Operator Pattern：用 Go 扩展 K8s 的最佳实践

×

吴学强

ApeCloud
KubeBlocks Maintainer & 研发总监

目录

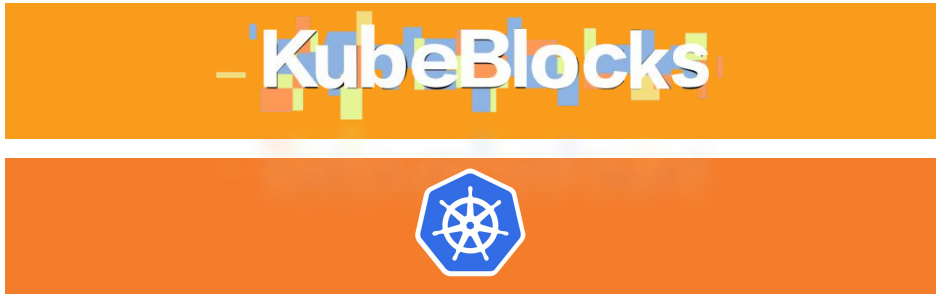
我们是谁

云猿生（ApeCloud）是一家提供数据库内核与管理平台的基础软件开发商.

云猿生于2022年5月份成立，总部坐落于杭州，并同期设立北京分公司。公司是云原生计算基金会（CNCF）会员企业，信通院数据库应用创新实验室成员，并入选杭州市2023准独角兽企业榜单。

KubeBlocks

基于 K8s 的多云、混合云DBPaaS管理平台 ， 支持MySQL、PostgreSQL、Redis、MongoDB、Kafka等开源数据库的自动化运维。

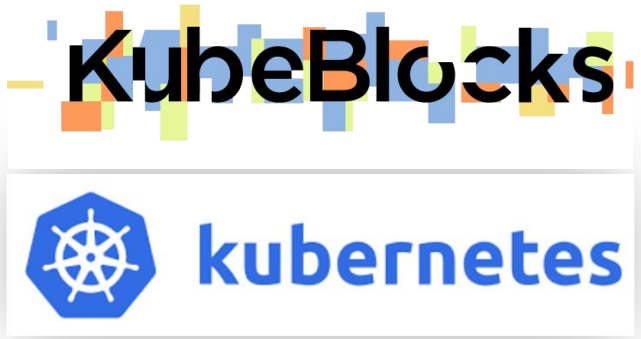


我是谁

毕业即创（shi）业

从被收购到卷王（si）

回到初（qi）心（dian）



第一部分

什么是 Operator

Operator 前世今生

TPR

K8s 1.1 版本中正式推出 TPR（ThirdPartyResource），首次尝试解决 K8s API 的扩展性问题，但存在诸多问题，Alpha 阶段既夭折

Operator

CoreOS 提出 Operator 概念，用于管理和运行基于应用程序领域的复杂有状态应用程序。

给出了用 TPR + controller-runtime 早期版本的 sample：etcd operator

CRD

K8s 1.9 版本发布，CRD进入 beta 阶段并正式取代 TPR；

controller-runtime 加入 K8s 社区并正式发布；

自此，CRD + controller-runtime 逐渐成为开发 operator 的首选

Operator Pattern

Operator Pattern 是官方定义的标准扩展机制，是 K8s Native Application；

Operator = CRD + control loop，i.e，Declaretive API + Automation；

kubebuilder + controller-runtime + helm

Operator Capability Levels

Level I	Level II	Level III	Level IV	Level V
Basic Install	**Seamless Upgrades**	**Full Lifecycle**	**Deep Insights**	**Auto Pilot**
Automated application provisioning and configuration management	Patch and minor version upgrades supported	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

GO

Installation of the workload
- Operator deploys an Operand or configures off—cluster resources
- Operator waits for managed resources to reach a healthy state
- Operator conveys readiness of application or managed resources to the user leveraging the status block of the Custom Resource

Configuration of the workload
- Operator provides configuration via the spec section of the Custom Resource
- Operator reconciles configuration and updates to it with the status of the managed resources

Upgrade of the managed workload
- Operand can be upgraded in the process of upgrading the Operator, or
- Operand can be upgraded as part of changing the CR
- Operator understands how to upgrade older versions of the Operand, managed previously by an older version of the Operator

Upgrade of the Operator
- Operator can be upgraded seamlessly and can either still manage older versions of the Operand or update them
- Operator conveys inability to manage an unsupported version of the Operand in the status section of the CR

