Navigating the app delivery landscape while solving everyday problems

Lei Zhang, Alois Reitbauer

Session abstract

KubeCon CloudNativeCon North America 2020 Virtual

Recently there was a lot of buzz around the CNCF landscape getting overly complex. There are more and more options to choose from which is good to see a growing ecosystem. However, this can sometimes also become overwhelming. In this session we walk through major challenges that people are facing and show how they can be solved with projects available today. We will look into scenarios from defining your application all the way to running it in production. Whether you are just about to start running applications on Kubernetes or want to improve your skills. This session will help you to identify what to consider when building app platforms, share good practices, compare different approaches and give you an interactive and quick tour through the CNCF landscape with a focus on delivering and running applications. You will also get some demo code samples to try everything hands on and have a reference to get started with your own projects.

Speakers

Lei Zhang

Staff Engineer - OAM/KubeVela, Alibaba

Lei is a co-maintainer of Kubernetes community, and co-chair of CNCF App Delivery SIG. Lei is co-leading engineering effort in Alibaba including Kubernetes and large-scale cluster management system. Before it, Lei worked for Hyper_ and Microsoft Research (MSR). Lei is a popular speaker... Read More →

Alois Reitbauer

Chief Technical Strategist, Dynatrace

Alois is an executive member of the technical staff at Dynatrace. He has been building monitoring and application management solutions for more than 15 years. Alois has successfully brought multiple products to market that are used by the biggest companies on the planet. His current... Read More →

about SIG App-Delivery

CNCF App Delivery SIG

The Application Delivery SIG focuses on delivering cloud native applications which involves multiple phases including building, deploying, managing, and operating. Additionally, the SIG produces supporting material and best practices for end-users, and provide guidance and coordination for CNCF projects working within the SIG's scope.

See our full charter here: https://github.com/cncf/toc/blob/master/sigs/app-delivery.md

Chairs

Alois Reitbauer, Bryan Liles and Lei Zhang.

Community

- Slack channel: #sig-app-delivery in CNCF workspace - https://cloud-native.slack.com/messages/CL3SL0CP5
- Mailing list: https://lists.cncf.io/g/cncf-sig-app-delivery/topics

Meetings

We have meetings every 1st and 3rd Wednesday of the month at 8:00 a.m. PST.

- Agenda and Notes: https://docs.google.com/document/d/1OykvqvhSG4AxEdmDMXilrupsX2n1qCSJUWwTc3I7AOs/edit#
- Zoom Meeting: https://zoom.us/j/7276783015
- Recordings of previous meetings: https://www.youtube.com/playlist?list=PLj6h78yzYM2OHd1Ht3jiZuucWzvouAAci

Mission

Consistent with the CNCF SIG definition, the mission of CNCF SIG App Delivery is:

- To collaborate on areas related to developing, distributing, deploying, managing and operating secure cloud-native applications with the target of delivering application in manner of cloud native.
- To develop informational resources including guides, tutorials and white papers to give the community an understanding of best practices, trade-offs, and value-adds regarding to application delivery.
- To identify suitable projects and gaps in the landscape, periodically updating the TOC with suggested actions in a structured manner. This includes helping the TOC with assessments and due diligence of prospective new projects.

Areas considered in Scope

SIG Application Delivery focuses on the following topics of the lifecycle of cloud-native applications:

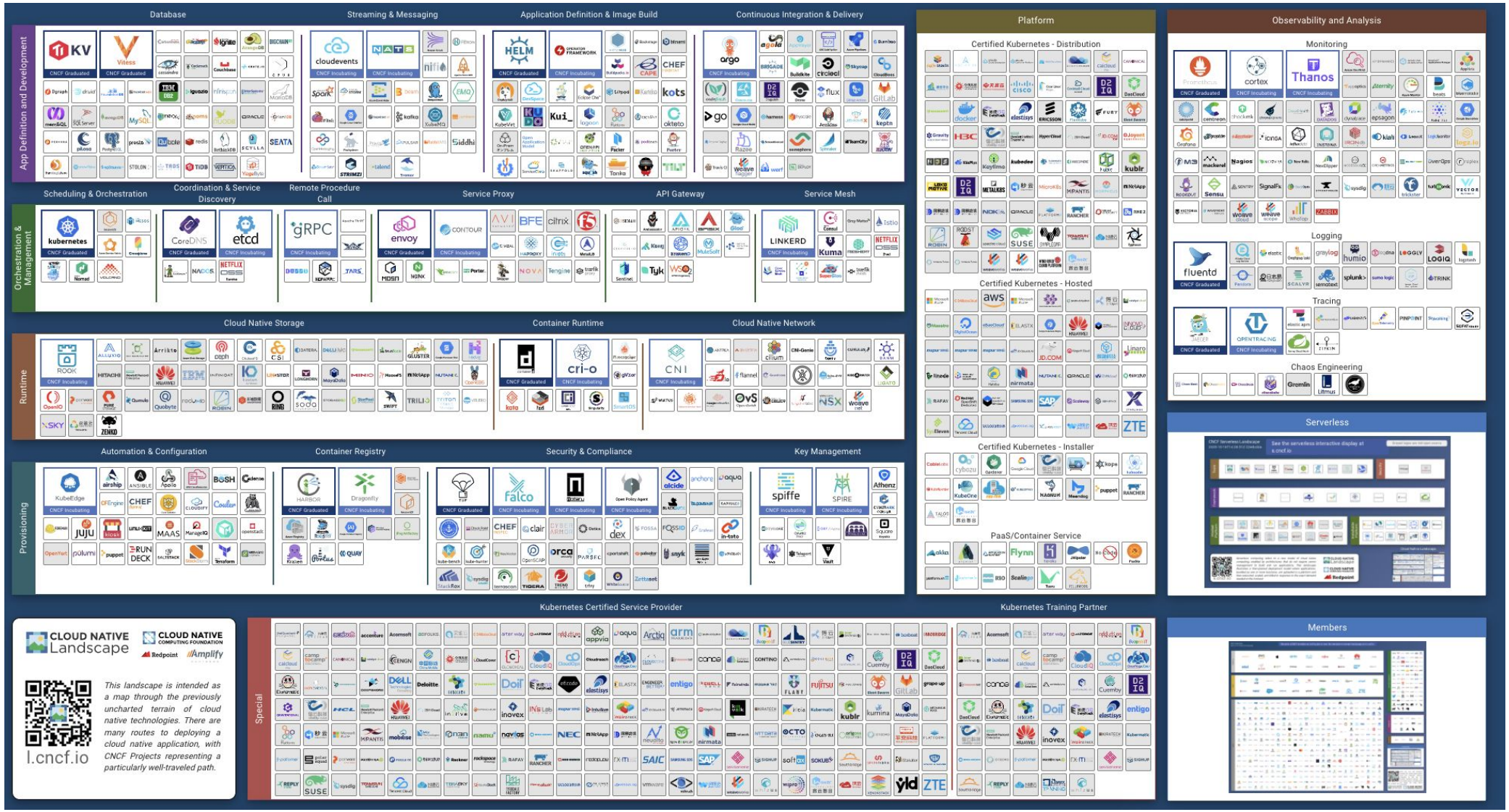
- Application definition, including description, parameter and configuration
- Guidance and practice for application design and development
- Application bundling and deployment
- Package management
- Application delivery workflow and strategy
- Configuration source driven workflow
- Release management

