Evolving Deep Learning Platform with Knative

Ti Zhou @tizhou86

Deep Learning Platform Introduction

Baidu Deep Learning Platform

Unified Platform for ML/DL

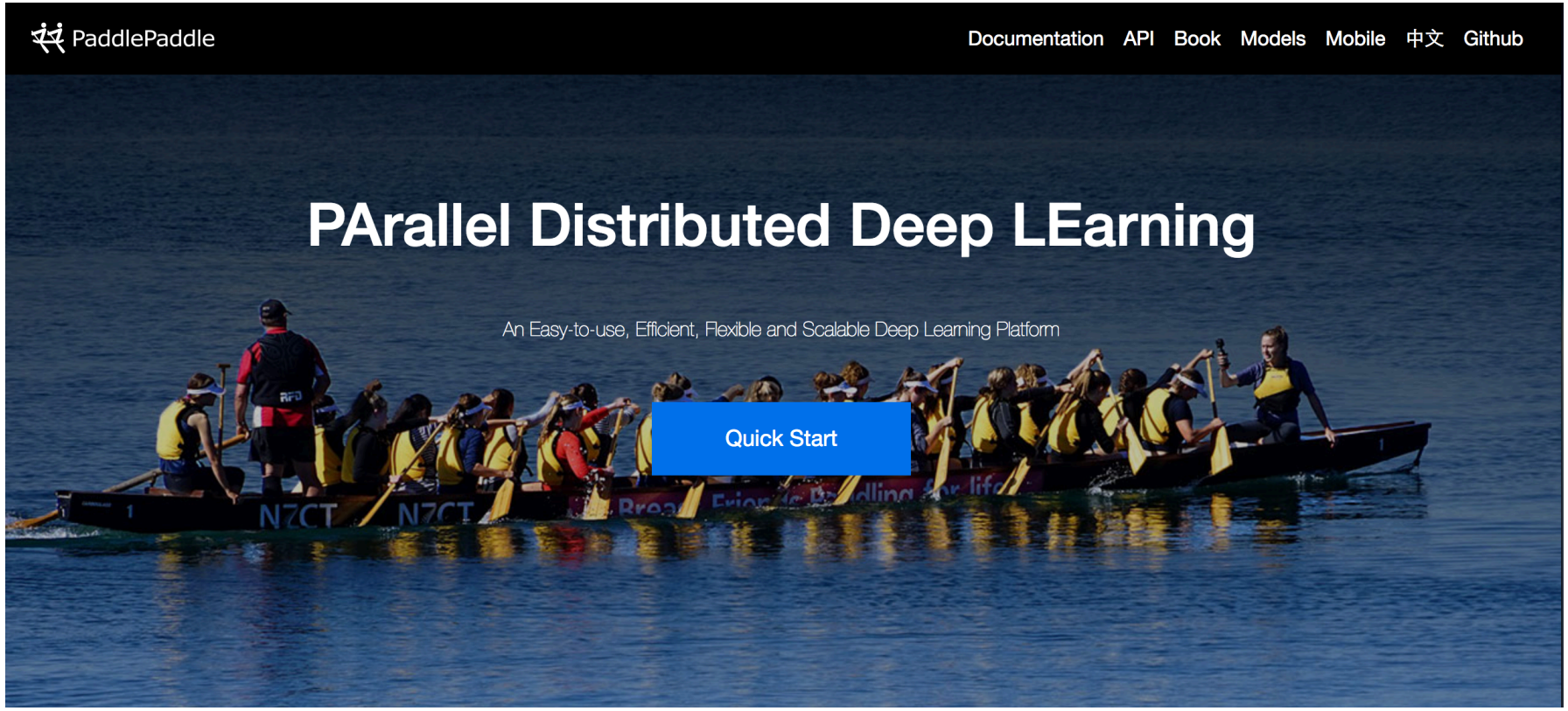
- Data Processing
- Model Training/Eval
- Model Inference

Advantage

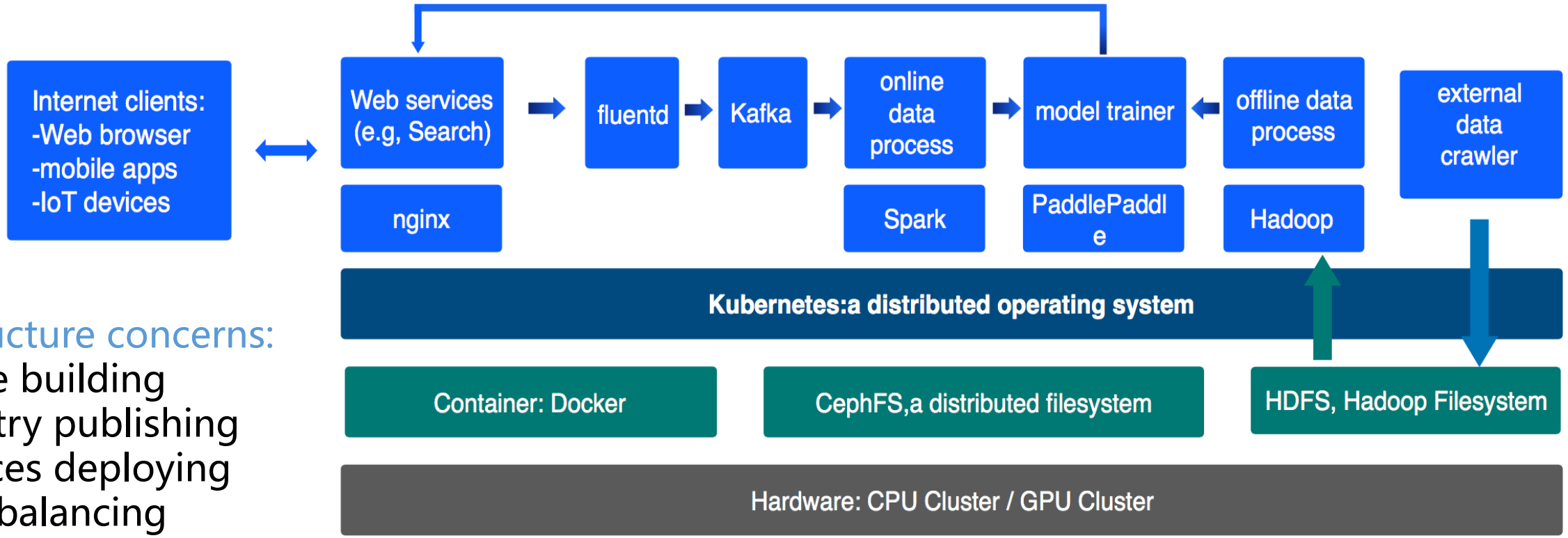
- Resource Utilization
- Model Pipeline
- Multi Tenant/Security

Business Support

- Search
- Ads
- Feeds
- NLP/Visual/Speech



Why DLP on Kubernetes



Infrastructure concerns:

- Image building
- Registry publishing
- Services deploying
- Load balancing
- Logging
- Monitoring
- Scaling

