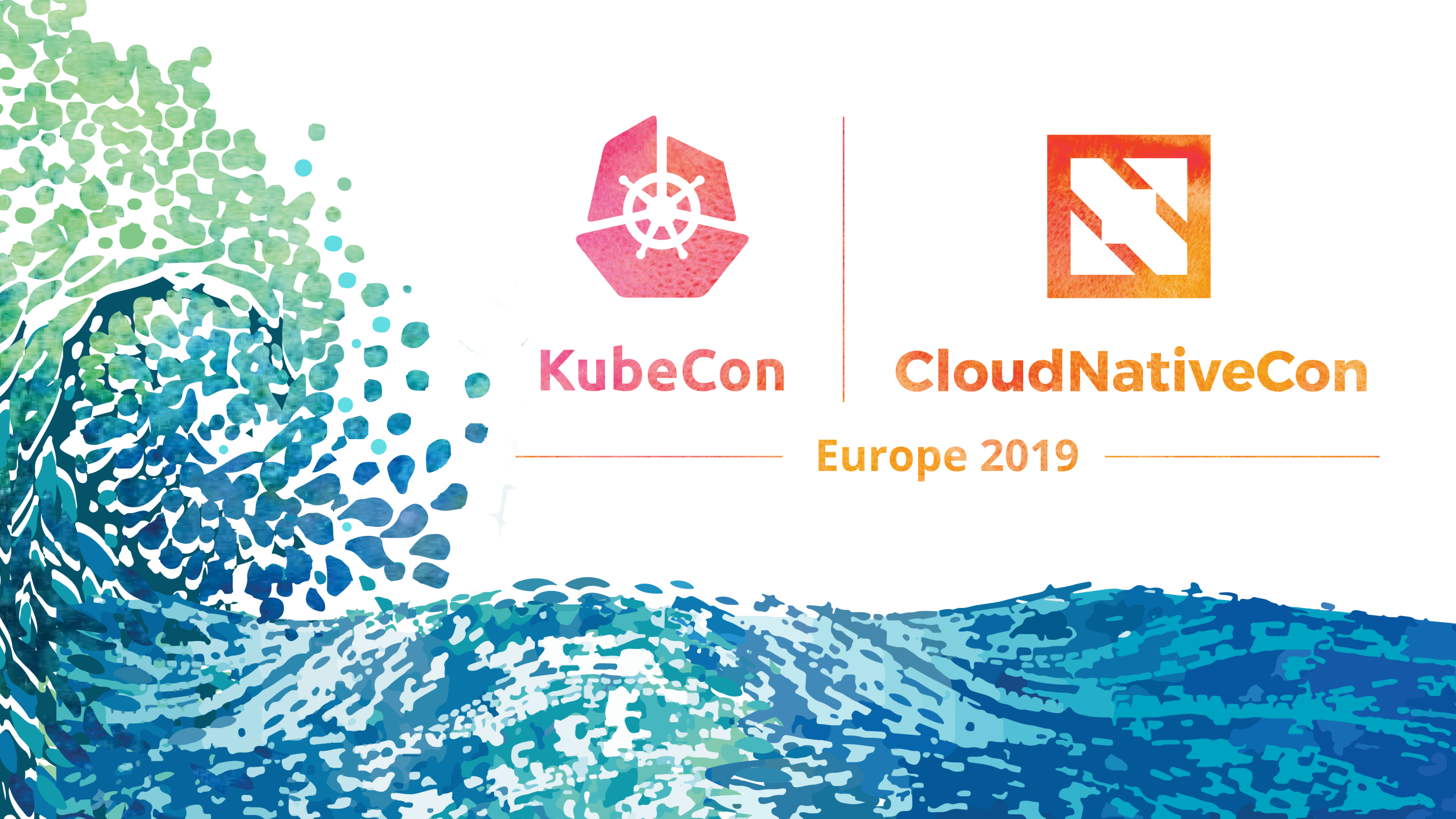

KubeCon
CloudNativeCon
Europe 2019

KubeCon | CloudNativeCon

Europe 2019

Manage CRDs and Operators in Practice

Zhen Zhang shouchen.zz@alibaba-inc.com

Wei Guo kira.gw@antfin.com

Alibaba's Journey to Cloud Native

2011: containerize

- LXC(T4)
- Rich Container
- script

2015: dockerize + smart scheduling

- Docker(Pouch)
- Rich Container with Dockerfile
- Swarm(Sigma)

