
deployment.apps/wordpress      1/1     1            1           5m

NAME                                   DESIRED   CURRENT   READY   AGE
replicaset.apps/mysql-77f7db94b7       1         1         1       5m
replicaset.apps/wordpress-7b4b47ff64   1         1         1       5m
```

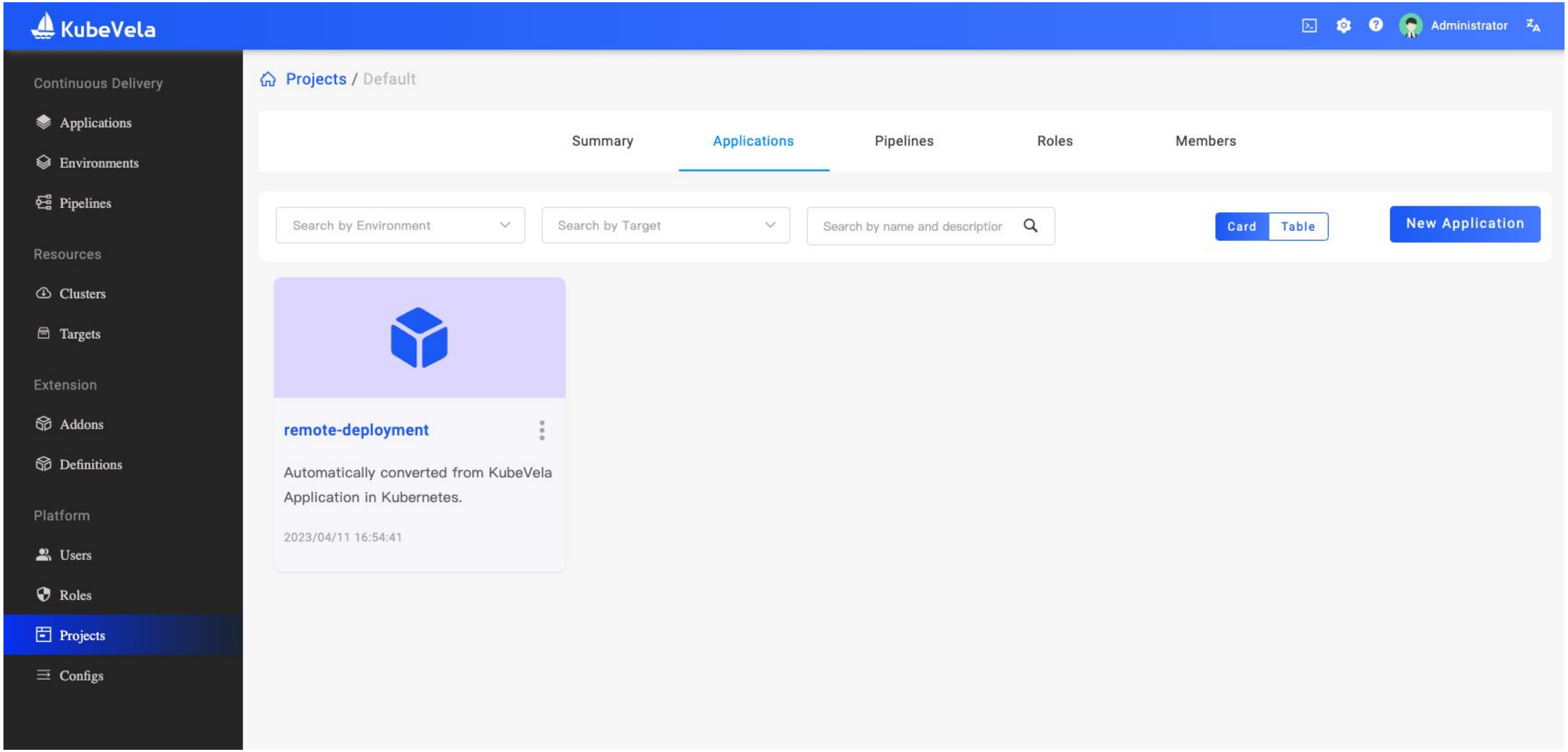
Defining component dependencies

- Option 1
  - Use the dependsOn clause on the components
  - But it is too coupled with the component definition

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
 name: my-wordpress-dep
 annotations:
   version: v1.0.0
spec:
 components:
 - name: wordpress
   type: webservice
   dependsOn:
   - mysql
   properties:
     image:  bitnami/wordpress:latest
     ...
 - name: mysql
   type: webservice
   properties:
     image: mysql:8.0.32
     ...
```