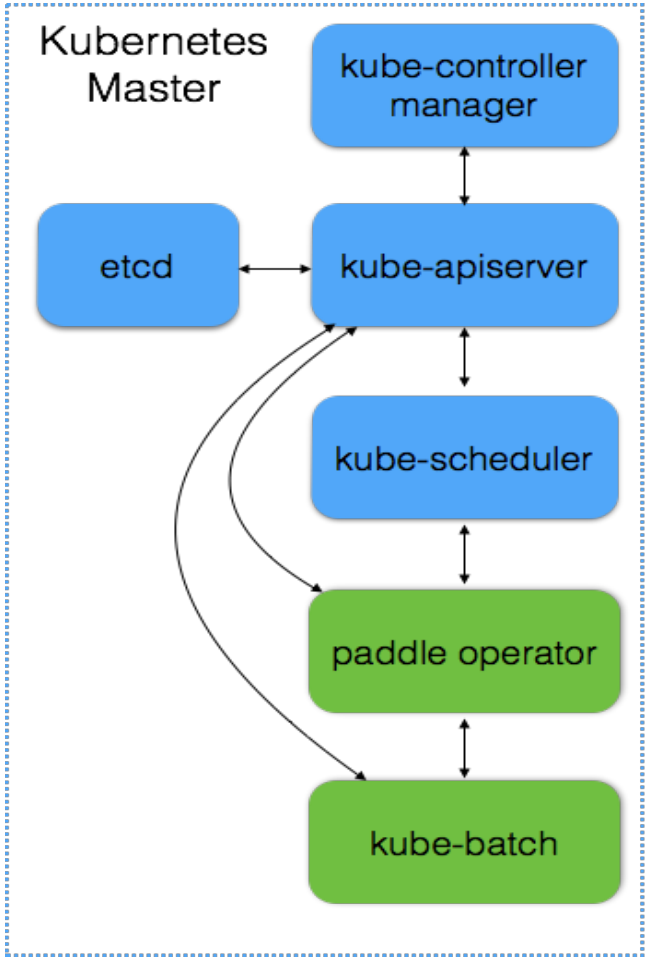
What we did

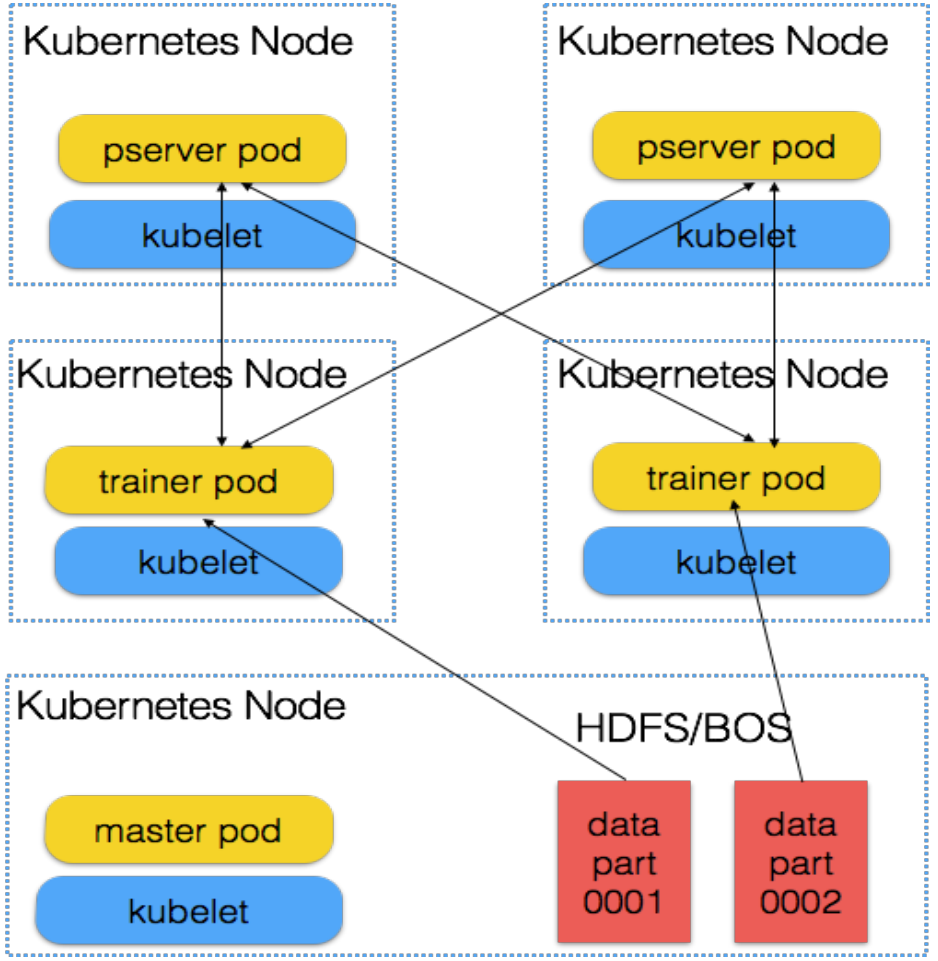
Features:

- Paddle Operator
- Kube-Batch

Abilities:

- Job-level scheduling
- Paddle Job CRD
- Job Fault Tolerant
- Job Auto Scaling





Why Serverless

Why serverless

Real world needs that original Kubernetes service cannot provide

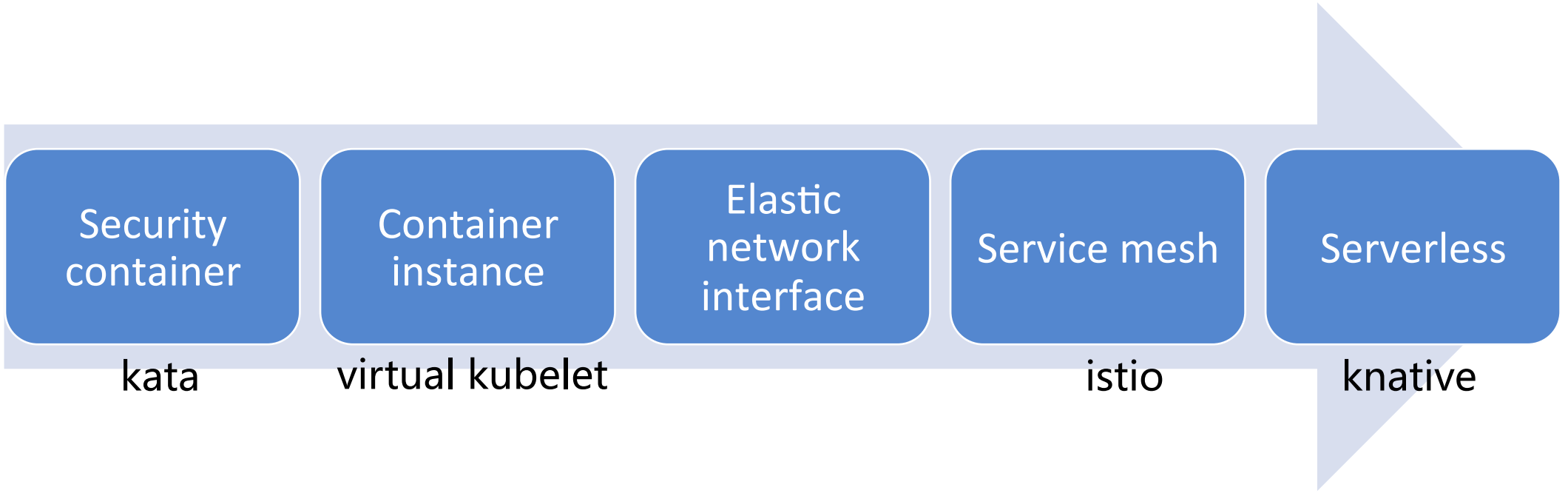
- No operation cost
- No need to pre-evaluate the resources
- Consistent extension
- Pay by usage

Issues that user concerns most

- Network paradigm
- Resource utilization
- Multi tenancy
- Security

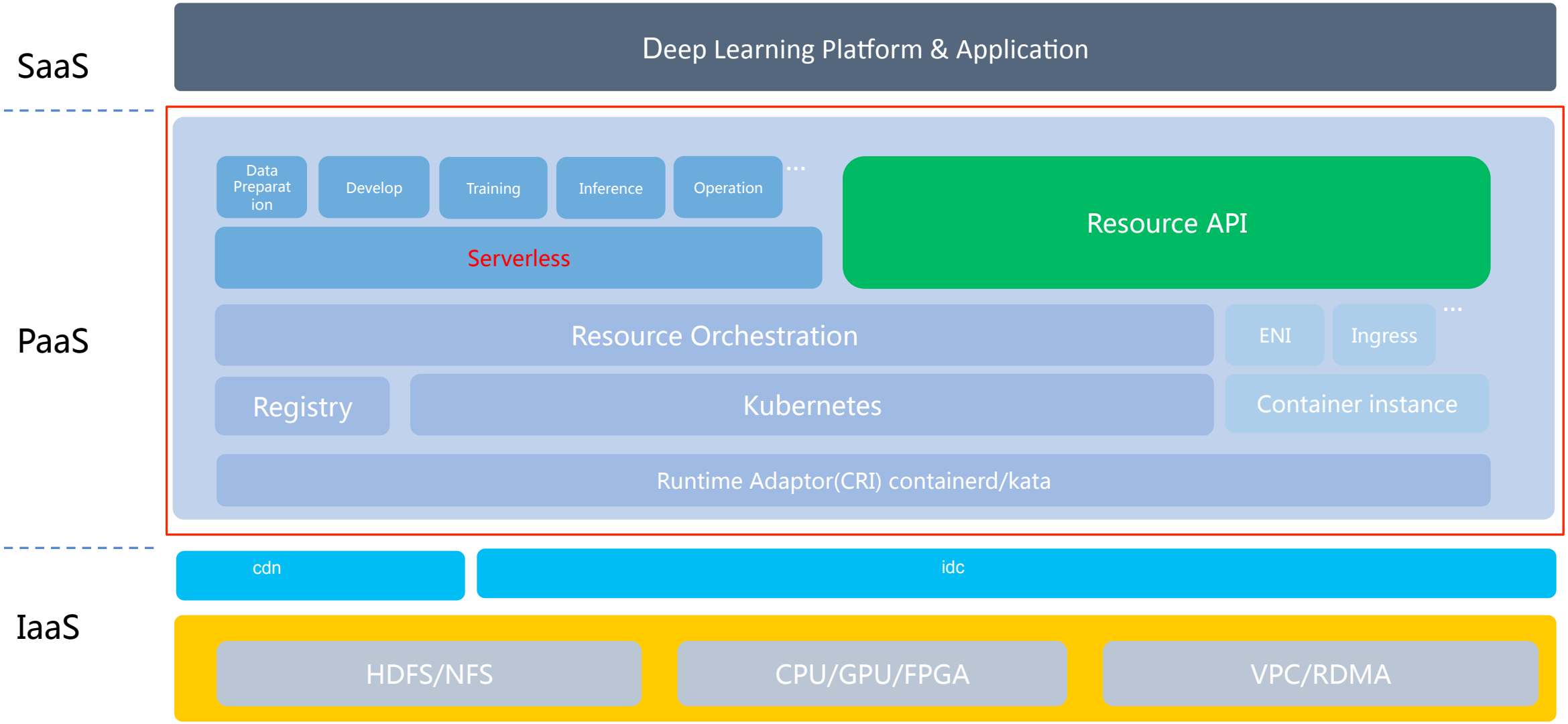
What’ s changed in 2018/2019

Infrastructure for serverless architecture is ready



Our current stack

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2019



Why Knative

What is Knative

Uniformed PaaS/FaaS/
Serverless user experiences

Features

- Automated builds
- Automatic scaling
- Traffic splitting
- Better deployment scenarios
- Event driven flows

