GOTC

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE , OPEN WORLD

「开源云原生计算时代」专场

本期议题：从 OpenKruise 看云原生应用负载发展趋势

王思宇（酒祝） 2021年07月10日

大纲

1. 云原生应用负载的定义与起源
2. 原生应用负载的现状与局限性
3. 从 OpenKruise 看未来的应用负载

1. 云原生应用负载的定义与起源

GOTC

什么是云原生应用负载?

操作方式:

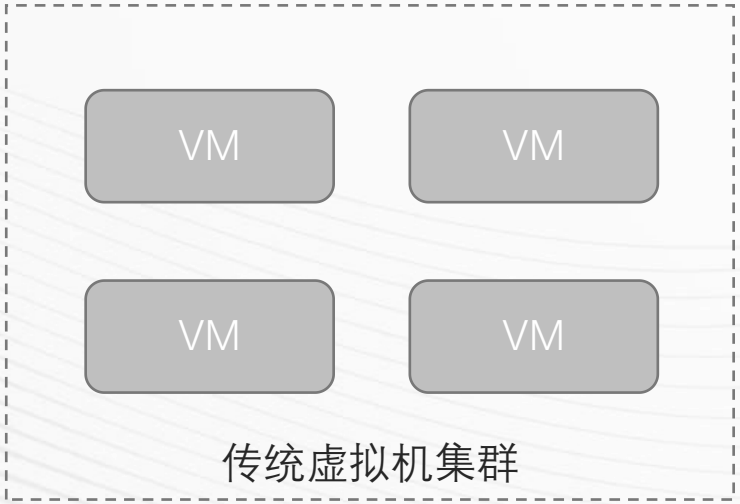
- 扩容几台VM
- 部署应用依赖环境
- 缩容哪几个VM
- 更新哪些VM中的软件包
- 巡检确保一致性

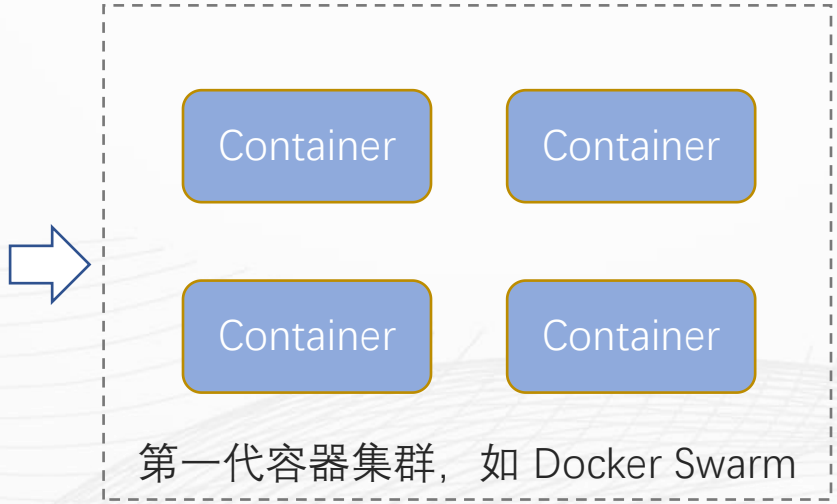
操作方式:

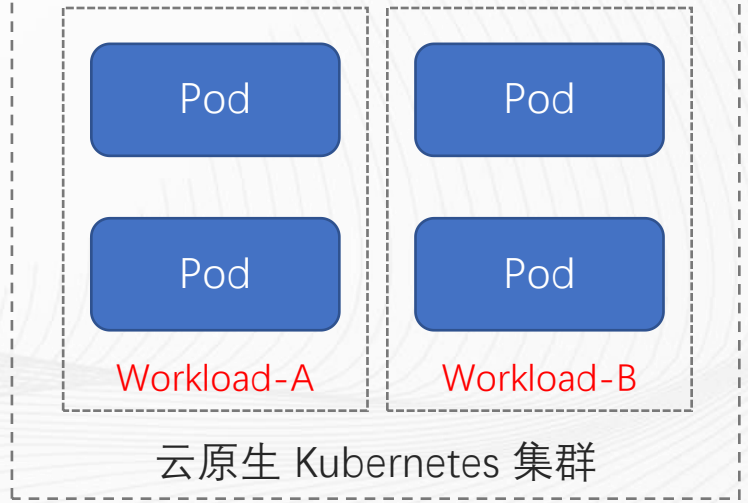
- 扩容几台容器（用某个应用镜像）
- 缩容哪几个容器
- 升级哪些容器到某个镜像
- 有宿主机故障，需要删除容器并重新扩一些新的实例

