# SOFAStack

## 单元化应用服务 LHC
## 技术白皮书

产品版本：AntStack Plus 1.11.0

文档版本：20220928

蚂蚁集团
ANT GROUP

# 法律声明

## 蚂蚁集团版权所有©2022，并保留一切权利。



## 商标声明



## 免责声明

# 通用约定

| 格式 | 说明 | 样例 |
| --- | --- | --- |
| 危险 | 该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。 | 危险<br>重置操作将丢失用户配置数据。 |
| 警告 | 该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。 | 警告<br>重启操作将导致业务中断，恢复业务时间约十分钟。 |
| 注意 | 用于警示信息、补充说明等，是用户必须了解的内容。 | 注意<br>权重设置为0，该服务器不会再接受新请求。 |
| 说明 | 用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。 | 说明<br>您也可以通过按Ctrl+A选中全部文件。 |
| > | 多级菜单递进。 | 单击**设置**> **网络**> **设置网络类型**。 |
| **粗体** | 表示按键、菜单、页面名称等UI元素。 | 在**结果确认**页面，单击**确定**。 |
| Courier字体 | 命令或代码。 | 执行 `cd /d C:/window` 命令，进入Windows系统文件夹。 |
| *斜体* | 表示参数、变量。 | `bae log list --instanceid`<br>*Instance_ID* |
| [] 或者 [a\|b] | 表示可选项，至多选择一个。 | `ipconfig [-all\|-t]` |
| {} 或者 {a\|b} | 表示必选项，至多选择一个。 | `switch {active\|stand}` |

# 目录

# 1.什么是单元化应用服务

## 产品背景

随着云原生技术在业界的持续升温，越来越多的金融客户希望能把应用系统搭建在拥有云原生能力的平台之上，利用各类云原生的红利迅速实现技术转型，促进业务的敏态发展和持续创新。

同时，由于金融业务的特点，必须具备高可靠、高可用、高安全、高稳定性，这对云原生在金融场景的落地提出了很大的挑战。蚂蚁 SOFAStack 已经打造了 AKS（Application Kubernetes Service），并且成功落地了国泰产险，但该场景为公有云，在可用性和稳定性方面可以借助公有云的能力，比如数据的多副本备份，ECS的宕机迁移等。

但在专有云场景下，SOFAStack 不再具备上述公有云的容灾恢复能力，这就要求我们在专有云的云原生架构上做出调整，在云平台本身满足金融级严苛场景之外，还要以产品化的方式输出蚂蚁集团十多年来在金融级分布式架构、容灾应急、高危防护、数据安全的能力。

单元化应用服务（LDC Hybrid Cloud，简称 LHC）产品致力于解决上述用户诉求，适用于以下四大场景：

- 同城双活
- 两地三中心
- 异地多活（单元化）
- 异构技术设施的混合云

从技术实现的角度，当各机房的 K8S 集群和 PaaS 入口都在统一平台（CAFE）下进行治理、管控后，LHC 将满足混合云（狭义上的“公有云+专有云”）以及更广义上的多云多集群联邦场景中的应用发布变更、配置同步、集群管理能力。

## 产品概述

SOFAStack 平台下的单元化应用服务（LDC Hybrid Cloud，简称 LHC），全面集成 Kubernetes，提供完整的多集群管控、跨集群发布运维、认证授权、容器网络、持久卷存储等方面的平台能力。在兼顾标准化一致性的 Kubernetes 能力的同时，亦将源自实践的应用全生命周期的发布部署能力通过产品化的形式交付，同时依托于社区多集群联邦架构，提供了跨多个Kubernetes集群的发布运维能力。

针对金融级场景下大规模分布式系统的特点，提供了丰富的发布策略以满足不同的场景，如：分组发布、Beta 发布、灰度发布等，帮助传统架构平滑过渡，适应金融技术风险保障需求，实现大规模金融级运维场景下的容器服务落地。

单元化应用服务致力于通过成熟的技术和最佳实践经验的支撑，使金融场景亦能从容地应对云原生应用开发、运维、架构的难题，同时针对金融场景必需的高可靠、高可用、高安全、高稳定性要求，单元化应用服务从架构上得以保证高可用，解决金融系统应用容器化转型的需求，使容器技术真正的大规模应用于金融行业生产环境里，帮助传统应用以更高效、低成本的方式迁移到云原生微服务、容器化体系架构。

# 2.产品优势

## 金融级发布

- 发布过程安全可靠，可重试，可灰度，可回滚，可溯源。
- 支持虚拟机和容器的混合发布，提供从虚拟机到容器的过渡方案。

## 运行时监控

- 业务自定义大盘随时关注业务动态。
- 实时监控应用基础指标，如 PV、Service（应用服务被调用）、SAL（调用外部服务）等。
- 全面收集基础资源指标，如 CPU、内存、IO 流量等。

## 微服务框架

- 深度集成蚂蚁 SOFA Mesh 做异构语言的服务注册发现及通信。

## 网络模式

- 支持 VPC 和 Overlay 两种网络模式。
- 支持负载均衡类型的 Service 和 Ingress。

## 高可用和容灾

- 支持同城双活、两地三中心容灾方案。
- 支持升级至 SOFAStack 沉淀多年的单元化高可用容灾方案。

# 3.产品架构

SOFAStack 提供基于 Kubernetes 的托管式容器应用服务，服务底层可无缝对接阿里云，裸金属服务器和其他 IaaS 平台（如 OpenStack），并针对不同平台，按照 CSI、CNI 标准提供存储、网络插件，使用 Open Service Broker 标准接入三方服务，如数据库、缓存服务、负载均衡等。