Knative components

KubeCon CloudNativeCon
OPEN SOURCE SUMMIT
China 2019

Build

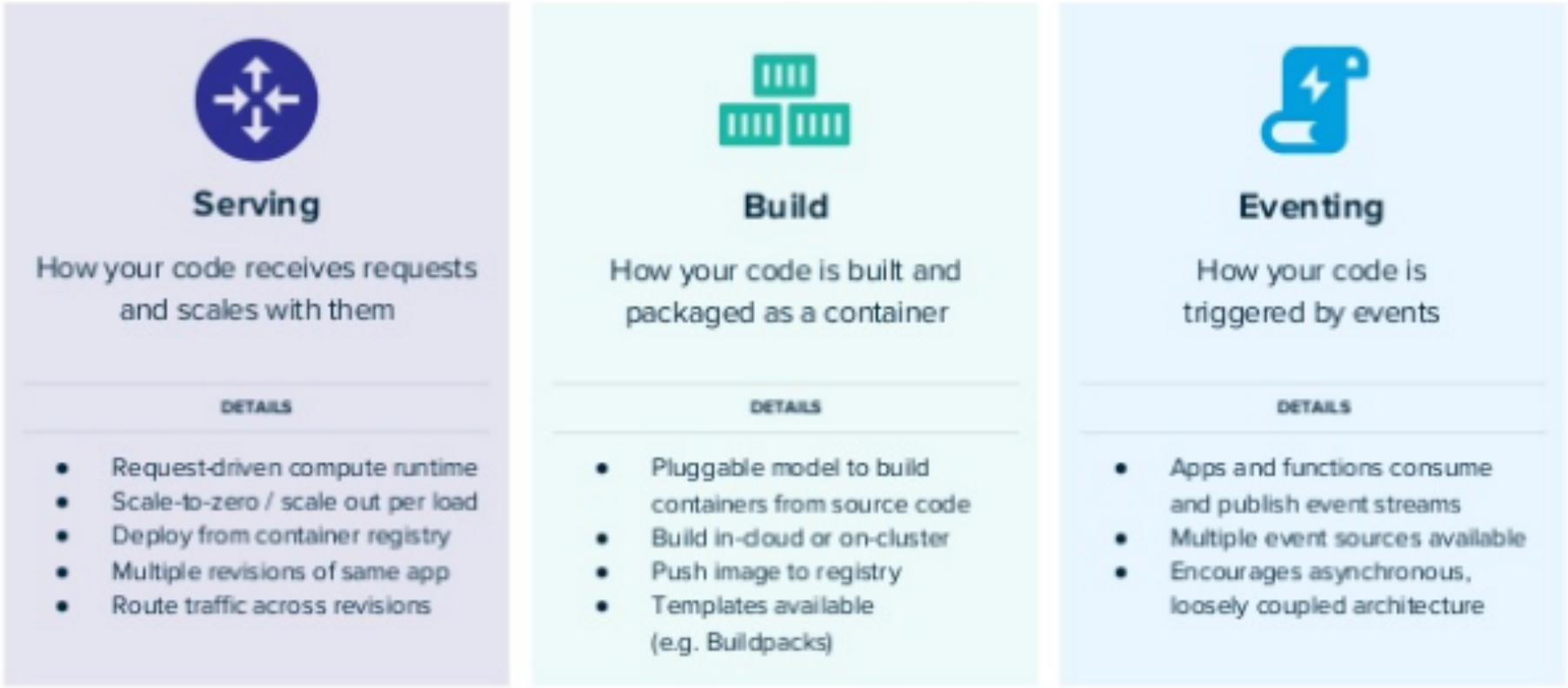
- Source to URL
- Build templates
- K8s ServiceAccount support