操作方式:

- 声明：应用 A 的镜像版本、需要的机器数
- 声明：应用 A 需要升级到某个新版本镜像
- 声明：守护进程 B 需要部署到所有宿主机上







1. 云原生应用负载的定义与起源

GOTC

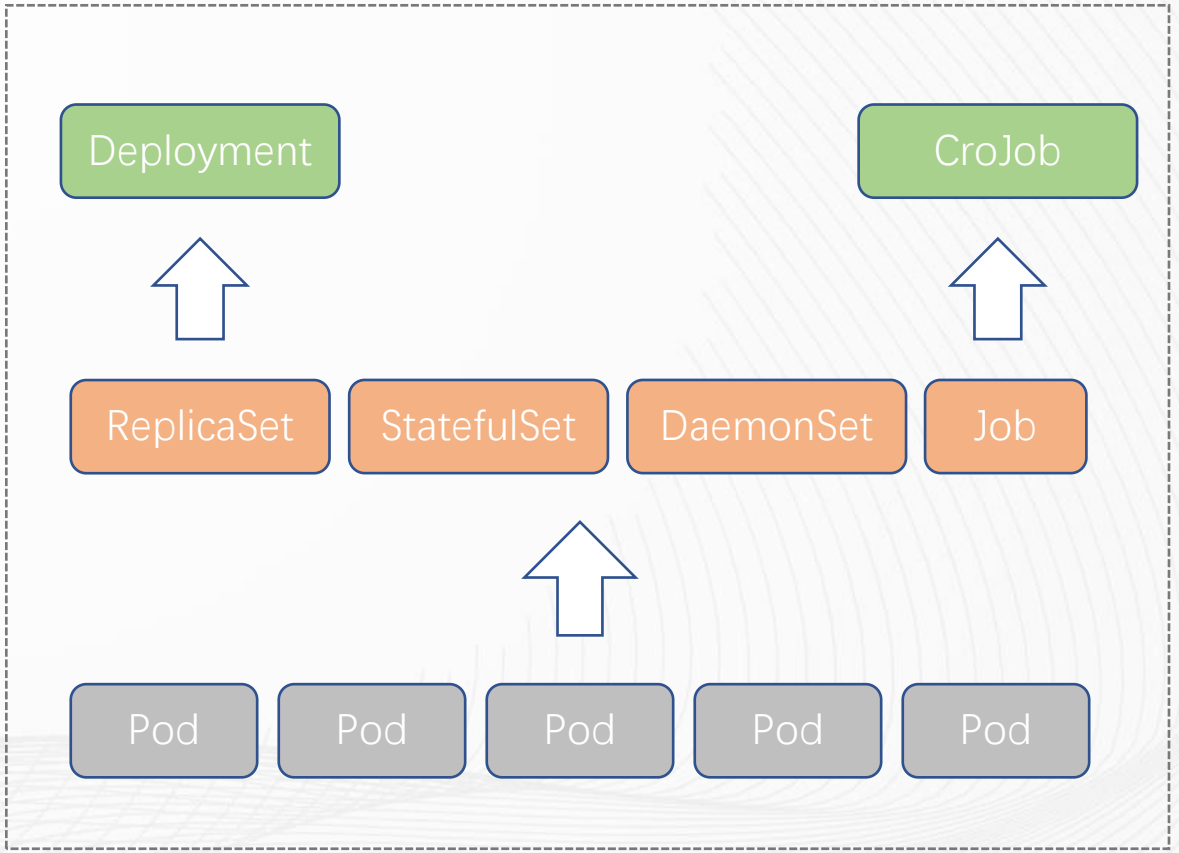
什么是云原生应用负载?