Lifecycle features
- Operator provides the ability to create backups of the Operand
- Operator is able to restore a backup of an Operand
- Operator orchestrates complex re—configuration flows on the Operand
- Operator implements fail—over and fail—back of clustered Operands
- Operator supports add/removing members to a clustered Operand
- Operator enables application—aware scaling of the Operand

Monitoring
- Operator exposing metrics about its health
- Operator exposes health and performance metrics about the Operand

Alerting and Events
- Operand sends useful alerts
- Custom Resources emit custom events

Auto—scaling
- Operator scales the Operand up under increased load based on Operand metric
- Operator scales the Operand down below a certain load based on Operand metric

Auto—Healing
- Operator can automatically heal unhealthy Operands based on Operand metrics/alerts/logs
- Operator can prevent the Operand from transitioning into an unhealthy state based on Operand metrics

Auto—tuning
- Operator is able to automatically tune the Operand to a certain workload pattern
- Operator dynamically shifts workloads onto best suited nodes

Abnormality detection
- Operator determines deviations from a standard performance profile

GOCN

DB Operator Day-2 Operations

Observerbility

日志、系统指标等采集、分析；监控配置与报警；性能指标收集与分析等等。

Backup & Restore

备份策略、备份方式、恢复方式、备份管理等等。

Patching & Upgrades

小版本升级、大版本升级、安全漏洞修复等等。

Disaster Recovery & High Availability

Failover/Switchover、多可用区、数据恢复等等。

Security & Compliance

访问控制、审计、安全链接、加密存储等等。

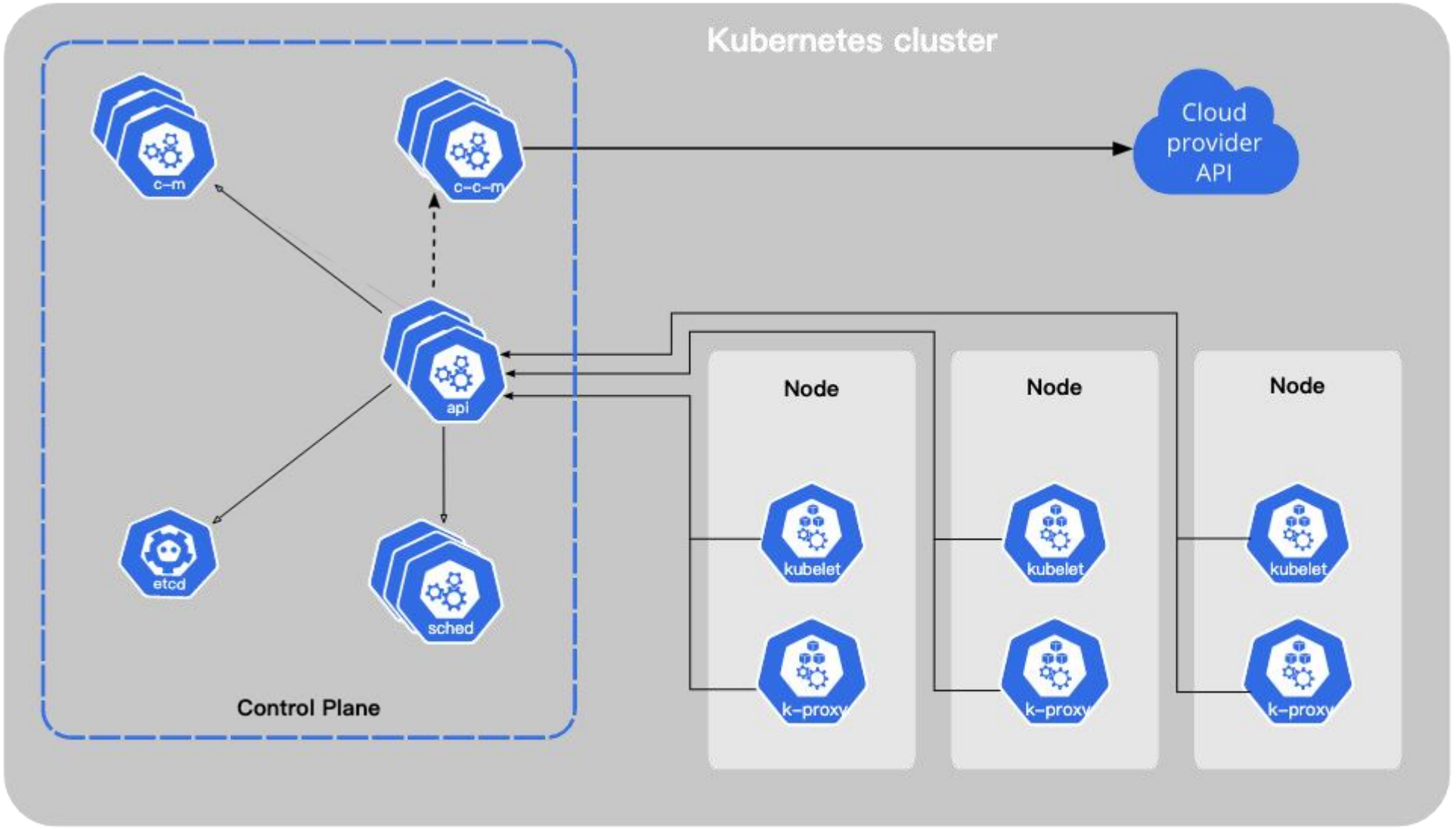
Data Migrations

迁移、同步、清洗、跨地域、灾备、多活等等。

第二部分

Operator 基础模型

K8s 架构



API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

kubelet

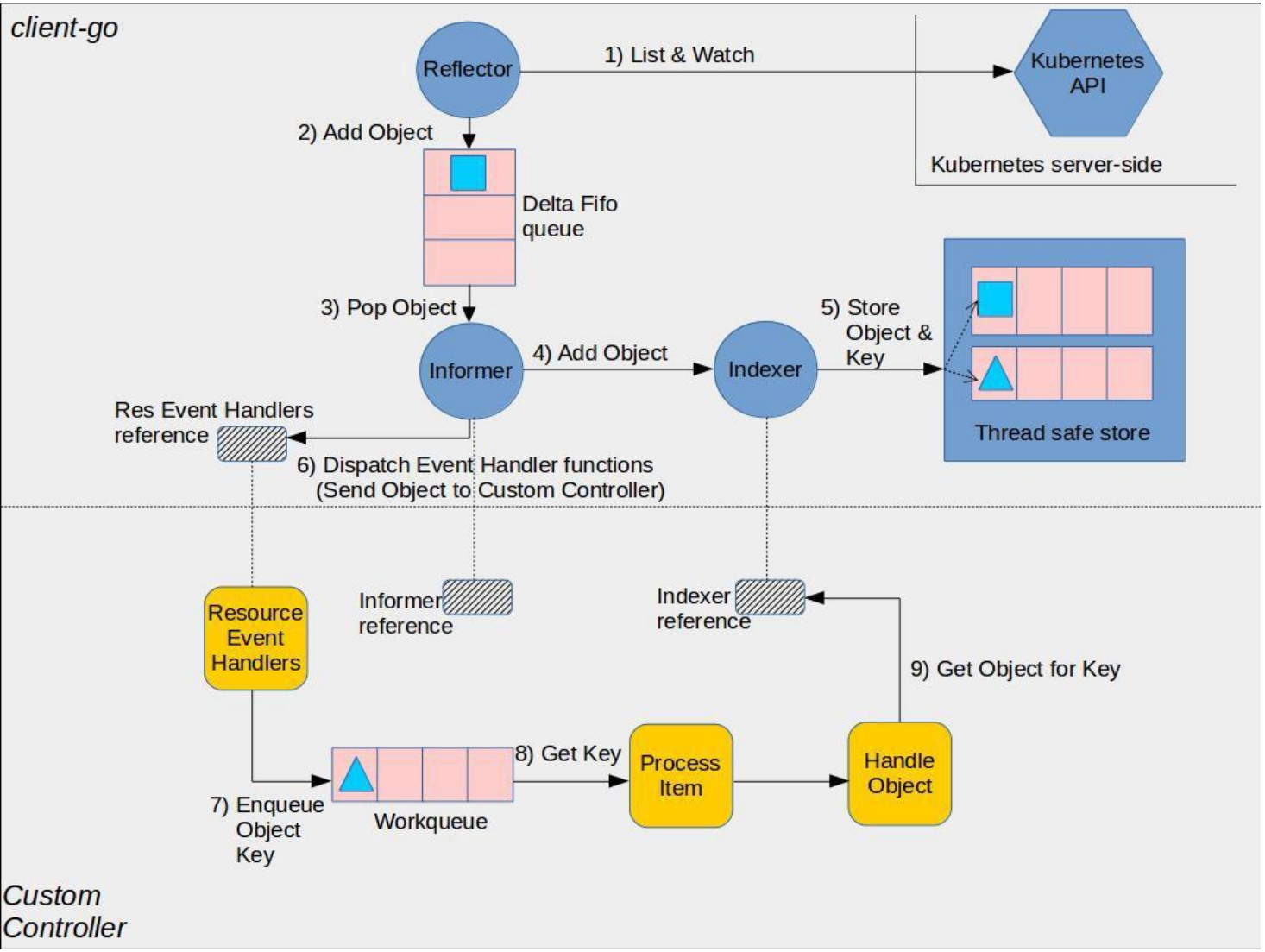