Serving

- Auto scaling
- Scale to zero
- Routing based on Istio

Eventing

- De-coupled pipelines
- Third party sources
- Cross platform



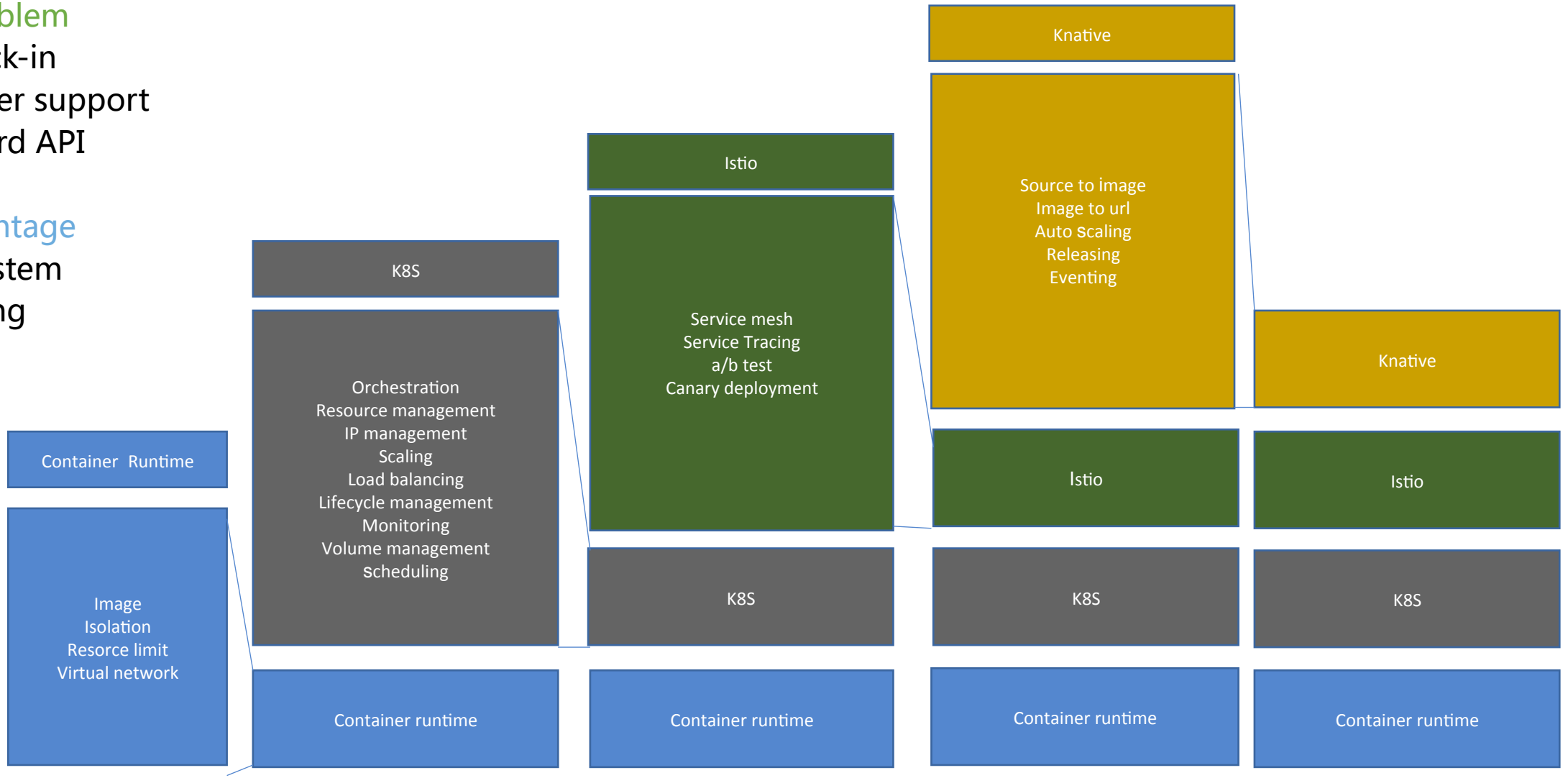
Why Knative

Common Problem

- Vendor lock-in
- No end user support
- No standard API

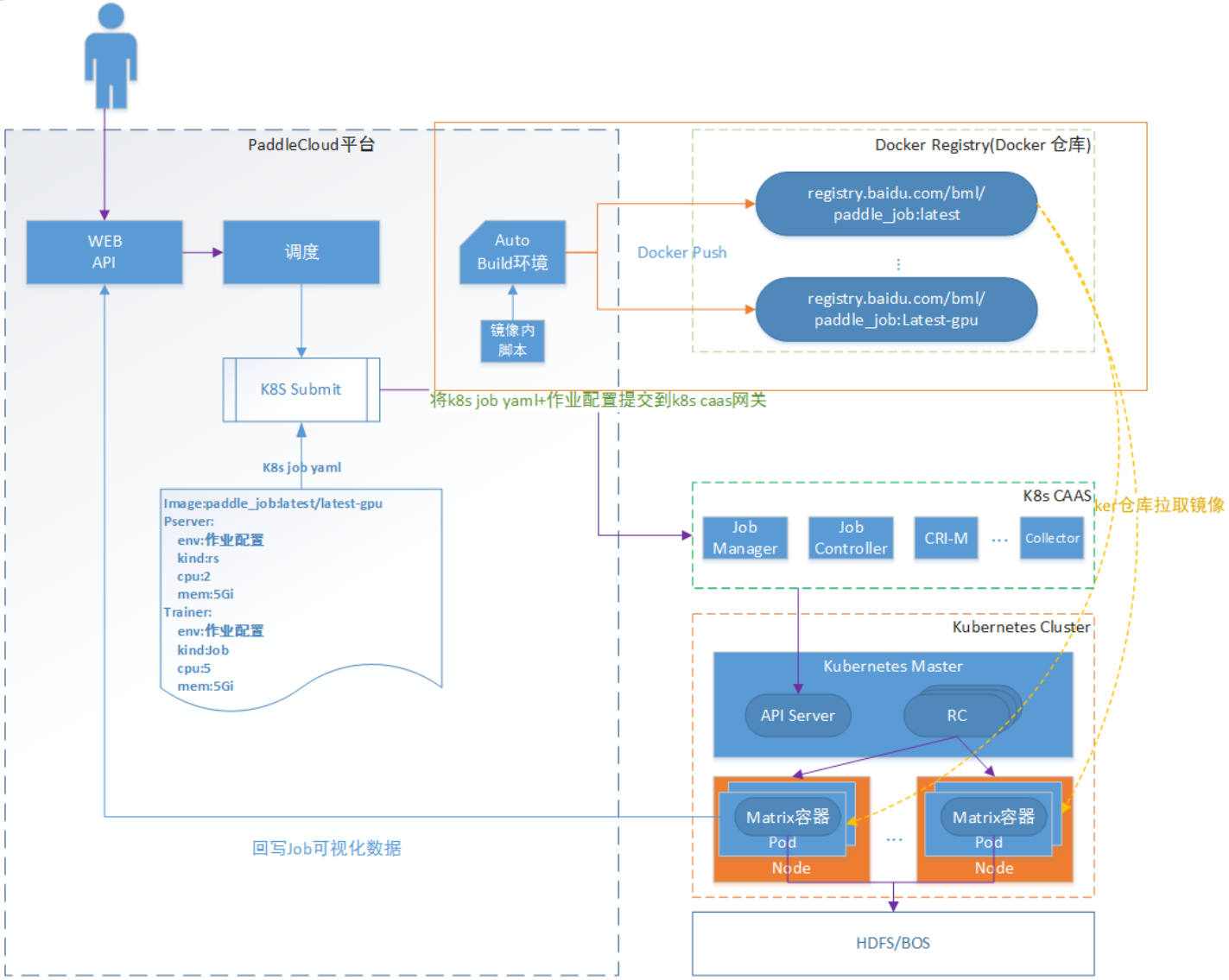
Knative Advantage

- K8S ecosystem
- De-coupling
- Adaptive



Knative Build for DLP

Our old architecture



Knative build – DLP example

KubeCon CloudNativeCon
OPEN SOURCE SUMMIT
China 2019

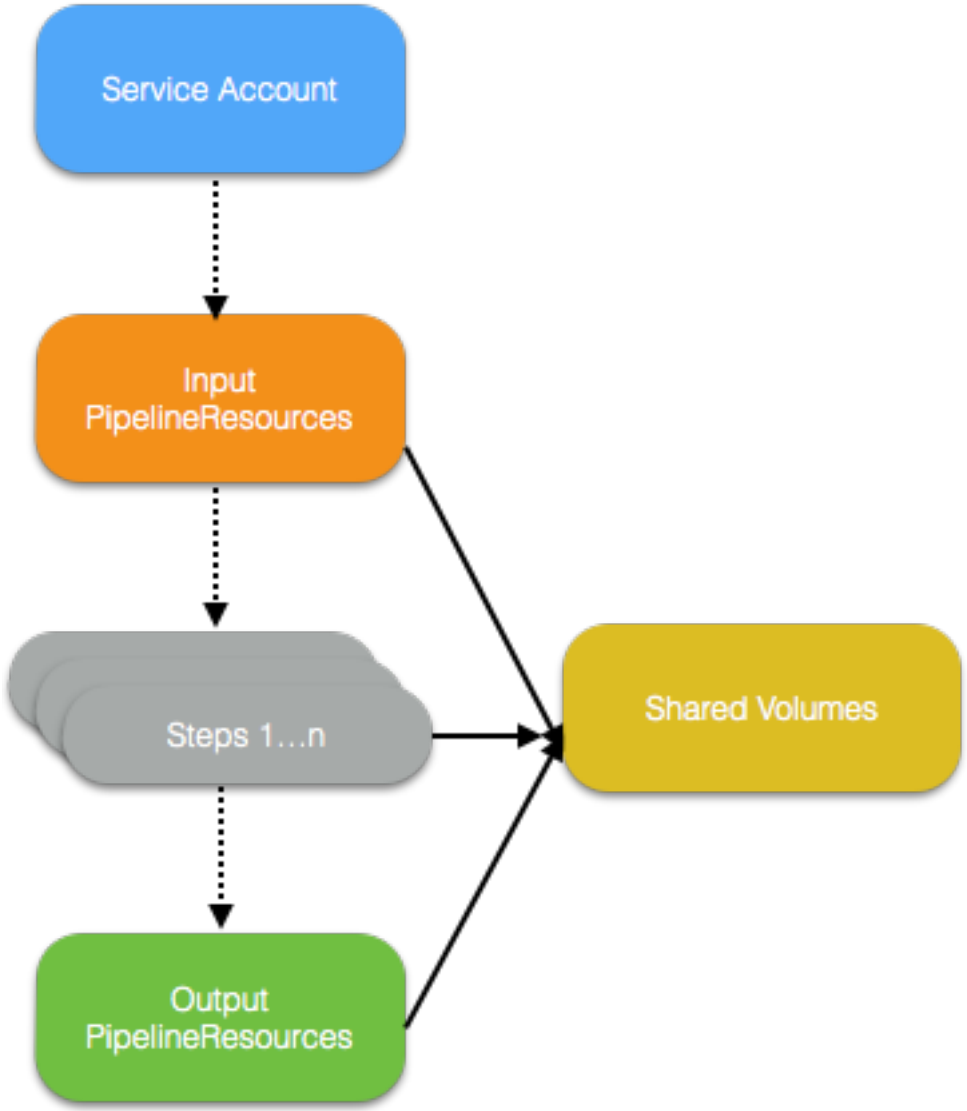
```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: dlp-image-pull-push
secrets:
  - name: dlp-image-pull-push-token-ed83a
imagePullSecrets:
  - name: xxx-xxx

apiVersion: v1
kind: Secret
metadata:
  name: dlp-image-pull-push-token-ed83a
  annotations:
    build.knative.dev/docker-0: https://
registry.baidu.com/v1/
type: kubernetes.io/basic-auth
stringData:
  username: xxx-xxx
  password: xxx-xxx
```

```
apiVersion: build.knative.dev/v1alpha1
kind: Build
metadata:
  name: dlp-job-build
spec:
  serviceAccountName: build-image-pull-push
  source:
    git:
      url: https://gitlab.baidu.com/dlp/build-paddle-job.git
      revision: master
  steps:
  - name: build-paddle-image
    image: registry.baidu.com/public/centos6u3-online:
1.0.12
    args: [ “/bin/bash”  “paddle-build", ”  run"]
  - name: build-paddle-job-image
     image: registry.baidu.com/bml/
centos6u3_paddle:v0.11.0
    args: [  ‘/bin/bash', ’  paddle-job-build', ’  run']
```

Tektoncd-pipeline with buildkit

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2019

```
apiVersion: tekton.dev/v1alpha1
kind: Task
metadata:
  name: paddle-git-to-image
spec:
  serviceAccount: "build-image-pull-push"
  inputs:
    resources:
      - name: paddle-source
        type: git
  outputs:
    resources:
      - name: builtImage
        type: image
  steps:
    - name: paddle-build-and-push
      image: registry.baidu.com/bml/
centos6u3_paddle:v0.11.0
      command:
        -   '/bin/bash', '   paddle-job-build', '   run'
```



Knative Serving for DLP

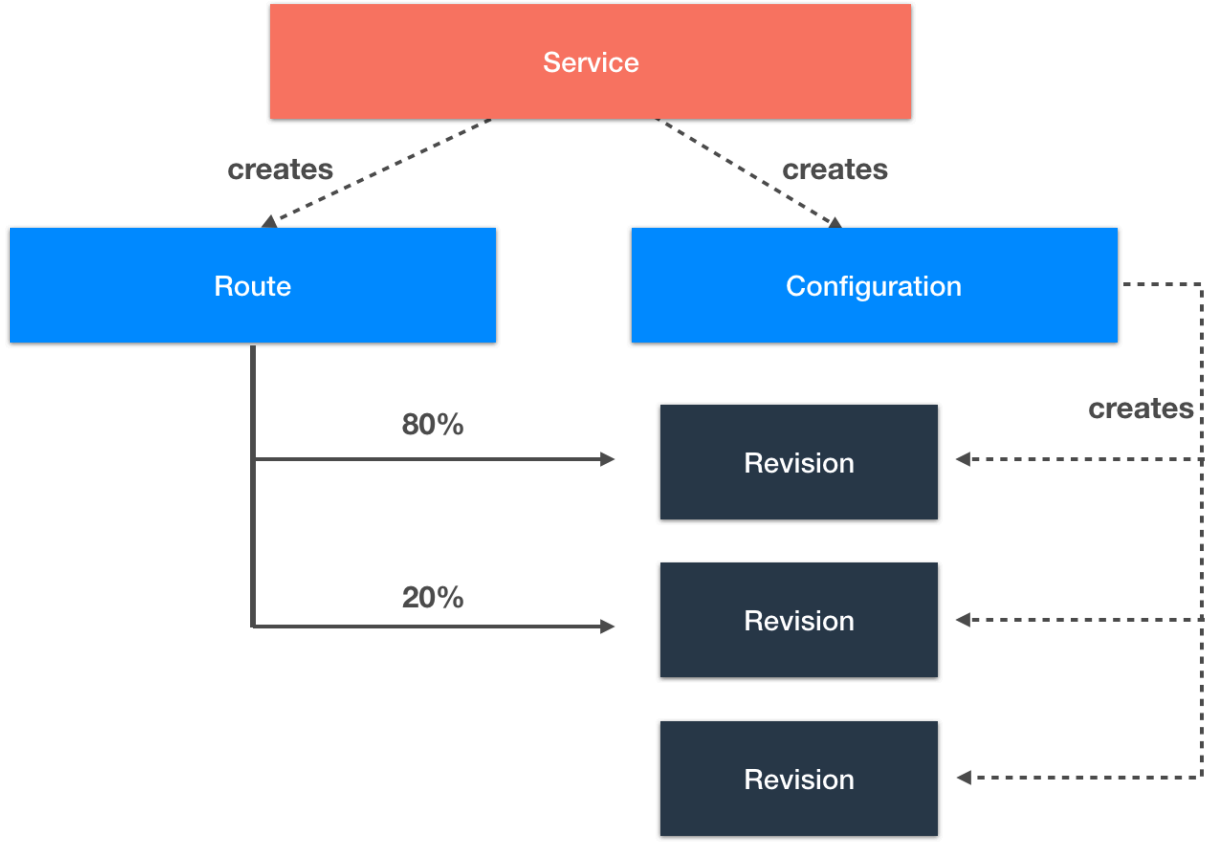
What is Knative serving

Serving Features:

- Network routing
- Flow control
- Upgrade policy
- Auto scaling

Resources when we start a service

- Knative Service
- Knative Configuration
- Knative Route
- K8S Deployment
- K8S HPA
- K8S Service
- Istio VirtualService



Knative serving – DLP example

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2019

```
apiVersion: serving.knative.dev/v1alpha1
kind: Configuration
metadata:
  generation: 1
  labels:
    serving.knative.dev/route: paddle-model-
serving-route
  name: paddle-model-serving-config
spec:
  generation: 1
  revisionTemplate:
    metadata:
    labels:
        app: greeter
    spec:
      container:
        image: registry.baidu.com/bml/paddle-model-
serving:1.0.1
```

```
apiVersion: serving.knative.dev/v1alpha1
kind: Route
metadata:
  generation: 1
  name: paddle-model-serving-route
spec:
  generation: 1
  traffic:
  - configurationName: paddle-model-serving-config
    percent: 100
```