The SIG will work on developing best practices, fostering collaboration between related projects, working on improving tool interoperability as well as proposing new initiatives and projects when blank spots in the current landscape are identified.

https://github.com/cncf/sig-app-delivery

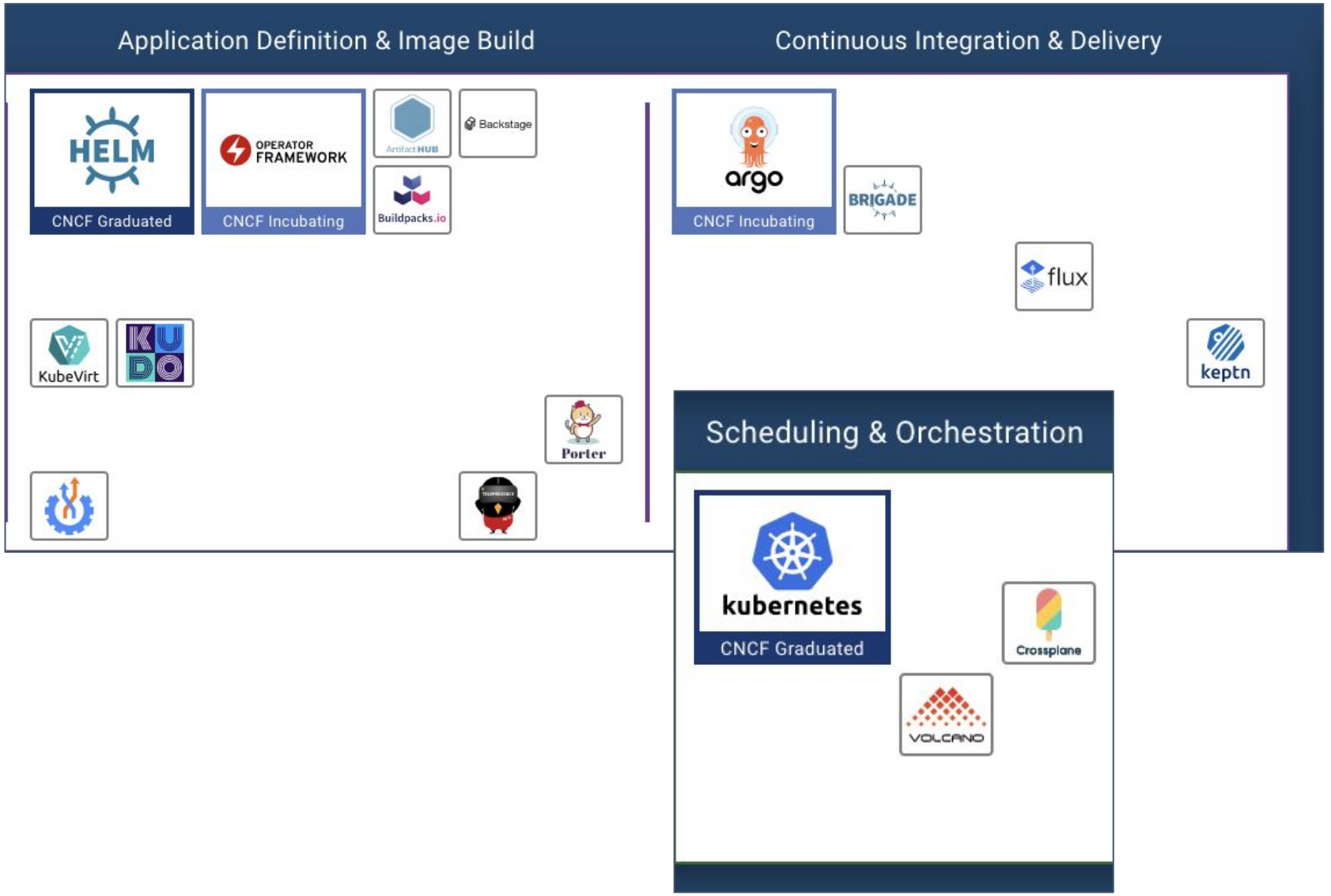
The CNCF landscape ...