2018: k8s with basic orchestration

- Docker(Pouch) + RunC
- Pod with single rich container
- K8S

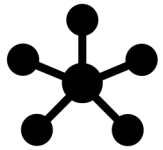
2019: cloud native

- Docker(Pouch) + RunC
- Pod with light container
- K8S

Why CRD and operator

Stateful App Management

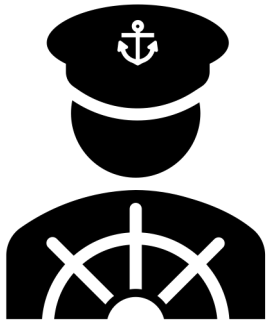
Enhancement



Integration with infrastructure

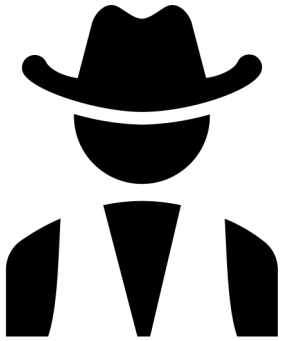
Opensource & innovation

Who are building CRD and operator



"Captain"

- us (from our own eyes)
- admin of your k8s clusters
- k8s and golang knowledge and experience
- Sheldon Cooper



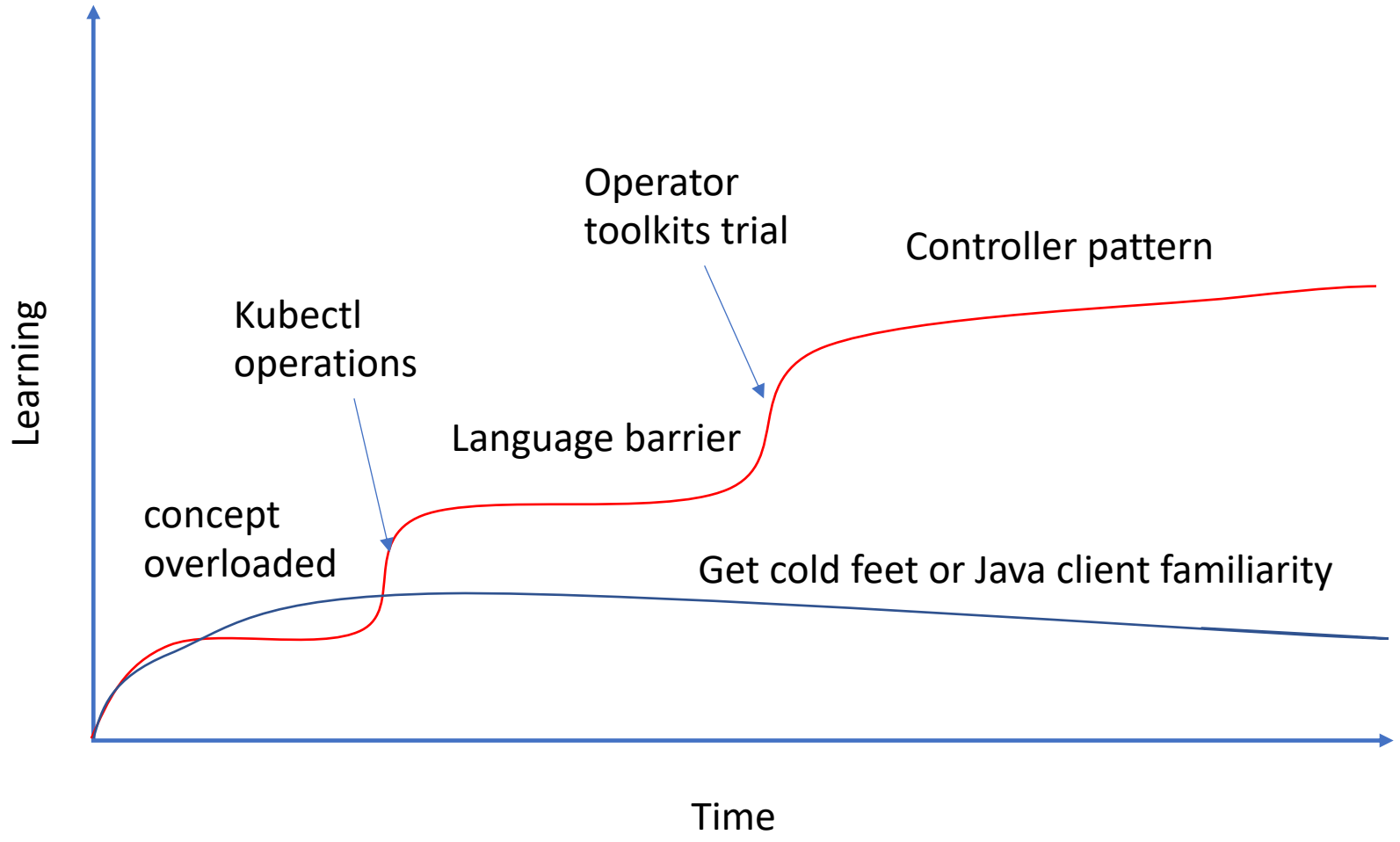
"Cowboy"