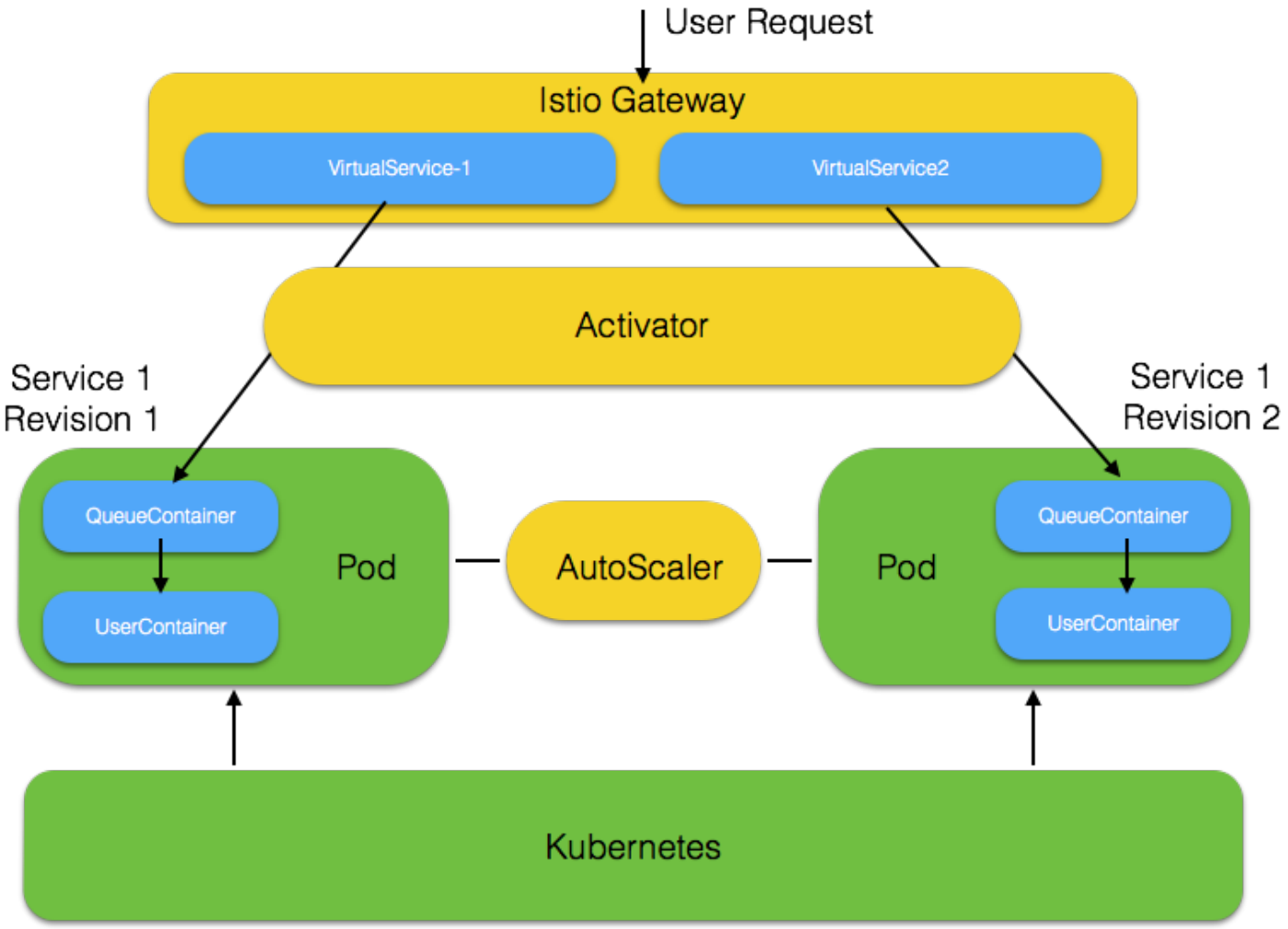
Knative serving - Ingress

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2019

Ingress Controller Features :

- Load Balancing
- Retrying
- TLS Terminating
- Route based on Header
- Header addon

```
apiVersion: serving.knative.dev/v1alpha1
kind: Route
metadata:
  name: paddle-model-serving
spec:
  traffic:
    - revisionName: paddle-model-serving-00001
      percent: 90
    - revisionName: paddle-model-serving-00002
      percent: 10
```

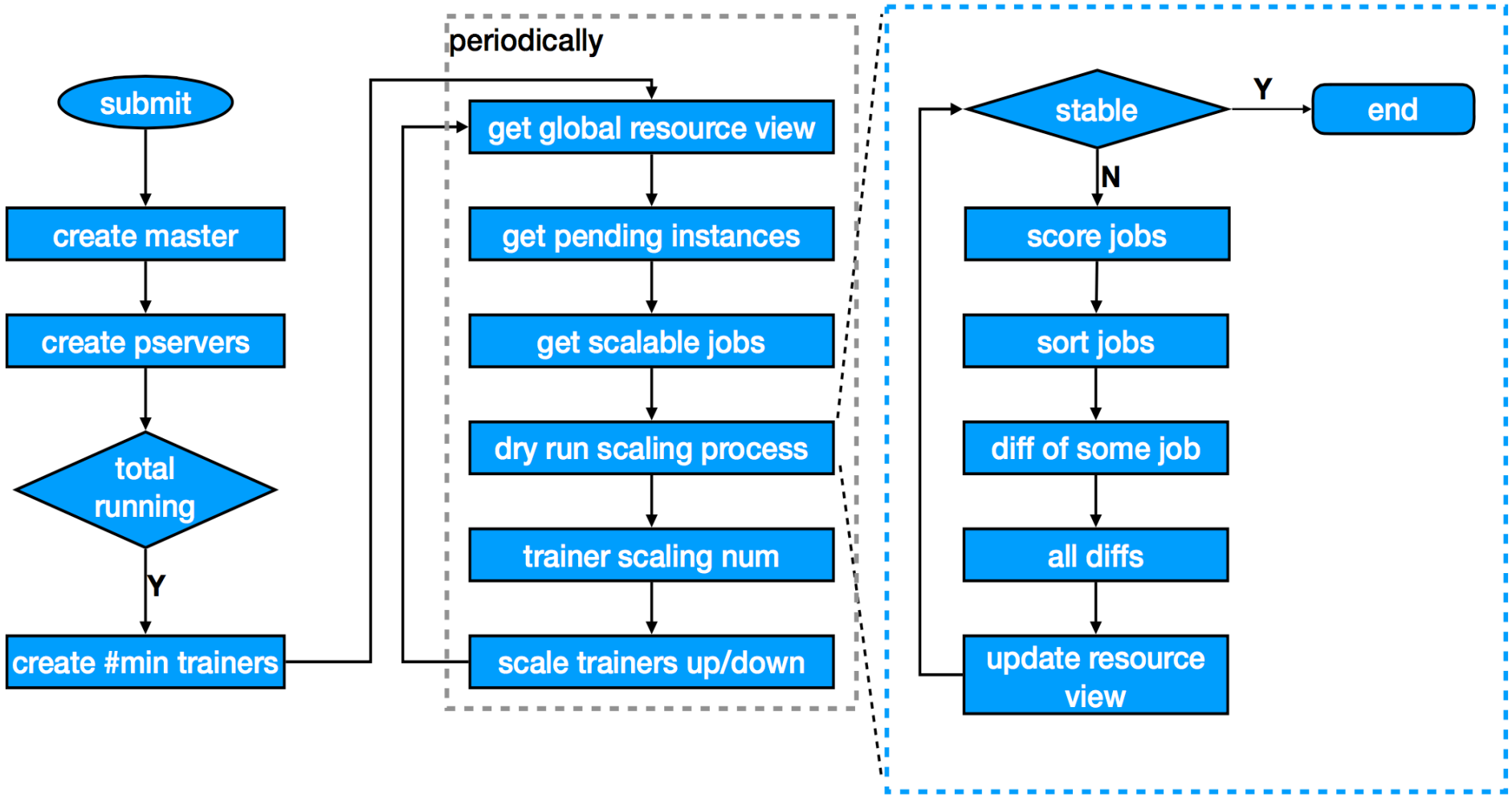


Knative serving – autoscale old solution

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2019

Old Solution:

- Starting with small instance number, if the cluster GPU utilization is low, scale up the number
- Paddle job specific, implemented in Paddle Operator



Knative serving – custom autoscale class

KubeCon CloudNativeCon
OPEN SOURCE SUMMIT
China 2019

```
apiVersion: serving.knative.dev/v1alpha1
kind: Service
metadata:
  name: paddle-model-autoscale
  namespace: default
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/class: kpa.autoscaling.knative.dev
        autoscaling.knative.dev/metric: concurrency
        autoscaling.knative.dev/target: "10”
        autoscaling.knative.dev/minScale: "1”
        autoscaling.knative.dev/maxScale: "100"
    spec:
      containers:
        - image: registry.baidu.com/paddle/paddle-model-serving:1.0.1
```