在产品层除了提供业界标准的云原生能力外，还提供具备金融级特性的发布运维能力，比如按序的无损发布策略，针对大规模分布式系统的批量发布运维能力，内置按机房或部署单元的高可用容灾拓扑，并集成金融科技的业务实时监控能力。

您可通过产品控制台或 Kubernetes 提供的标准 Kubectl 工具接入容器产品。三方产品亦可通过容器应用服务开放的 OpenAPI 或者 Kubernetes 标准的 API 集成容器应用服务的产品能力，打造各种解决方案，服务用户。

# 4.功能原理

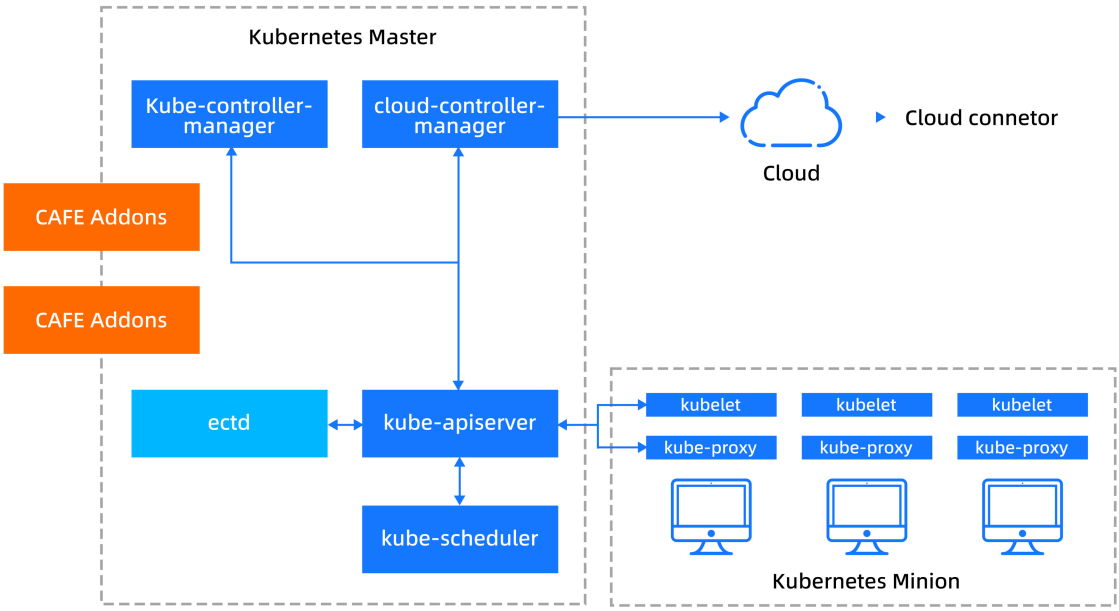
## 集群管理

集群（Cluster）管理用于对集群以及集群内部的资源进行统一管控。具体分为控制平面和数据平面两部分：

- **控制平面**

  对外暴露管控 API 以操作集群本身以及其中的各种资源。

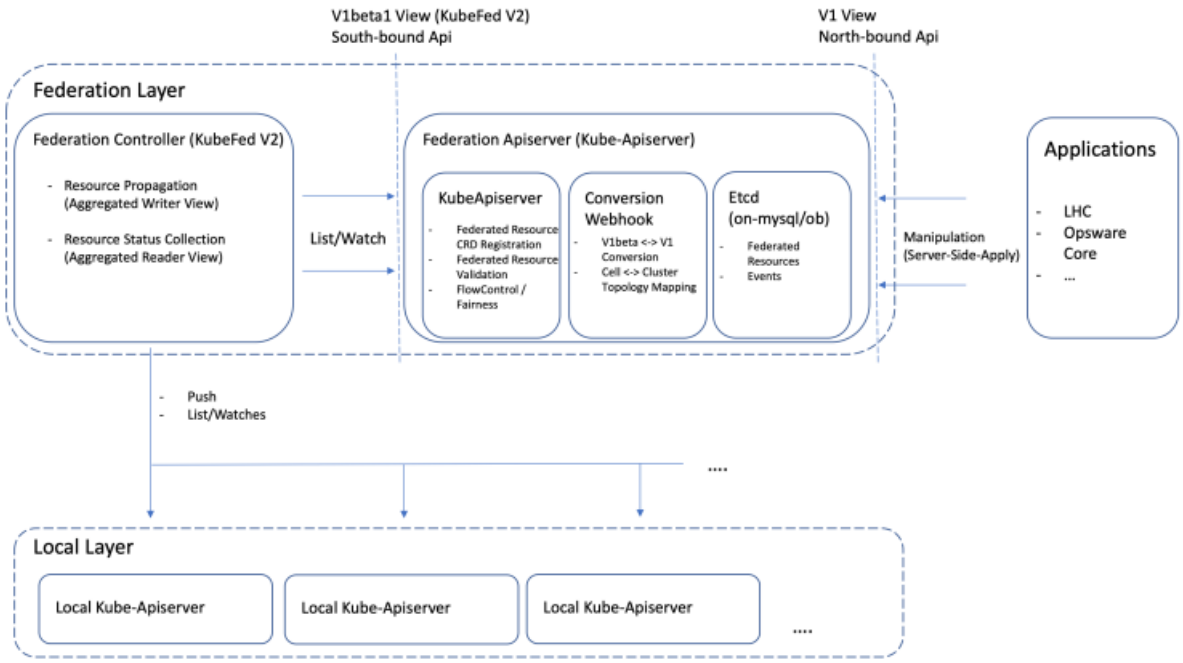
- **数据平面**
  - 狭义上是由一台或者多台 ECS（物理机）组成的有网络边界的资源隔离单位。
  - 广义上除了 ECS（物理机）外，还包括其他资源，比如 GPU/FPGA、负载均衡、网络存储等。