- Other teams (from our own eyes)
- Expert of some PaaS or maintainer of other's k8s cluster
- Limited k8s and golang knowledge and experience
- Always mess sth up with insane ideas

"Business Man"

- Expert of applications
- In depth knowledge about apps
- No k8s or golang knowledge and experience but very curious
- Lots of legacy
- Some are forced to write operators

Problems: learn to write operators



Solutions: speed up learning

1. Easy and early access to test cluster
2. Direct document reading
3. Direct operator toolkit selection
4. Easy access to sample operators (your local awesome operators)
5. Learn controller patterns
6. Make CRD and operators standards and best practice

Problems: improper naming

CRD naming

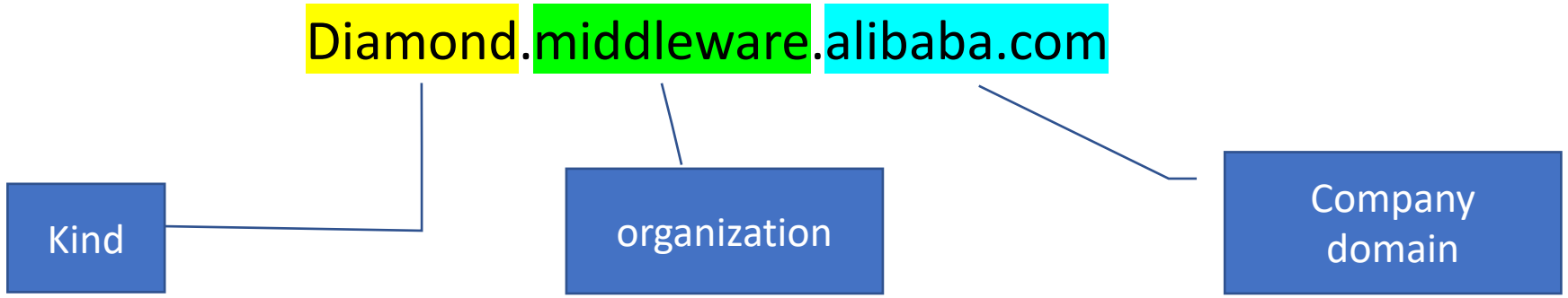
Deployment	Naming collision with core types
PetSet.extensions	unclear group meaning

Labels naming

Region.	Conflict with standard label failure-domain.beta.kubernetes.io/region

Solutions: crd standards

- CRD naming conventions



- Predefined resource category: apps, data, auth etc.
- Company wide common labels
- All apps CR must provide scale subresource and RolloutControlDefinition