VelaUX

VelaUX



remote-deployment

Automatically converted from KubeVela Application in Kubernetes.

2023/04/11 16:54:41

Installing VelaUX

doc/**04.install_velaux.md**

https://github.com/napptive/kubecon-23-oam-kubevela-tutorial

VelaUX installation

- Provided as an addon

```
vela addon enable velaux
```

- Open the dashboard

```
Initialized admin username and password: admin / VelaUX12345

To open the dashboard directly by port-forward:

   vela port-forward -n vela-system addon-velaux 9082:80

Select "local | velaux | velaux" from the prompt.
```

KubeCon | CloudNativeCon
Europe 2023

Application workflows

Application workflows

doc/**05.app_workflows.md**

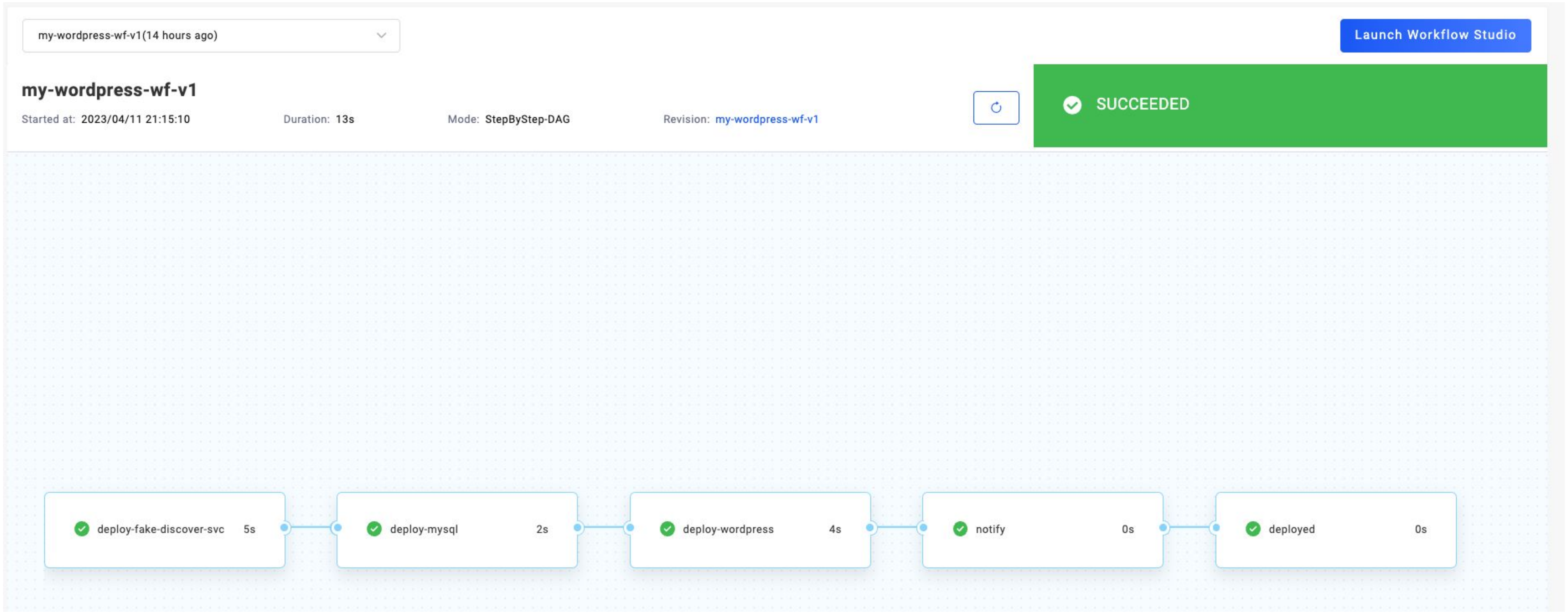
https://github.com/napptive/kubecon-23-oam-kubevela-tutorial