图中 CAFE Addons 是蚂蚁集团做的扩展，用于提供一些高阶的云原生工作负载和现有 PaaS 层模型云原生化。

## 多集群架构

目前在专有云场景下，以采用 KOK 模式为例，用户的 K8s 集群被中枢托管，即 KOK 集群的控制面组件部署在中枢 VPC，Federation 控制面和每个 Local 集群都在同一个 VPC，不存在任何网络连通性问题。因此，在专有云场景下，我们直接采用社区原生 Federation 架构，如下图所示：




如上图所示，目前 Fed 的整体架构采用层级（Hierarchical）+ 推送（Push）的架构。其组件构成包括以下几部分：

- **Federation Controller**：从抽象的角度来看，此 Controller 组件提供的能力较为集中化：
  - 提供多集群聚合“写”视图。
  - 提供多集群聚合“读”视图。
- **Federation Apiserver**：此 Apiserver 组件提供的能力定位包含如下几点：
  - 南向能力（面向基础设施）：一个 KubeFed V2 的适配器，使用到 Controller 的资源聚合读取或写入能力。
  - 北向能力（面向应用/LDC）：提供部署单元（Cell）到物理集群（Cluster）的映射。

## 镜像管理

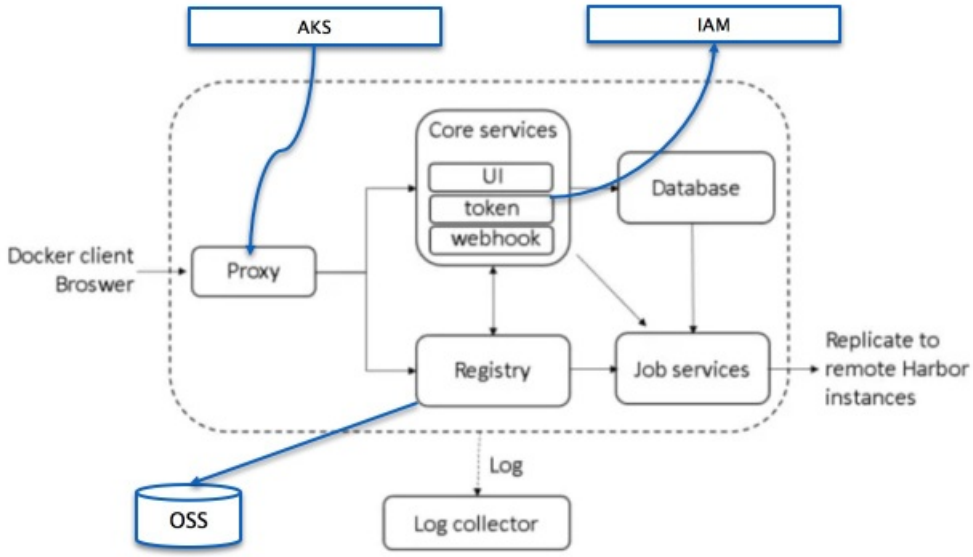
镜像管理分为三个部分：

- 用户镜像管理
- 应用镜像构建
- 镜像技术栈

## 用户镜像管理

用户镜像管理基于开源 harbor 进行包装，额外提供以下能力：

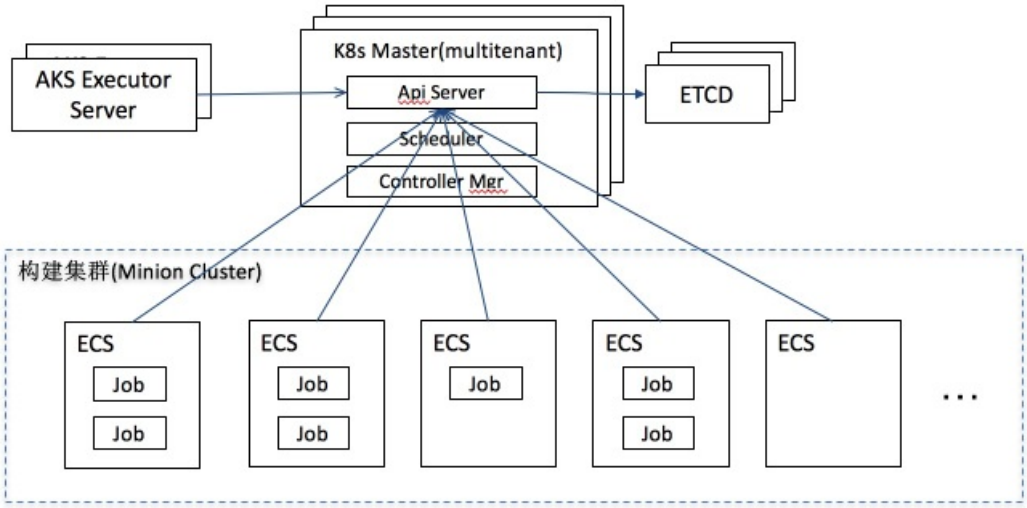
- 和 PaaS IAM（身份与认证管理系统）打通，可使用 PaaS 账号进行登录来管理镜像。
- 底层存储可基于 OSS、S3 或者本地磁盘。




## 应用镜像构建

镜像构建有以下特点：

- 采用 Kubernetes 进行构建集群管理，并采用 Job workload 来运行构建进程。
- 使用 Docker in Docker 的方式进行构建，每次构建均在容器内，构建后即焚，保证环境的一致性。
- 源代码支持从 gitlab 获取，版本 8.8 及以上。