KubeCon CloudNativeCon North America 2020 Virtual



There is even a puzzle ...

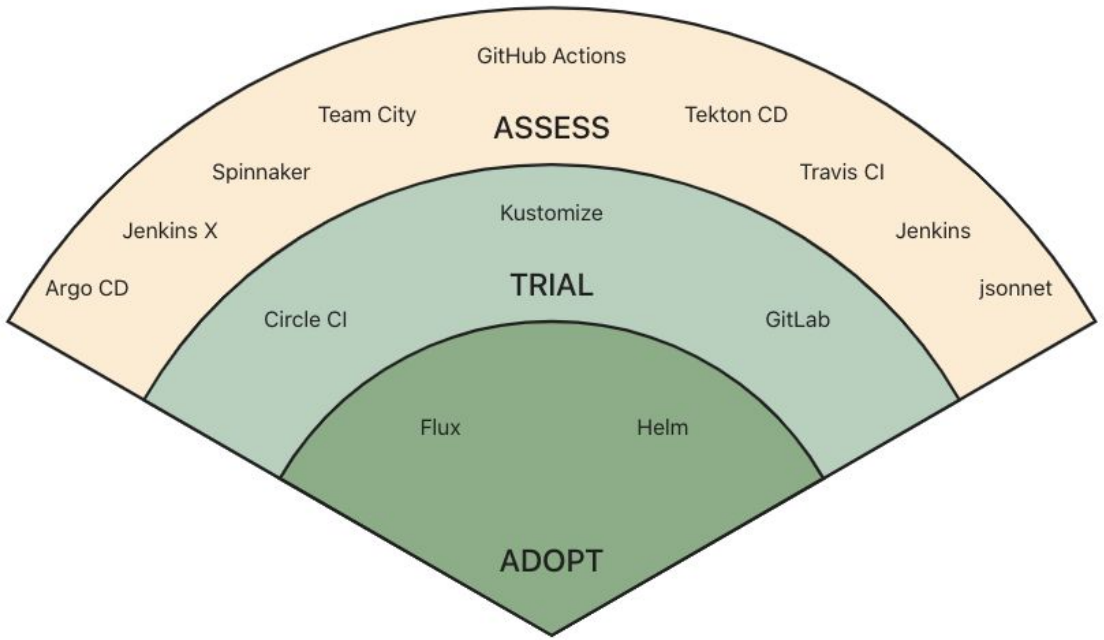
… let us look at CNCF app-delivery only



The CNCF tech radar

KubeCon CloudNativeCon North America 2020 Virtual

Continuous Delivery, June 2020



The Themes

1. Publicly available solutions are combined with in-house tools

Many end users had tried up to 10 options and settled on adopting 2-4. Several large enterprise companies have built their own continuous delivery tools and open sourced components, including LunarWay's release-manager, Box's kube-applier, and stackset-controller from Zalando. The public cloud managed solutions on the CNCF landscape were not suggested by any of the end users, which may reflect the options available a few years ago.

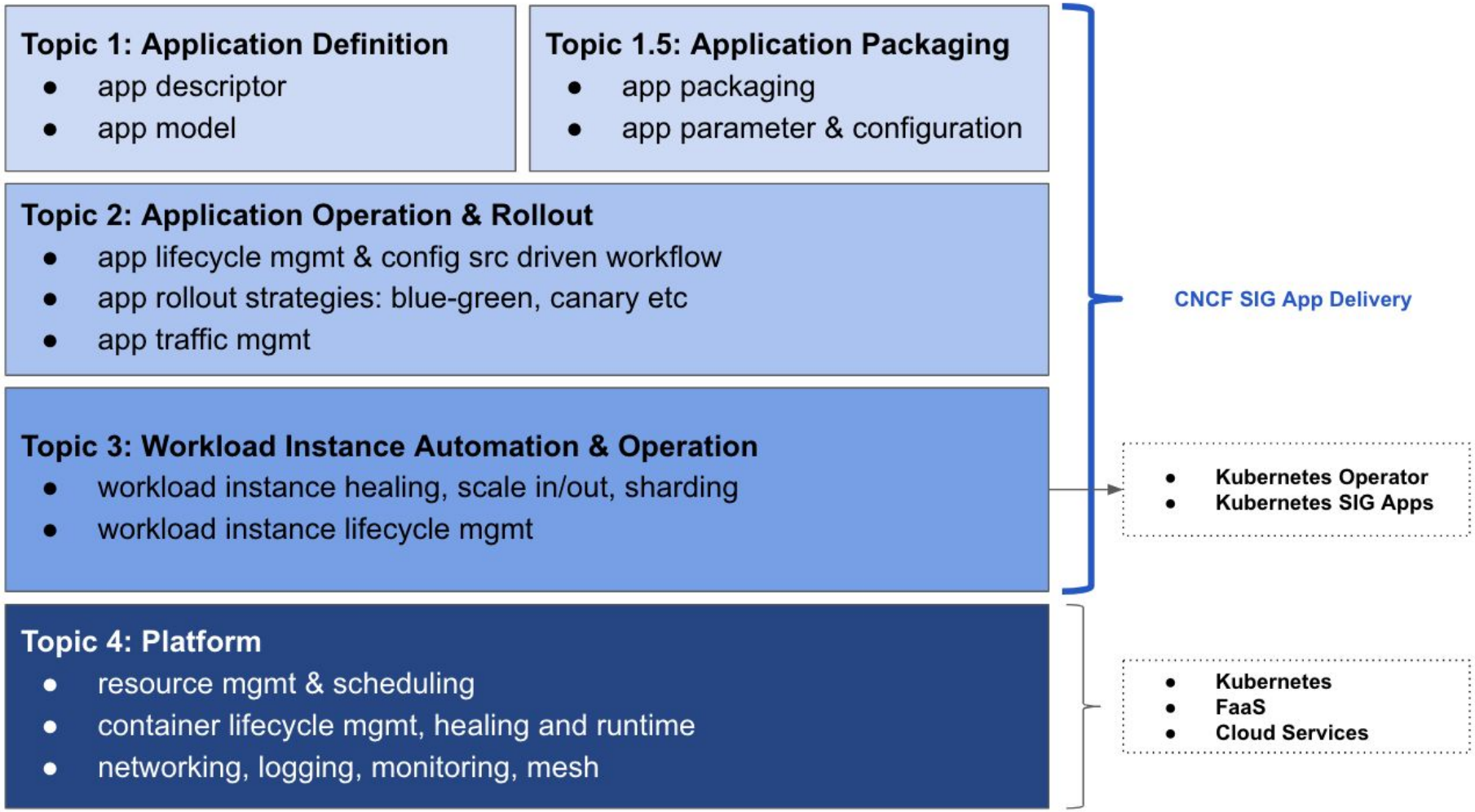
2. Helm is more than packaging applications