Kubernetes 终态语义:

- 用户定义期望状态与要求（spec）
- 组件（Controller）按用户的要求自动化地执行工作，并上报实际状态（status）

应用负载 or 工作负载（Workload）：

- 一般是指用来（直接或间接）定义一组工作实例（Pod）终态运行的资源类型
- 原生 Kubernetes 提供的内置 Workload 主要集中在 apps 和 batch 两个组下



1. 云原生应用负载的定义与起源

第一个云原生 Workload -- ReplicationController

Kubernetes 1.0 版本：

```
kind: ReplicationController
spec:
  replicas: 1
  selector:
    xx: xx
  template:
    # pod template for this workload
status:
  replicas: 1
```

最朴素的部署需求：

1. 应用 Pod 模板：定义镜像、环境变量、挂载卷等
2. 需要的实例数量
3. 标签选择器

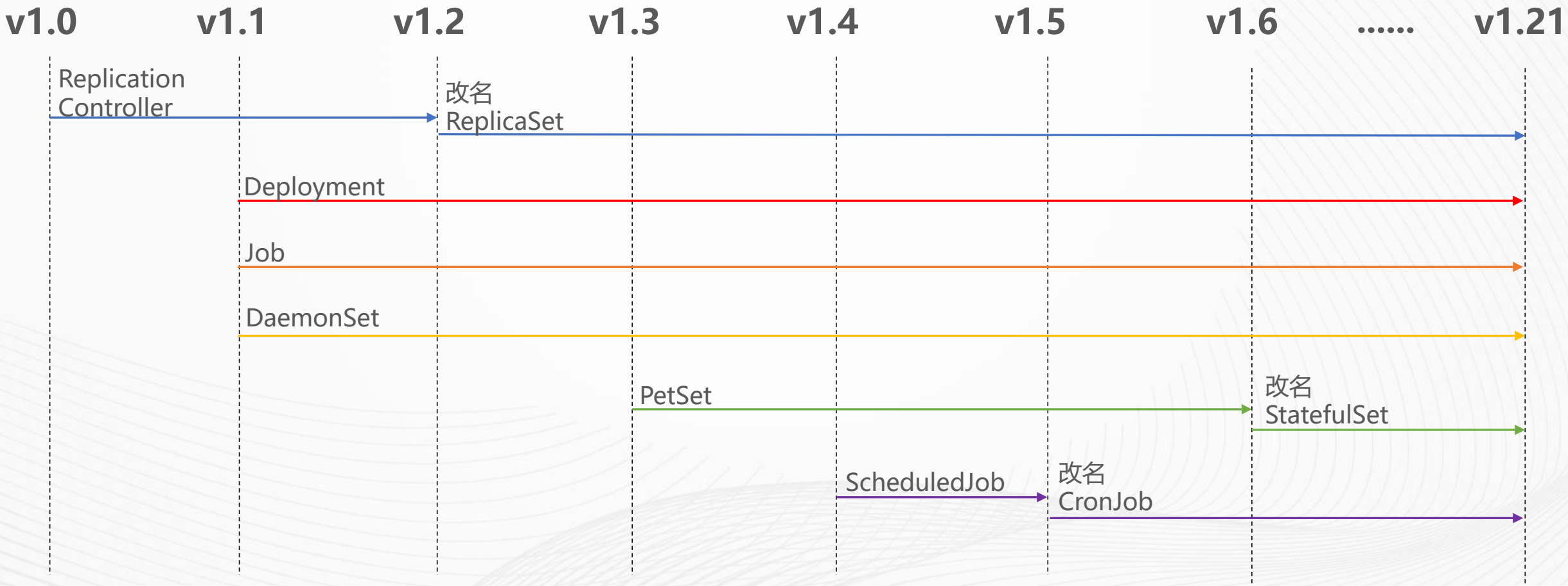
最简洁的状态上报：

1. 当前 Pod 实例数

2. 原生应用负载的现状与局限性

GOTC

原生 Workload 发展史

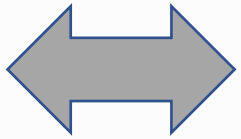


2. 原生应用负载的现状与局限性

“保守”基础能力与“先进”生产力的矛盾

原生 Workload 宗旨：

- 通用性
- 稳定性
- 基础部署能力
- 不倾向于做较大改动



来自生产环境的需求：

- 多样化的业务部署发布要求
- 追求效率、性能
- 使用上灵活可控
- 垂直领域（游戏、直播...）的场景诉求

2. 原生应用负载的现状与局限性

GOTC

原生 Workload 的功能迭代

生产需求（举例）	状态	说明
Deployment、ReplicaSet 执行缩容时可以指定 Pod 删除	Merged	从 1.5 版本开始有用户提出需求，直到 1.21 版本才加入了 deletion-cost 策略（alpha）