## 镜像技术栈

镜像技术栈为用户提供基础镜像管理的能力，涵盖官方提供的镜像技术栈以及用户自定义的镜像技术栈。后续在 创建应用服务 时，您可以上传应用代码包，并选择所需的基础镜像，一键构建出目标镜像 Binary to Image（B2I），无需编写 dockerfile，真正实现无感镜像化。

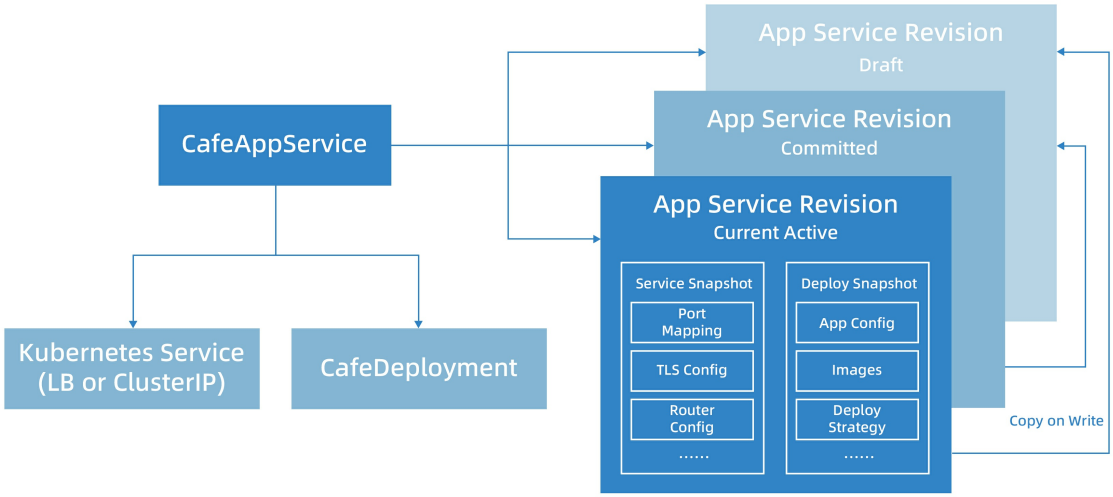
## 应用发布

应用发布部分分为两个部分：应用服务管理和发布变更管理。

## 应用服务管理

SOFAStack 定义了 CafeAppService 的概念，是应用服务在容器世界里的一个抽象，目前包含了 deployment、service 以及中间件的配置信息，是一个 namespace 级别的资源。

CafeAppService 使用 Immutable Infrastructure 的思想进行设计，定义了 Revision 对象做版本控制。用户每次的修改都是一个 Revision，发布一个应用本质上是发布该应用的一个 Revision，故可做到快速的弹性扩缩容，并且可以方便回滚到之前发布成功过的 Revision。




## 发布变更管理

发布变更是 PaaS 平台提供的重要能力。对于客户来说，每次发布必须要有据可查，而且要保证安全无损。这里，我们将蚂蚁的安全生产理念融入其中，在产品层面上提供“可灰度、可回滚（应急）、可监控”的能力。

为了做到上述能力，SOFAStack 提供了发布单的概念，并定义了一个原生的 CRD：CafeDeployment。

## 发布单

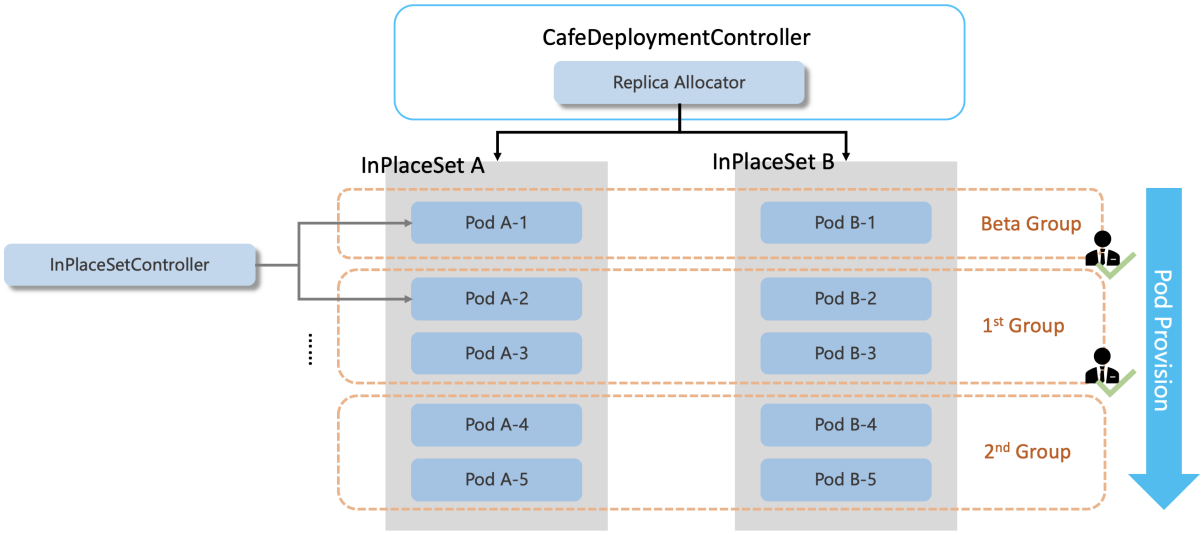
主要用途如下：

- 做应用发布的审查记录，用于统计分析、故障复盘回顾等。
- 协调多个应用的发布顺序，这是由于某些业务对系统的可靠性要求高，尤其在涉及资金的主链路。
- 不少系统由于业务原因存在依赖关系，必须做有序发布。