kube-proxy

Scheduler

Control plane

Node

Cache Informer 机制



Cache 如何获取到本地（内存中）

Informer 启动后会通过 reflector 的 list & watch 机制获取某种资源的全量 objects。list 可以简单理解为一个 HTTP GET 请求，watch 为一个 HTTP/2 长连接

Cache 如何保持与 API Server 一致性

list & watch 机制中，list 获取 API Server 中数据的一份快照，并记录 ResourceVersion 版本信息，watch 从 ResourceVersion 开始，获取后续的增量数据。

watch 通过网络异步（asynchronous）获取增量数据，所以 cache 提供的是最终一致性（eventual consistency）。

期间遇到网络、API Server 报错等异常时，会有重试机制

Controller-runtime 的 Informer

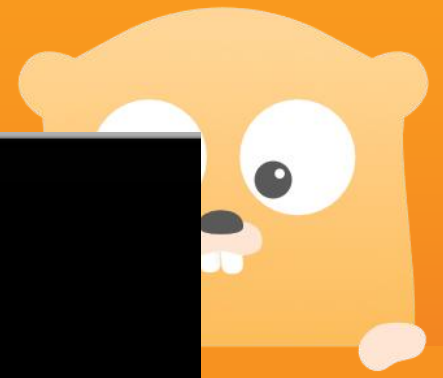
增加一段逻辑：如果上层 GET 某个 object 没有对应的 informer，controller-runtime 会马上为其增加 informer 并完成初始化

Cache 注意事项

Cache 中的对象都保存在内存中，如果对象很多，内存占用会比较大，所以一方面要根据单个对象大小以及总得对象规模来评估 controller 内存消耗。

另一方面 informer 提供了同类型对象的共享机制，降低内存开销

近距离感受 list & watch 机制

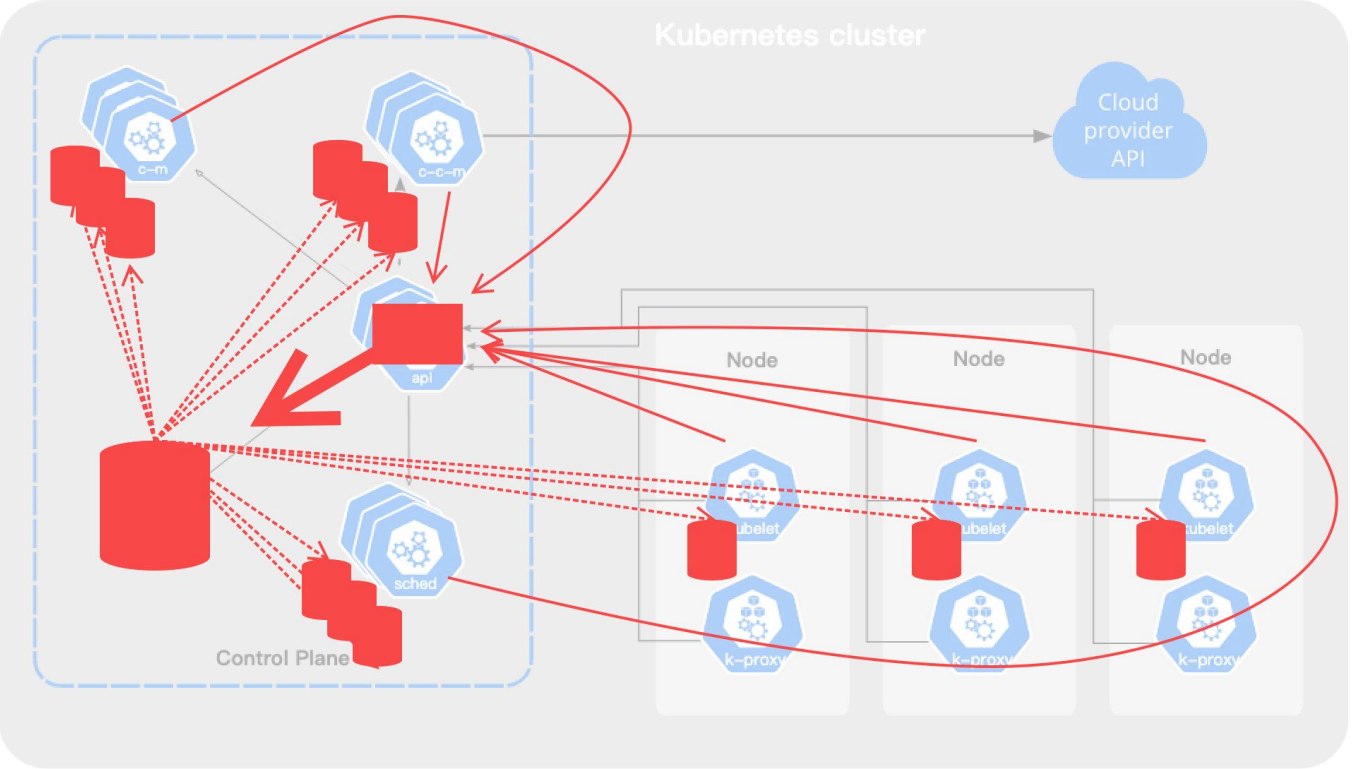


```
free6om@apecloud ~/G/kubeblocks (main)>
```

```
free6om@apecloud ~/G/p/rsm (master)>
```