Defining component dependencies

- Option 2
  - Define a deployment workflow
  - Manage not only dependencies but intermediate processes
    - Pre-loading data
    - Send requests to other systems
    - Slack notifications
    - And much more

```
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
 name: my-wordpress-wf
spec:
 components:
 - name: wordpress ...
 - name: mysql ...
 - name: discoversvc ...
 workflow:
   steps:
     - name: deploy-fake-discover-svc
       type: apply-component
       properties:
         component: discoversvc
     - name: deploy-mysql
       type: apply-component
       properties:
         component: mysql
     - name: deploy-wordpress
       type: apply-component
       properties:
         component: wordpress
     ...
```

Application workflows



my-wordpress-wf-v1(14 hours ago)

my-wordpress-wf-v1

Started at: 2023/04/11 21:15:10 Duration: 13s Mode: StepByStep-DAG Revision: my-wordpress-wf-v1

SUCCEEDED

deploy-fake-discover-svc	5s
deploy-mysql	2s
deploy-wordpress	4s
notify	0s
deployed	0s

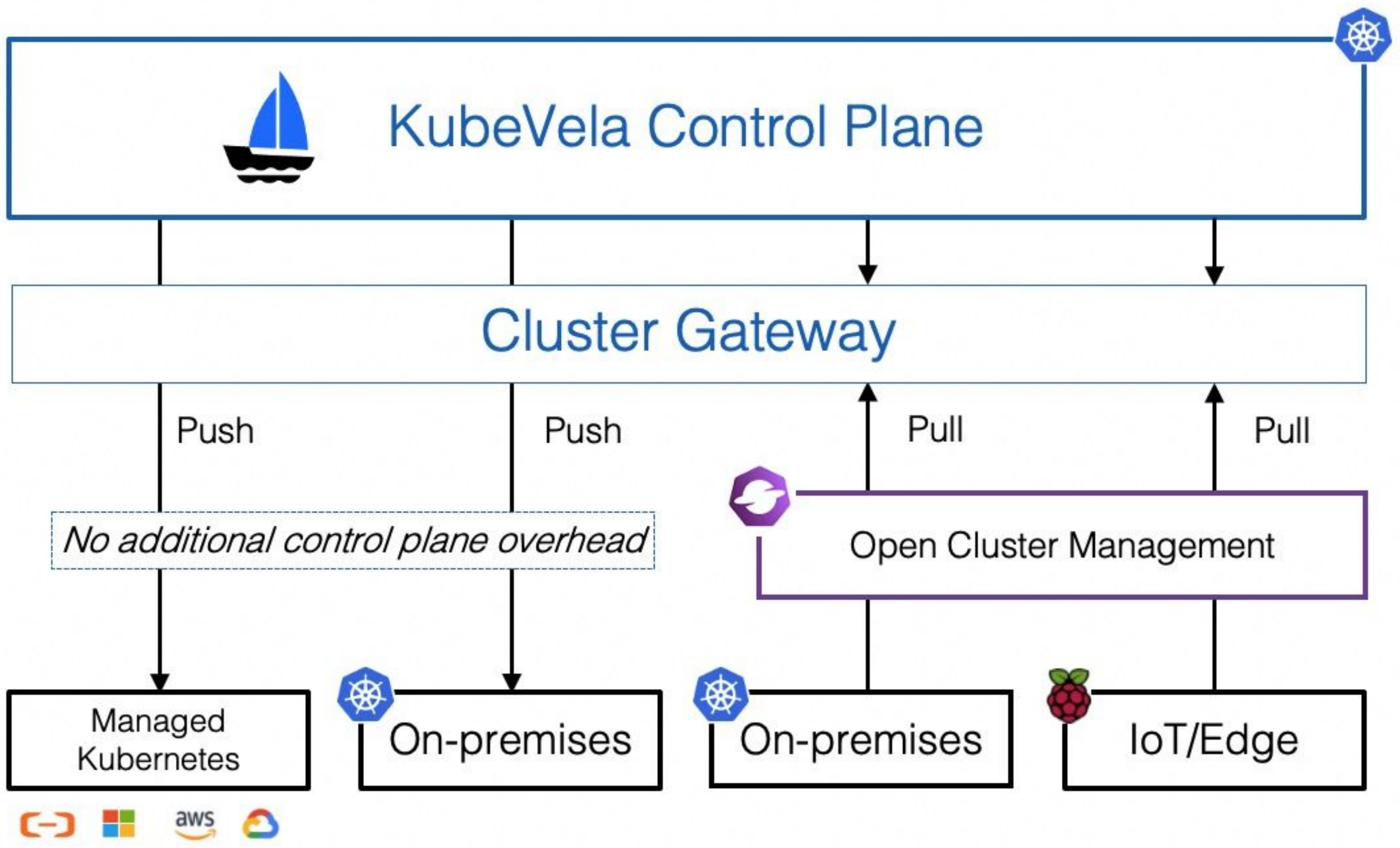
Multi-cluster

Multi-cluster deployments

- Oriented to continuous delivery use cases
- Set the target clusters in the application
- When combined with workflows and addons we can provision clusters and managed services before deployment