## CafeDeployment

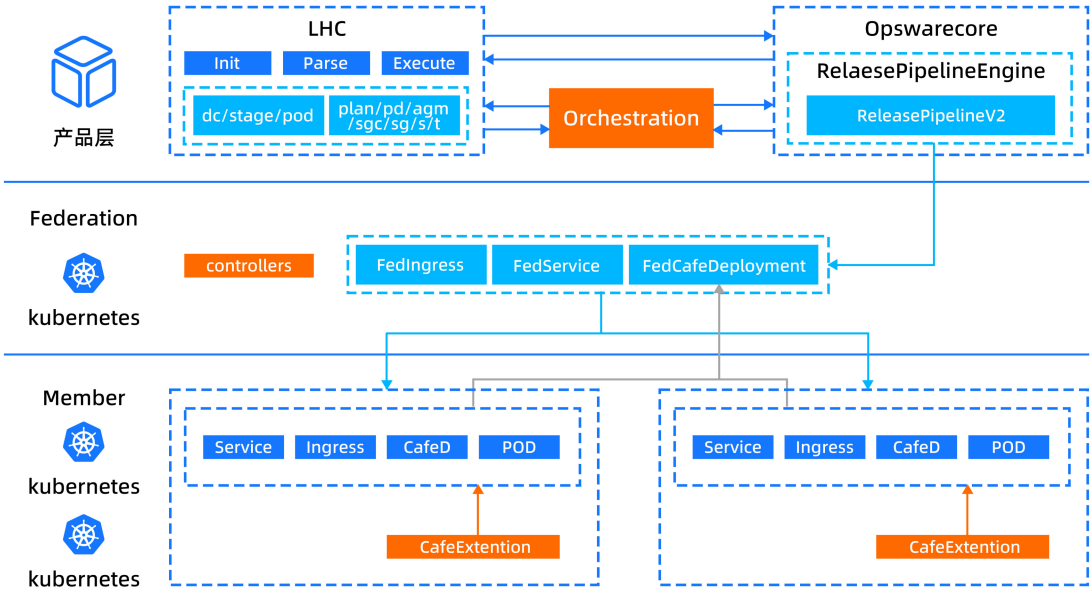
该 CRD 拥有原地升级（InplaceSet）能力，即升级过程中 Pod 的 IP 保持不变（仅在替换镜像场景下生效），可和经典的运维监控体系做无缝集成。

除此之外，相比社区的 deployment，还具备 beta 验证、自定义分组策略、分组暂停、引流验证的能力。




## 发布调用链路

调用链路架构图如下所示：




发布链路贯穿产品层、Fed 集群、Member（Local）集群三层：

- **产品层**

  产品层涉及到 LHC、Opswarecore 和 Orchestration 三个系统。

  i. LHC 内部（容器服务和发布服务两个模块）创建发布单并下发 DAG 到 Orchestration 驱动发布流程执行。

  ii. 当执行到应用服务节点时，LHC 会组装 ReleasePiepline 并调用 Opswarecore 接口下发。

  iii. Opswarecore 会将 ReleasePiepline 解析成 DAG 下发 Orchestration 驱动发布流水线执行。

  iv. 随着执行流程推进，Opswarecore 会将状态通过事件调用通知 LHC，LHC 以此更新发布单的状态。

- **Fed 集群**

  Opswarecore 执行 RelesePipeline 时，会根据执行流程下发 Fed 资源到 Fed 集群，同时监听 Fed 资源及时感知 Fed 资源的状态变化。

- **Local 集群**

  Fed 集群向 Local 集群做资源的下发和调和，Local 集群会向 Fed 集群汇聚状态，此处主要是把状态汇聚到 FedCafeDeployment，而 FedCafeDeployment 状态的变化会让 Opswarecore 监听到。

## 云原生资源管理

除了上述的应用服务、发布单、CafeDeployment 这些高阶工作负载之外，单元化应用服务平台还提供对 Kubernetes 自带的资源进行管理。

## 命名空间

命名空间分为统默认的命名空间和用户自定义的命名空间。

- **系统默认的命名空间**
  - default：在不指定命名空间时，默认使用 default。
  - kube-public：用来部署公共插件、容器模板等。
  - kube-system：部署系统组件。
- **用户自定义的命名空间**

  例如开发环境、联调环境和测试环境分别创建对应的命名空间。

## 配置

配置分为配置项（ConfigMap）和保密字典（Secret）。

- **配置项**

  一种用于存储工作负载所需配置信息的资源类型，包括文本类型的 key-value 形式，也可以有 key-文件形式。

- **保密字典**

  一种用于存储工作负载所需要认证信息、密钥的敏感信息等的资源类型。

## Service

Service 是 Kubernetes 自带的核心资源，用于抽象向多个 Pods 访问的接口，类似于负载均衡的概念。目前支持 Cluster IP、Loadbalancer、Node Port、Headless、External DNS 类型。