StatefulSet 支持并行发布（当前只能 one by one 串行升级）	Reviewing	2018.12 提出 KEP，2019.8 KEP 合并、提交代码 PR，至今仍在 reviewing
Pod 发布支持 原地升级（in-place update）	Rejected	sig/apps 评估功能上对现有 Workload API 和 controller 改动太大，且会打破 Pod 不可变

2. 原生应用负载的现状与局限性

GOTC

如何看待原生 Workload 的局限性?

From **Matt Farina** and **SIG Apps:**

Upstream Kubernetes won't likely accept every new type of controller. The direction is generally to have them installed as 3rd party controllers. The bar to move something into core Kubernetes is now very high.

Some successful features may make it into Kubernetes core. Others will likely always be recommended to be handled as 3rd party.

SIG Apps does not believe there should be one home for "all workload-related CRD ideas". They can happen in many places and innovation can happen under many projects.

3. 从 OpenKruise 看未来的应用负载

GOTC

OpenKruise：社区最成熟的扩展应用负载项目

https://github.com/openkruise/kruise

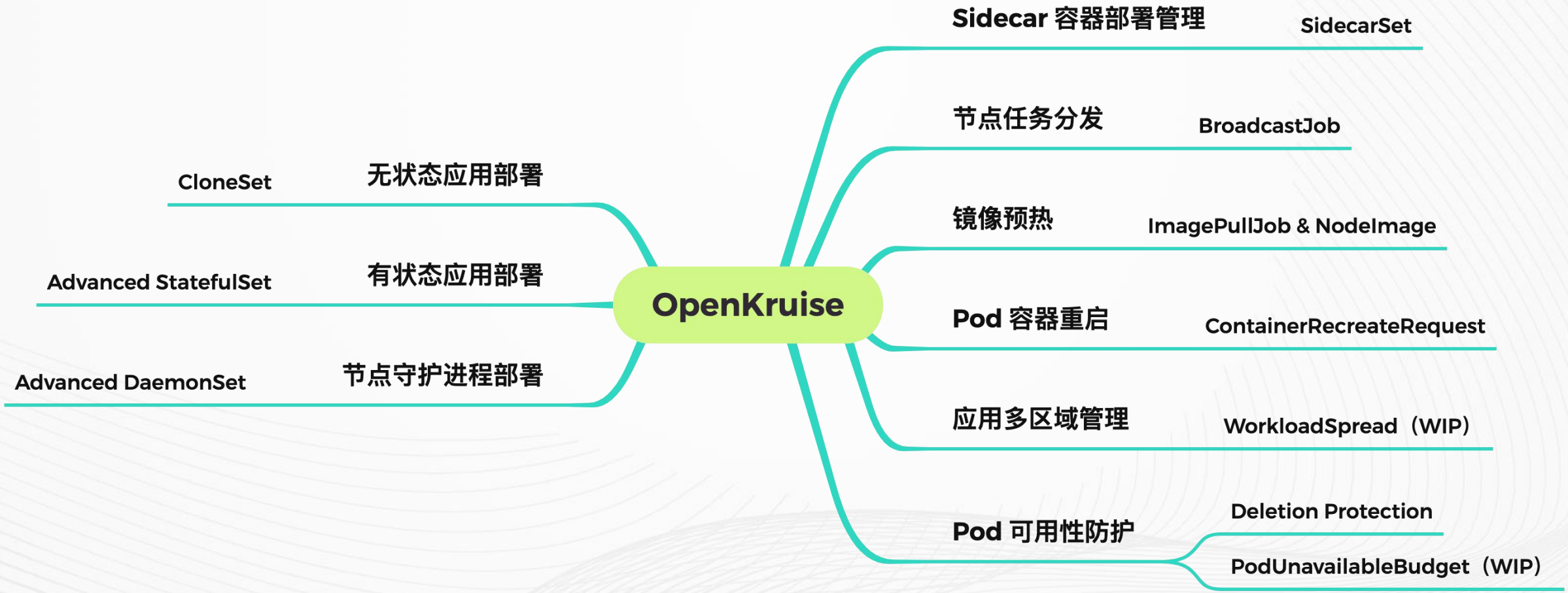
Kruise 是 Cruise 的谐音，'K' for Kubernetes，寓意 Kubernetes 上应用自动化巡行

