KubeCon

CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon | CloudNativeCon

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

AI_dev

China 2024

# Inplace-update: the past, present and future

Zhen Zhang & Yuxin Yuan Alibaba Cloud & OpenKruise maintainer

# Inplace-update: the origin

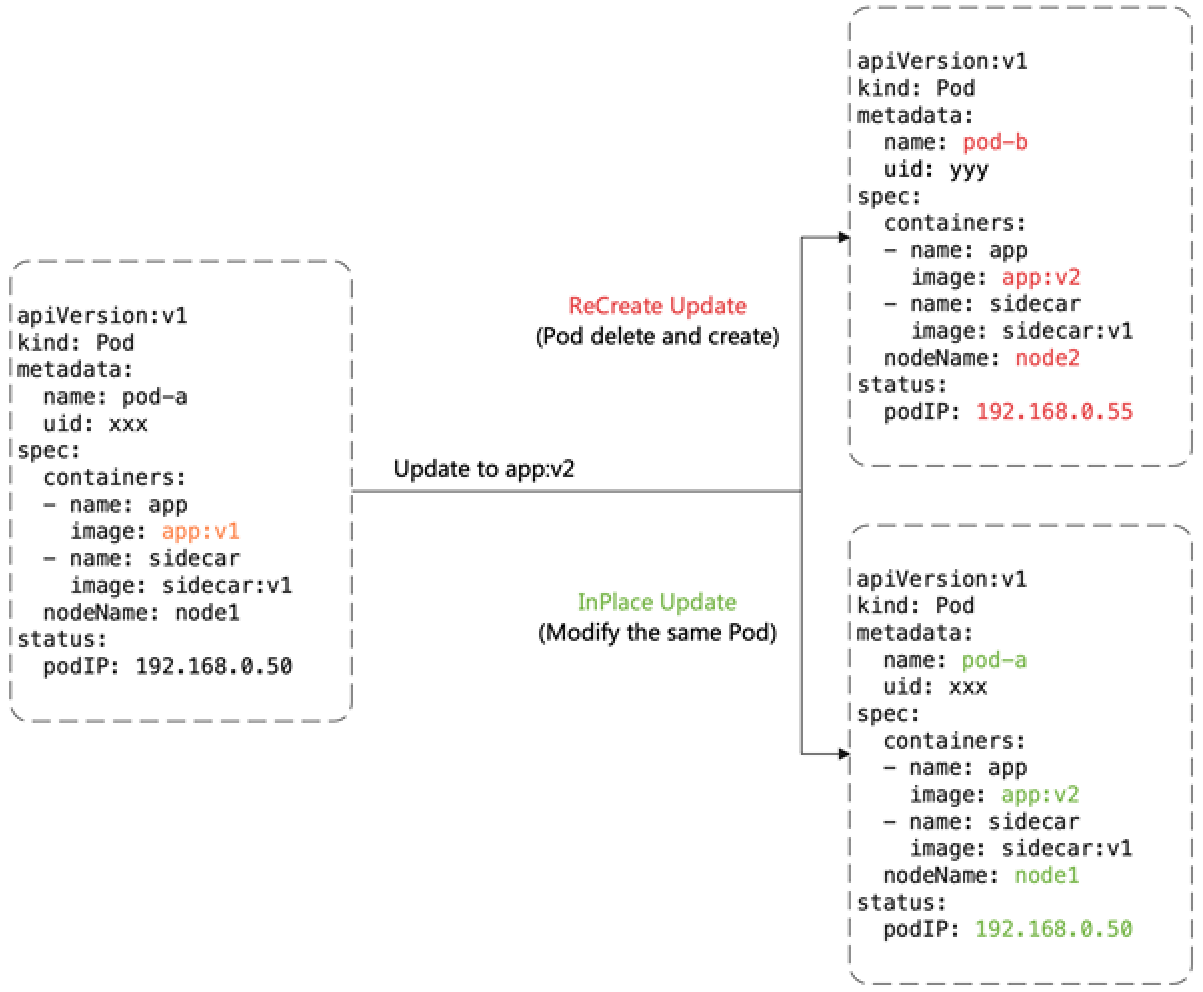
2015 天猫双11全球狂欢节
2015 TMALL 11.11 GLOBAL SHOPPING FESTIVAL
双11全球狂欢节
¥69,352,939,491
$ 10,904,207,335
无线成交占比:69%
淘宝异地多活
Taobao Cross-region Availability
支付宝异地多活
Alipay Cross-region Availability
阿里云服务全球客户
双11直播间
2015天猫双11全球狂欢节交易额达到683亿元

1. Cluster is locked after the load test
2. Resource is highly consolidated
3. Image of rich container is huge
4. Service discovery of large scale pod recreate is challenging

Recreate rolling update is not an option !

And we are not alone

# In-place Update Concept



```
apiVersion:v1
kind: Pod
metadata:
  name: pod-a
  uid: xxx
spec:
  containers:
  - name: app
    image: app:v1
  - name: sidecar
    image: sidecar:v1
  nodeName: node1
status:
  podIP: 192.168.0.50
```

Update to app:v2

ReCreate Update
(Pod delete and create)

```
apiVersion:v1
kind: Pod
metadata:
  name: pod-b
  uid: yyy
spec:
  containers:
  - name: app
    image: app:v2
  - name: sidecar
    image: sidecar:v1
  nodeName: node2
status:
  podIP: 192.168.0.55
```

InPlace Update
(Modify the same Pod)

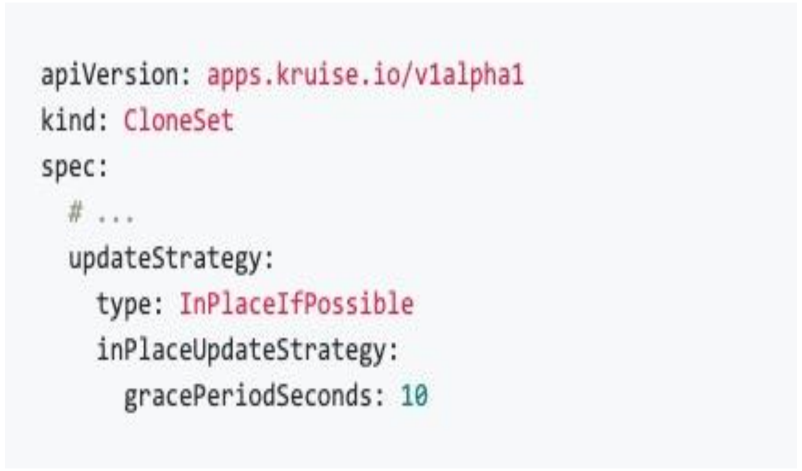
```
apiVersion:v1
kind: Pod
metadata:
  name: pod-a
  uid: xxx
spec:
  containers:
  - name: app
    image: app:v2
  - name: sidecar
    image: sidecar:v1
  nodeName: node1
status:
  podIP: 192.168.0.50
```

- Avoid additional cost of scheduling, allocating IP, allocating and mounting volumes, load volumes with data
- Faster image pulling, because of we can re-use most of image layers pulled by the old image and only to pull several new layers
- When a container is in-place updating, the other containers in Pod will not be affected and remain running.