## PV（Persistent Volume）存储资源

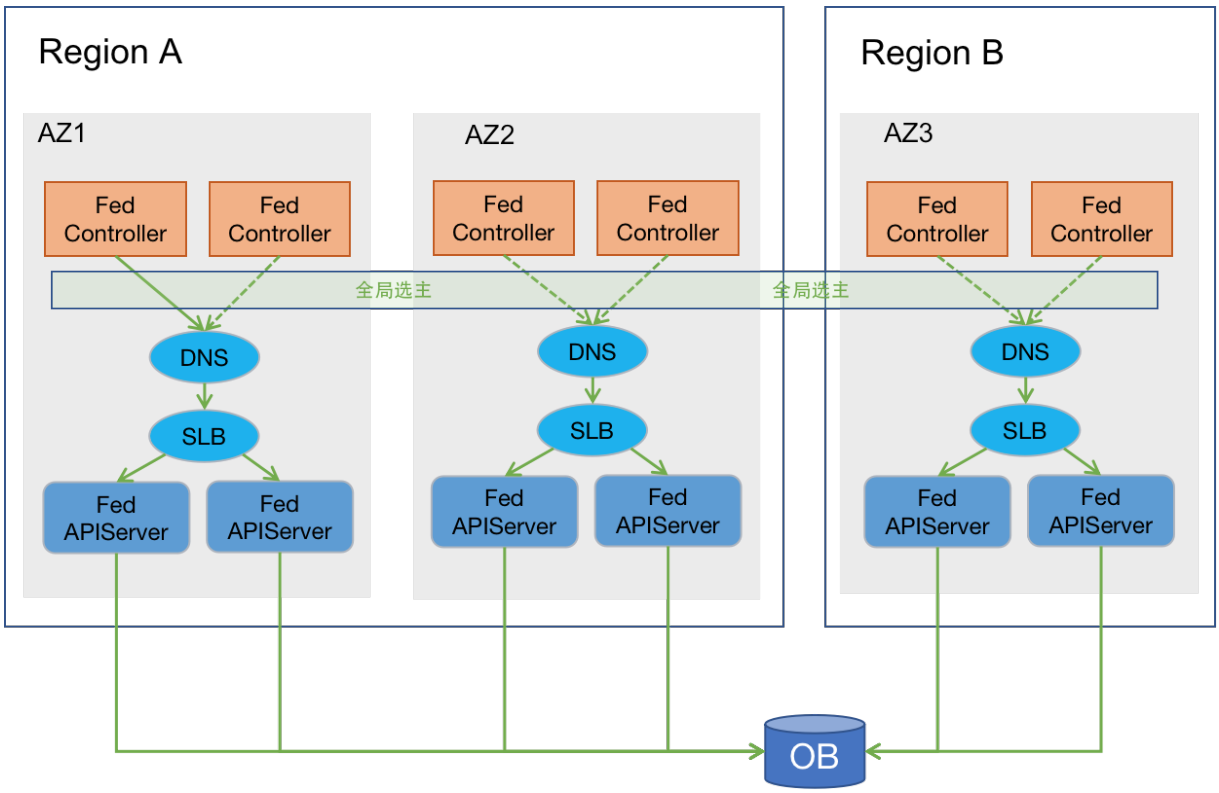
PV 提供跨 Pod、跨应用、跨 node 的存储服务，是一个用来描述后端持久化的一块存储的模型。PV 有两种 Provision 形态：Static 和 Dynamic。

## 工作负载（Workload）

工作负载包括 Kubernetes 原生的 Deployment、StatefulSet,、DaemonSet、Job、Pod。

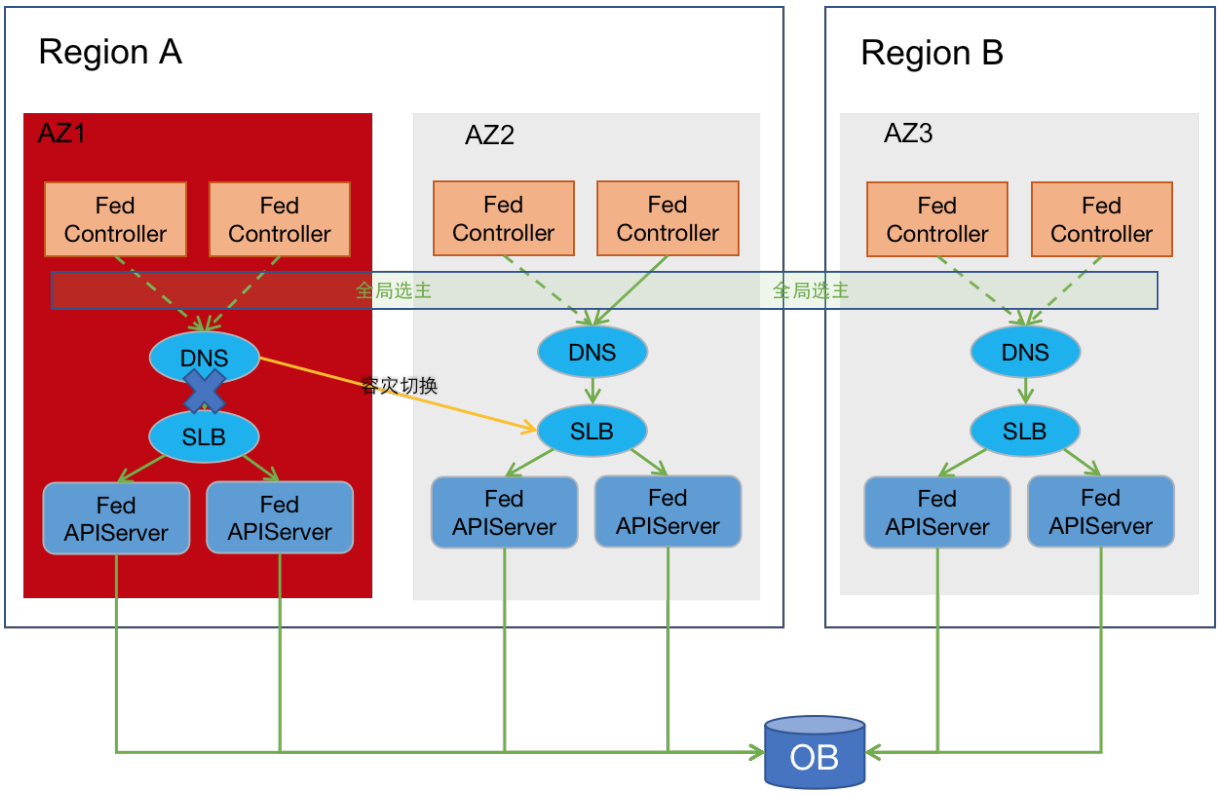
## Federation

## 高可用




1. 每个机房部署多副本 Fed Controller 和 Fed APIServer，保证单机房内高可用。
2. Fed APIServer 访问全局 OB 保证数据一致性。
3. 每个机房内通过独立的域名和 VIP 访问 Fed APIServer，保证常规访问机房收敛。