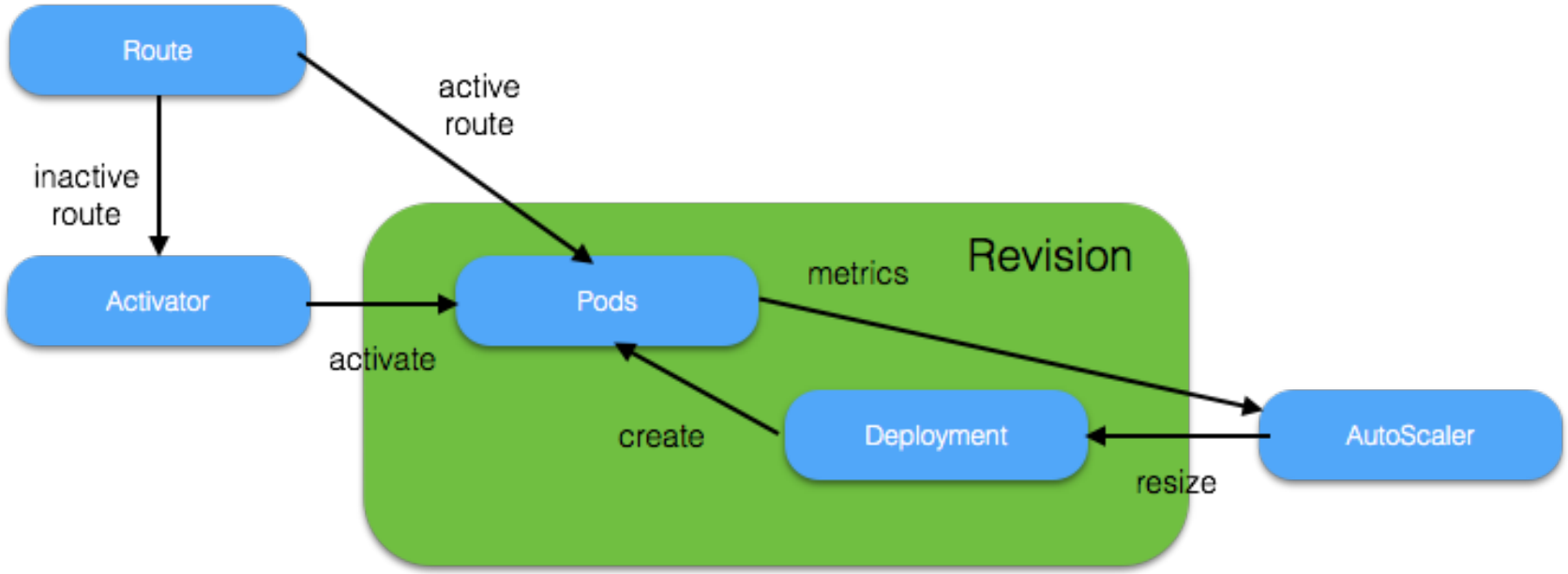
your own reconciler and autoscaling system

autoscaling.knative.dev/class
hpa.autoscaling.paddle.baidu.com

Knative serving – cold start

Cold Start Solution:

- Reduce side-car injection latency, with Istio 1.0.2 release, reducing the Envoy programming time
- Reduce image pull latency with container instance pool, container can be provisioned in advanced
- Immediately scaling up when the autoscaler gets stats from activator.



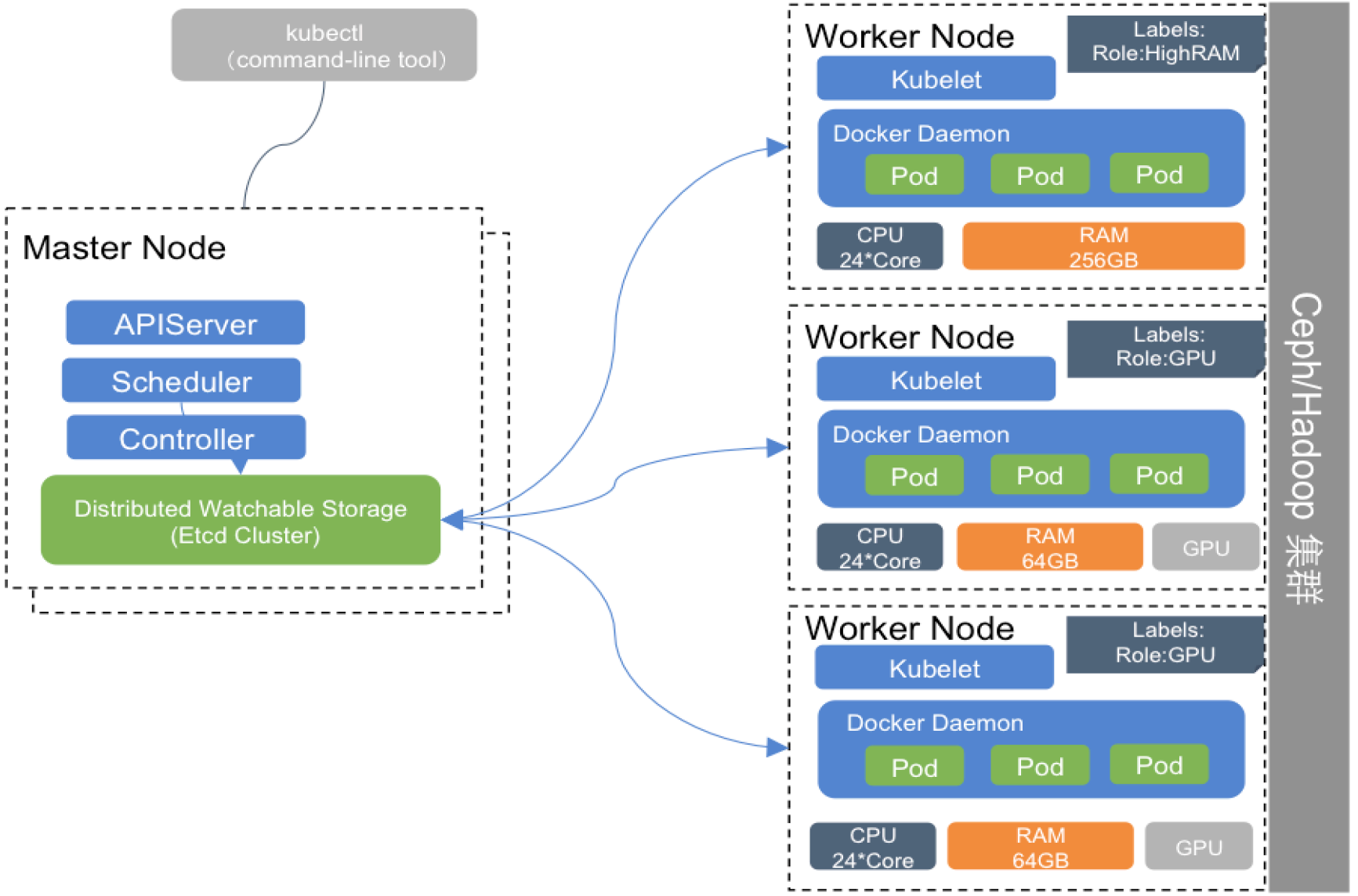
Knative Serving with Container Instance

Knative with container instance – old solution

KubeCon CloudNativeCon
OPEN SOURCE SUMMIT China 2019

Old Solution:

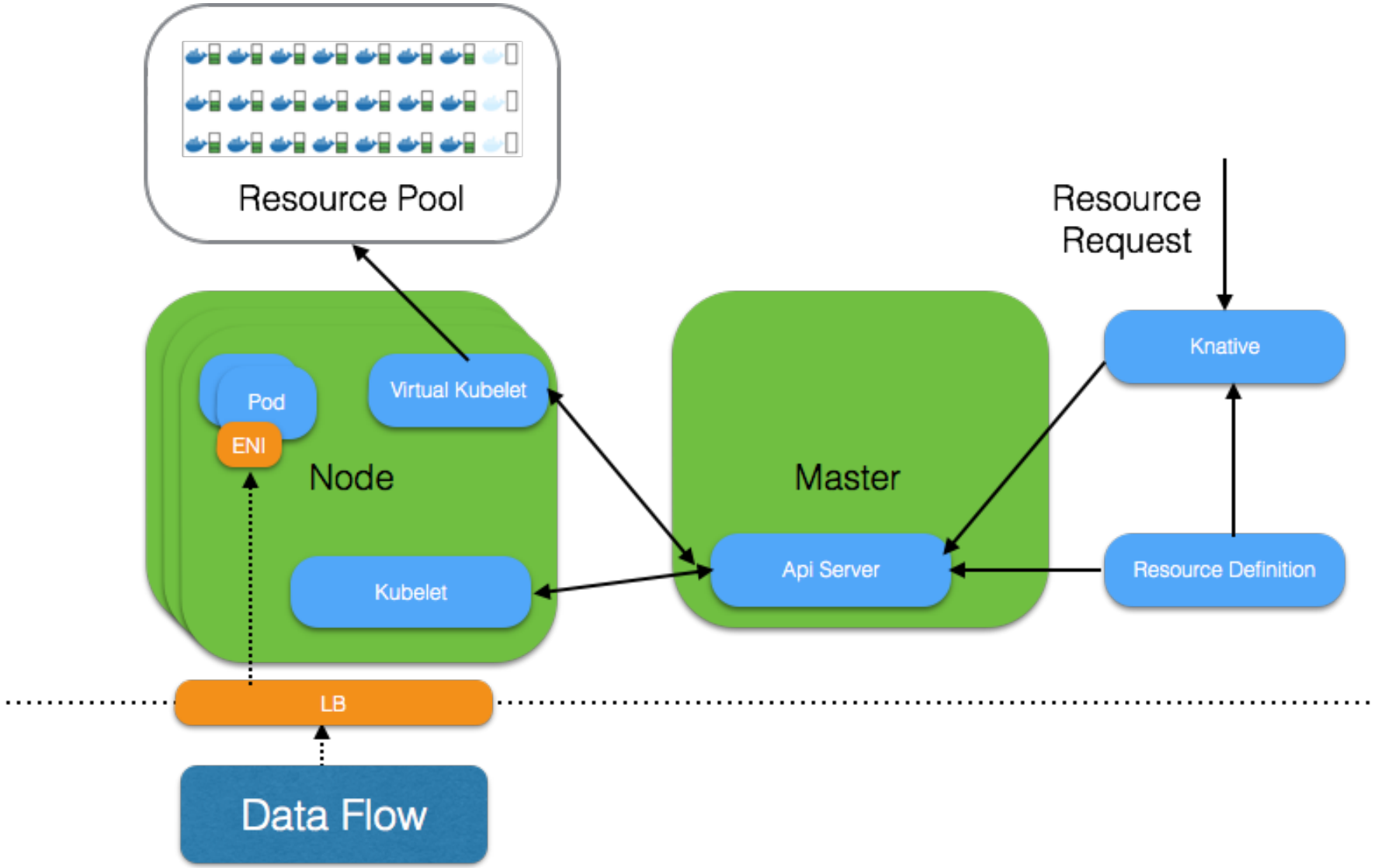
- Use Label & Node Affinity
- Use Extended Resource for Hetergeneous GPUs(K40, P40, V100, etc) and Network(Inifiniband, 100GEth)
- Scheduler: GPU Binpack by default
- Multi-tenancy on k8s namespace level