Upstream [KEP: In-place rolling update]

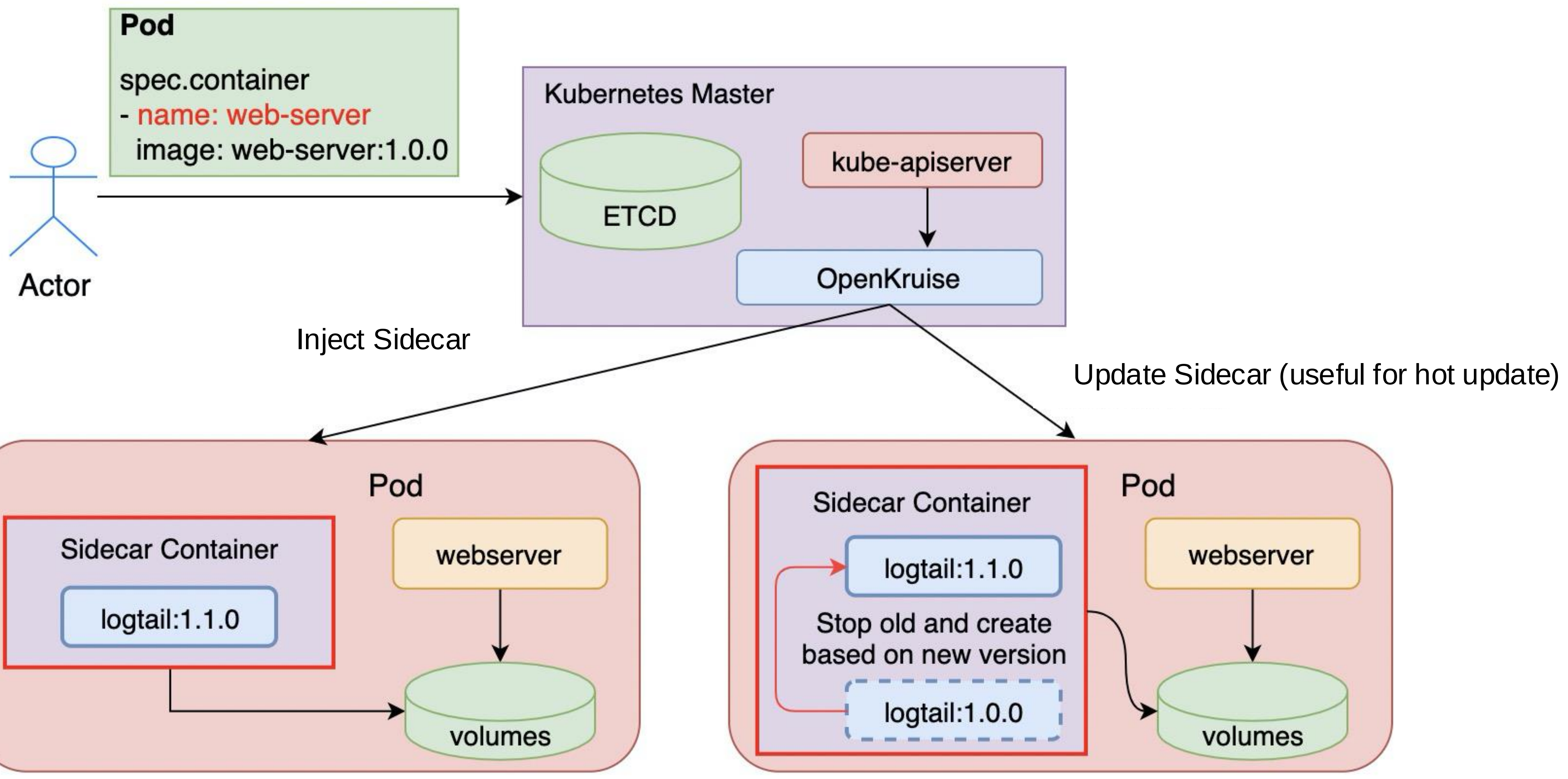
# Inplace-update & Immutable infra.

- Immuable Infra with replacable part
  - ~~Pod~~ Container as the element of Immuable infra
  - Only inplace-update if possible
  - Limit the replacable part to:
    - Image
    - Resource



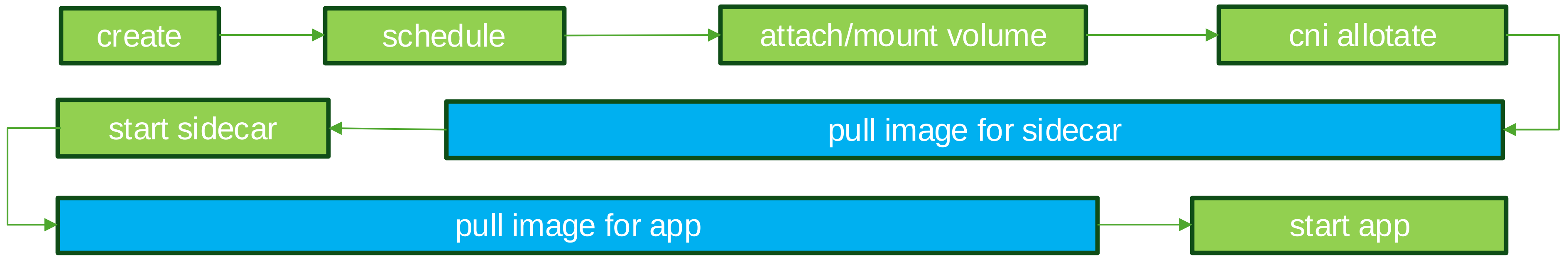
```yaml
apiVersion: apps.kruise.io/v1alpha1
kind: CloneSet
spec:
  # ...
  updateStrategy:
    type: InPlaceIfPossible
    inPlaceUpdateStrategy:
      gracePeriodSeconds: 10
```

## Use case – Sidecar management

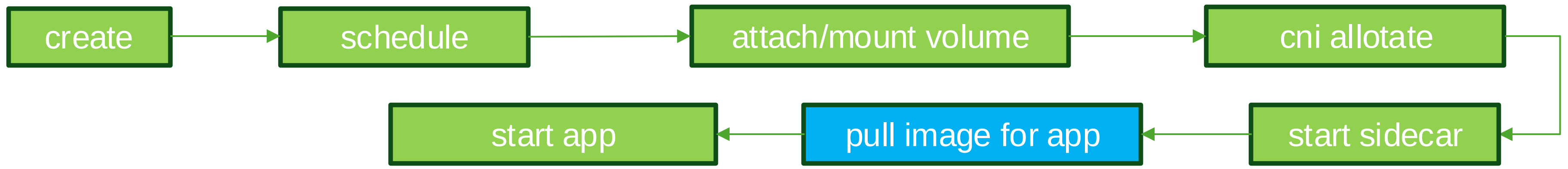


# Use case: Image pre-download

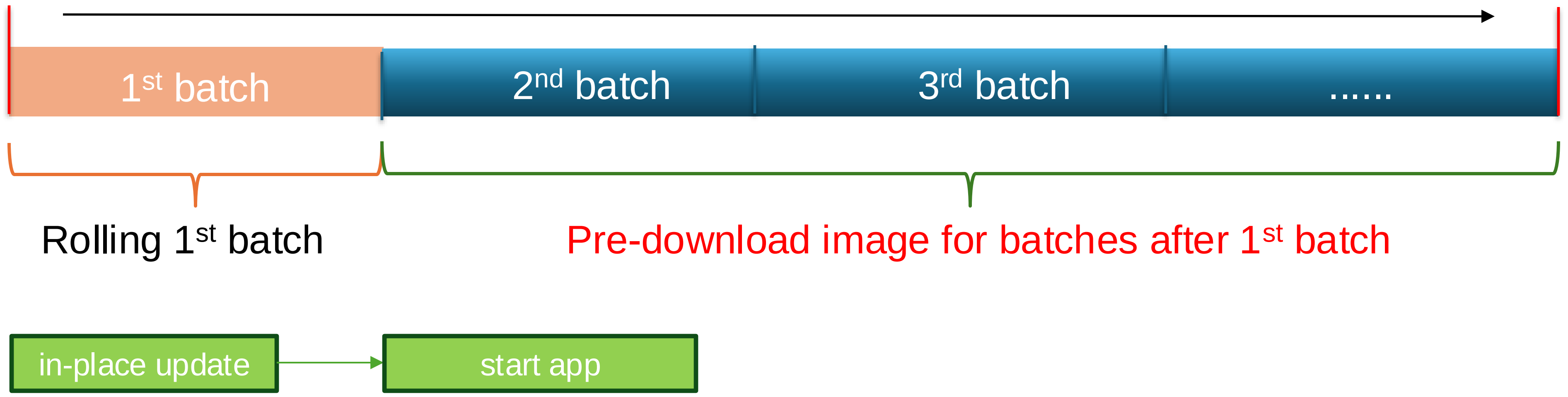
(1) Vanilla Pod creation：



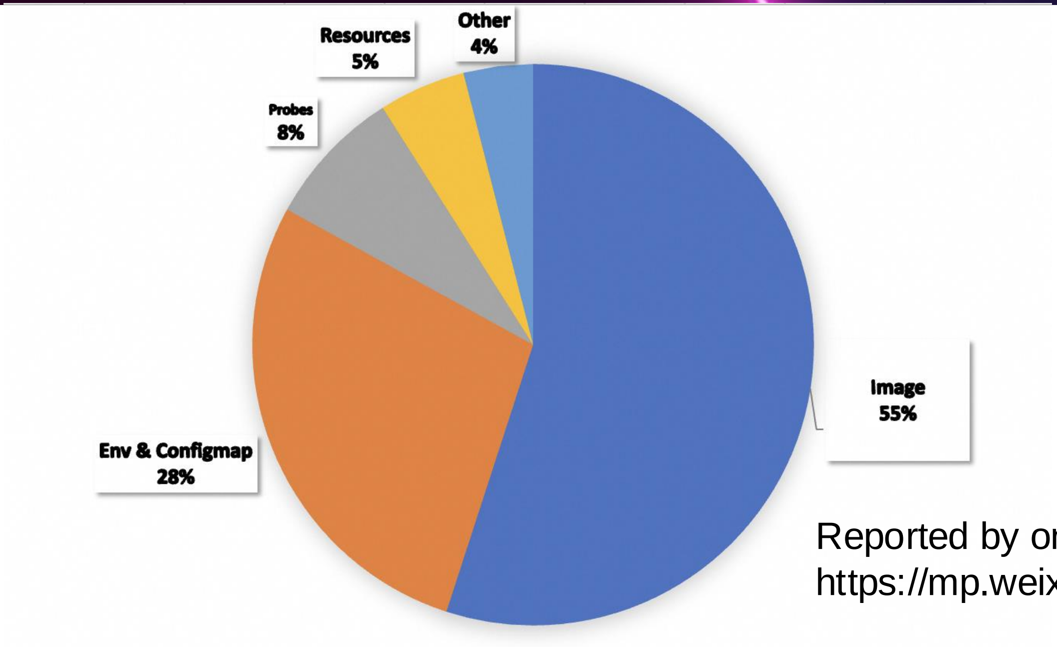
(2) Pod creation with image pre-download



(3) inplace-update with Pre-download



# challenges and solution: updatable fields



| fields | Mutable | Reported | Trigger update |
|---|---|---|---|
| Image | Y | Y | Y |
| Annotation | Y | N | N |
| Environment Command Argument | N | N | N |
| Resource | Y(since 1.27) | Y | Y |

Reported by one user:
https://mp.weixin.qq.com/s/hRvZz_bZfchmP0tkF6M2OA

Immutable:
1. Change Apiserver 😈
2. Wire with downward API 😇

Not trigger update
1. Update along with image
2. Kruise-daemon trigger update

Limit the updatable fields is by design

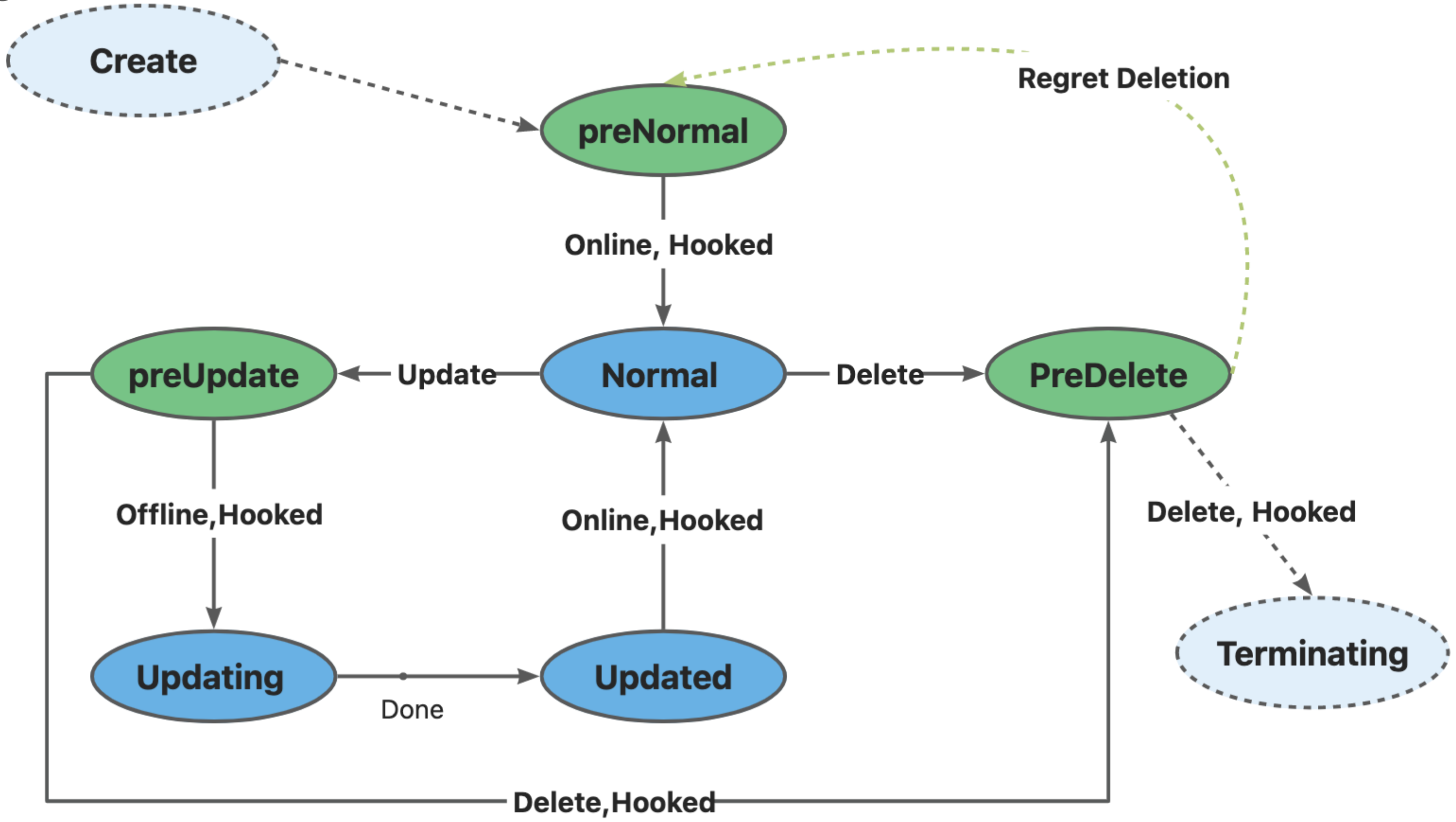
```
apiVersion: apps.kruise.io/v1alpha1
kind: CloneSet
spec:
 template:
  metadata:
   annotations:
    app-config: "... the real env value ..."
  spec:
   containers:
   - name: app
    image: app-image:v1
    env:
    - name: APP_CONFIG
     valueFrom:
      fieldRef:
       fieldPath: metadata.annotations['app-config']
```

# challenges and solution: pod lifecycle

**Usecases:**

- Remove traffic by mark pods not ready before inplace-update
- Ensure graceful shutdown before pod deletion
- Ensure graceful startup before pod becomes available

```
apiVersion: apps.kruise.io/v1alpha1
kind: CloneSet
spec:
 lifecycle:
   inPlaceUpdate:
     markPodNotReady: true
   preDelete:
     finalizersHandler:
     - example.io/unready-blocker
   preNormal:
     finalizersHandler:
     - example.io/unready-blocker
```



## challenges and solution: restart counts

- Limitation: Restart counts includes inplace-update count

Restart count since last update = restart count – restart count before last update

reported in pod status by kubelet

Recorded in pod annotations by kruise-manager before inplace-update

```
apiVersion: v1
kind: pods
metadata:
  annotations:
        apps.kruise.io/restart-count-before-last-update: '{"nginx":{"revision":"sample-d97f89dcf","restartCount":1}}'
        apps.kruise.io/pod-restart-count-before-last-update: "2"
```

## Any chance to upstream ?

- Top concerns
  - Security (Kritis & Notary)
  - Complexity
  - Immuable Conviction
- Opputunity
  - Good news: Inplace vertical pod scaling merged.
  - Bad news: 5 years to merge, still alpha and not integrated by workload

## New opportunity : resource resizing

### Use Cases

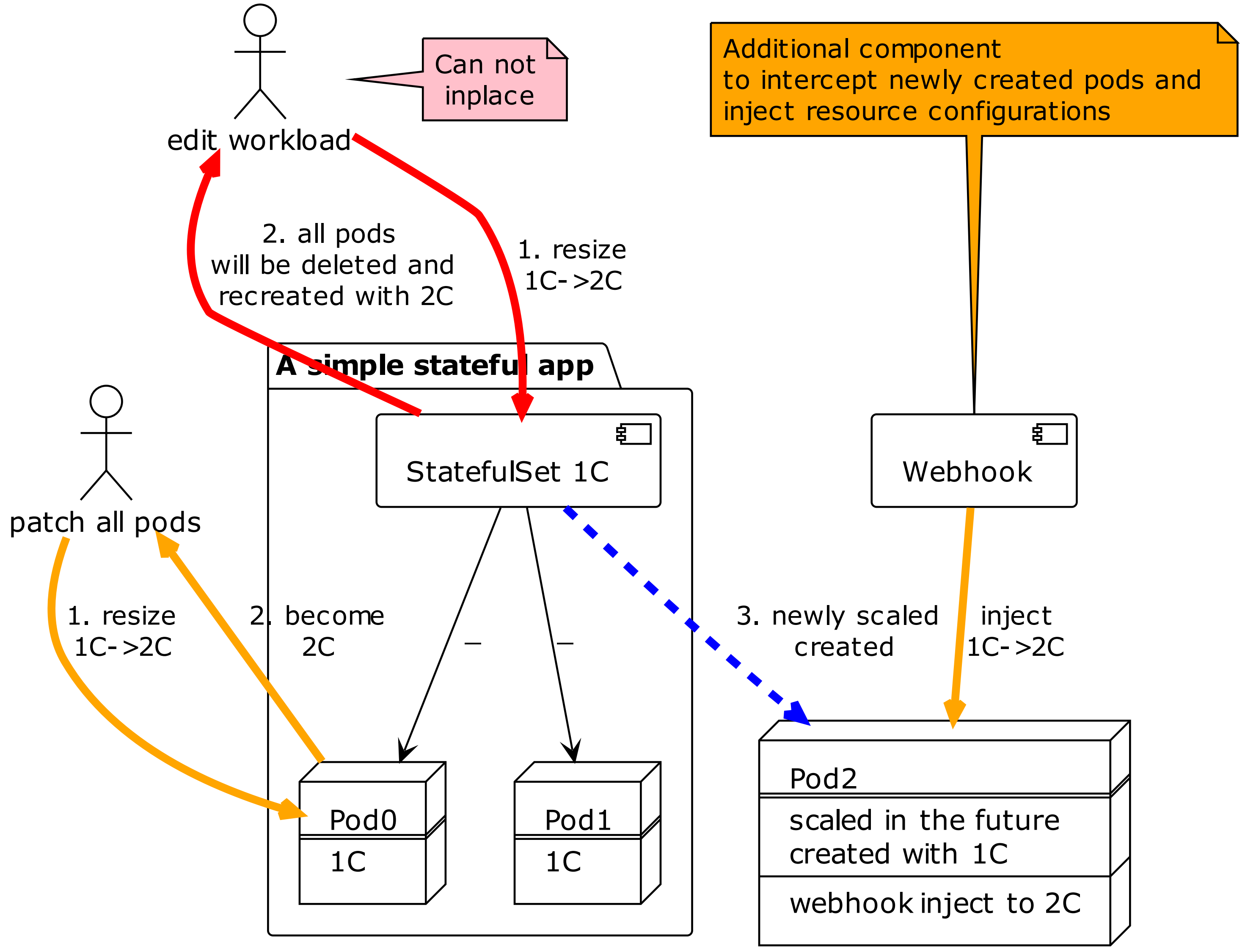
- load handled by the Pod has increased significantly, and current resources are not sufficient,
- load has decreased significantly, and allocated resources are unused,
- resources have simply been set improperly.

# New opportunity : resource resizing

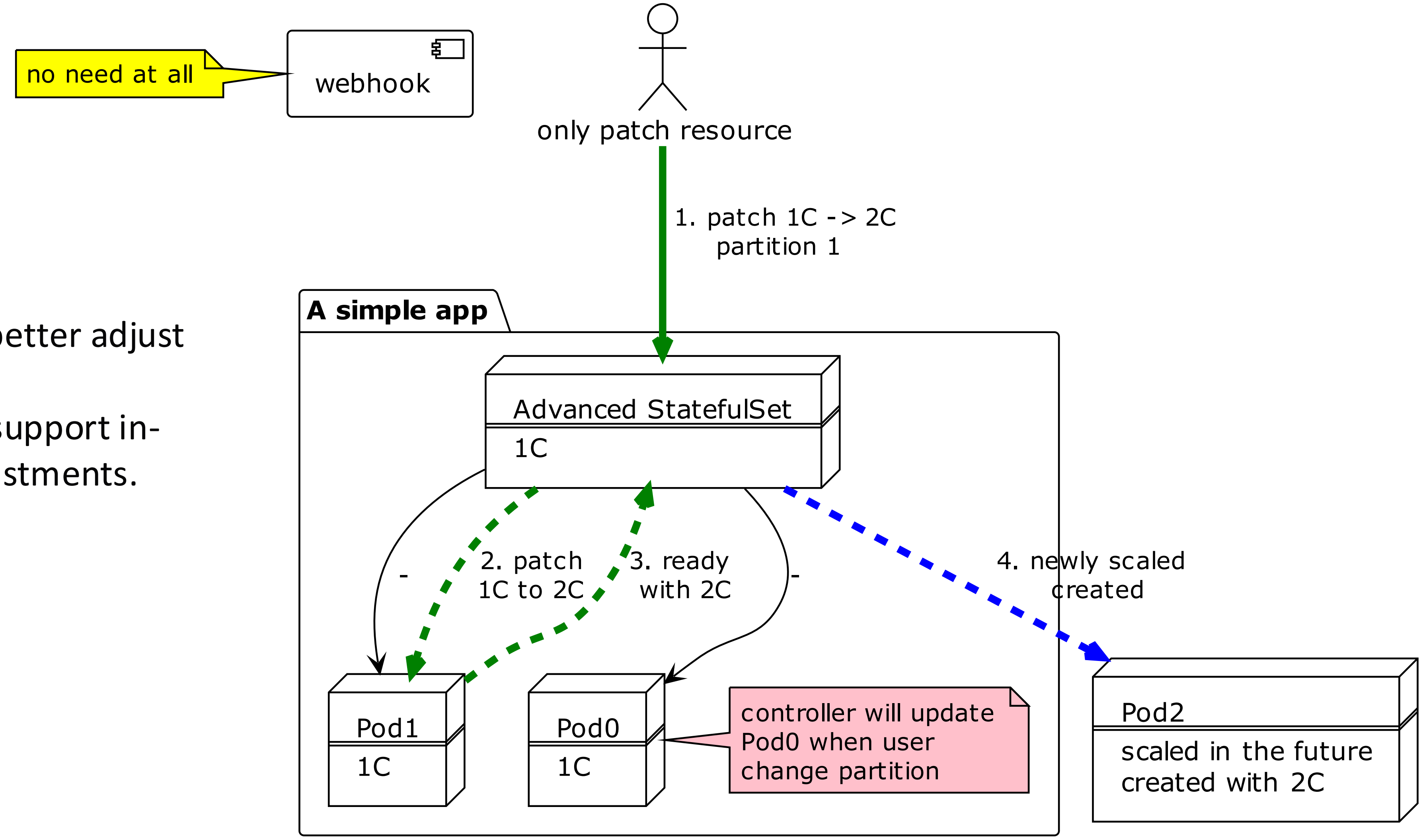
## Background

**Kubernetes 1.27: In-place Resource Resize for Kubernetes Pods (alpha)**

However, users cannot directly utilize this capability through workloads;
directly modifying the resources of workload will still result in the reconstruction of pods.



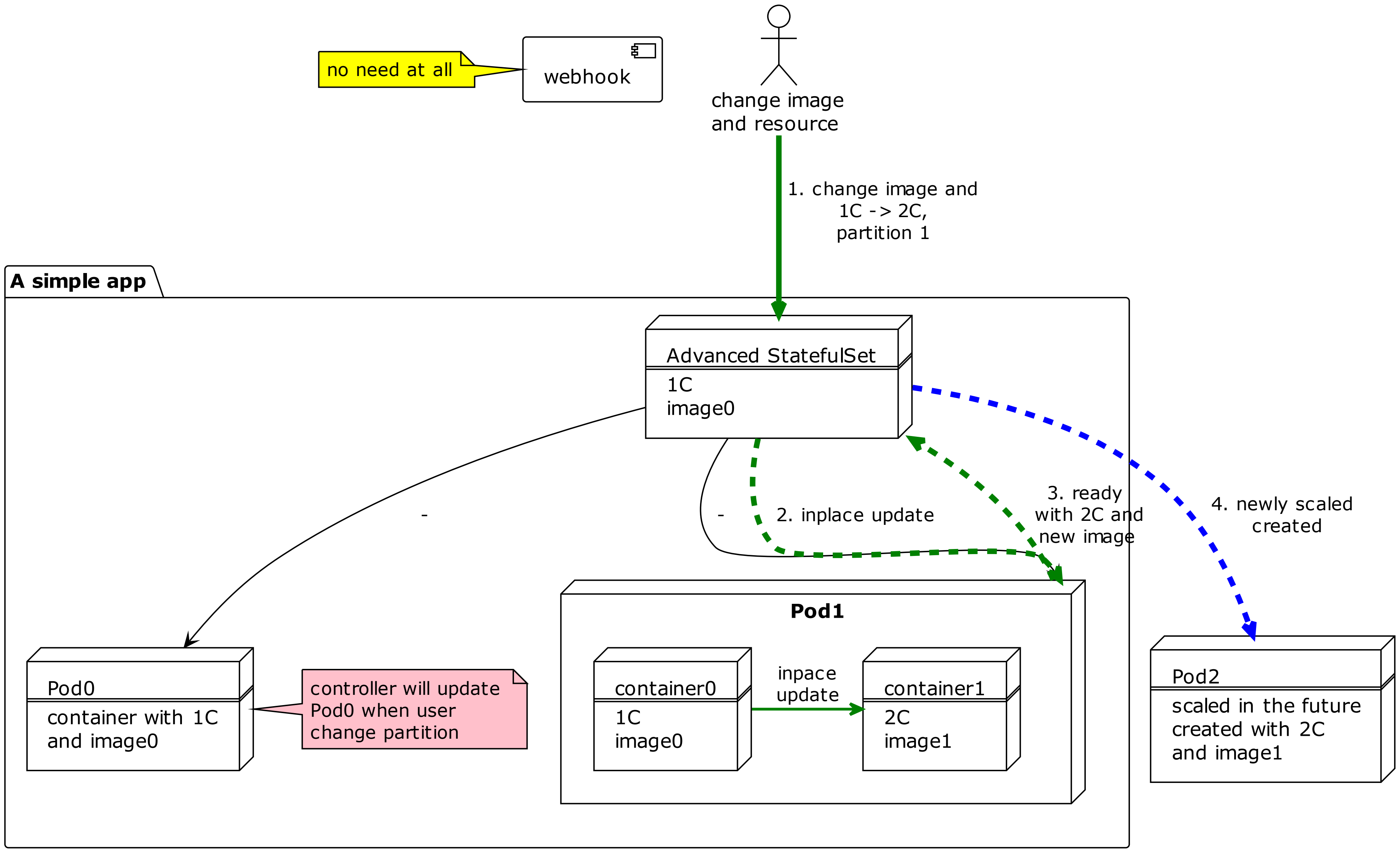
# New opportunity : resource resizing



## Solution

How can we enable users to better adjust resource configurations?
-- Let our workloads natively support in-place resource allocation adjustments.

# New opportunity : resource resizing


no need at all
webhook
change image and resource
1. change image and 1C -> 2C, partition 1
A simple app
Advanced StatefulSet
1C
image0
-
-
2. inplace update
3. ready with 2C and new image
4. newly scaled created
Pod1
container0
1C
image0
inplace update
container1
2C
image1
Pod0
container with 1C and image0
controller will update Pod0 when user change partition
Pod2
scaled in the future created with 2C and image1

# New opportunity : volume resizing

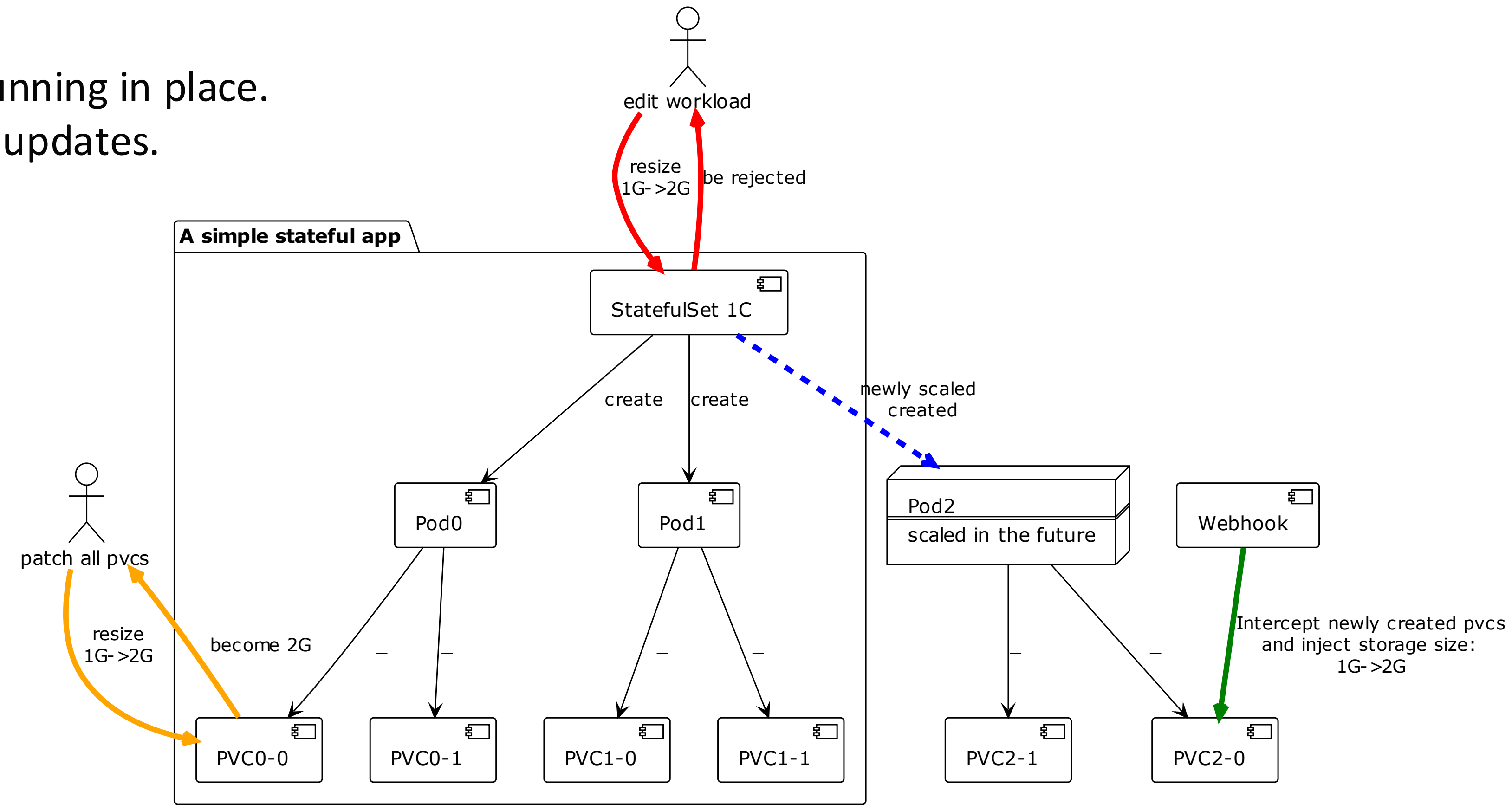
## Use Cases:

- Expand PVC size while the pod is running in place.
- Expand PVC size during pod rolling updates.

## Background :

**In Kubernetes v1.11+, Volume expansion is enabled by default.**

However, StatefulSet does not permit modification of this field.
Advanced StatefulSet allows modification but only takes effect for newly created PVCs.

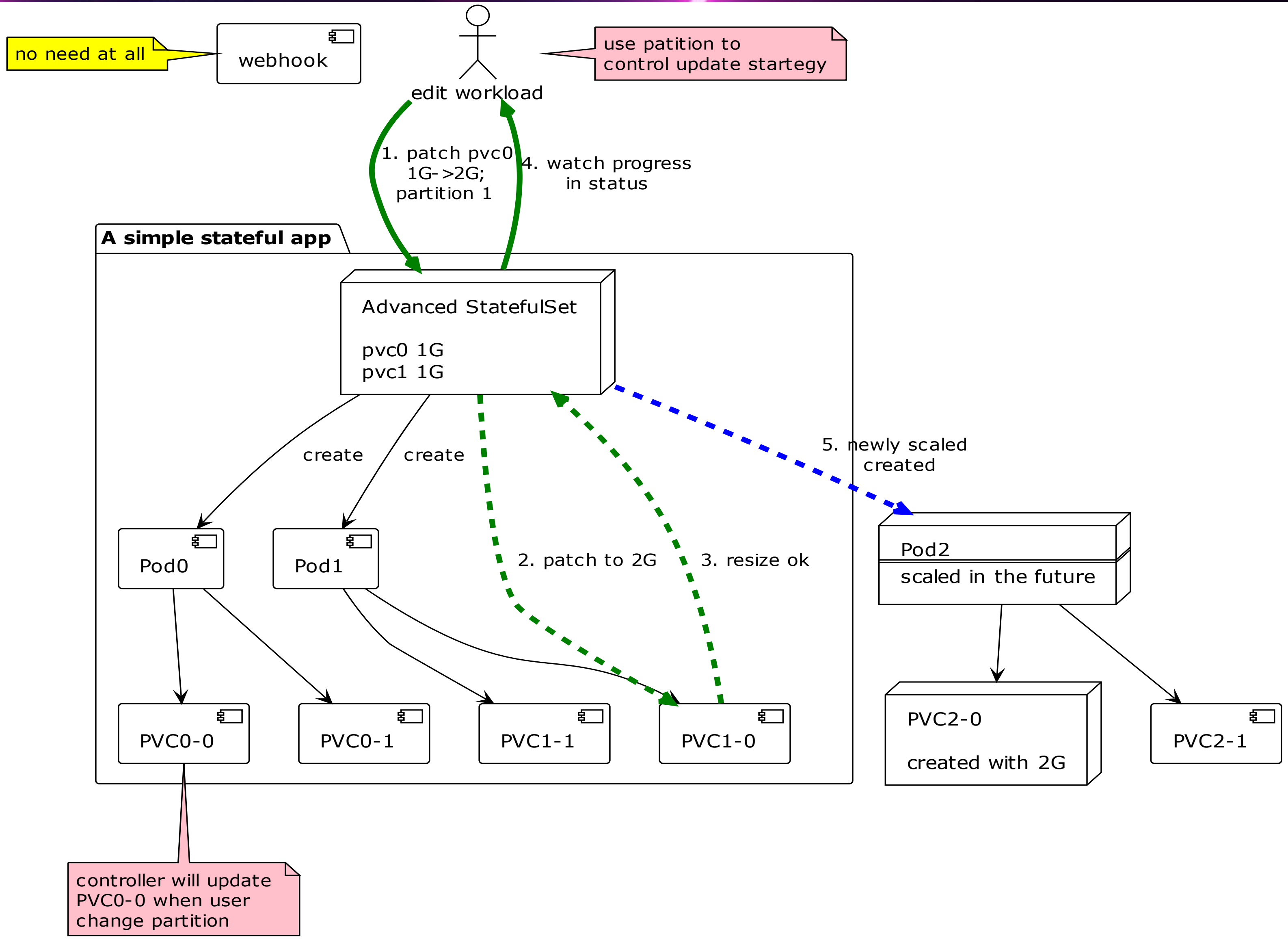


# New opportunity : volume resizing

**Solution ：**

How can we enable users to better resize pvc?

-- Let our workloads natively support in-place volume resizing.



## New opportunity : volume changes

### Use Cases

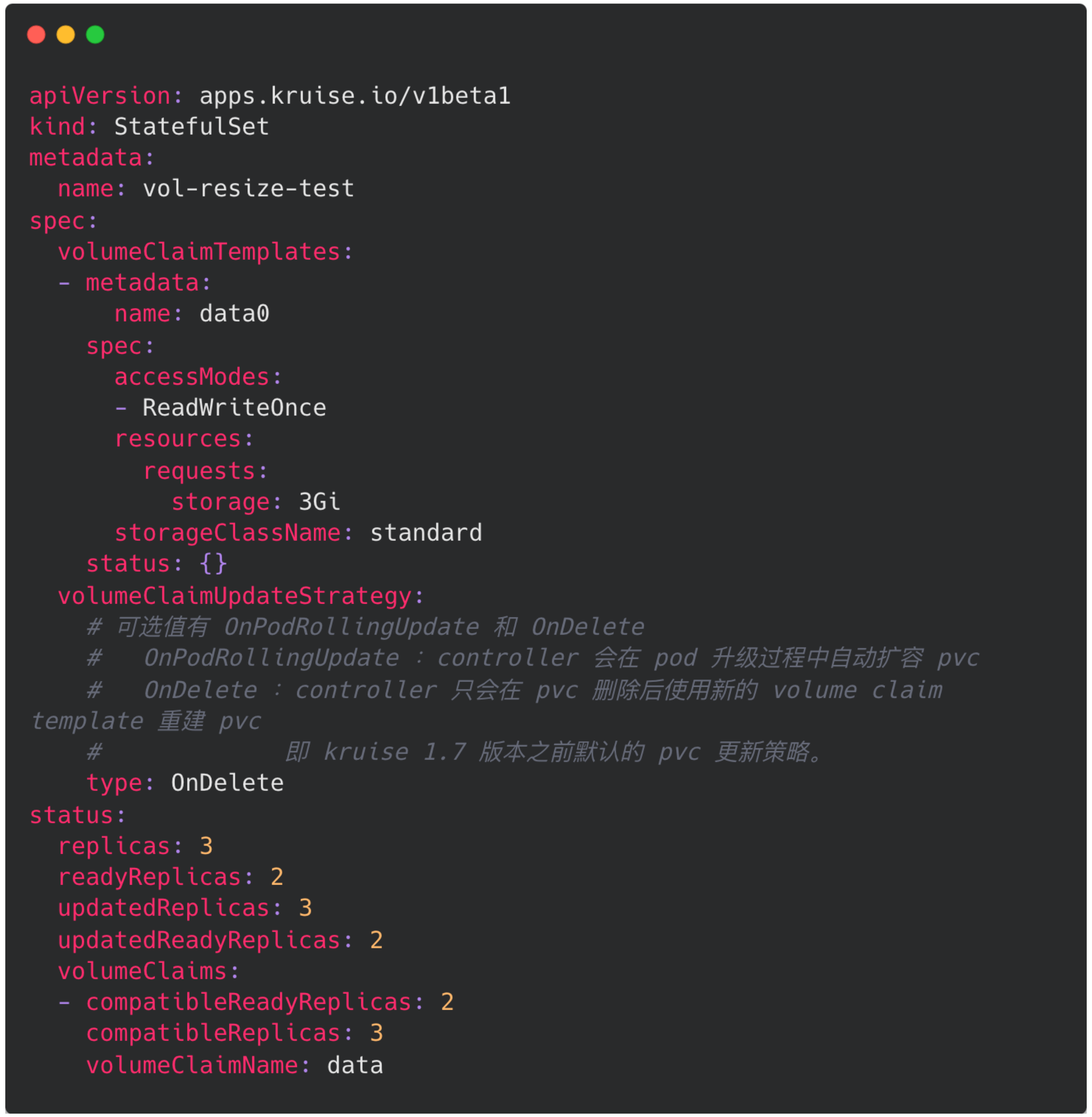
- Modify other PVC configurations, such as reducing PVC size.
    - Users are willing to ensure data synchronization on their own
    - Use snapshot backup and restore methods to reconstruct PVCs.
- Use volume snapshots to restore PVCs managed by the workload.

**Not only Resize?**

# New opportunity : volume changes

## Solutions

- OnDelete allows editing templates of pvc and recreate newest pvc when user delete it.
- New PVC will be newest pvc



```yaml
apiVersion: apps.kruise.io/v1beta1
kind: StatefulSet
metadata:
  name: vol-resize-test
spec:
  volumeClaimTemplates:
  - metadata:
      name: data0
    spec:
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 3Gi
      storageClassName: standard
    status: {}
  volumeClaimUpdateStrategy:
    # 可选值有 OnPodRollingUpdate 和 OnDelete
    #   OnPodRollingUpdate : controller 会在 pod 升级过程中自动扩容 pvc
    #   OnDelete : controller 只会在 pvc 删除后使用新的 volume claim
template 重建 pvc
    #             即 kruise 1.7 版本之前默认的 pvc 更新策略。
    type: OnDelete
status:
  replicas: 3
  readyReplicas: 2
  updatedReplicas: 3
  updatedReadyReplicas: 2
  volumeClaims:
  - compatibleReadyReplicas: 2
    compatibleReplicas: 3
    volumeClaimName: data
```

## New opportunity : volume changes

### Background

**In Kubernetes v1.17， VolumeSnapshot was introduced.**
**It can be used with the PVC's dataSource to create new PVC from snapshots.**

However, users cannot directly use snapshots to restore PVCs through the workload, and manual handling is also inconvenient.

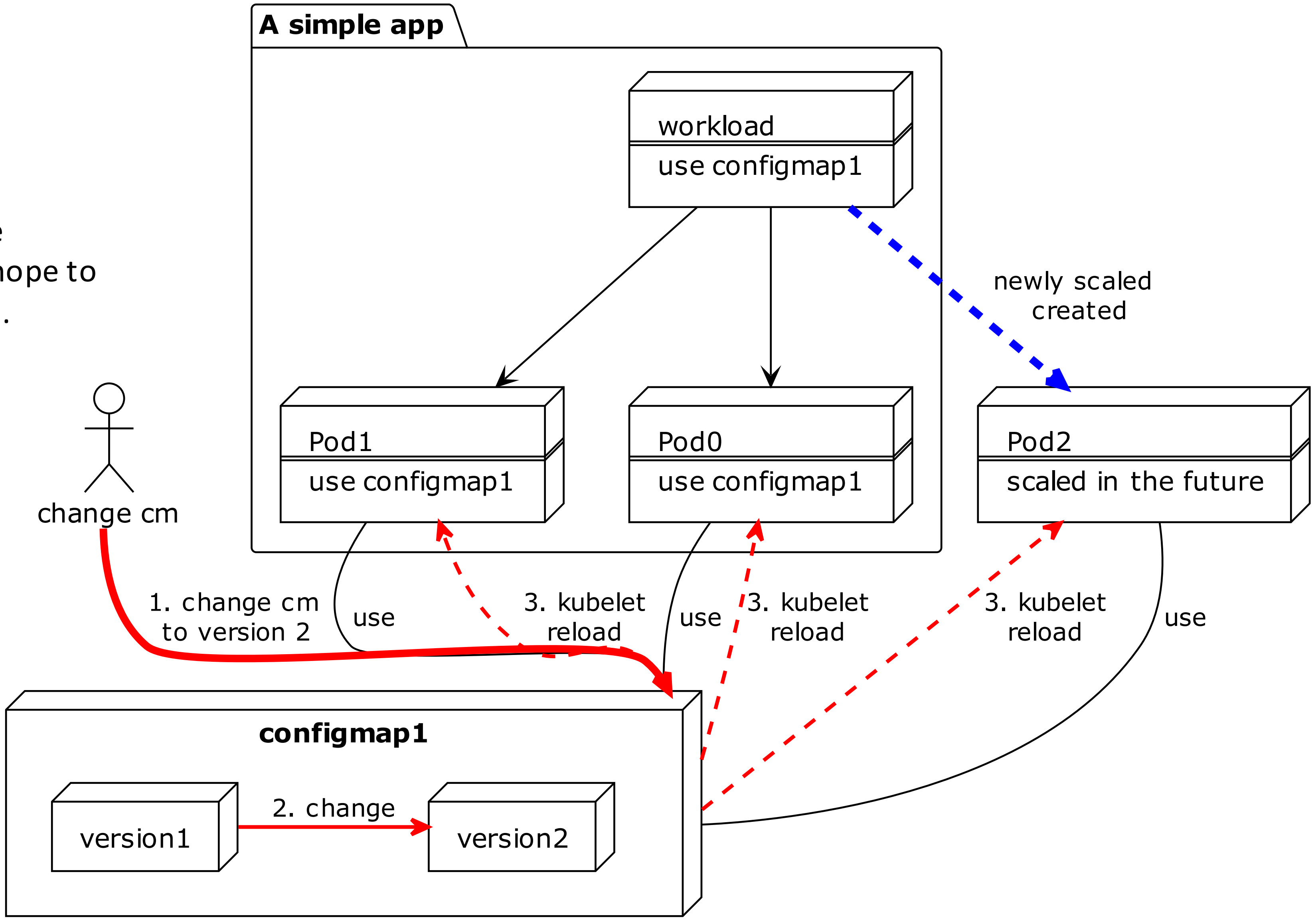
### Solutions

- [Plan] Define VolumeSnapshot information in VolumeClaimTemplates， maybe in annotations.
- [Plan] Implement a volume claim update strategy type about auto snapshot and restore from it.

# New opportunity : progressive configmap reload

## Use Cases

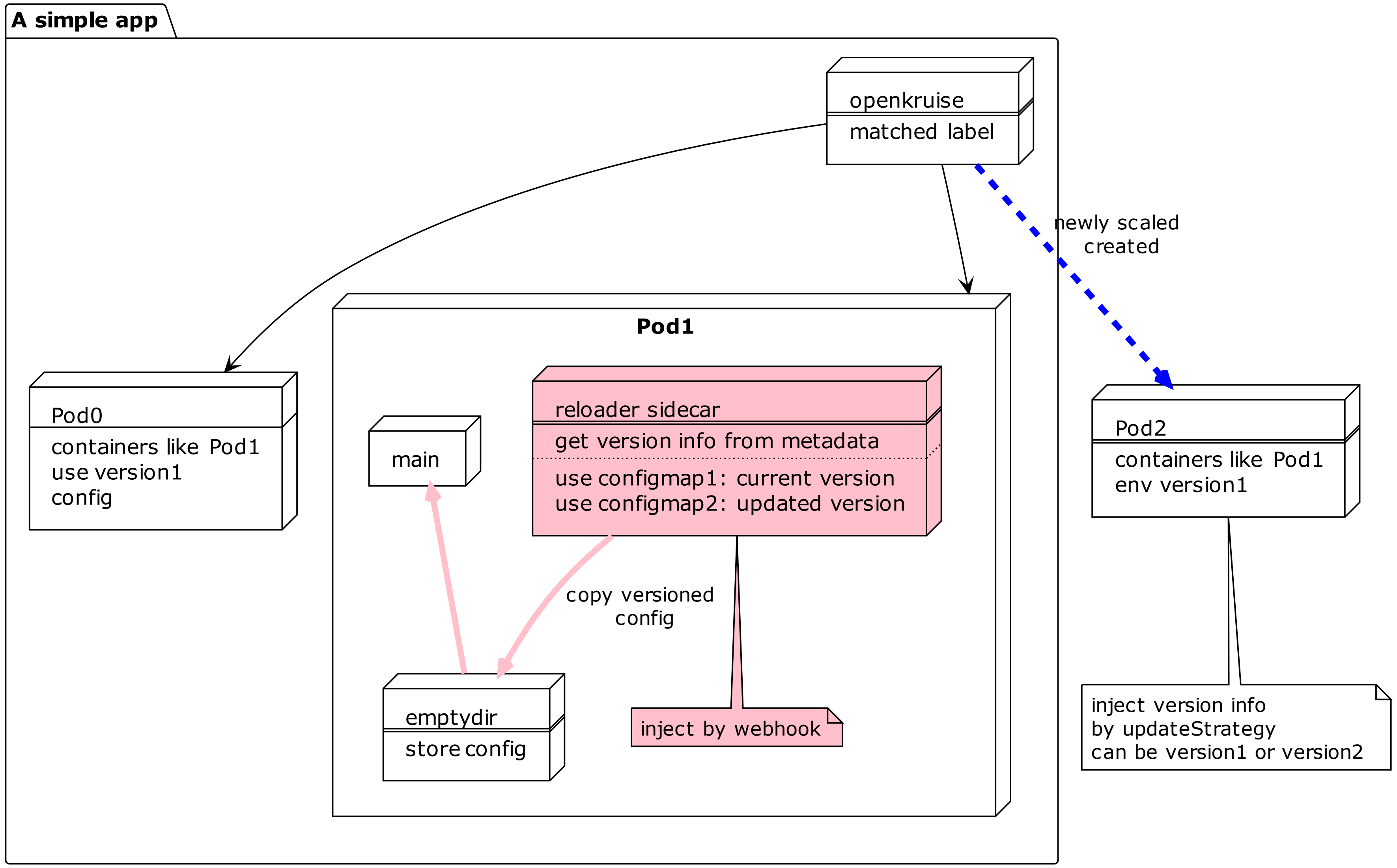
- Users use ConfigMaps to store configuration information and hope to implement a progressive reload.



# New opportunity : progressive configmap reload

## Solution:

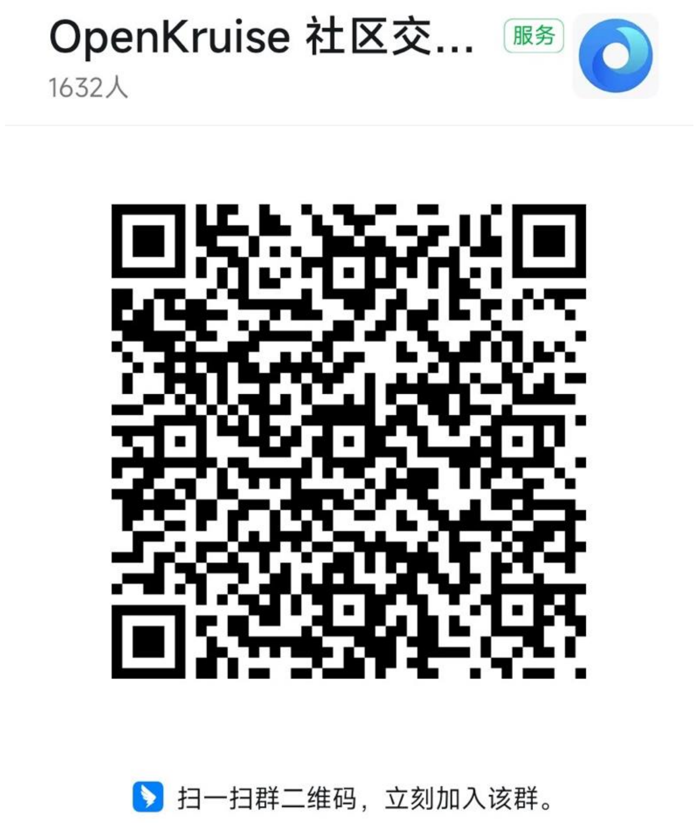
1. Support canary
2. Support label selector



# Keep In Touch

- Wechat Group: openkruise
- Dingding Group

https://kubernetes.slack.com/archives/openkruise



OpenKruise 社区交... 服务

1632人

扫一扫群二维码，立刻加入该群。