While Helm has not positioned itself as a Continuous Delivery tool (it's the Kubernetes package manager first), it's widely used and adopted as a component in different CD scenarios.

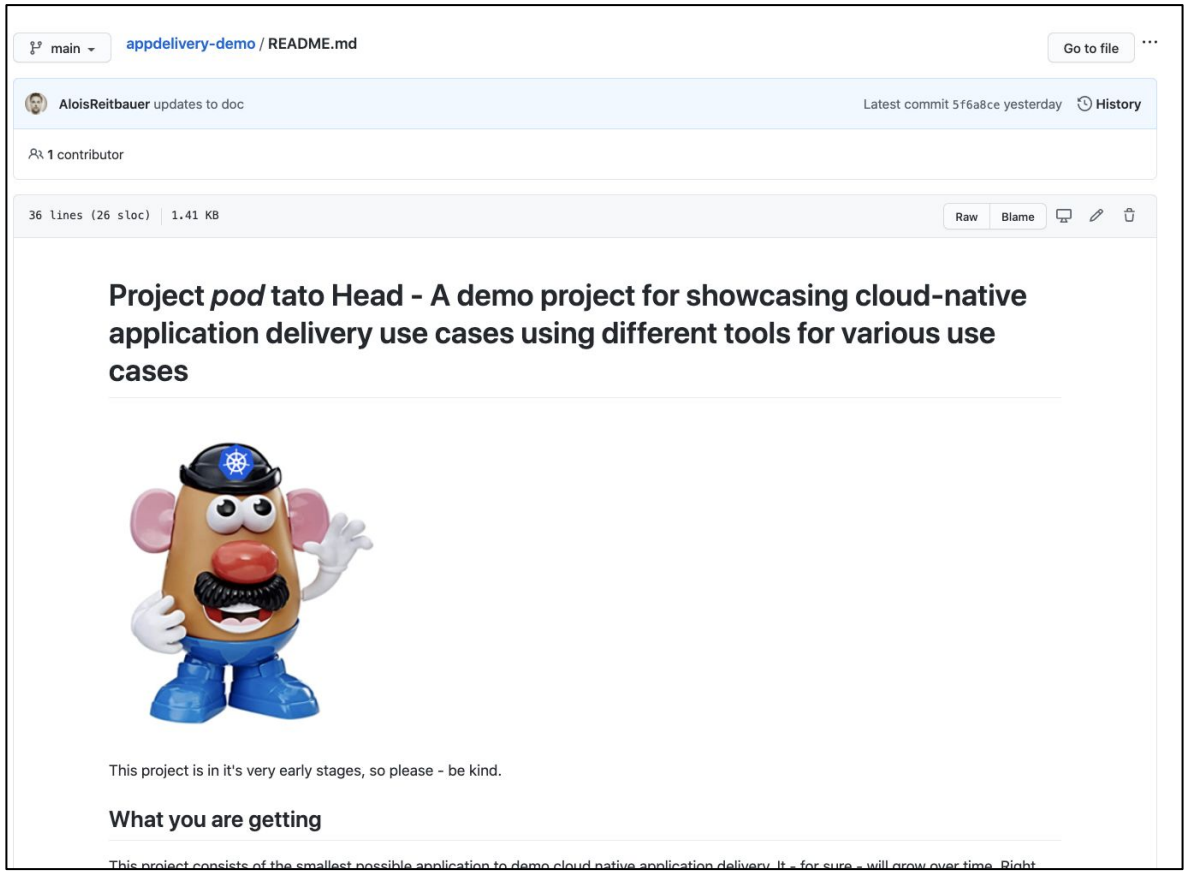
3. Jenkins is still broadly deployed, while cloud native-first options emerge.

Jenkins and its ecosystem tools (Jenkins X, Jenkins Blue Ocean) are widely evaluated and used. However, several end users stated Jenkins is primarily used for existing deployments, while new applications have migrated to other solutions. Hence end users who are choosing a new CD solution should assess Jenkins alongside tools that support modern concepts such as GitOps (for example, Flux).