Knative with container instance

Advantage:

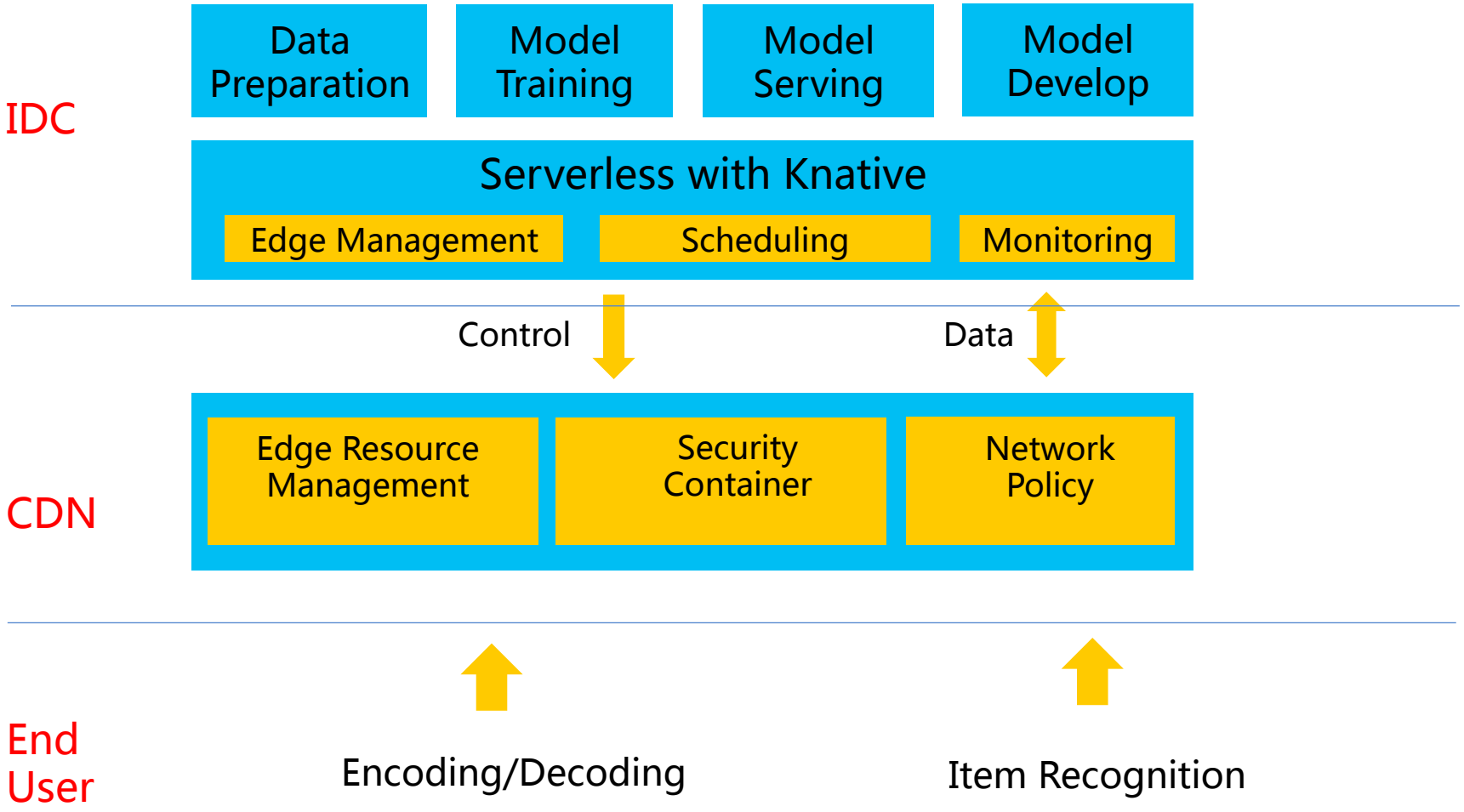
- Cost savings of around 30%
- High density deployment
- No need for labels & taints
- No resource fragmentation
- Resource Reusable