common pitfalls and best practise

1. Package management through operator

2. Reinvent the wheels

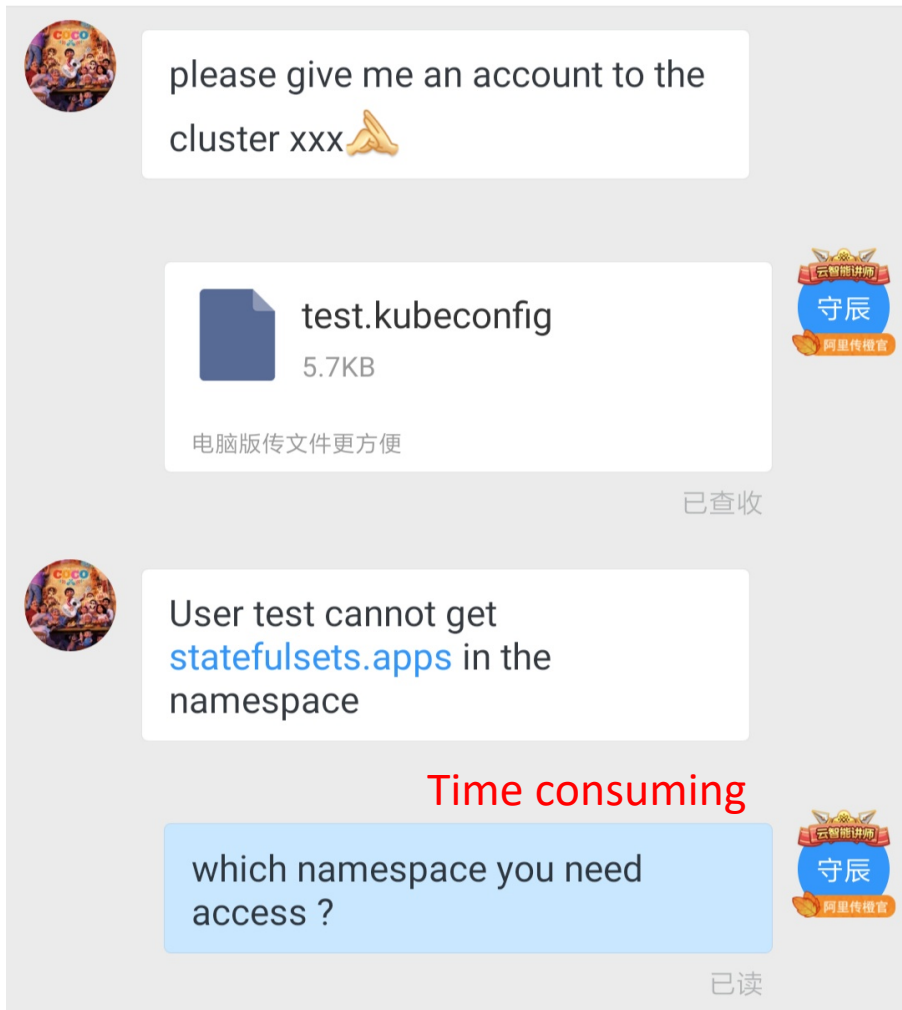
3. imperative design

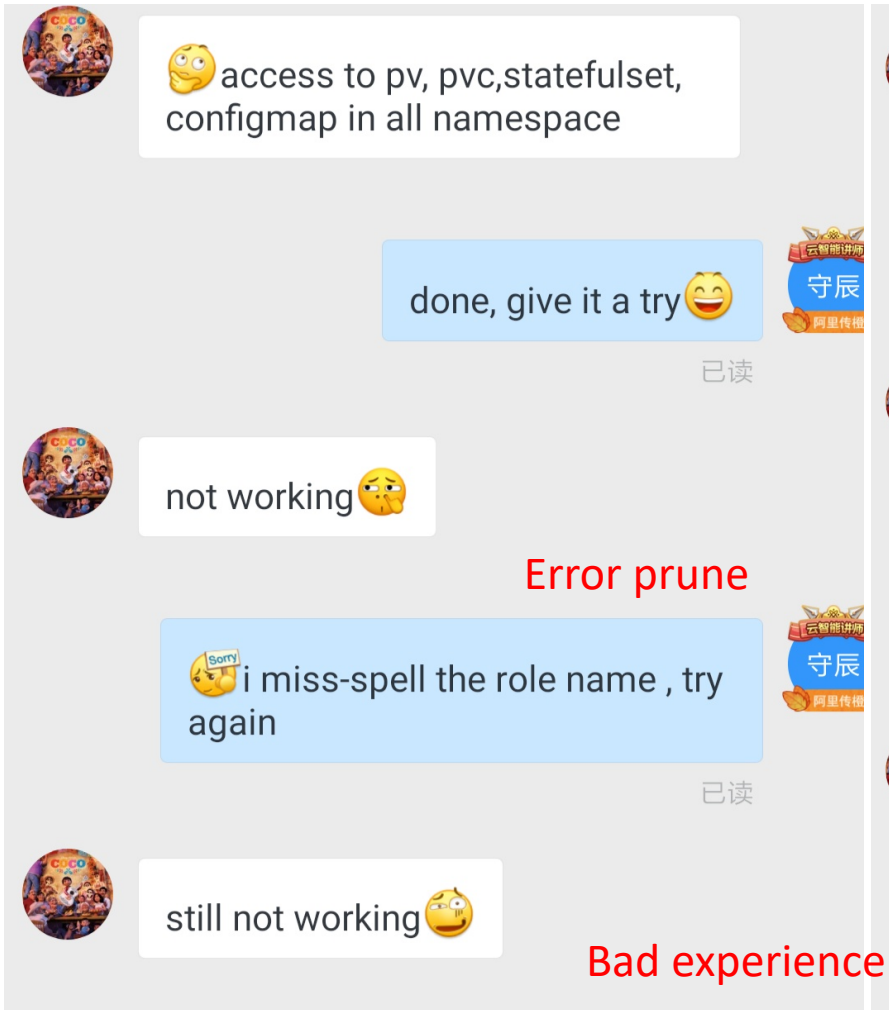
1. Package management through helm, create operator only if app specific logic required

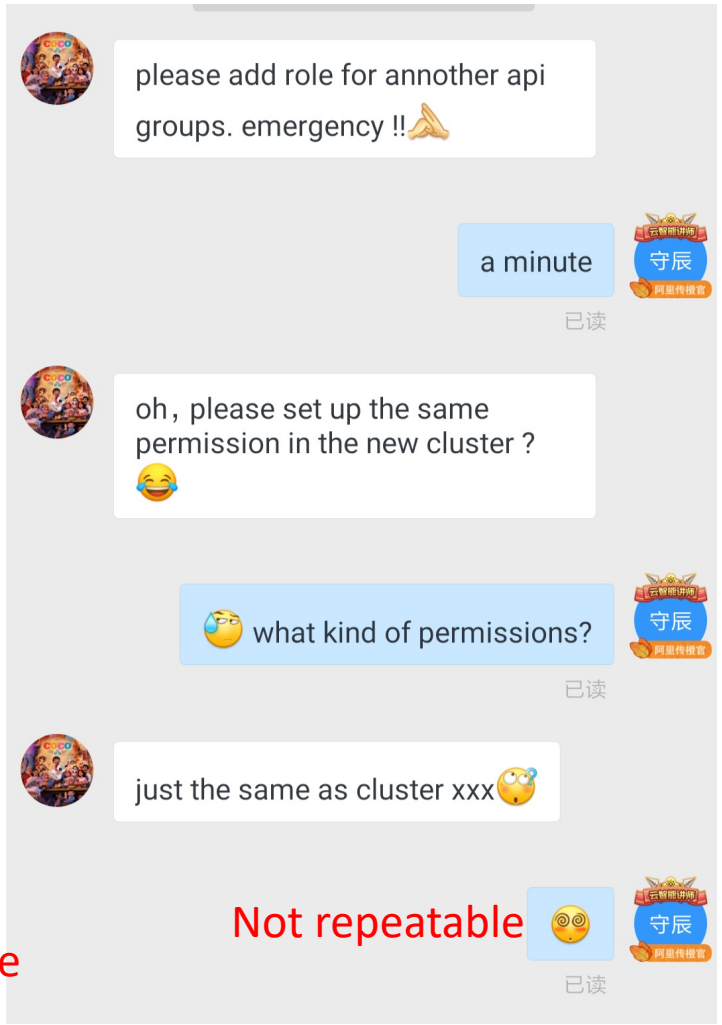
2. Use k8s existing types if any (statefulset, configmap)