The App Delivery Reference Model



Introducing podTato Head



*pod*tato Head is:

- the simplest cloud-native project
- a way to explore different delivery tools
- a funny name for a GitHub project

https://github.com/cncf/podtato-head

Let us start super, super simple

Using Manifest yaml files

The most basic way to deploy an application is by using Kubernetes manifest files

```yaml
apiVersion: v1
kind: Service
metadata:
  name: helloservice
  namespace: demospace
spec:
  selector:
    app: helloservice
  ports:
  - name: http
    port: 9000
    protocol: TCP
    targetPort: 9000
  type: LoadBalancer
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloservice
  namespace: demospace
spec:
  selector:
    matchLabels:
      app: helloservice
  template:
    metadata:
      labels:
        app: helloservice
    spec:
      terminationGracePeriodSeconds: 5
      containers:
      - name: server
        image: aloisreitbauer/hello-server:v0.1.1
        imagePullPolicy: Always
        ports:
        - containerPort: 9000
        env:
        - name: PORT
          value: "9000"
```

What if we want more flexibility

Using Helm templates


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: helloservice
  namespace: {{ .Release.Namespace }}
spec:
  selector:
    matchLabels:
      app: helloservice
  template:
    metadata:
      labels:
        app: helloservice
    spec:
      terminationGracePeriodSeconds: 5
      containers:
      - name: server
        image: {{ .Values.imageName}}
        ports:
        - containerPort: 9000
        env:
        - name: PORT
          value: "9000"
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: helloservice
  namespace: {{ .Release.Namespace }}
spec:
  type: ClusterIP
  selector:
    app: helloservice
  ports:
  - name: http
    port: 9000
    protocol: TCP
    targetPort: 9000
  type: LoadBalancer
```

```yaml
imageName: aloisreitbauer/hello-server:v0.1.2
```

We want changes to be applied automatically

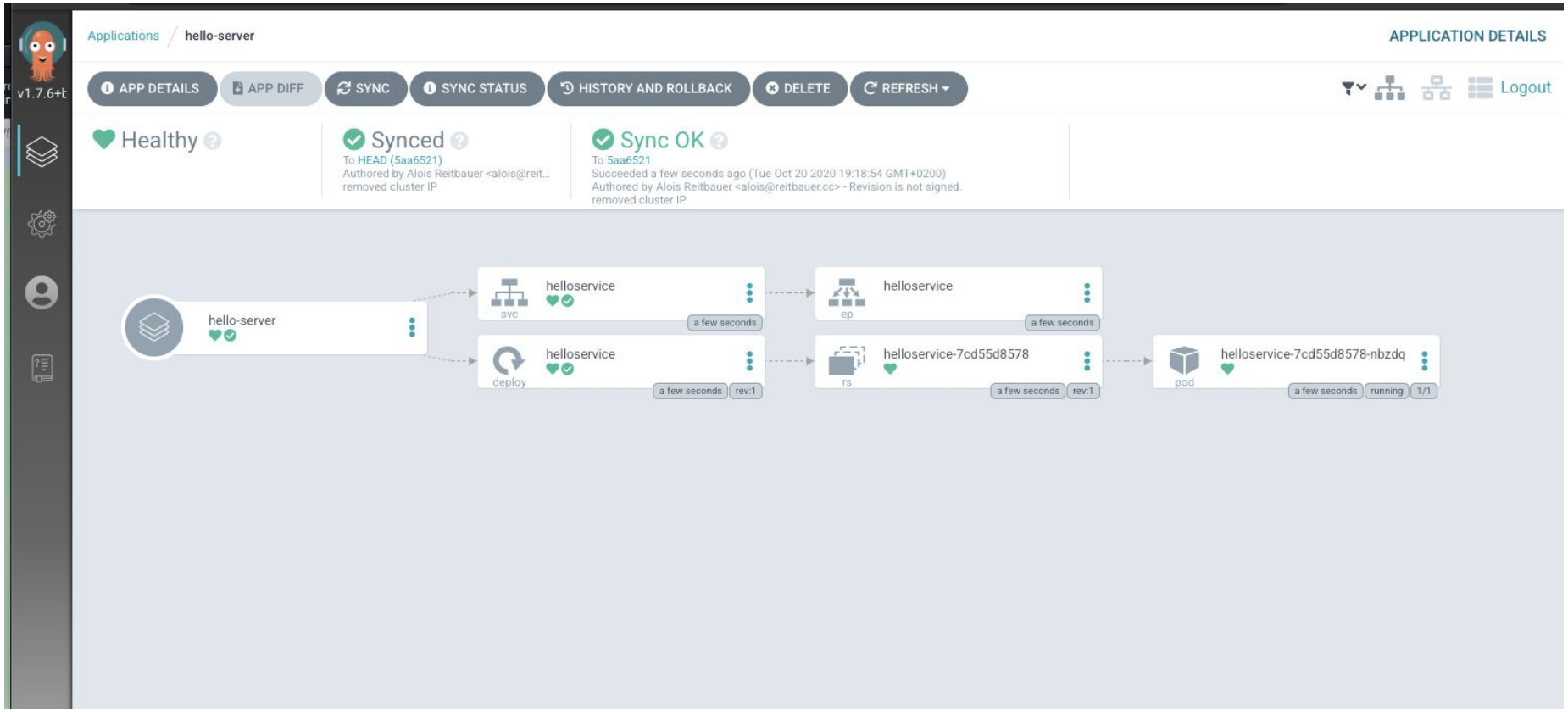
Moving towards GitOps with Flux

KubeCon CloudNativeCon North America 2020 Virtual

flux

We want to see what happens

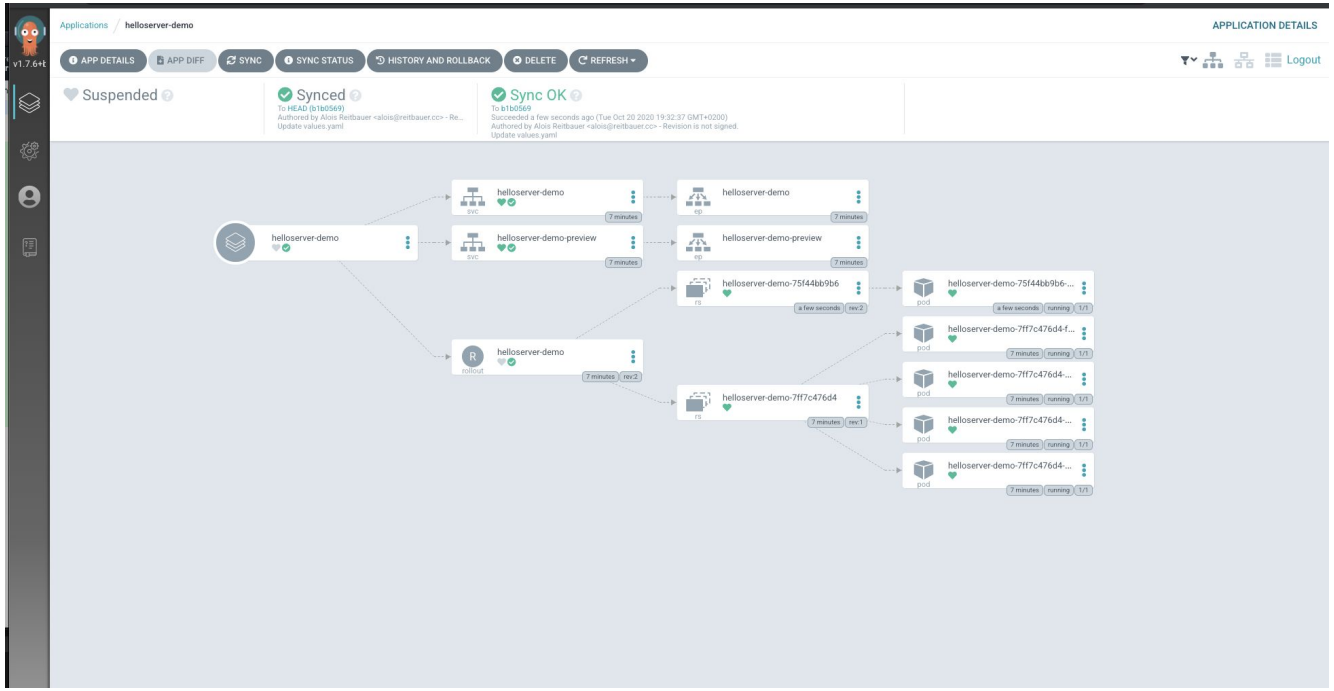
GitOps with ArgoCD



We want to get more control over the rollout

Using Argo Rollouts