00:00:02 / 01:00:00 结束录制

Cache 本质及开发建议



相信 Cache

相信 cache 最终能提供所有的你想要的数据版本，不会丢、也不会错

时刻记着 cache 是最终一致

在做任何读（get、list）操作时，不能假设读到的是最新版本，也不能假设一次 reconcile 中始终会读到同一个版本

避免写后读

同一个 controller，在一次 reconcile 中，避免写（create、update、delete）完一个对象后马上去读（get、list）最新版本，等controller-runtime 触发下一次 reconcile

遵循惯例开发模式

即 controller 用读 cache，UT 中不用 cache

附加题：Stale Cache 情况下 情况下 Operator 正确性如何保证

Cache\Planned Action		Create	Update	Delete
latest				
stale	c−lag	−−		
	u−lag	−−		
	d−lag		−−	−−

问题抽象

本地 cache 中的对象有两种可能，即及时（latest）与过期（stale），我们生成的执行计划有3种可能的动作，即 Create、Update 和 Delete。

进一步的，stale 对象意味着本地 cache 落后于 API Server 中对象若干版本，也就是说有一段增量更新还没有复制过来，那么这段增量可以用一个 c/u/d （即 create/update/delete）排列组合来表达。

形式化的，将“+”运算定义为集合 {c，u，d} 上的一个二元运算，其目的是将连续两个操作转化成一个操作，可以看出 c+u=c，u+c=u，c+d=d， d+c=c， u+d=d，d+u=u。那么可以得出，集合 {c，u，d} 中元素排列组合构成的串，可以用一个元素表达，也就是说 stale 对象跟 API Server 之间相差一个 c/u/d 操作，我们用 c−lag，u−lag 和 d−lag 来表示。

当本地 cache 为 latest 时，Plan Action 都能达到预期目的。

当 c−lag 时，API Server 中有该对象，cache 中无该对象。此时 Plan 只应该是 Update 或 Delete 两种 Action，但因本地无 cache，所以 Update 实际变成了 Create，执行时会报“StatusReasonAlreadyExists”错误，与预期不符；Delete 实际不会生成，意味着操作丢失，与预期不符。

当 u−lag 时，API Server 与 cache 中都有该对象，但版本不同。此时 Plan 只应该是 Update 或 Delete 两种 Action，结果与预期相符。

当 d−lag 时，API Server 中无该对象，cache 中有该对象。此时 Plan 只应该是 Create 一种 Action，但因 cache 中有该对象，所以 Create 变成了 Update，执行时会报“StatusReasonNotFound”错误；当新 Spec 中无该对象时，Plan 会错误生成 Delete Action，执行时同样会报对象不存在错。