- 阿里云开源的云原生应用自动化套件
- CNCF 托管的 Sandbox 项目
- SIG Apps 官方认可的 3rd party 应用负载项目
- 阿里巴巴经济体上云全面使用的部署基座

3. 从 OpenKruise 看未来的应用负载

GOTC

OpenKruise 功能一览



3. 从 OpenKruise 看未来的应用负载

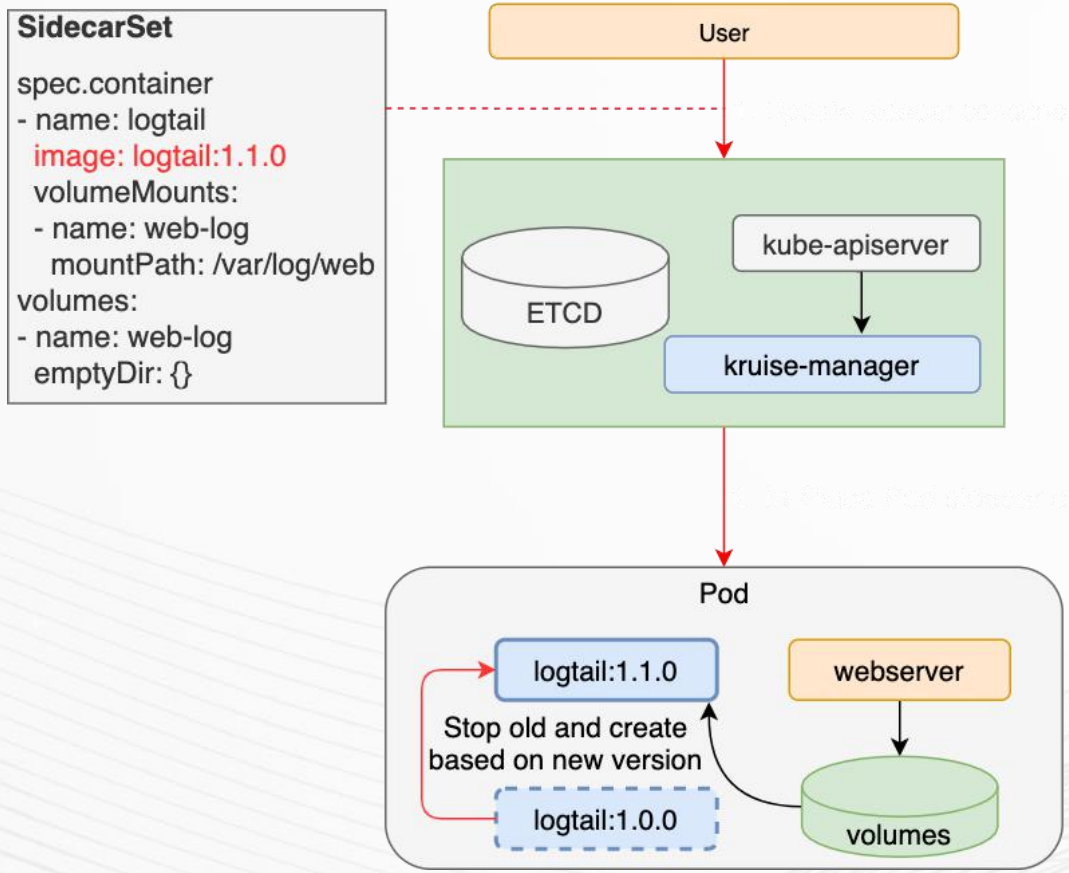
GOTC

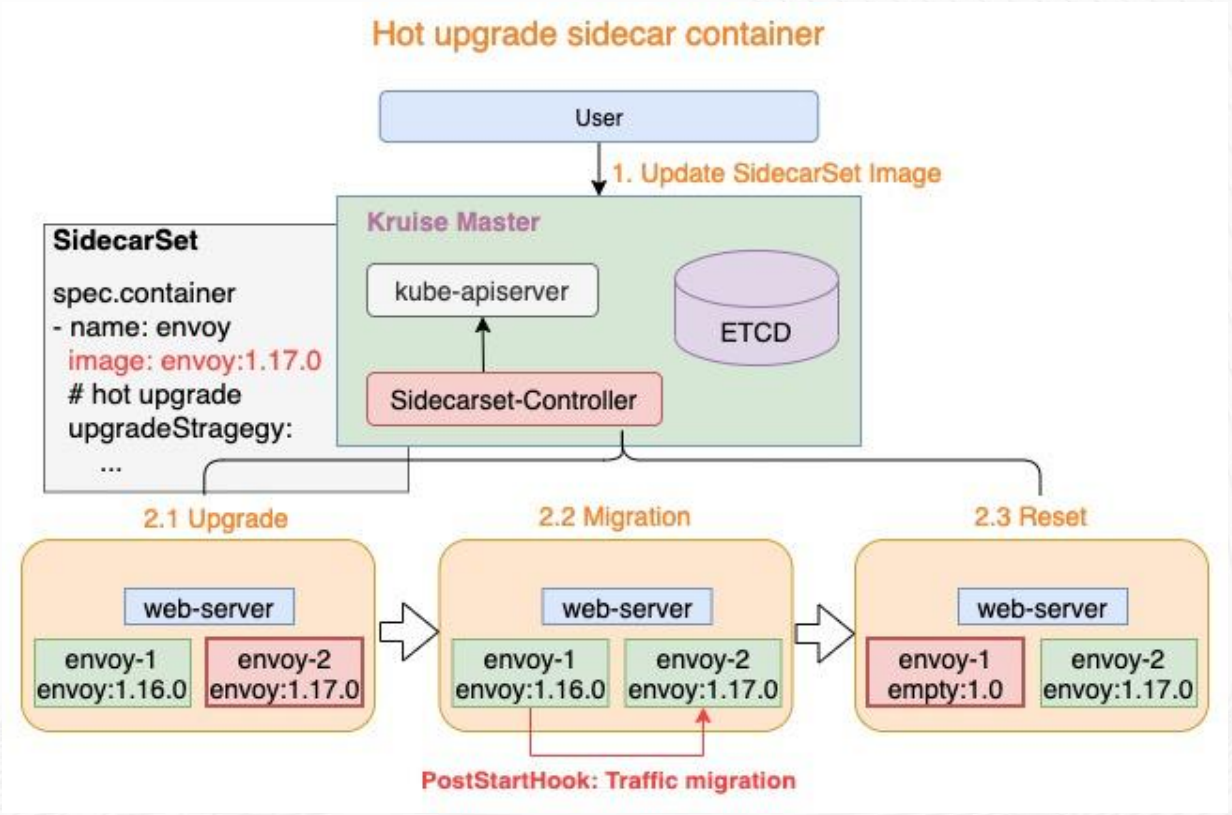
规模化：CloneSet vs Deployment 差异性（部分）

功能分类	功能	CloneSet	Deployment
伸缩	支持 PVC per Pod	可配置 pvc template 实现	不支持
	指定 Pod 缩容	支持 3 种方式： scale strategy，pod deletion label，deletion cost	支持 deletion cost（since v1.21）
	先扩再缩（主动迁移）	通过 maxSurge+maxUnavailable 弹性迁移	不支持
	根据 topology 打散缩容	优先根据 Pod Topology Spread Constraints 打散，其次根据单机实例数打散	不支持/非打散
发布	Pod 原地升级	当前支持修改 image 原地升级，后续支持 env from annotations/labels 原地升级	不支持
	灰度（分批）发布	通过 partition 控制灰度数量/比例	不支持
	发布顺序可配置	可配置 priority 优先级、scatter 打散发布策略	不支持
	原地升级提前镜像预拉取	灰度升级过程中，优先在后续节点上预先拉取新镜像，提高发布效率	不支持