## 容灾

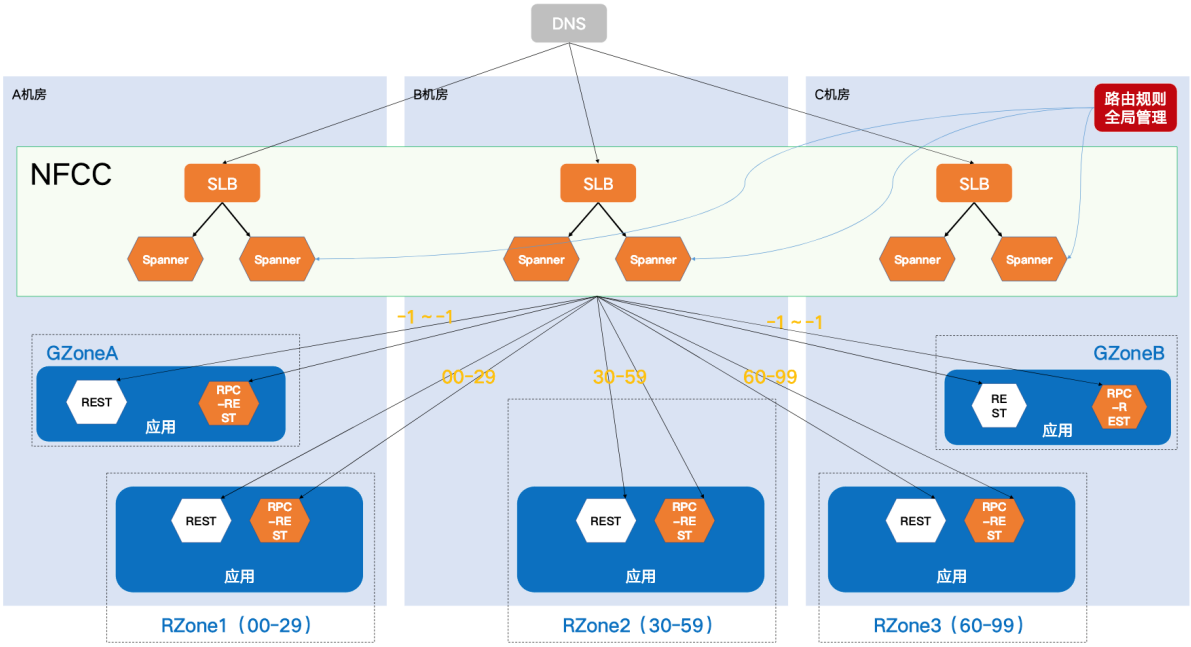


- **同城容灾**
  - Fed Controller，依赖全局选主，自动切换到其他机房。
  - Fed APIServer，需要切换域名关联到其他机房的 VIP。
- **异地容灾**：同上。

## 统一接入层管理

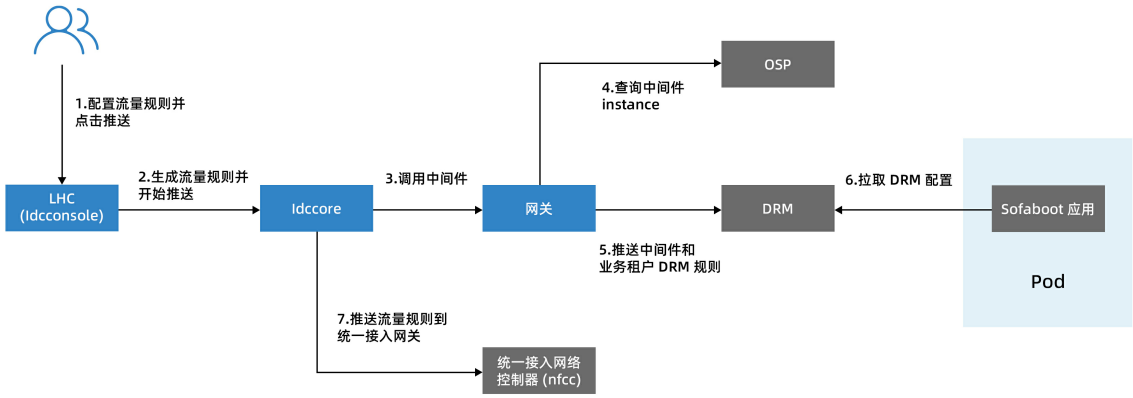
统一接入层管理为应用服务提供统一接入 spanner 集群和 spanner 实例的管理。用户可以在应用服务-访问配置为应用提供统一接入配置，在流量管理界面配置和推送流量统一接入规则。

- **spanner 集群**：承担统一接入和单元化网关的角色，在每个 K8s 集群内 spanner 集群平行发布运维，多个平行的 spanner 集群会在 nfcc 侧联合成一个“联邦单元化”集群，为一个逻辑的集群实体，任何一个单元化实例（LDC instance）都必须落在一个完整的逻辑单元化集群上。
- **单元化实例**：为一个 VIP 独立，单元化 spanner 集群共享的逻辑实例，在多个集群的情况下，该实例应该包含多个 VIP，需要前置域名作为统一入口。用户重点需要关注的是规则和后端服务器，无需关注端口的配置。




## 流量管理

单元化应用服务提供统一接入网关路由和中间件的流量规则管理和推送能力。流量规则管理可以管理各个部署单元的流量占比。在容灾场景下，可以通过将灾难发生的机房所属的部署单元流量切到正常机房，以保证业务访问正常并对外提供服务。