3. declarative design

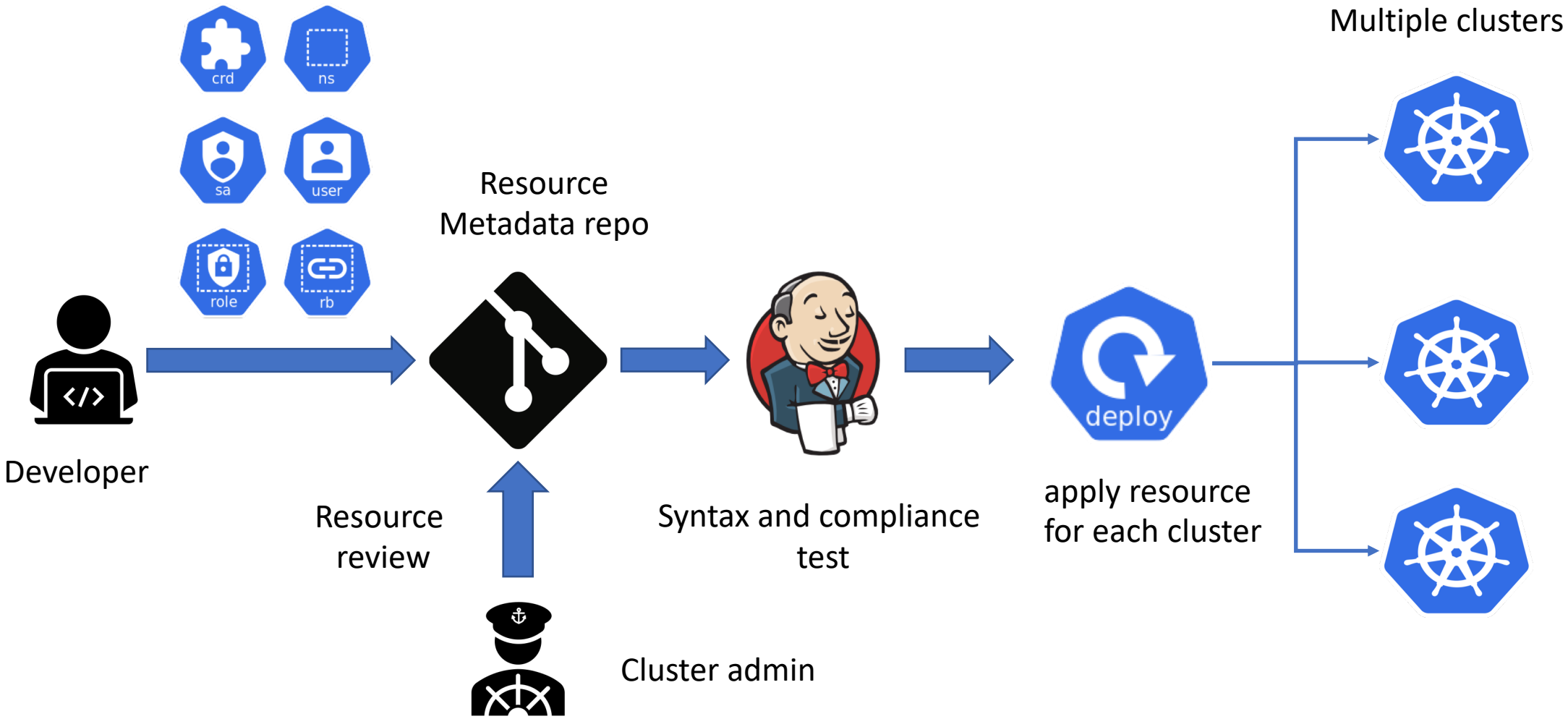
Problems: manual access control







Solutions: CI/CD based access review



review & test

Common test problem

1. Missing kubeconfig or service account
2. Incorrect role or rolebinding name
3. Conflict name with other crd or core types
4. Install in namespace kube-system

Common review problems

1. Incorrect CRD scope
2. Excess permission
3. No update the operator list
4. Duplicate operators
5. Webhook for core types (pods & nodes)
6. Delete crd without clear resources (danger!)

kubeconfig & webhook issuer

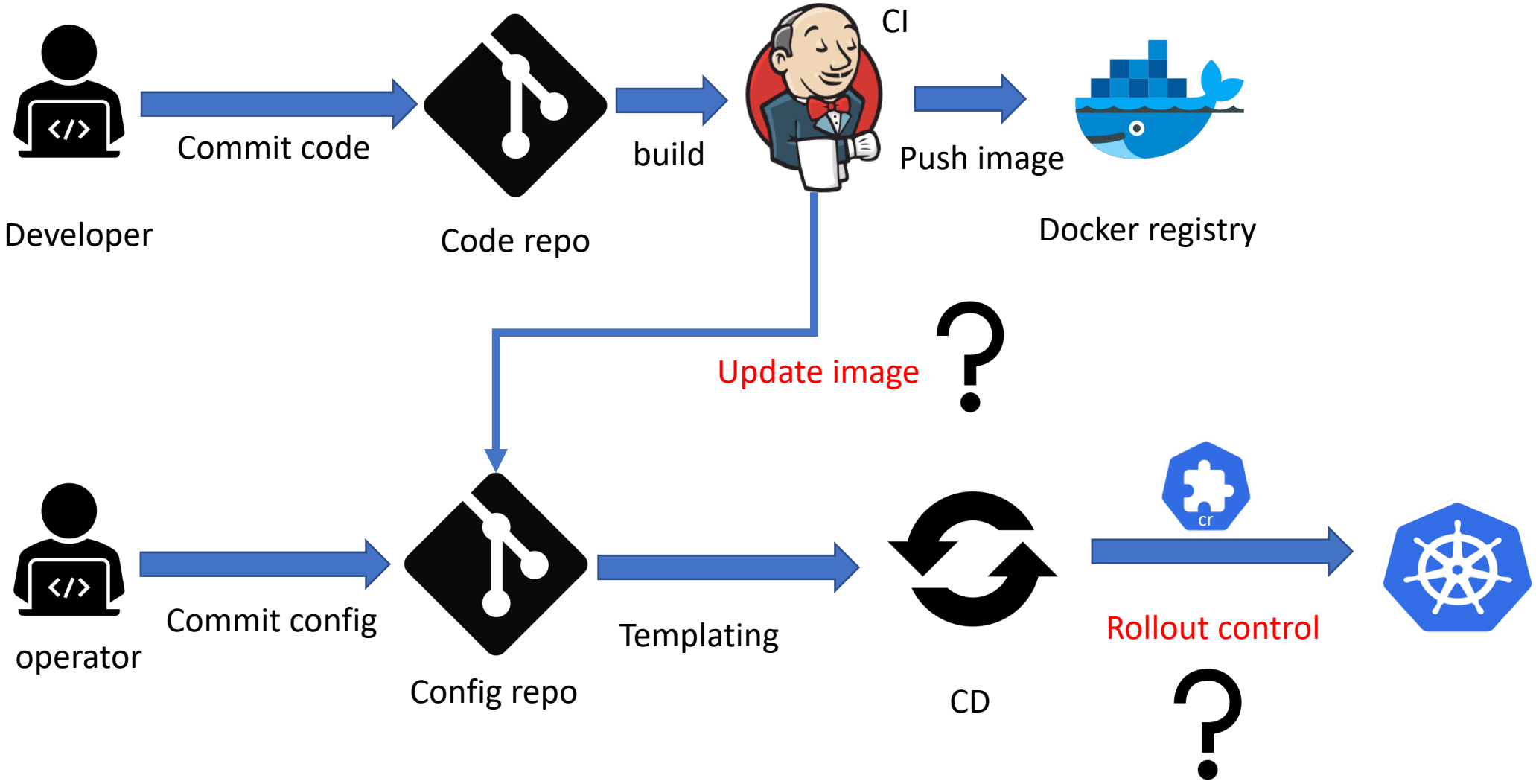
Automatic kubeconfig issue

```
kind: KubeconfigRequest
metadata:
  name: siteops
spec:
  username: siteops
  groups:
  - paas
  email: sample@alibaba-inc.com
```

Automatic webhook cert generation

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: webhook-account
  annotations:
    webhook.alpha.sigma.ali/gen-serving-cert: "true"
```

Problems: integration with CICD pipeline



Solutions: common helm values

Values.yaml

```
images:
  main: nginx
  sidecar1: logagent
```

cr.yaml

```
template:
  spec:
    containers:
    - name: main
      image: {{index .Values.images "main" }}
    - name: logagent
      image: {{index .Values.images "logagent" }}
```

No crd-install required

Solutions: common rollout control


```
apiVersion: apps.sigma.ali/v1alpha1
kind: RolloutControlDefinition
metadata:
    name: diamond-control-def
spec:
   controlResource:
      apiVersion:  apps.middleware.ali/v1alpha1
      resource: Diamond
   specPath:
      paused: .spec.paused
      partitions: .spec.strategy.rollingUpdate.partitions
      maxUnavailable: .spec.strategy.rollingUpdate.maxUnavailable
   statusPath:
      replicas: .status.replicas
      readyReplicas: .status.readyReplicas
      updatedReplicas: .status.updatedReplicas
      observedGeneration: .status.observedGeneration
```

```
apiVersion: apps.sigma.ali/v1alpha1
kind: RolloutControl
metadata:
   name : diamond-control
spec
   resource: diamond1
   paused: “true”
   updateStrategy:
        partitions: 10
        maxUnavailable: 30%
status:
  replicas: 100
  readyReplicas: 100
  updatedReplicas: 100
  ObservedGeneration: 10
```


KubeCon
CloudNativeCon
Europe 2019
Questions?