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
 name: remote-deployment
spec:
 components:
   # app components
 policies:
   - name: target-clusters
     type: topology
     properties:
       clusters: ["cluster-1", "cluster-2"]
```

Multi-cluster deployments

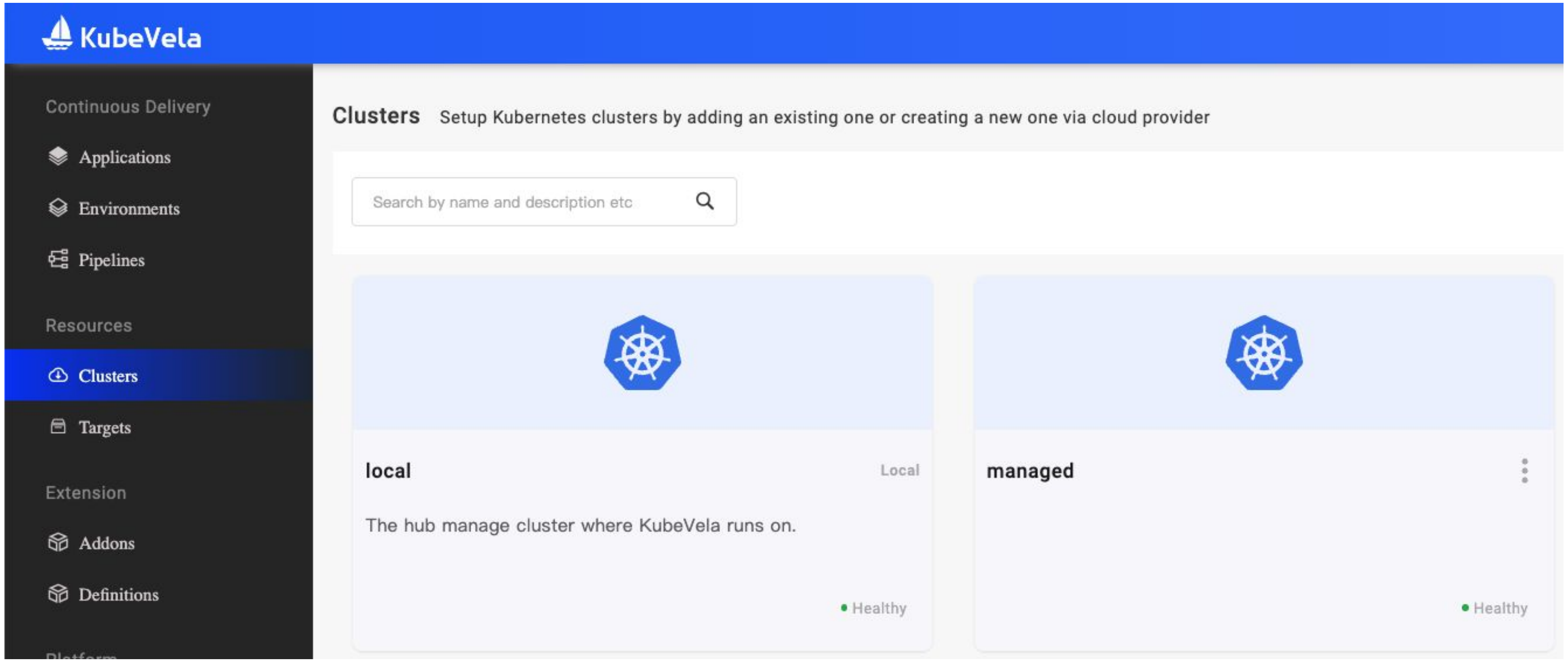


Multi-cluster deployments

doc/**06.multicluster.md**

https://github.com/napptive/kubecon-23-oam-kubevela-tutorial

Multi-cluster install



Clusters Setup Kubernetes clusters by adding an existing one or creating a new one via cloud provider

local Local

The hub manage cluster where KubeVela runs on.

● Healthy

managed

● Healthy

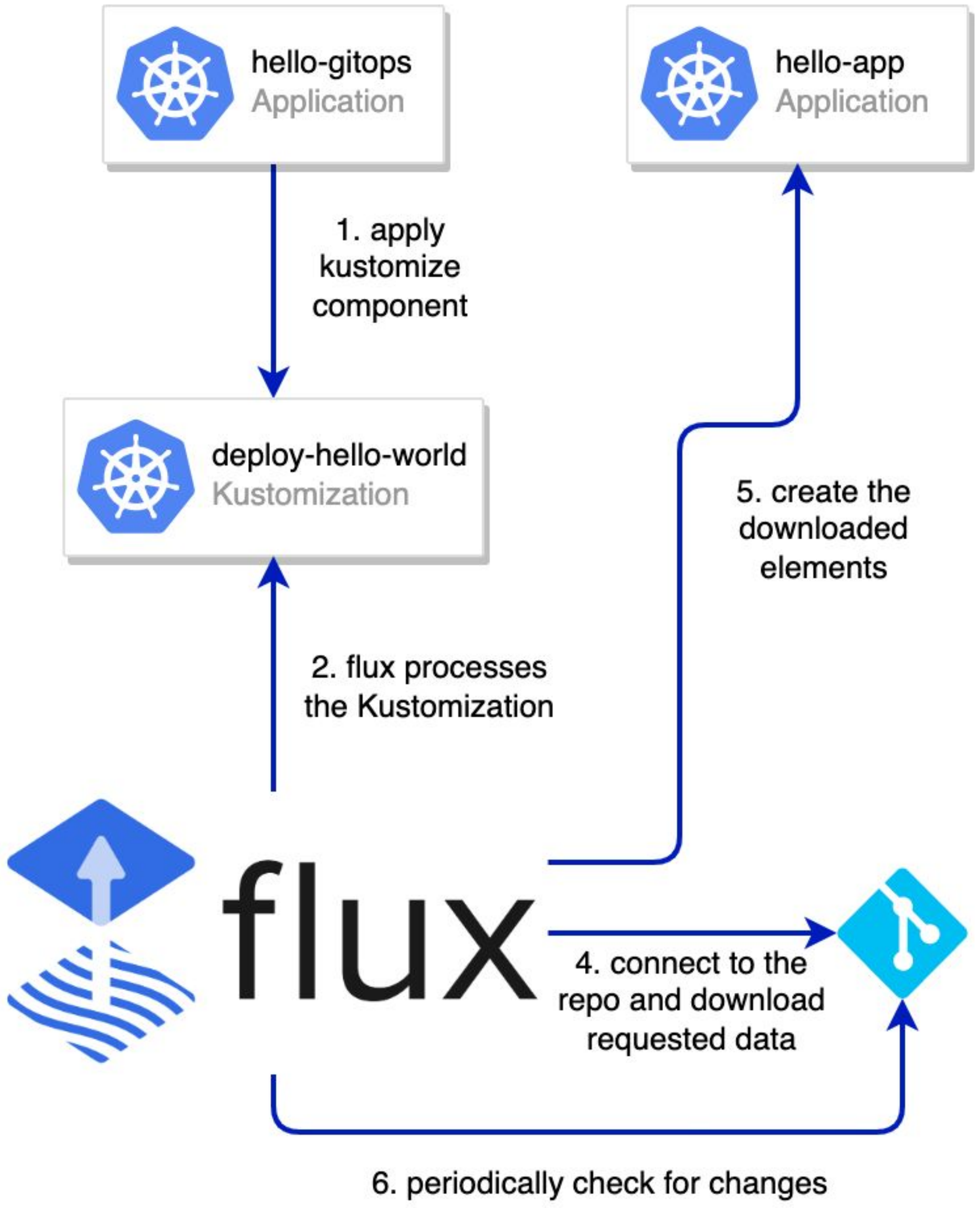
GitOps with FluxCD

Enabling FluxCD

- The FluxCD addon provides a set of new component definitions
  - Kustomize
  - Helm
- Helm components are a good way to progressively migrate apps to OAM
- Kustomize components enables GitOps deployments

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
 name: helm-redis
spec:
 components:
   - name: redis
     type: helm
     properties:
       repoType: "helm"
       url: "https://charts.bitnami.com/bitnami"
       chart: "redis"
       version: "16.8.5"
       values:
         master:
           persistence:
             size: 16Gi
         replica:
           persistence:
             size: 16Gi
```

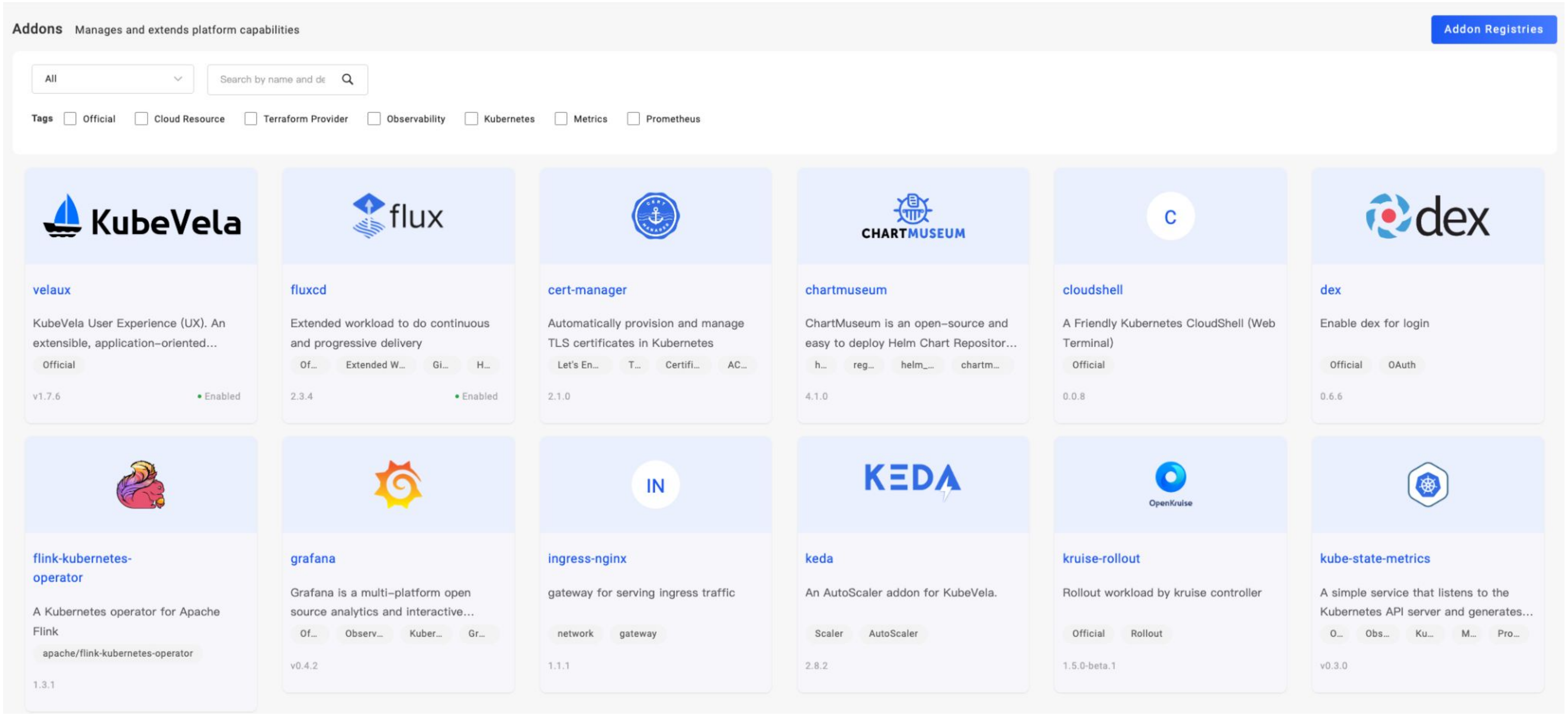
GitOps hello world


hello-gitops
Application
hello-app
Application
1. apply kustomize component
deploy-hello-world
Kustomization
5. create the downloaded elements
2. flux processes the Kustomization
flux
4. connect to the repo and download requested data
6. periodically check for changes

```yaml
apiVersion: core.oam.dev/v1beta1
kind: Application
metadata:
 name: hello-gitops
spec:
 components:
 - name: deploy-hello-world
   type: kustomize
   properties:
     targetNamespace: kubecon
     repoType: git
     # replace it with your own repo url to explore further
     url: https://github.com/napptive/kubecon-23-oam-kubevela-tutorial
     # replace it with your git secret if it's a private repo
     # secretRef: git-secret
     # the pull interval time, set to 30s for demo purposes
     pullInterval: 30s
     git:
       # the branch name
       branch: initial_commit
     # the path to sync
     path: ./scenarios/gitops/target
```

Extra content

Addon catalog



Addons Manages and extends platform capabilities

Addon Registries

All

Search by name and de

Tags Official Cloud Resource Terraform Provider Observability Kubernetes Metrics Prometheus

velaux
KubeVela User Experience (UX). An extensible, application-oriented...
Official
v1.7.6 ● Enabled

fluxcd
Extended workload to do continuous and progressive delivery
Of... Extended W... Gi... H...
2.3.4 ● Enabled

cert-manager
Automatically provision and manage TLS certificates in Kubernetes
Let's En... T... Certifi... AC...
2.1.0

chartmuseum
ChartMuseum is an open-source and easy to deploy Helm Chart Repositor...
h... reg... helm_... chartm...
4.1.0

cloudshell
A Friendly Kubernetes CloudShell (Web Terminal)
Official
0.0.8

dex
Enable dex for login
Official OAuth
0.6.6

flink-kubernetes-operator
A Kubernetes operator for Apache Flink
apache/flink-kubernetes-operator
1.3.1

grafana
Grafana is a multi-platform open source analytics and interactive...
Of... Observ... Kuber... Gr...
v0.4.2

ingress-nginx
gateway for serving ingress traffic
network gateway
1.1.1

keda
An AutoScaler addon for KubeVela.
Scaler AutoScaler
2.8.2

kruise-rollout
Rollout workload by kruise controller
Official Rollout
1.5.0-beta.1

kube-state-metrics
A simple service that listens to the Kubernetes API server and generates...
O... Obs... Ku... M... Pro...
v0.3.0