1.配置流量规则并
点击推送
LHC
(Idcconsole)
2.生成流量规则并
开始推送
Idccore
3.调用中间件
网关
4.查询中间件
instance
OSP
DRM
6.拉取 DRM 配置
Sofaboot 应用
Pod
5.推送中间件和
业务租户 DRM 规则
7.推送流量规则到
统一接入网关
统一接入网络
控制器 (nfcc)

# 5.基础术语

| 中文 | 英文 | 说明 |
| --- | --- | --- |
| 混合云 | Hybrid Cloud | 混合云狭义上指“公有云 + 私有部署”的混合形态，通过平台能力抽象统一、自动化部署、配置管理等方面的技术和产品，淡化开发和运维人员对底层基础设施的关注，使应用和数据能够在混合数据中心环境中进行部署和运维。 |
| 单元化工作空间 | LDC workspace | 提供单元化能力，可用于同城双活及异地容灾场景。您可以通过单元化工作空间组对用户资源进行隔离，不同工作空间组下的集群彼此隔离。 |
| 工作空间组 | WorkspaceGroup | 工作空间（Workspace）在多地域的扩展，在多地域内对资源进行分组隔离管理。多个地域的网络可以通过专线高速通道实现互通。 |
| 逻辑单元 | Zone | 一个单元被称为一个 Zone，有 3 种不同类型：RZone、GZone、CZone。单元的特点如下：<br>• 同一个应用在每个单元中拥有独立使用的资源。<br>• 同一个应用的业务在不同单元中按水平方向拆分。<br>• 不同单元处理的业务分片不重叠。 |
| 单元化架构 | LDC architecture | 应用层按照数据层相同的拆片维度，将整个请求链路收敛在一组服务器中，从应用层到数据层就可以组成一个封闭的单元。<br>数据库只需要承载本单元的应用节点的请求，大大节省了连接数。“单元”可以作为一个相对独立整体来挪动，甚至可以将部分单元部署至异地。 |
| 部署单元 | Cell | 部署单元（Cell），是指一个能完成所有业务操作的自包含集合，在这个集合中包含了所有业务所需的所有服务，以及分配给这个单元的数据。<br>单元化架构就是将单元作为部署的基本单位，在全站所有机房中部署数个单元，每个机房里的单元数目不定，任意一个单元都部署了系统所需的所有应用，数据则是全量数据按照某种维度划分后的一部分。 |
| 应用服务 | Application service | 该概念和 经典应用服务 中的应用服务概念一致。 但由于容器有其特殊性，LHC 中的应用服务会包含一些额外的元数据信息，比如容器规格配置、镜像、调度策略、日志配置等。 |

| 中文 | 英文 | 说明 |
| --- | --- | --- |
| 镜像 | Image | 镜像是应用包，将配置和相关软件等打在一起的二进制包，并且符合 Docker Image 规范。镜像可以来自任何可被 LHC 网络访问到的镜像中心，对于私有镜像中心，需要在 LHC 中配置相应的访问信息。 |
| 构建 | Build | 构建用于描述从应用源代码到制作出镜像过程的配置信息，包括源代码地址、分支信息、源镜像访问信息、目标镜像信息、Dockerfile 位置信息等。 |
| 集群 | Cluster | LHC 中集群用于描述您所创建的一个工作负载集群，由多个节点组成。 |
| 节点 | Node | 节点表示一台装了 Docker 和 Kubelet，用以运行应用负载的物理机或者虚拟机。 |
| 容器组 | Pod | Kubernetes 中最小的部署及管理单元。一个 Pod 由多个相关的并且共享磁盘的容器组成。 |
| 命名空间 | Namespace | 命名空间和 Kubernetes 中相应的概念保持一致，用于表示一个逻辑隔离的空间，会将 Pod、Service、ReplicaSet 等元素隔离，但通常来说，网络不隔离。 |
| 原地升级 | Inplace upgrade | 原地升级是指应用服务中 Pod 的更新方式。发布后 Pod 的 IP 通常和发布前无法保持一致，所在的节点也可能发生变化。该更新方式在镜像替换时不会导致 Pod 删除。 |
| 标签 | Label | Kubernetes 的原生概念，用于给相应的资源打上标签，做聚合或者匹配。 |
| 污点 | Taint | Kubernetes 的原生概念，用于给节点做污点标记，通常用于 Pods 的调度策略。<br>与之相对应的概念为：容忍（tolerance），若 Pods 上有相对应的 tolerance 标记，则可以容忍节点上的污点，并调度到该节点。 |
| 保密字典 | Secret | Kubernetes 的原生概念，用于存储用户的加密内容。 |
| 应用容器 | Container | 应用程序所运行的隔离工作空间，通常是 Docker 容器或者 Pouch 等兼容 CRI 接口的具有隔离能力的沙箱工作空间。 |

| 中文 | 英文 | 说明 |
| --- | --- | --- |
| 工作负载 | Workload | 应用程序运行态的载体及其上层聚合。通常包括：Pod、Deployment、StatefulSet、DeamonSet、Job 等。 |
| 配置项 | Configmap | Kubernetes 的原生概念，用于存储用户的配置信息。 |
| 存储类型 | Storage Class | Kubernetes 的原生概念，通常由系统管理员定义，用于指定所支持的存储类别，不同的类别会有不同的存储 SLA、备份策略等差异性。 |
| 存储卷 | Persistent Volume | Kubernetes 的原生概念，表示一个由系统管理员创建好的存储资源。 |
| 存储卷声明 | Persistent Volume Claim | Kubernetes 的原生概念，一个存储卷声明绑定一个存储卷。 |