Knative Serving on Edge

Edge solution with Knative

KubeCon CloudNativeCon
OPEN SOURCE SUMMIT
China 2019



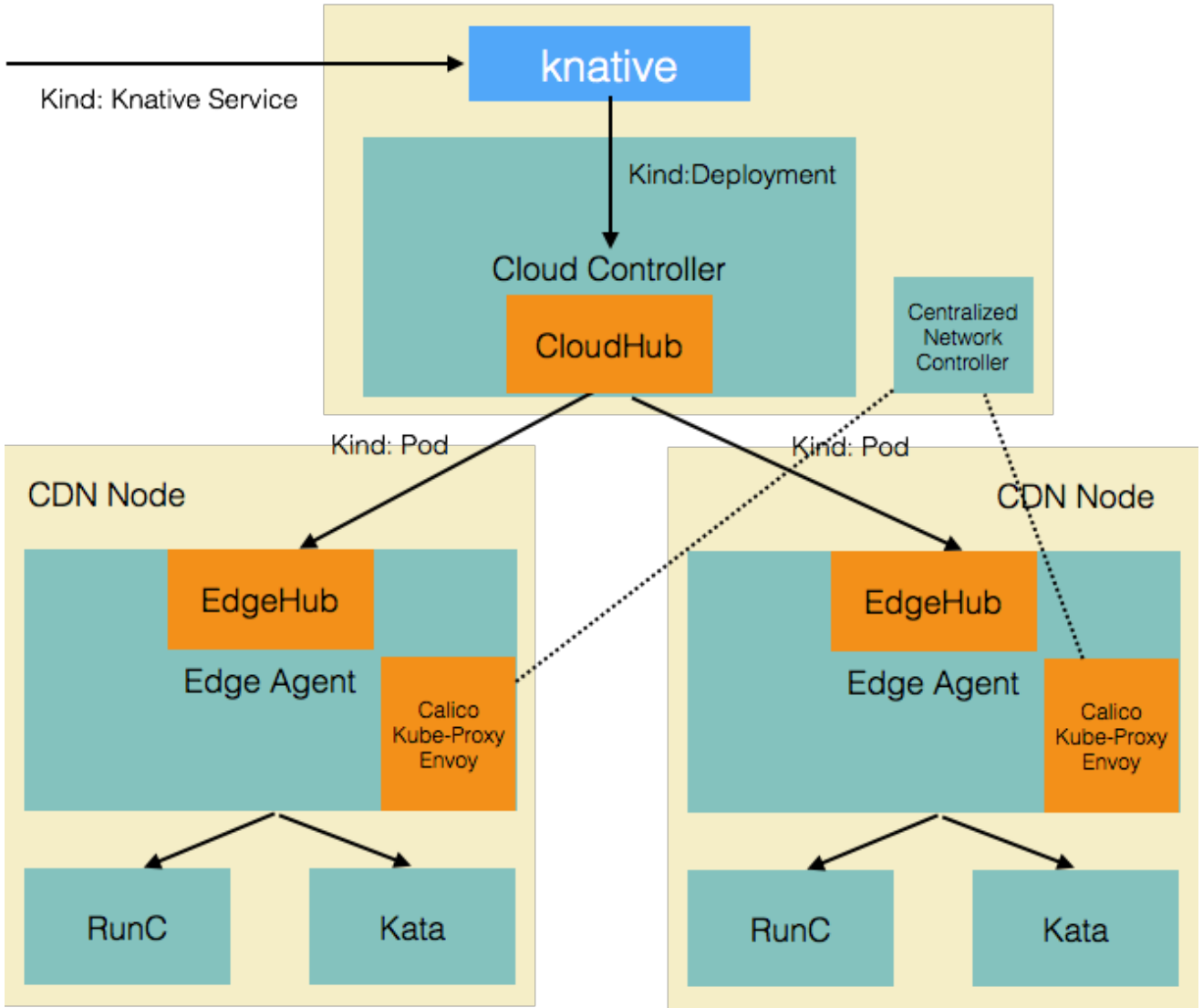
Compute and Network on Edge

Compute

- Knative Service
- Deployment
- Pod

Network

- Kube Proxy
- Calico Agent
- Envoy

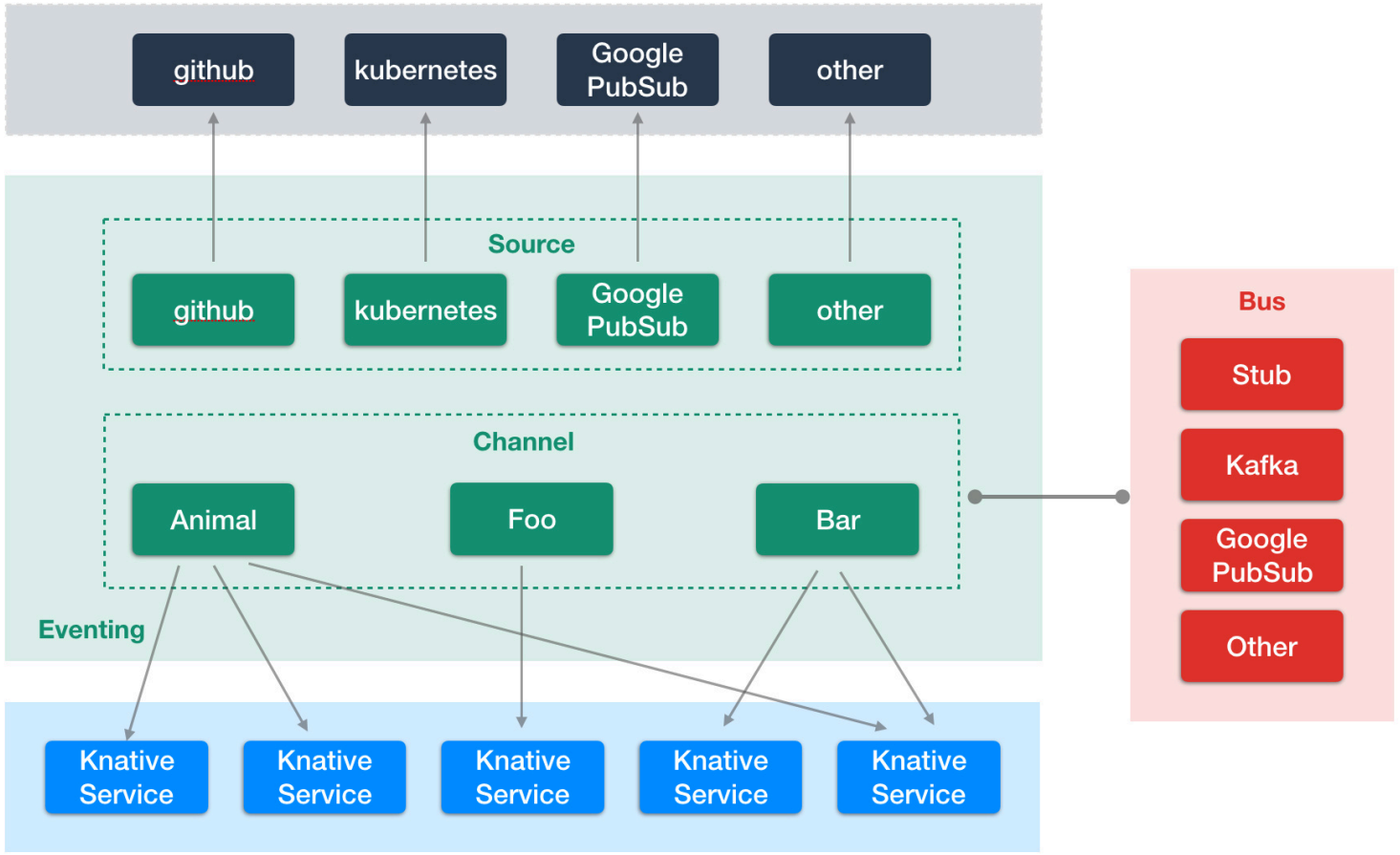


Knative Eventing for DLP

Knative eventing

Advantage:

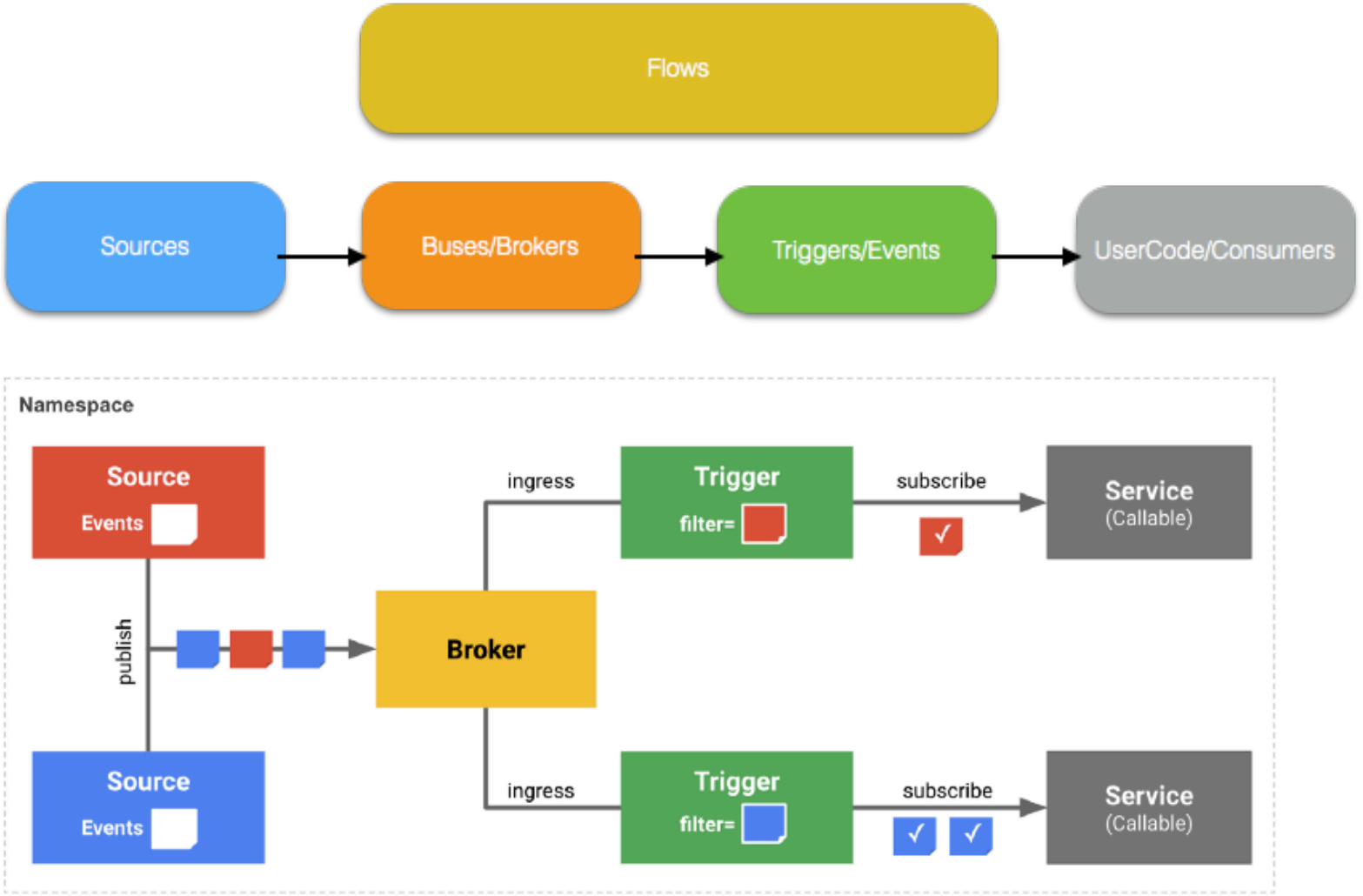
- Universal subscription, delivery and event management
- Building loosely coupled event-driven systems with advanced objects
- Declarative binding between event generators and event usage services
- Extend from several events to streaming
- Custom event pipeline for connecting to existing systems



Knative eventing – DLP example

KubeCon CloudNativeCon
OPEN SOURCE SUMMIT
China 2019

```
apiVersion: flows.knative.dev/v1alpha1
kind: Flow
metadata:
  name: k8s-event-to-serving-flow
  namespace: default
spec:
  serviceAccountName: paddle-sa
  trigger:
    eventType: dev.knative.k8s.event
    resource: k8sevents/
dev.knative.k8s.event
    service: k8sevents
    parameters:
      namespace: default
  action:
    target:
      kind: Route
      apiVersion: serving.knative.dev/
v1alpha1
      name: paddle-model-serving
```



Knative eventing - CloudEvents

KubeCon CloudNativeCon OPEN SOURCE SUMMIT China 2019

Producer:

```
c := cloudevents.NewClient(
        "http://localhost:8080",
        cloudevents.Builder{
            Source:      “https://github.com/paddle/pkg#cloudevents-paddle",
            EventType: "dev.knative.cloudevent.paddle",
            Encoding: cloudevents.BinaryV01,
        },
    )

if err := c.Send(data); err != nil {
            log.Printf("error sending: %v", err)
        }
```

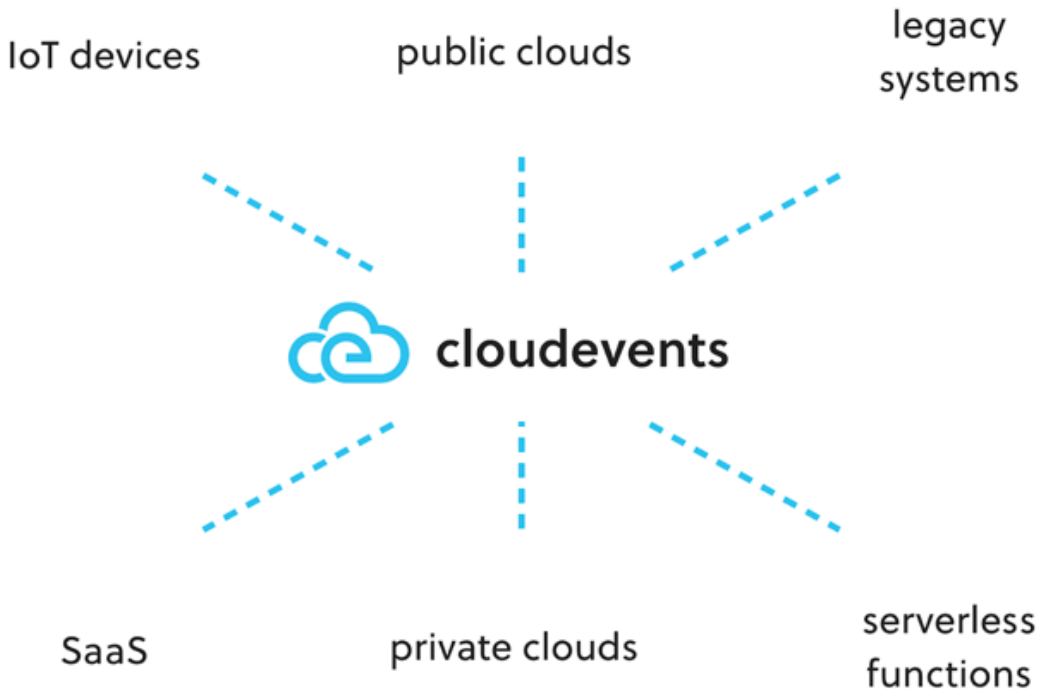
Consumer:

```
func handler(ctx context.Context, data *PaddleJob) {
    metadata := cloudevents.FromContext(ctx)
}
```

Experience:

Components are loosly coupled.

Need to handle event tracing.




KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2019