```yaml
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: {{ template "helloservice-demo.fullname" . }}
  labels:
    app: {{ template "helloservice-demo.name" . }}
    chart: {{ template "helloservice-demo.chart" . }}ku
    release: {{ .Release.Name }}
    heritage: {{ .Release.Service }}
spec:
  replicas: {{ .Values.replicaCount }}
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: {{ template "helloservice-demo.name" . }}
      release: {{ .Release.Name }}
  strategy:
    canary:
    #blueGreen:
      # activeService: {{ template "helloservice-demo.fullname" . }}
      # previewService: {{ template "helloservice-demo.fullname" . }}-preview
      steps:
      - setWeight: 20
      - pause: {duration: 10}
      - setWeight: 40
      - pause: {duration: 10}
      - setWeight: 60
      - pause: {duration: 10}
      - setWeight: 80
      - pause: {duration: 10}
  template:
    metadata:
      labels:
        app: {{ template "helloservice-demo.name" . }}
        release: {{ .Release.Name }}
    spec:
      containers:
        - name: {{ .Chart.Name }}
          image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
          imagePullPolicy: {{ .Values.image.pullPolicy }}
          ports:
            - name: http
              containerPort: 9000
```



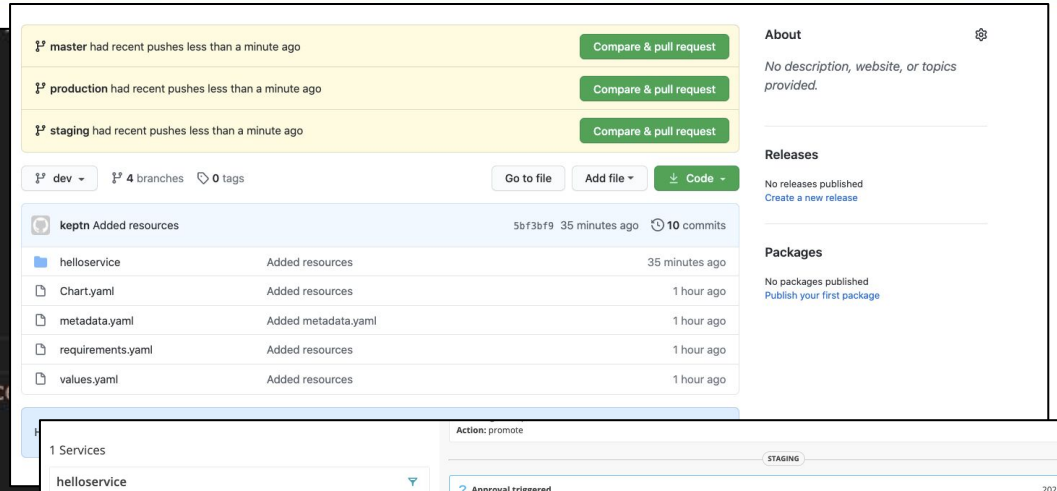