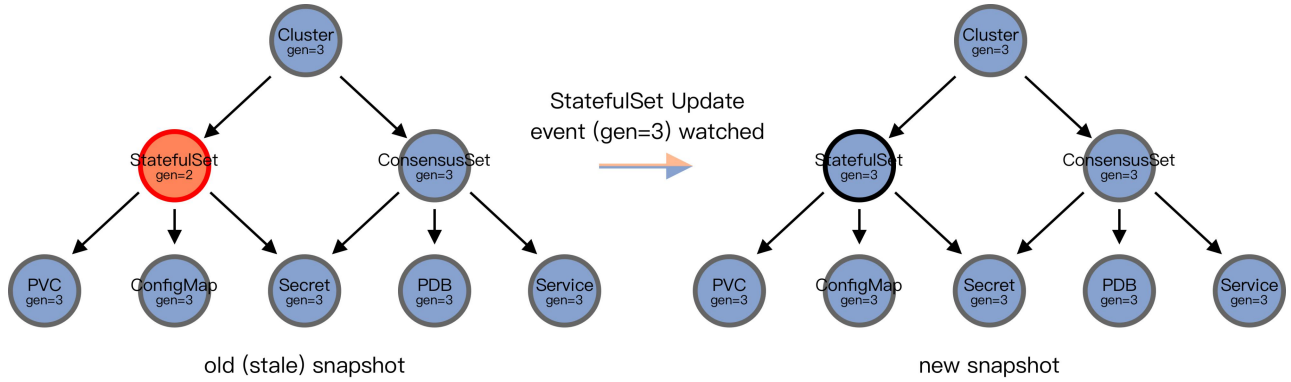
根据上述分析，stale cache 确实会有问题，如何补救？先看一个 stale 对象。

如下图所示，某个版本为3（gen=3）的集群（Cluster）中有一个 stale 对象，即 StatefulSet（gen=2）。

Controller−runtime 的 Reconcile 过程是一个 EDA 模型，当该 stale 对象的更新到达 cache 时，controller−runtime 会发送一个事件（Event）给到 owner controller（也就是我们的 cluster controller）。这时该对象处于 latest 状态，根据表格，Plan Action执行后符合预期。对于其它已经处于 latest 状态的对象，再次 Update 并不会有其它影响，所以stale 对象被成功补救回来。

这个过程可以推广到多个 stale 对象。

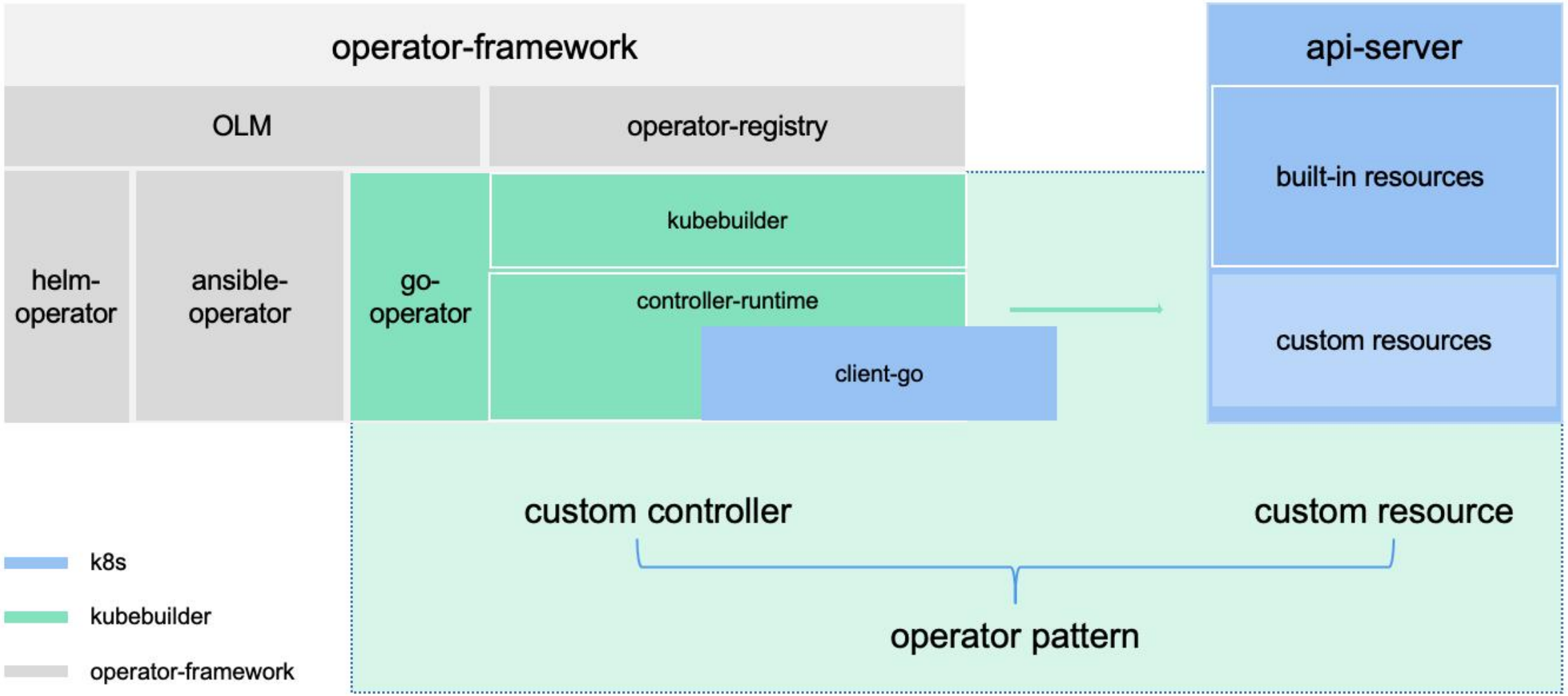
所以最终 stale cache 下能保证 operator 的正确性，前提是 operator 要收到所有对象的事件。