3. 从 OpenKruise 看未来的应用负载

深度化：使用 SidecarSet 定义与升级 sidecar 容器





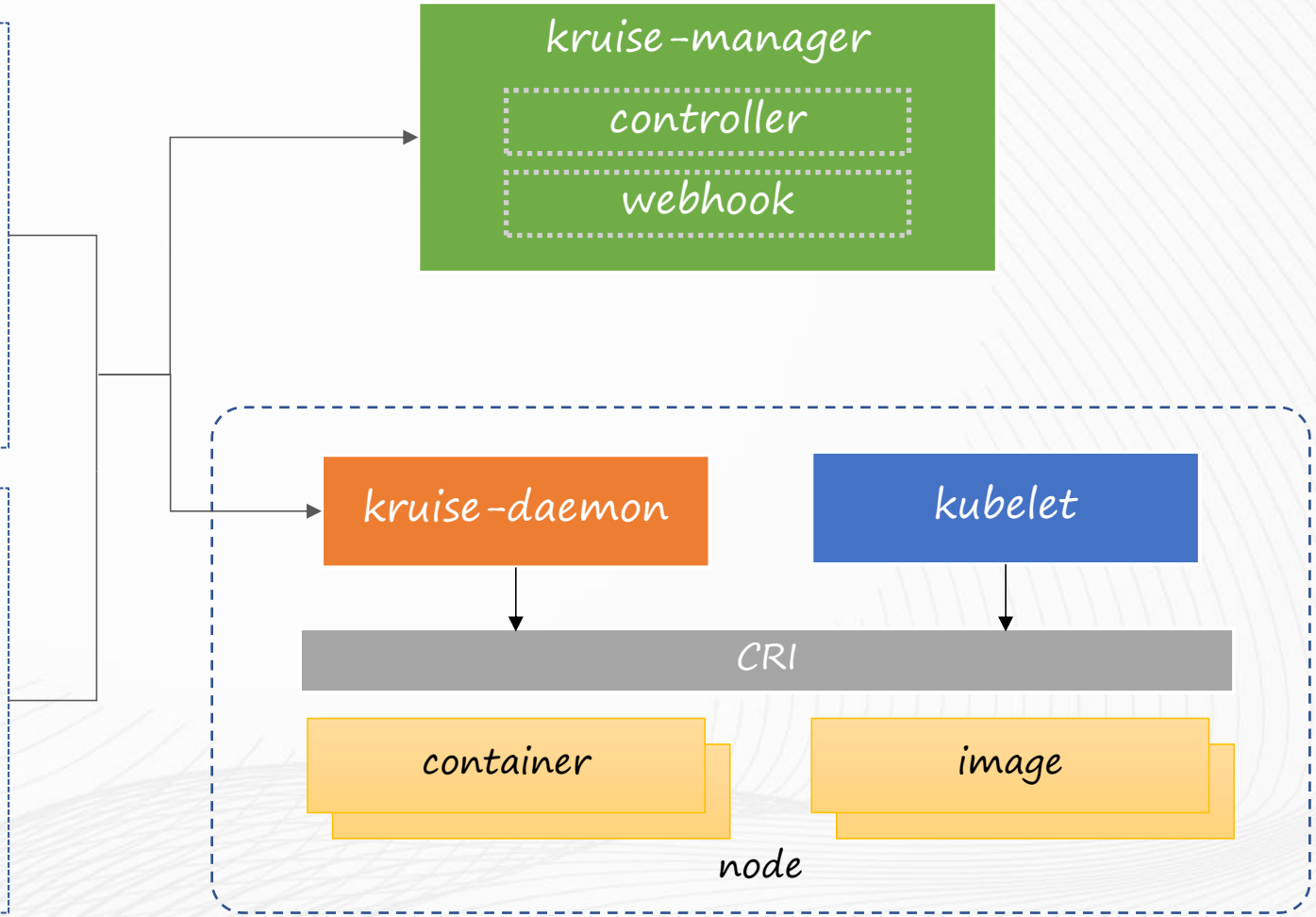
3. 从 OpenKruise 看未来的应用负载

GOTC

全面化：利用 daemon 实现 Pod 容器重启与镜像预热

```yaml
apiVersion: apps.kruise.io/v1alpha1
kind: ContainerRecreateRequest
spec:
  podName: pod-xxx
  containers:
  - app
  - sidecar
  strategy:
    # ...
```

```yaml
apiVersion: apps.kruise.io/v1alpha1
kind: ImagePullJob
spec:
  image: hub/app:latest
  parallelism: 10
  selector:
    # ...
  pullPolicy:
    # ...
```



3. 从 OpenKruise 看未来的应用负载

GOTC

抽象化：[WIP] 应用多区域管理

Pod Topology Spread Constraints
（从 Kubernetes 1.19 提供 [stable] ）：

```yaml
topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: topology.kubernetes.io/zone
    labelSelector:
      matchLabels:
        foo: bar
```

缺点：

- **只能按 topology 均等打散，不支持比例、不支持优先级、不支持动态选择**
- **调度器无法干预缩容，依赖 workload 自身实现**

```yaml
apiVersion: apps.kruise.io/v1alpha1
kind: WorkloadSpread
spec:
  targetRef:
    # a Deployment or CloneSet
  subsets:
  - name: subset-a
    requiredNodeSelectorTerm:
      # zone-a
    maxReplicas: 10 | 30%
  - name: subset-b
    # ...
```

特性：

- **无需修改 Workload or Pod，可直接配合使用**
- **多个拓扑或区域维度，按比例打散，或按固定数量分配**
- **支持多区域调度顺序选择，在前一个 subset 数量满足或资源不足时选择下一个 subset**
- **通过 deletion-cost 自动设置 Workload 缩容优先级**

3. 从 OpenKruise 看未来的应用负载

GOTC

总结：未来的云原生应用负载发展趋势 -- 规模化、深度化、全面化、抽象化。在原生 Kubernetes 应用负载基础上，通过 3rd party 扩展出符合更多真实生产业务场景的高级能力。

- 以 OpenKruise 为例，它来自阿里巴巴多年来容器化、云原生的技术沉淀，既是阿里经济体上云的部署基座，也是紧贴上游社区标准、适应互联网规模化场景的最佳实践之一。

- 社区用户包括：阿里巴巴集团、蚂蚁集团、Lyft、携程、OPPO、斗鱼TV、苏宁、Boss直聘、申通、小红书、有赞、欢聚集团 等 30+ 国内外企业。

https://github.com/openkruise/kruise | https://openkruise.io

微信号：

社区钉钉群：

扫一扫群二维码，立刻加入该群。

GOTC

THANKS

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE