# 94Kubernetes 系列（八十八）Karmada 多云、多集群 Kubernetes 编排：第 1 部分

应用程序容器化是构建和部署软件应用程序的现代方式。多年来，Kubernetes 已成为容器编排的最佳平台之一。单个集群易于设置和管理，并提供 Kubernetes 的基本功能，但缺乏 Kubernetes 著名的典型弹性和高可用性。在许多情况下，单个集群不足以有效管理所有组件的负载。因此，我们需要多个集群来更好地分配工作负载和资源，因此需要一种多集群解决方案。

在本文中，我们将讨论什么是多集群设置、为什么需要它以及 Karmada(https://karmada.io/) 如何允许我们跨多个 Kubernetes 集群和云运行容器化应用程序。

## 什么是 Kubernetes 多集群?

多集群是一种在多个 Kubernetes 集群上或跨多个 Kubernetes 集群部署应用程序的策略。这有助于我们提高应用程序的可用性、隔离性和可扩展性。多集群对于确保遵守不同且相互冲突的法规也很重要，因为可以调整各个集群以遵守地理法规。通过单独的开发团队将应用程序部署到隔离的集群并有选择地公开哪些服务可用于测试和发布，软件交付的速度和安全性也可以得到提高。

## 多集群应用架构

多集群应用程序可以通过两种基本方式构建：

**1. 复制**

在此模型中，每个集群都运行应用程序的完整副本。这种简单但功能强大的方法使应用程序能够在全球范围内扩展，因为应用程序可以复制到多个区域或云中，并且用户流量可以路由到最近或最合适的集群。与健康感知的全局负载均衡器相结合，该架构还支持故障转移。

**2. 按服务划分**

在此模型中，应用程序被分为多个组件或服务，并将它们分布在多个集群中。这种方法在应用程序各部分之间提供了更强的隔离，但代价是增加了复杂性。

## 多集群 Kubernetes 的好处

- 提高可扩展性和可用性
- 应用隔离
- 安全与合规性

到目前为止，我们已经了解了什么是多集群 Kubernetes、为什么我们需要它以及它如何帮助我们部署具有可扩展性、可用性、隔离性、安全性和合规性的应用程序。现在我们将了解 Karmada 如何帮助我们将多集群 Kubernetes 编排到多云中。

## Karmada 简介

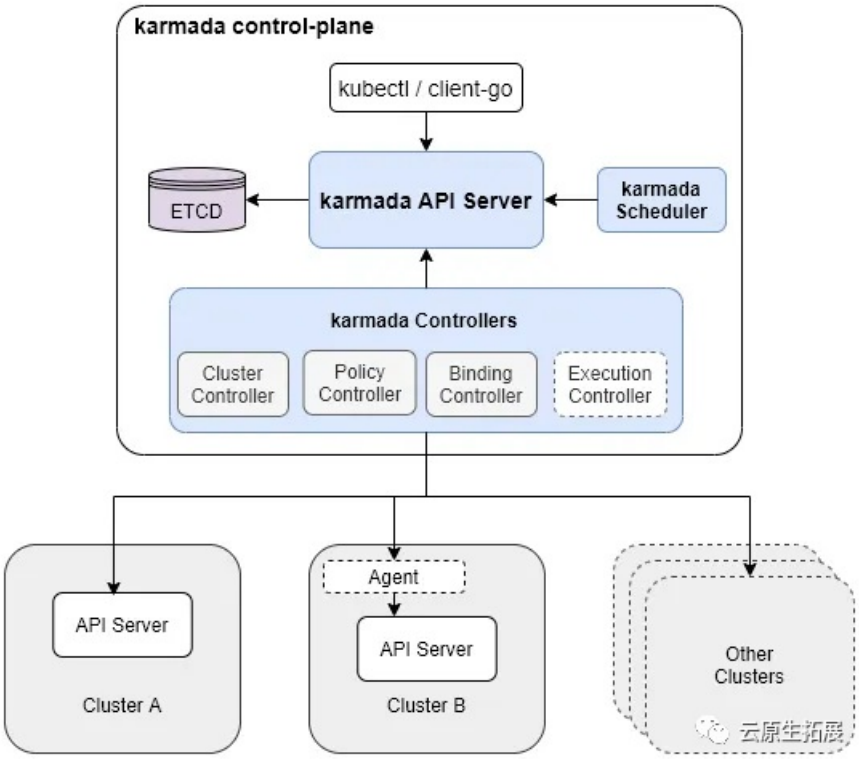
Karmada (Kubernetes Armada) 是一个 Kubernetes 管理系统，使我们能够跨多个 Kubernetes 集群和云运行云原生应用程序，而无需更改我们的应用程序。通过使用 Kubernetes 原生 API 并提供高级调度功能，Karmada 实现了真正开放的多云

Kubernetes。

Karmada 旨在为多云和混合云场景中的多集群应用管理提供统包自动化，具有集中式多云管理、高可用性、故障恢复和流量调度等关键功能。

## Karmada 架构

Karmada 的架构在很多方面与单个 Kubernetes 集群的架构相似。它们都有一个控制平面、一个 API 服务器、一个调度程序和一组控制器。



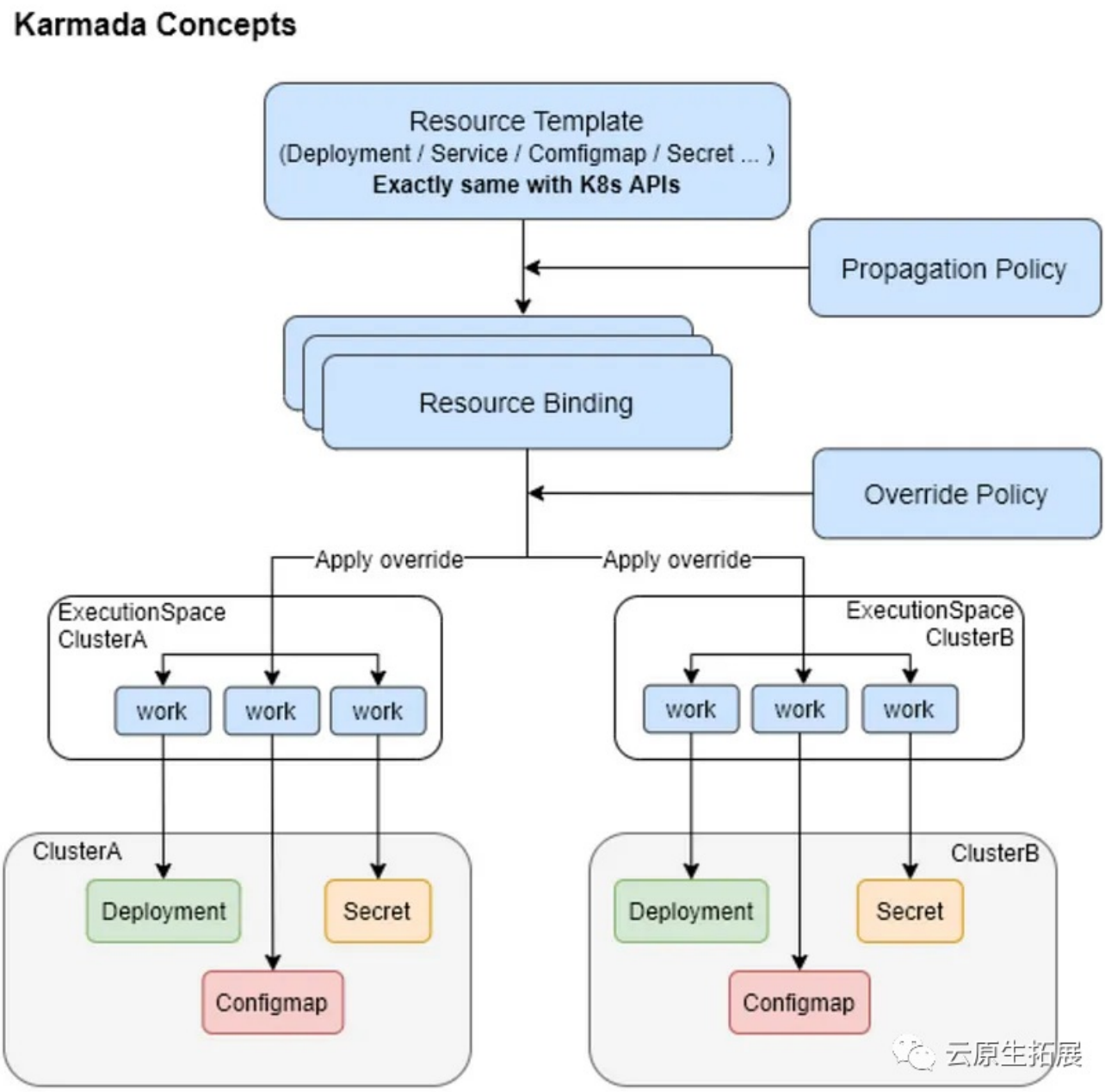

Karmada 控制平面由以下组件组成：

- **Karmada API Server** 提供 Kubernetes 原生 API 和 Karmada 扩展的策略 API。
- **Karmada Scheduler** 重点关注故障域、集群资源、Kubernetes 版本以及集群中启用的附加组件，实现多维度、多权重、多集群的调度策略。
- **Karmada 控制器管理器** 运行各种控制器，这些控制器监视 Karmada 对象，然后与底层集群的 API 服务器通信以创建常规 Kubernetes 资源。
- **ETCD** 存储 karmada API 对象，API 服务器是所有其他组件与之通信的 REST 端点，Karmada 控制器管理器根据您通过 API 服务器创建的 API 对象执行操作。

## Karmada 相关概念

在这里我们将讨论 Karmada 中的关键概念。

**1. 资源模板：** Karmada 使用 Kubernetes Native API 定义作为联合资源模板，以便轻松与 Kubernetes 已采用的现有工具集成。 **2. 传播策略：** Karmada 提供独立的传播（放置）策略 API 来定义多集群调度和传播要求。它支持策略：工作负载的 1:n 映射。用户无需在每次创建联合应用程序时指示调度约束。 **3. 覆盖策略：** Karmada 提供独立的覆盖策略 API，专门用于集群相关配置的自动化。例如，根据成员集群区域覆盖镜像前缀。



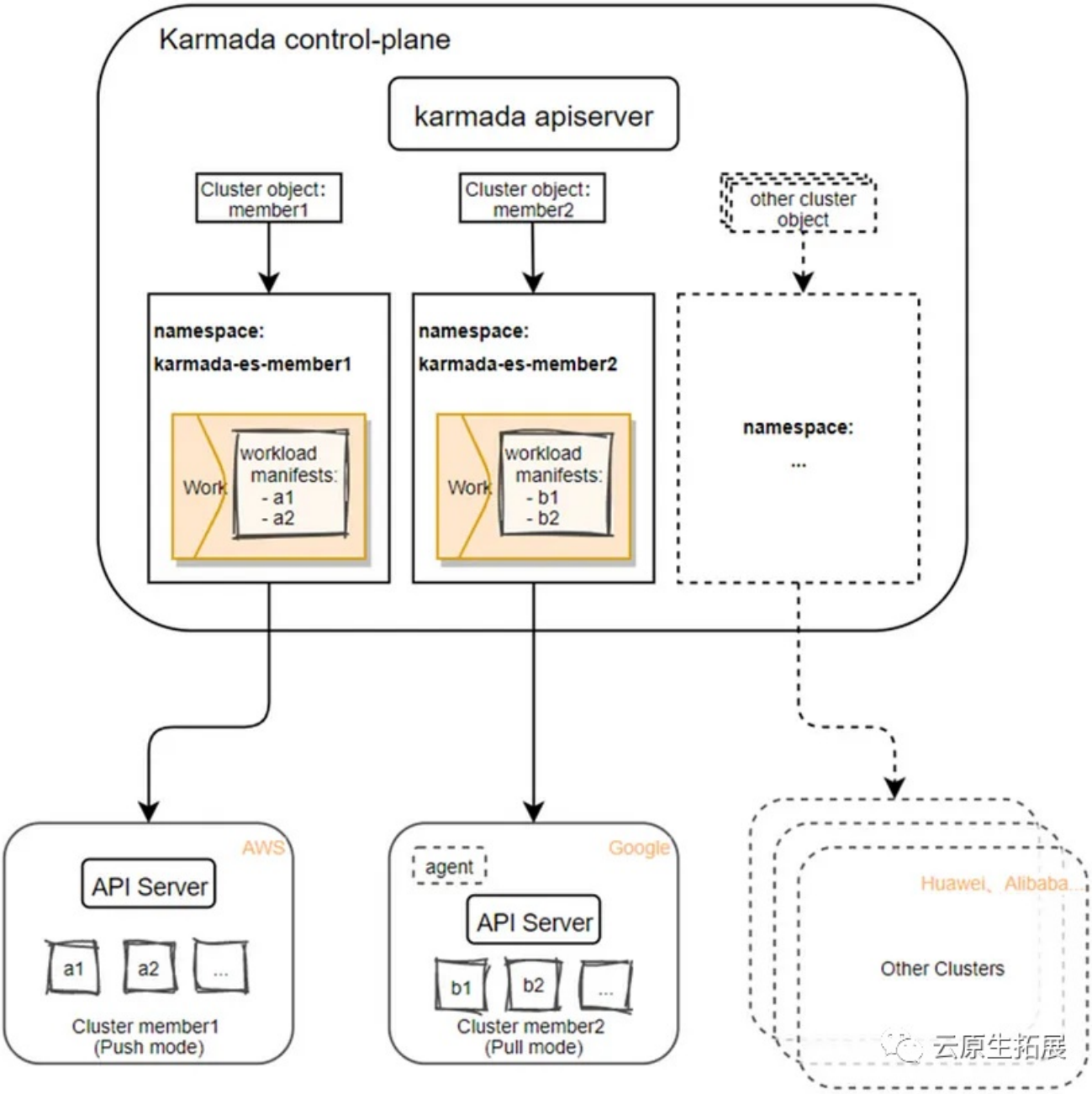

## Karmada 的主要特点

### 跨云多集群多模式管理

1. 通过为每个集群创建一个命名空间（前缀为 karmada-es-* ）来安全隔离

2. Karmada 支持与目标集群的多种模式（推式和拉式模式）连接。在 Push 模式下，Karmada 直接连接到目标集群的 kube-apiserver，而在 Pull 模式下，目标集群中有一个代理组件，Karmada 将任务委托给代理组件。

3. 多云支持（仅当符合 Kubernetes 规范时）。



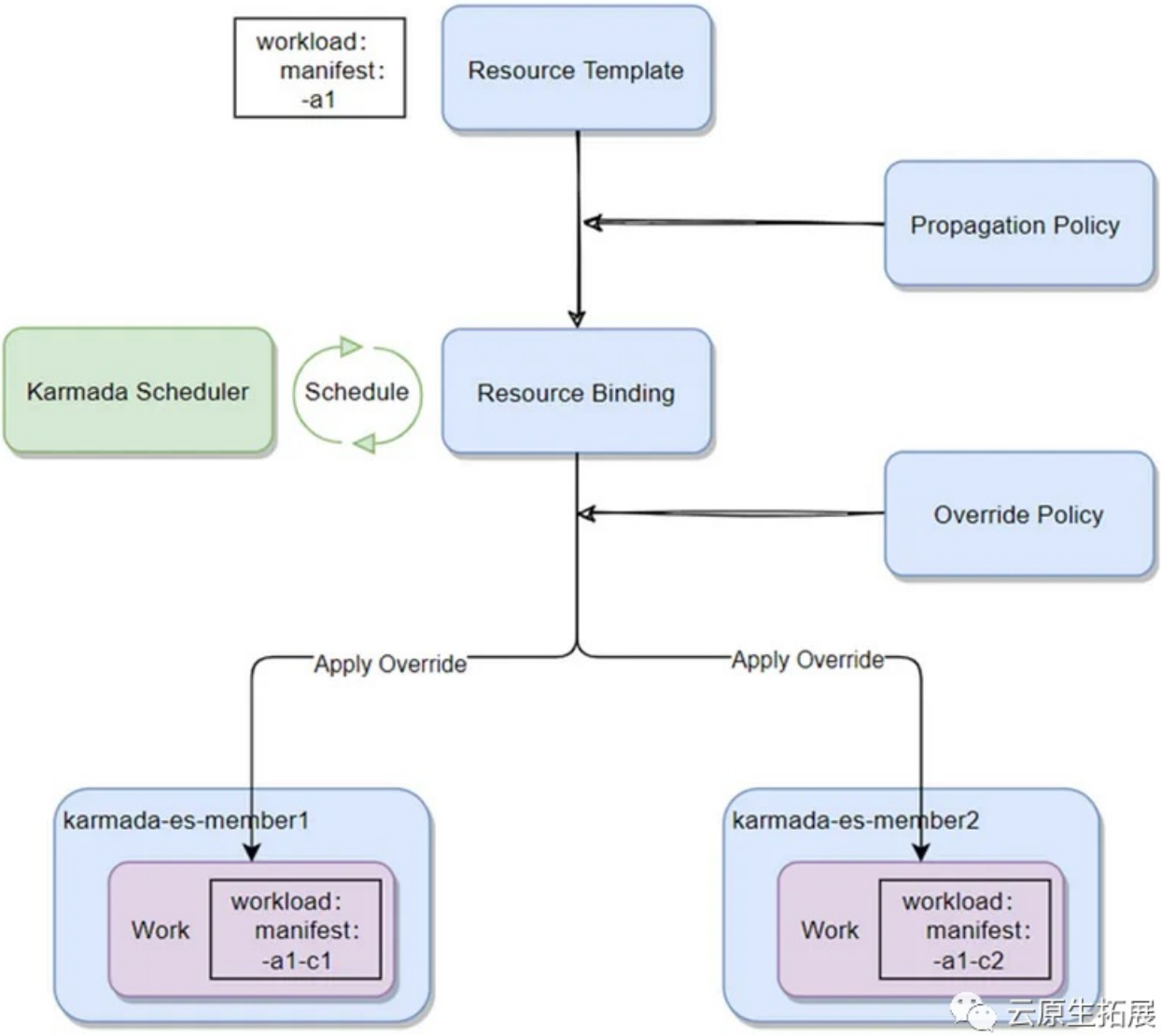

## 多策略多集群调度

1. Karmada 具有在不同的调度策略（例如 ClusterAffinity、Tolerations、SpreadConstraint 和 ReplicasScheduling）下将工作负载分配到各种集群的功能。

2. Karmada 通过利用覆盖策略支持每个集群具有不同的应用程序配置。

3. Karmada 具有重新调度功能，可以根据成员集群中实例状态的变化触发工作负载重新调度。

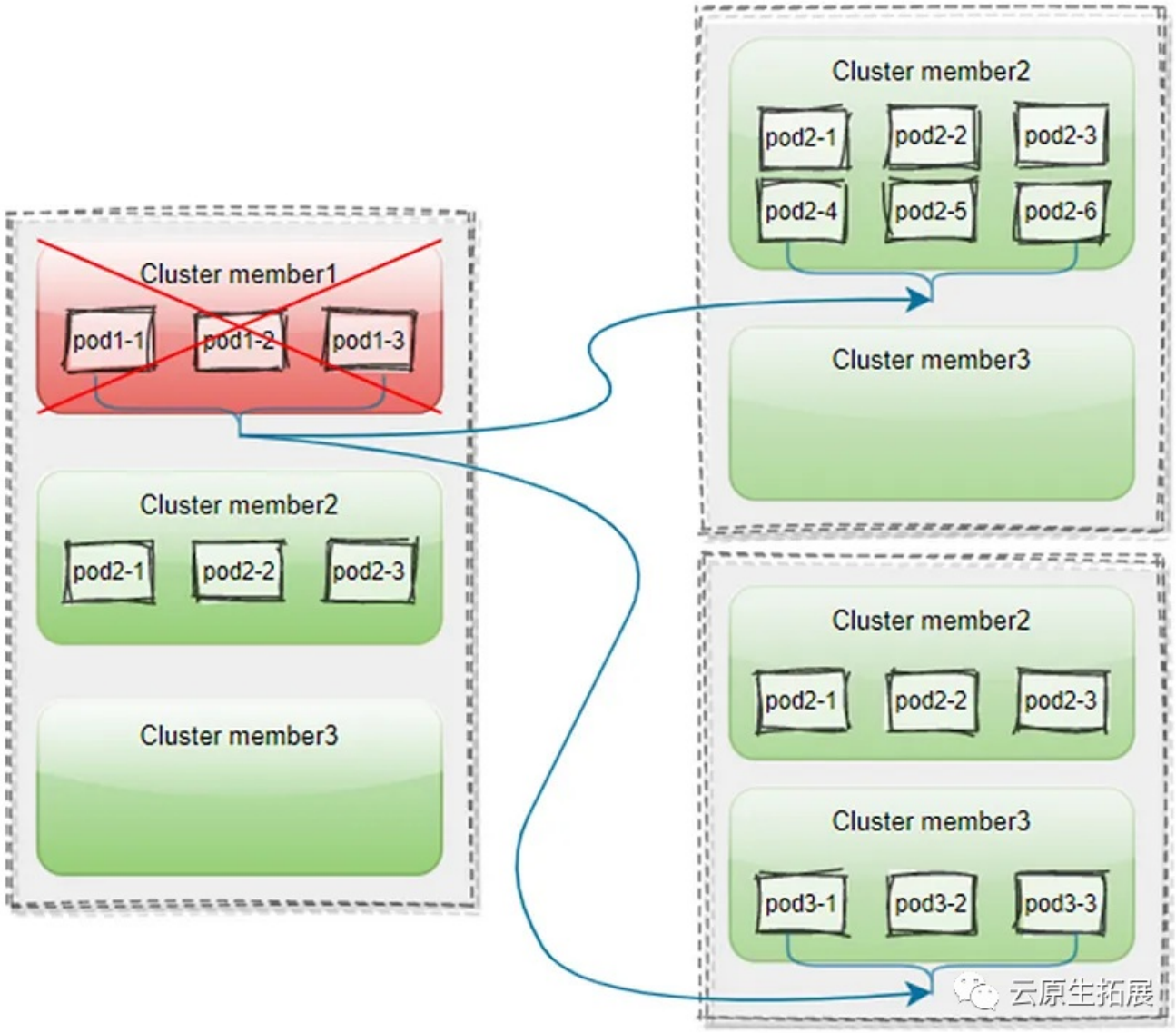
与 k8s 调度非常相似，Karmada 支持不同的调度策略。整体调度流程如下图所示：




## 应用程序的跨集群故障转移

- 集群故障转移：Karmada支持用户设置分配策略，在集群故障后自动以集中或分散的方式迁移故障集群副本。
- 集群污点：当用户为集群设置污点并且资源分配策略无法容忍该污点时，Karmada 也会自动触发集群副本的迁移。
- 服务不中断：在副本迁移过程中，Karmada可以保证服务副本不归零，从而保证服务不会中断。

Karmada 支持集群故障转移，一个集群故障将导致副本故障转移，如下所示：



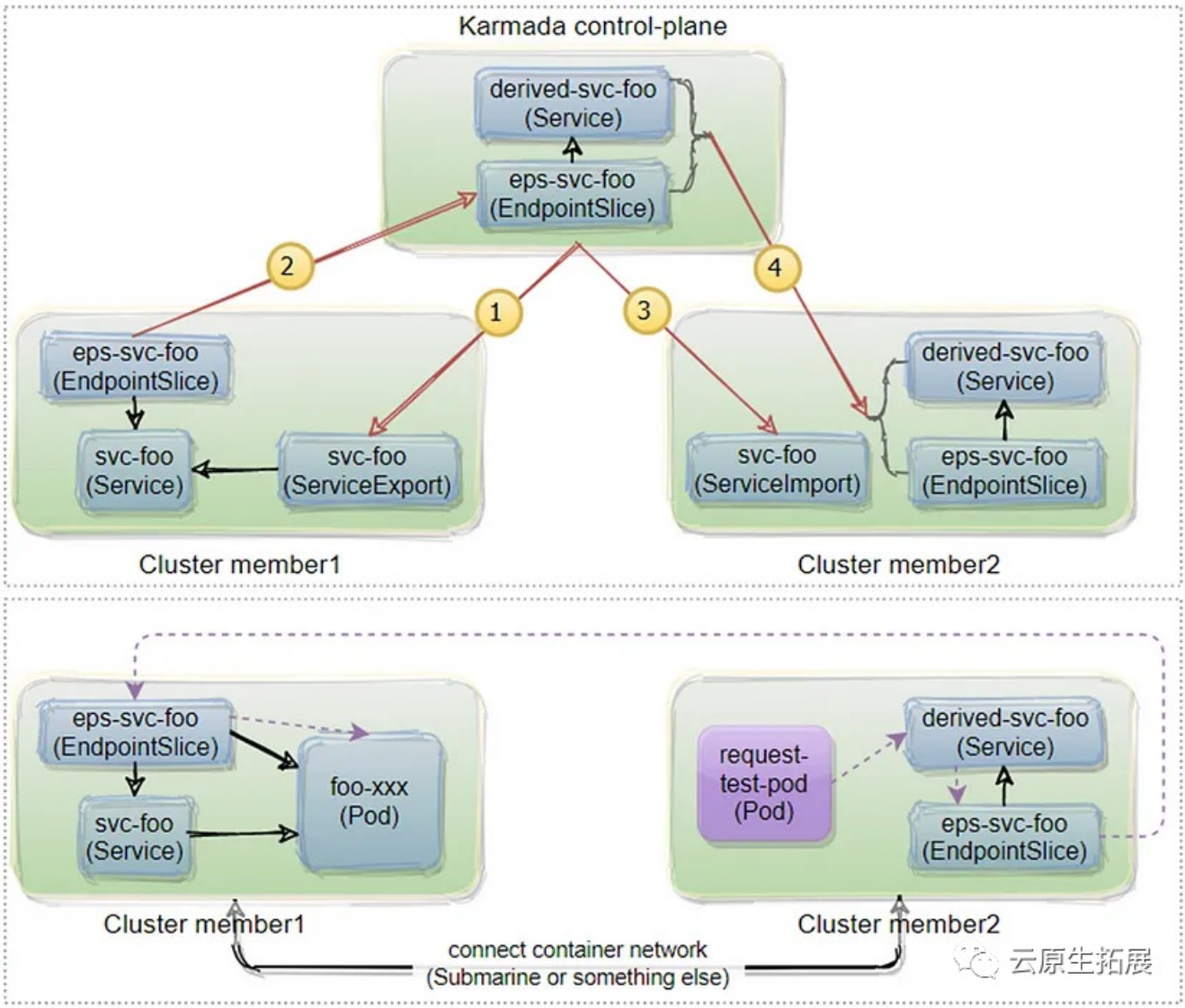

用户已加入 Karmada 中的三个集群： `member1` 、 `member2` 和 `member3` 。名为 `Foo` 的 Deployment 有6个副本，部署在karmada控制平面上。使用 PropagationPolicy 将部署分发到集群 `member1` 和 `member2` 。

当集群 `member1` 发生故障时，集群上的 Pod 实例将被驱逐并迁移到集群 `member2` 或新集群 `member3` 。这种不同的迁移行为可以通过 `PropagationPolicy` / `ClusterPropagationPolicy` 的副本调度策略 `ReplicaSchedulingStrategy` 来控制。

## 跨集群服务治理

1. 多集群服务发现：通过 `ServiceExport` 和 `ServiceImport` ，实现跨集群服务发现。
2. 多集群网络支持：使用 `Submariner` 或其他相关开源项目打通集群之间的容器网络。

用户可以使用 Karmada 启用跨集群的服务治理：




在上图中，我们使用 `submarine` 来连接成员集群之间的网络。

应用程序 `foo` 及其服务 `svc-foo` 与 `ServiceExport` 资源一起部署到 `member1` 集群。在 `member2` 集群中，创建了 `ServiceImport` 资源，现在 `karmada` 将服务 `svc-foo` 导入到名为 `derived-svc-foo` 的 `member2` 中。现在 `member2` 集群中的任何应用程序都可以在内部调用此服务端点来访问 member1 的服务 `svc-foo` 。

---

在本部分中，我们了解了多集群 Kubernetes 设置的需求、架构、策略、多集群 Kubernetes 的优势以及 Karmada 如何在多集群、多云 Kubernetes 设置中编排工作负载。此外，我们还了解了 Karmada 的架构、其关键概念和特征。

在下一部分中，我们将亲自体验 Karmada 的一些演示，以及它如何将工作负载编排到多个集群中以及更实用的内容。