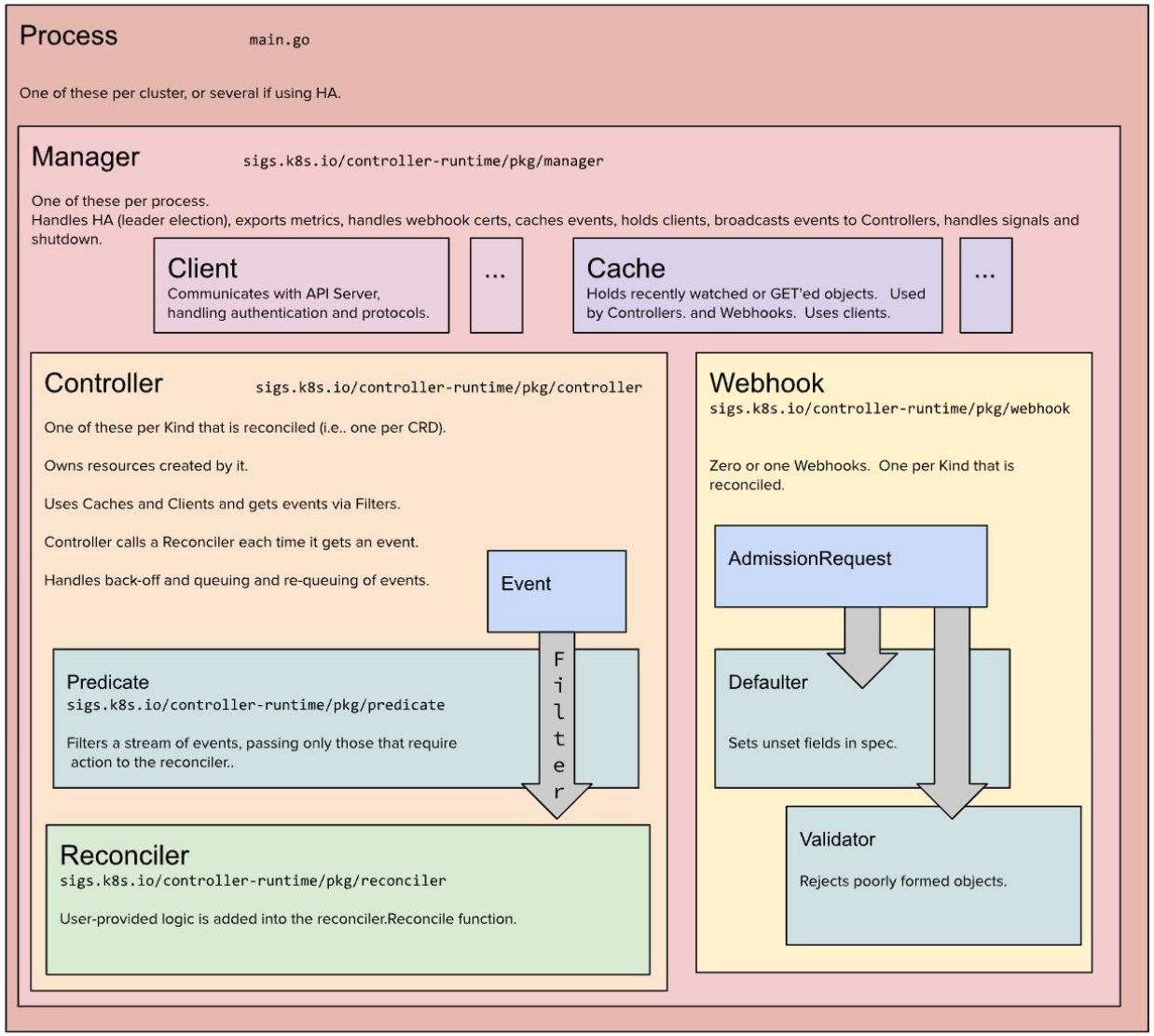
第三部分

Operator 最佳实践

Operator 开发常见概念关系



Kubebuilder 框架模型



Setup 阶段接口

```
// For defines the type of Object being *reconciled*, and configures the ControllerManagedBy to respond to create / delete /
// update events by *reconciling the object*.
// This is the equivalent of calling
// Watches(&source.Kind{Type: apiType}, &handler.EnqueueRequestForObject{}).
func (blder *Builder) For(object client.Object, opts ...ForOption) *Builder {...}

// Owns defines types of Objects being *generated* by the ControllerManagedBy, and configures the ControllerManagedBy to respond to
// create / delete / update events by *reconciling the owner object*.
//
// The default behavior reconciles only the first controller-type OwnerReference of the given type.
// Use Owns(object, builder.MatchEveryOwner) to reconcile all owners.
//
// By default, this is the equivalent of calling
// Watches(object, handler.EnqueueRequestForOwner([...], ownerType, OnlyControllerOwner())).
func (blder *Builder) Owns(object client.Object, opts ...OwnsOption) *Builder

// Watches defines the type of Object to watch, and configures the ControllerManagedBy to respond to create / delete /
// update events by *reconciling the object* with the given EventHandler.
//
// This is the equivalent of calling
// WatchesRawSource(source.Kind(scheme, object), eventhandler, opts...).
func (blder *Builder) Watches(object client.Object, eventhandler handler.EventHandler, opts ...WatchesOption) *Builder
```

Reconcile Loop(Objects Generation)

```
// SetControllerReference sets owner as a Controller OwnerReference on controlled.
// This is used for garbage collection of the controlled object and for