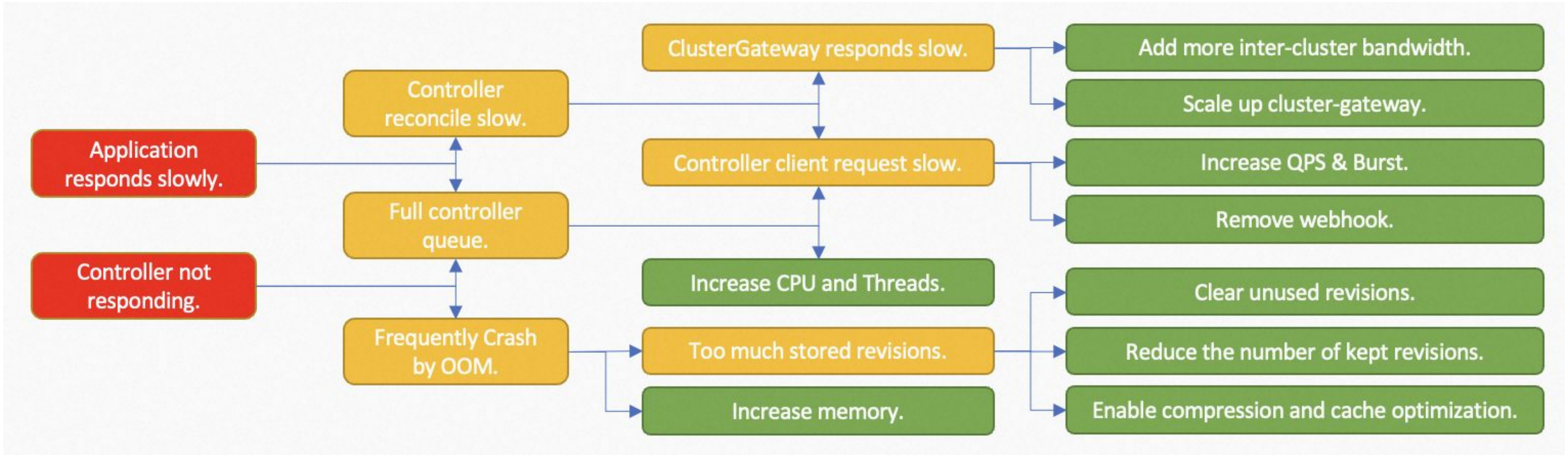
Multi-tenant deployments

- KubeVela can be integrated with the underlying K8s RBAC

```
helm upgrade --install kubevela kubevela/vela-core
             --create-namespace -n vela-system
             --set authentication.enabled=true --set authentication.withUser=true
             --wait
```

- Applications will be deployed impersonating the requesting user to guarantee that role limitations are enforced

KubeVela performance troubleshooting


Application responds slowly.
Controller not responding.
Controller reconcile slow.
Full controller queue.
Frequently Crash by OOM.
ClusterGateway responds slow.
Controller client request slow.
Increase CPU and Threads.
Too much stored revisions.
Increase memory.
Add more inter-cluster bandwidth.
Scale up cluster-gateway.
Increase QPS & Burst.
Remove webhook.
Clear unused revisions.
Reduce the number of kept revisions.
Enable compression and cache optimization.

Further reading

- OAM
  - https://oam.dev/
  - https://github.com/oam-dev/spec
- KubeVela
  - https://kubevela.io/
  - https://github.com/kubevela/kubevela
  - #kubevela on the CNCF Slack
  - BiWeekly meetings English & Chinese
- Napptive
  - https://napptive.com
  - https://docs.napptive.com

KubeCon | CloudNativeCon
Europe 2023

Thanks!

KubeCon | CloudNativeCon
Europe 2023

Please scan the QR Code above
to leave feedback on this session

KubeCon | CloudNativeCon

Europe 2023