```
NAME                                          KIND        STATUS      AGE    INFO
⟳ helloserver-demo                            Rollout     ॥ Paused    7m11s
├──# revision:2
│  └──⧉ helloserver-demo-75f44bb9b6           ReplicaSet  ✔ Healthy   31s    canary
│     └──□ helloserver-demo-75f44bb9b6-7bspd  Pod         ✔ Running   31s    ready:1/1
└──# revision:1
   └──⧉ helloserver-demo-7ff7c476d4           ReplicaSet  ✔ Healthy   7m11s  stable
      ├──□ helloserver-demo-7ff7c476d4-fgqt6  Pod         ✔ Running   7m11s  ready:1/1
      ├──□ helloserver-demo-7ff7c476d4-kfcg6  Pod         ✔ Running   7m10s  ready:1/1
      ├──□ helloserver-demo-7ff7c476d4-vfns5  Pod         ✔ Running   7m10s  ready:1/1
      └──□ helloserver-demo-7ff7c476d4-whm89  Pod         ✔ Running   7m10s  ready:1/1
```

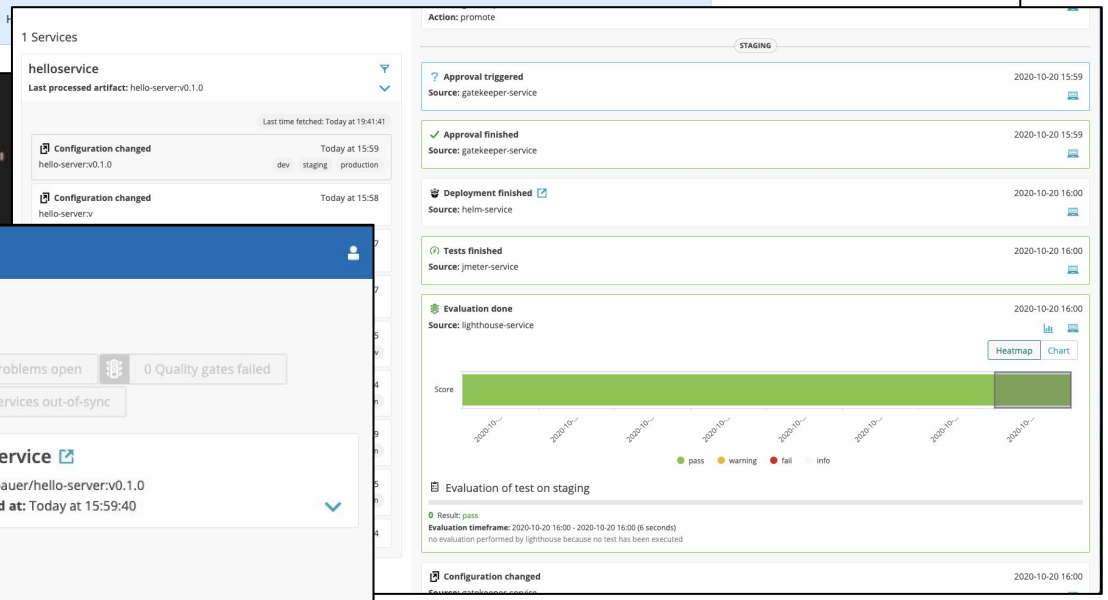
What if I want to orchestrate a multi-stage environment

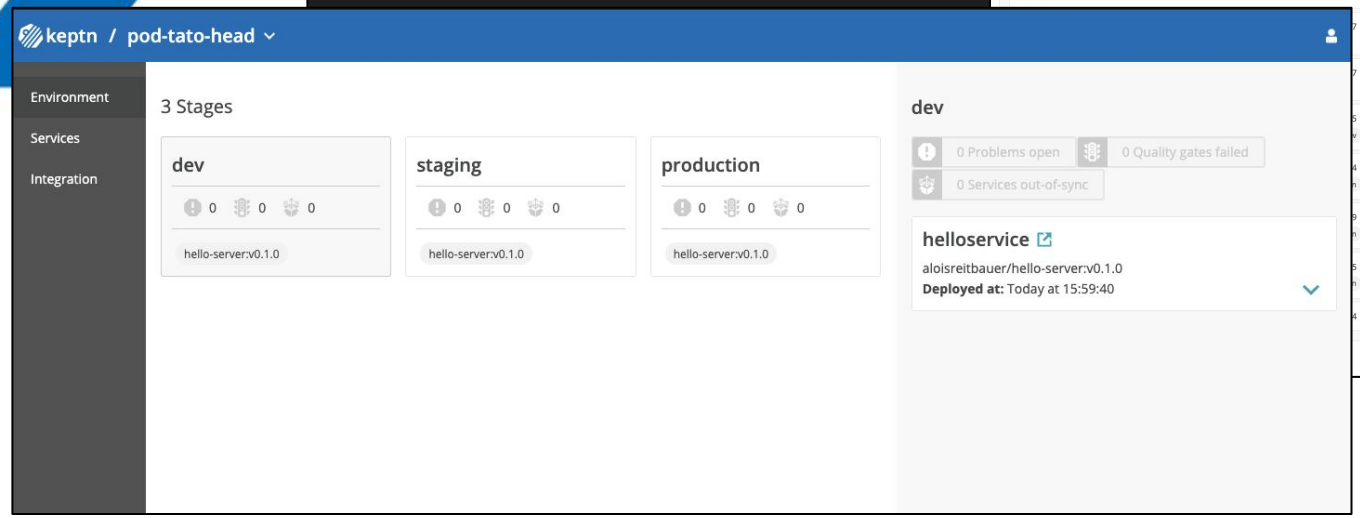
Complex releases with Keptn

KubeCon CloudNativeCon North America 2020 Virtual

```
stages:
  - name: "dev"
    approval_strategy:
      pass: "automatic"
      warning: "automatic"
  - name: "staging"
    approval_strategy:
      pass: "automatic"
      warning: "automatic"
    deployment_strategy: "blue_green_servic
  - name: "production"
    approval_strategy:
      pass: "manual"
      warning: "manual"
    deployment_strategy: "blue_green_service"
```







[Hand over to Harry]

What if I want to ship hiding all the details?

Package as a Helm-Operator