// reconciling the owner object on changes to controlled (with a Watch + EnqueueRequestForOwner).
// Since only one OwnerReference can be a controller, it returns an error if
// there is another OwnerReference with Controller flag set.
func SetControllerReference(owner, controlled metav1.Object, scheme *runtime.Scheme) error {...}

// SetOwnerReference is a helper method to make sure the given object contains an object reference to the object provided.
// This allows you to declare that owner has a dependency on the object without specifying it as a controller.
// If a reference to the same object already exists, it'll be overwritten with the newly provided version.
func SetOwnerReference(owner, object metav1.Object, scheme *runtime.Scheme) error {...}
```

Owns + SetControllerReference 配合，保证收到所有事件

实战：build an operator from scratch

a walkthrough of kubebuilder tutorial：building CronJob

实验描述：

Kubernetes 中有一个支持用 Cron 表达式运行定时任务的对象叫 CronJob，本次实验会用 Kubebuilder 构建一个 Operator，重新实现 CronJob 的功能（实验链接）。本次实验目标：

1. 通过实验，对 Operator 有一个真实体感，加深基础知识理解
2. 实验中覆盖 Kubebuilder 框架大部分功能特性，以便对 kubebuilder 有一个全面熟悉和了解
3. 实验以实现一个生产环境可用的 Operator 为目标，以便整个过程更加接近实际的 Operator 开发

技能点：

1. 熟悉 kubebuilder operator 工程结构
2. 熟悉 K8s Declaretive API 如何设计
3. 熟悉 CR（custom resource）相关事件如何获取
4. 熟悉 Operator Control Loop（即 Reconcile 函数） 如何实现
5. 熟悉如何生成二级资源（Managed Resource）
6. 熟悉如何写 UT
7. 熟悉如何制作 Helm Chart

课后思考题：

1. 如何不启动 WebHook？
2. API 只要用 Go 写就可以了吗，需要修改对应的 YAML 文件吗？
3. Setup 阶段，用 Watch 函数取代 Owns 并实现同样的目的。
4. Reconcile 函数为什么不区分 Create、Update 或 Delete 事件类型？
5. 如果我本次reconcile 时创建了一个二级资源对象，下次reconcile时如何知道该对象已创建？
6. UT中 Read（Get/List）也直接访问 API-Server 的好处是什么？

扫码加入 KubeBlocks 社区

Q&A

问・答

HANGZHOU APECLOUD CO., LTD.