```
operator-sdk init --plugins=helm --domain=helloservice --group=helloservice-demo --version=v1alpha1 --helm-chart ../charts/
```

OPERATOR
FRAMEWORK

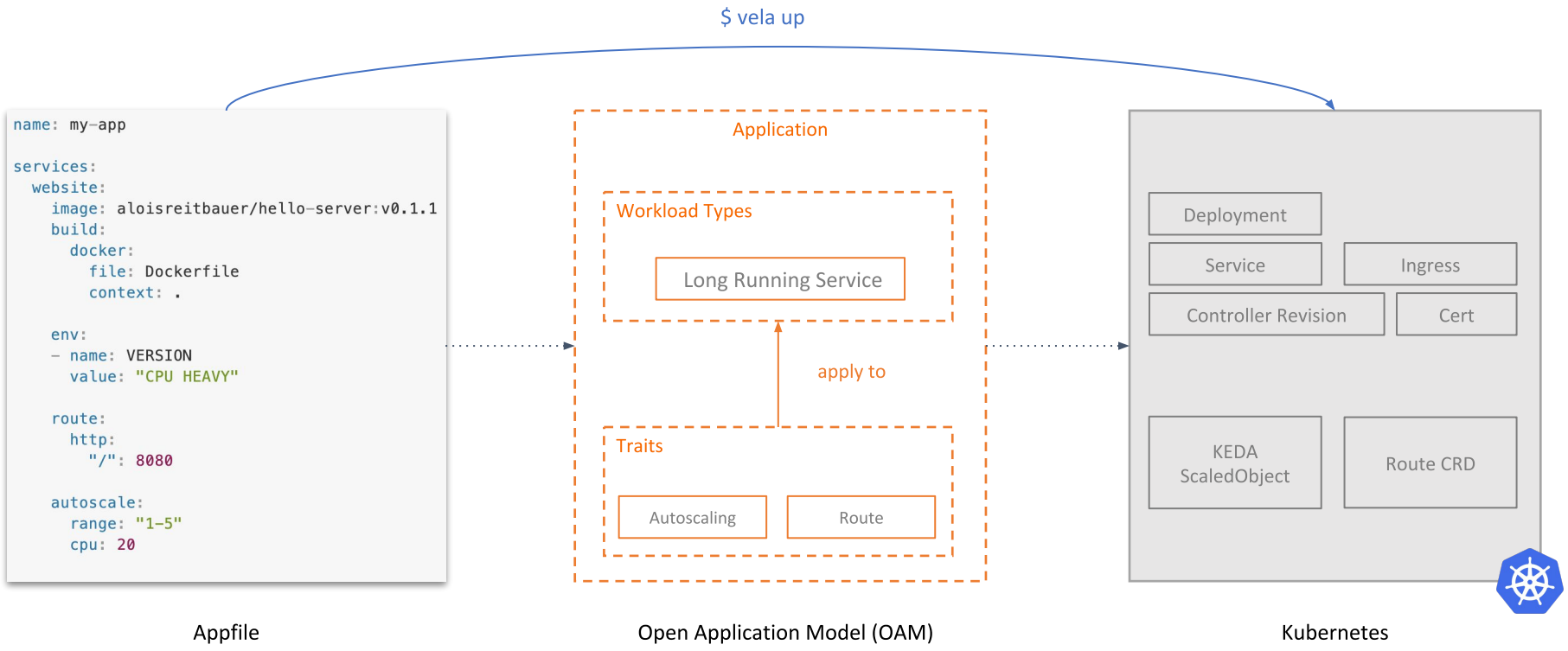
```yaml
apiVersion: helloservice-demo.helloservice/v1alpha1
kind: Helloserver
metadata:
  name: helloserver-sample
spec:
  # Default values copied from <project_dir>/helm-charts/helloserver/values.yaml
  imageName: aloisreitbauer/hello-server:v0.1.1
```

What if I want a Heroku but on Kubernetes?

Ship app using KubeVela

Open App Model | KubeVela

- KubeVela
  - The extensible app platform based on Kubernetes
- Open Application Model (OAM)
  - The model to define applications in developer-centric primitives



Ship app using KubeVela

- KubeVela is **highly extensible**!

Add a new capability!

```yaml
apiVersion: core.oam.dev/v1alpha2
kind: TraitDefinition
metadata:
  name: canaries.flagger.app
spec:
  appliesToWorkloads:
    - podspecworkload.standard.oam.dev
    - deployments.apps
  definitionRef:
    name: canaries.flagger.app
  workloadRefPath: spec.targetRef
  revisionEnabled: true
```

$ kubectl apply -f trait-def.yaml

Done!

```yaml
name: my-app

services:
  website:
    image: aloisreitbauer/hello-server:v0.1.1
    build:
      docker:
        file: Dockerfile
        context: .

    env:
    - name: VERSION
      value: "CPU HEAVY"

    route:
      http:
        "/": 8080

    autoscale:
      range: "1-5"
      cpu: 20

    rollout:
      strategy: canary
      analysis:
        interval: 30s
        threshold: 10
        maxWeight: 50
        stepWeight: 10
        maxReplicas: 4
```

$ vela up

Your newly added capability!

What is next for podTato head？

https://github.com/cncf/podtato-head

Extending **podTato head** for more complex use case:

- multiple services
- stateful workloads
- databases
- cross-stage secrets
- external dependencies
- ……

Add additional tools and use cases

- (Chaos) Testing
- App operations automation

What is next for SIG App Delivery?

- The community: https://github.com/cncf/sig-app-delivery
- We have meetings every 1st and 3rd Wednesday of the month at 8:00 a.m. PST.

"Show me the code, not whitepapers!"

- More real-world demonstrations to answer the question of:
  - *What project X is used for?*
- Series tech blog about:
  - *Deep dive project Y*
  - *How company Z is using project X to solve real-world problems?*
- Other plans:
  - CNCF Radar for application delivery


KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual