字节跳动基于KubeAdmiral的分布式云原生多云多集群管理技术实践

×

李汉波

火山引擎
分布式云原生高级技术专家

目录

第一部分

分布式云原生行业现状，痛点及方案

分布式云原生行业现状

分布式云定义：

中国信息通信研究院（以下简称“中国信通院”）2019年在ITU-T联合立项《分布式云全局管理框架》国际标准，旨在进一步明确分布式云标准定义及关键管理要素，进一步明确分布式云标准定义：**分布式云是一种将云服务按需部署到不同地理位置，提供统一管理能力的云计算模式**。分布式云落地形态可表现为中心云、区域云和边缘云。

分布式云与当前云计算概念主要区别在于：**摒弃了公有云、私有云、混合云、多云等分类，将地理位置作为考量因素**，为用户提供不同位置的云资源统一管理平面。

分布式云原生定义：

分布式云原生是指通过云原生技术统一多云技术栈，提供业务价值的设计模式。

趋势分析：

- Gartner 2020/2021 十大顶级战略趋势之一，2025年超过50%的组织将使用分布式云实现业务转型
- 2021Flexera云状态报告显示，92%的企业选择多云战略，企业平均使用2.6个公有云+2.7个私有云
- IDC预测：到2023年，50%的中国企业应用将部署在容器化的混合云/多云环境中，以提供敏捷的，无缝的部署和管理体验
- 信通院“2021云计算十大关键词”之三: **云原生，混合云，边缘计算**

为什么需要分布式云原生统一管理

集群规模及稳定性

- **单一集群规模受限**
- **单一大集群故障风险大**
- **单一云厂商锁定，容灾难**

01

多云多集群运维成本高

- 多云环境异构，云厂商资源管理差异
- 多云集群管理入口分散，缺乏统一运维方案
- 多集群孤岛，无法统一调度，集群利用率低
- 自建调度逻辑，成本高

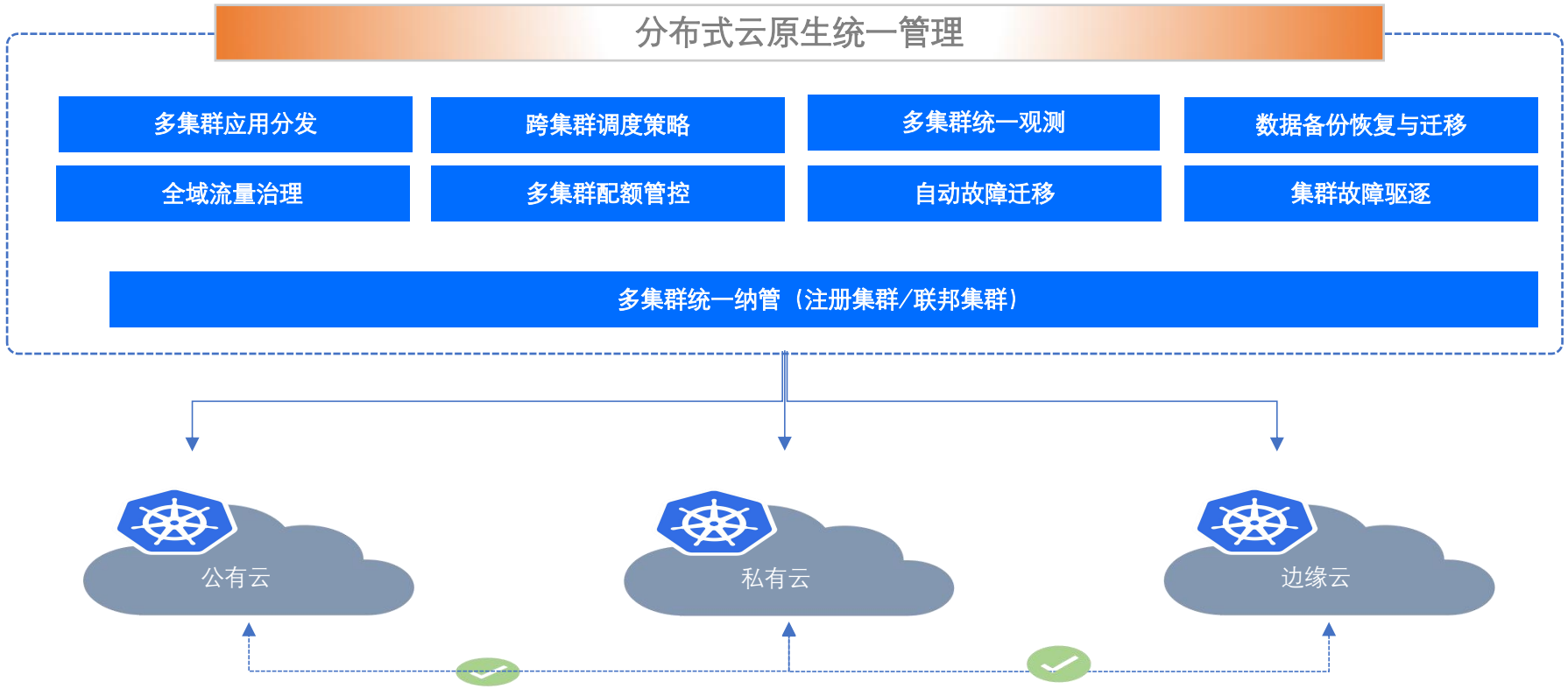
02

多云容灾困难

- 跨云迁移困难
- 多云流量如何调度
- 集群故障迁移困难

03

分布式云原生整体架构方案



第二部分

字节跳动基于KubeAdmiral的集群联邦技术实践

字节跳动云原生发展历程

现有

500+

生产集群，基础设施深度云原生化

2016

TCE 启动建设

TCE 为内部应用提供快捷高效的应用部署方案

2017

微服务架构

完成核心业务微服务迁移，并在 TCE 之上构建服务框架、Mesh、监控告警等基础设施

2019

“推广搜”云原生

把“推广搜”的物理机服务与在线服务进行全面融合，实现统一容器化调度

2020

离在线调度融合、存储云原生

融合资源管理形态，简化供应链选型；优化运维效率，开启数据库、缓存等存储系统的云原生化改造

2021

云原生联邦演进

资源多云到应用多云，实现全场景应用编排和资源管理的标准化和统一化

KubeFed—>KubeAdmiral

联邦集群管理节点数

210,000+

最大集群节点数上万，实现大规模集群落地

拥有

100,000+

在线微服务，敏捷化构建能力持续增强

平均每日变更数高达

30,000

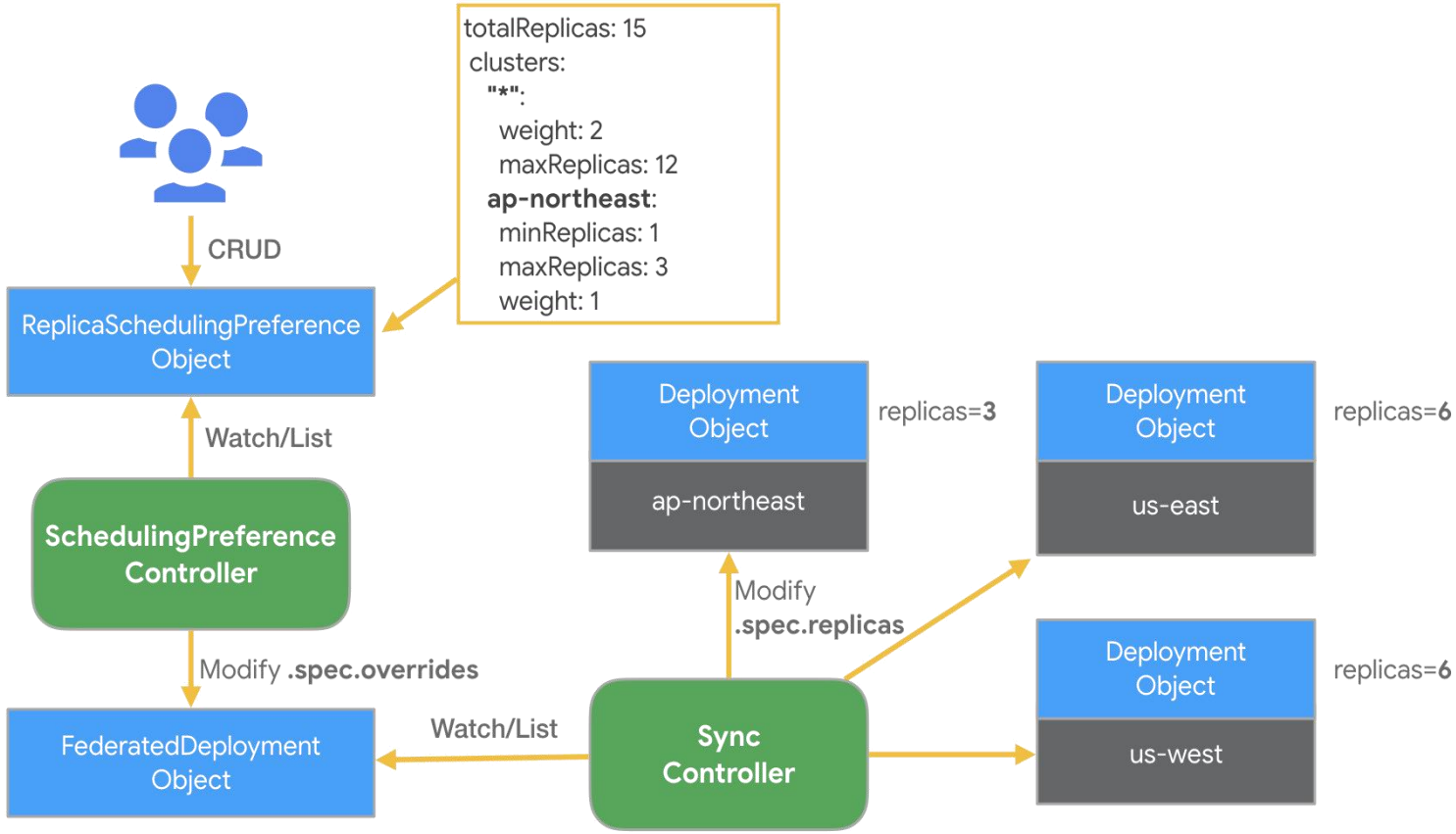
次，云原生为前线业务提供稳定性保障

离线任务数

140 M+

通过云原生混部，大规模节省企业资源成本

以 KubeFed (Federation v2) 为基础的集群联邦




```yaml
kind: FederatedDeployment
spec:
  overrides:
  - clusterName: ap-northeast
    clusterOverrides:
    - path: /spec/replicas
      value: 3
  - clusterName: us-east
    clusterOverrides:
    - path: /spec/replicas
      value: 6
  - clusterName: us-west
    clusterOverrides:
    - path: /spec/replicas
      value: 6
  placement:
    clusters:
    - name: ap-northeast
    - name: us-east
    - name: us-west
  template:
    ...
```

KubeFed V2 面临的问题

接入难度高

由于联邦层暴露的是“联邦化”的对象，而不是 Kubernetes 原生对象，上层服务往往需要针对联邦集群做特殊适配

调度能力弱

只支持 RSP 静态副本调度，导致成员集群水位不均，对其他调度策略只能在程序中特殊处理，扩展性差

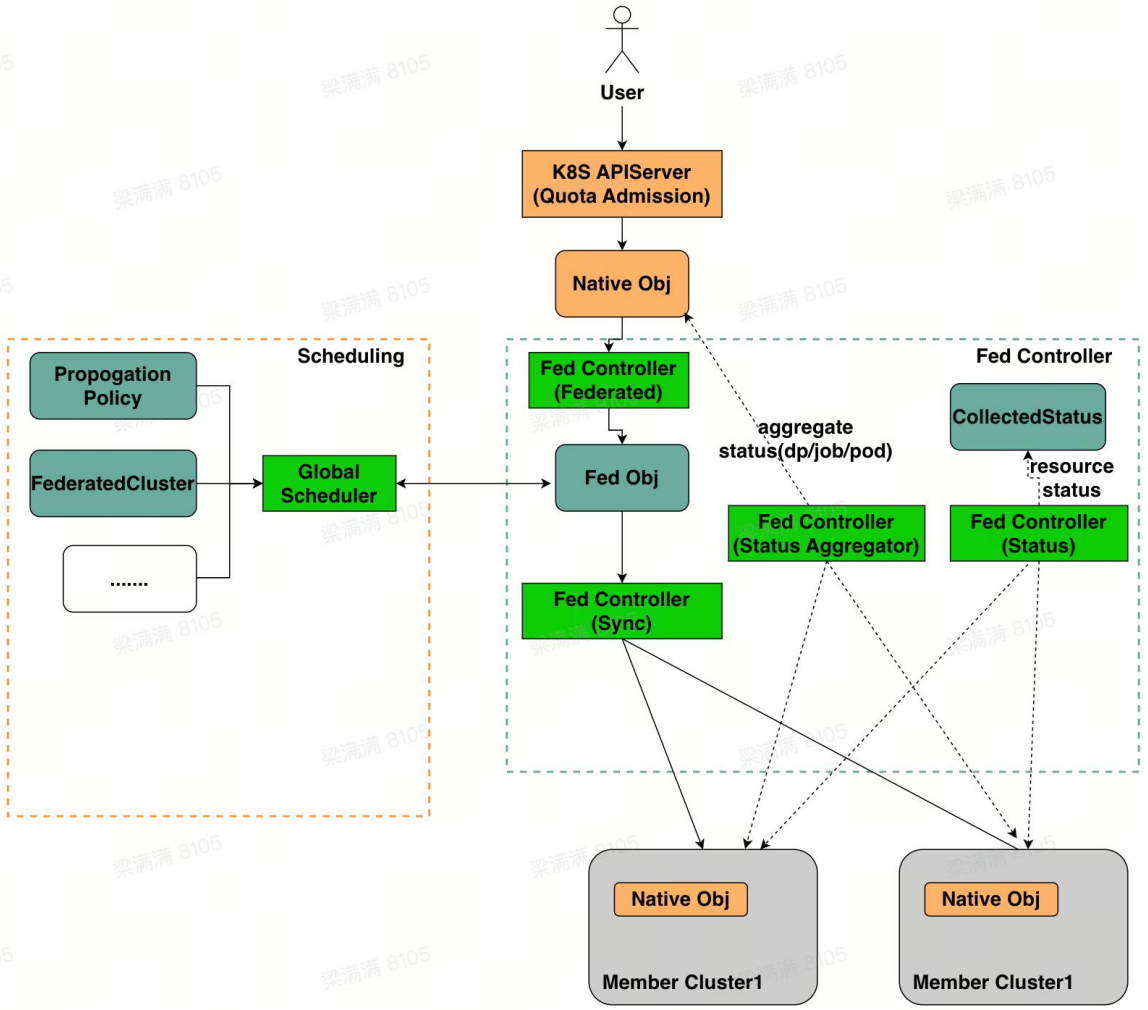
状态聚合能力差

只支持把资源的status字段采集上来，无法自定义采集字段，也没有对状态进行多云多集群语义的转换与汇聚

业务连续性

扩缩容过程中，可能产生 Rebalance，会影响业务稳定性；同时，故障迁移能力较弱

超大规模集群联邦 KubeAdmiral



https://github.com/kubewharf/kubeadmiral

- 兼容Kubernetes 的标准 API 以及自定义 CRD
- 调度器和调度算法可扩展，支持丰富的调度分发策略
- 支持差异化覆写策略
- 支持接管用户单集群已有资源，无损迁移
- 提供状态采集框架，自定义状态采集字段
- 更强大的故障自动迁移能力，涵盖集群故障迁移，应用故障迁移，污点迁移等场景
- 支持联邦HPA，可以低成本接管单集群HPA（原生/自定义）

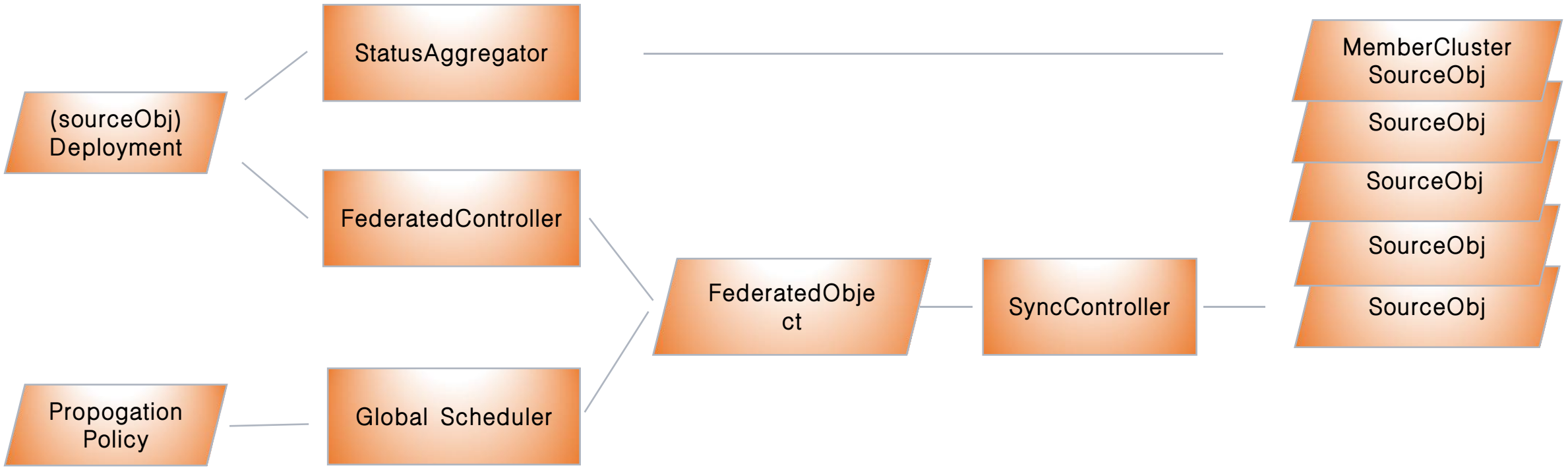
降低接入难度：支持原生 API 和 CRD

引入 FederatedController，负责将原生的资源转换为 FederatedObject

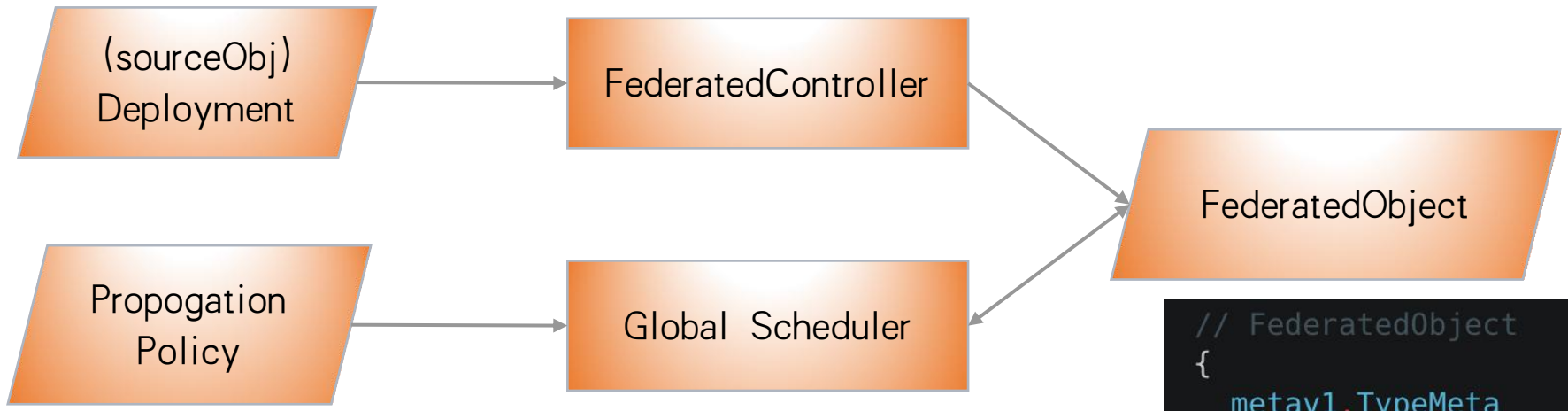

```
// FederatedObject
{
  metav1.TypeMeta
  metav1.ObjectMeta
  Spec
      GenericPlacementSpec
      GenericOverrideSpec
      Template
      RevisionHistoryLimit
      RetainReplicas
  Status
}
```

增强状态聚合能力：自定义采集字段，汇聚status

通过引入 StatusAggregator，负责将子集群资源的状态聚合到原生资源，针对不同的资源可以添加聚合 Plugin



提高调度扩展性：Global Scheduler（1）




```
// FederatedObject
{
  metav1.TypeMeta
  metav1.ObjectMeta
  Spec
      GenericPlacementSpec
      GenericOverrideSpec
      Template
      RevisionHistoryLimit
      RetainReplicas
  Status
}
```

提高调度扩展性：Global Scheduler（2）

Global Scheduler 的 Framework 和 Plugin：

- Filter：根据子集群的状态、亲和性、资源、是否支持 CRD 等等过滤不符合条件的 Member Cluster
- Score：对符合条件的 Members Cluster 打分，例如资源、亲和性等
- Selector：最多分发到多少个集群
- Replica：面向类似 Deployment 提供的副本数调度

提高调度扩展性：Global Scheduler（3）

Global Scheduler 支持丰富的调度策略模板，用户可在应用中指定所需要的调度策略模板，也支持用户使用自定义的扩展调度器

```
type PropagationPolicySpec struct {
    SchedulingProfile           // 灵活的指定 Scheduling Profile
    SchedulingMode              // 是否为副本数调度
    StickyCluster               // 仅在首次调度，适合有状态服务
    ClusterSelector             // 类似 NodeSelector
    ClusterAffinity             // 类似 NodeAffintiy
    Tolerations                 // 污点与容忍配置
    MaxClusters                 // 最多可分发到多少个子集群
    Placements                  // 用户指定集群或者静态权重
    DisableFollowerScheduling   // 是否开启依赖调度
}
```

```
apiVersion: core.kubeadmiral.io/v1alpha1
kind: PropagationPolicy
metadata:
  name: mypolicy # adjust name for your policy
  namespace: users-namespace # specify your namespace to apply policy
spec:
  schedulingMode: Duplicate # schedulingMode could be Divide or Duplicate
  disableFollowerScheduling: false
  placement:
    - cluster: ClusterID-01 # Modify the selected cluster ID to propagate the resource
      #preferences:
        #weight: 40
        #minReplicas: 1
        #maxReplicas: 3
    - cluster: ClusterID-02 # Modify the selected cluster ID to propagate the resource
      #preferences:
        #weight: 40
        #minReplicas: 1
        #maxReplicas: 3
  #clusterSelector:
    #region: beijing
    #az: zone1
  #clusterAffinity:
    #- matchExpressions:
      #- key: region
        #operator: In
        #values:
          #- beijing
      #- key: provider
        #operator: In
        #values:
          #- volcengine
  #tolerations:
```

提高调度扩展性：Global Scheduler（4）

Global Scheduler 支持通过http协议与外部插件交互，用户可以自行编写并部署定制化的调度逻辑：

- SchedulerPluginWebhookConfiguration：配置scheduler plugin访问地址
- SchedulingProfile：配置scheduler plugin启用顺序
- PropagationPolicy：指定使用的scheduling profile

```yaml
apiVersion: core.kubeadmiral.io/v1alpha1
kind: SchedulerPluginWebhookConfiguration
metadata:
  name: my-plugin
spec:
  payloadVersions:
    - v1alpha1
  urlPrefix: "http://127.0.0.1:50051"
  filterPath: 'filter'
  scorePath: 'score'
  selectPath: 'select'
```

```yaml
apiVersion: core.kubeadmiral.io/v1alpha1
kind: SchedulingProfile
metadata:
  name: sp-webhook
spec:
  plugins:
    filter:
      disabled:
      - name: '*'
      enabled:
      - name: my-plugin
        type: Webhook
    score:
      disabled:
      - name: '*'
      enabled:
      - name: my-plugin
        type: Webhook
    select:
      disabled:
      - name: '*'
      enabled:
      - name: my-plugin
        type: Webhook
```

```yaml
apiVersion: core.kubeadmiral.io/v1alpha1
kind: PropagationPolicy
metadata:
  name: pp-webhook
spec:
  schedulingProfile: sp-webhook
  schedulingMode: Divide
  maxClusters: 10
```

动态调度：基于集群水位

KubeFed V2问题：

- 副本调度策略 RSP 只能为每个 member 集群设置静态权重
- 静态权重很难做到集群部署率一致，在个别集群部署率很高时，很容易产生pending

基于集群水位的动态调度

- 进行权重计算时，同时考虑集群的资源总量 allocatable 和当前可用资源量 available
- 各个子集群总资源为：T1，T2，...Tn，各个子集群可用资源为：A1，A2，...An
- 则副本数比例为：R1，R2，...，Rn

$$R_i = Min(A_i/Sum(A), 1.4 * T_i/Sum(T))$$ # 1.4 为 SRE 最佳经验比例

After normalization: $r_i = R_i \ / \ Sum(R) \ * \ R_{total}$

效果

- 所有 member 集群的部署率都维持在 95% 以上，部分大规模 member 集群部署率可以达到 98%

重调度：避免业务中断

KubeFed V2问题：

30 个实例分布在 A B 2个 member 集群，集群权重1:1，现在用户增加一个集群C，同时缩容到 15 个实例：

- 用户预期：停止 15 个实例
- 按权重调度：停止 20 个实例，启动 5 个实例，瞬时可用实例5+5+0（启动之前），低于15

集群	A	B	C
实例数	15	15	0

缩容：30 -> 15

集群	A	B	C
瞬时实例数	5	5	0
实例数	5	5	5

30 个实例分布在 AB 2个 member 集群，集群权重1:1，现在用户增加一个集群C，同时扩容到36 个实例：

- 用户预期：启动 6 个实例
- 按权重调度：启动 12 个实例，停止 6 个实例，瞬时可用实例12+12+0（启动之前），低于30

集群	A	B	C
实例数	15	15	0

扩容：30 -> 36

集群	A	B	C
瞬时实例数	12	12	0
实例数	12	12	12

重调度：避免业务中断

当前方案：重调度策略增加avoidDisruption= true

avoidDisruption在扩缩容时特殊处理，以避免正常运行的pod因为权重的变化被迁移

- 先按权重算每个集群desired副本数，后结合当前副本数current计算最终分配副本数
- false: 直接用desired作为最终结果，可能会引起集群间的迁移，但最终分布完全遵循权重
- true：避免反向的pod移动（如果是扩容，则没有集群会缩容，反之亦然），但最终分布不完全遵循权重
  - 副本数不变时，直接使用current（不进行重分布）
  - 缩容时，只缩容current > desired的集群
  - 扩容时，只扩容current < desired的集群

集群	A	B	C
实例数 current	15	15	0

缩容：30 -> 15

集群	A	B	C
desired	5	5	5
final	7	8	0

扩容：30 -> 36

集群	A	B	C
desired	12	12	12
final	15	15	6

字节内场实践 - 流量协调、自动容灾

流量调度实现异常容器识别和规避

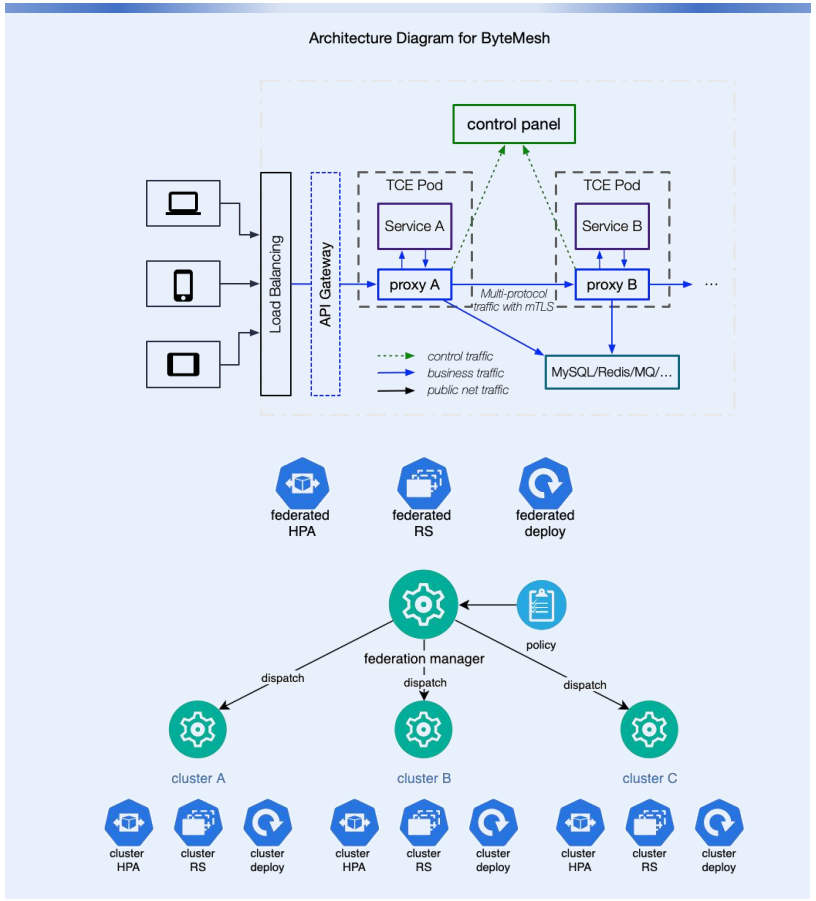
字节内部容器间的调用全量切入到 service mesh 体系，由其提供流量的代理和稳定性的保证

超时重试	服务限流	服务降级	负载均衡	流量切换
下游容器因为各种原因产生性能抖动，框架自动重试	容器配置 QPS 上限，请求量过高时自动拒绝，实现进程自我保护	下游容器错误率升高后，自动减少请求量，必要时直接丢弃请求	根据下游错误率和延迟指标，自动实现多容器间的负载均衡	集群或机房整体不可用时，自动将整体流量切换到正常集群或机房

集群联邦实现异常容器迁移和重建

集群联邦实现多集群联合，突破单集群规模上限，在集群间实现容器的自动化迁移

入口统一	故障切换	资源均衡
业务无感知接入，仍然看到单集群视角，优化使用体验	集群或者机房故障时，联邦自动实现全量容器的整体迁移	自动在多个集群实现容器平衡调度，减少单集群故障影响面



第三部分

从集群联邦到分布式云原生多云多集群管理实践

字节跳动分布式云原生整体能力

基于K8s和容器技术，让用户进行云原生应用部署和管理时感受不到云厂商、地域、流量的限制的PAAS服务。

统一集群管理

- 支持纳管多地域，多基础设施K8S集群
- 提供统一集群管理入口
- 支持单集群应用迁移到多集群（联邦）
- 应用备份，恢复与迁移
- 多集群RBAC统一认证

统一应用分发

- 兼容原生K8S API和资源/CRD语义
- 支持联邦/非联邦两种形态
- 支持多集群灵活调度：动态/跟随
- 支持集群差异化复写

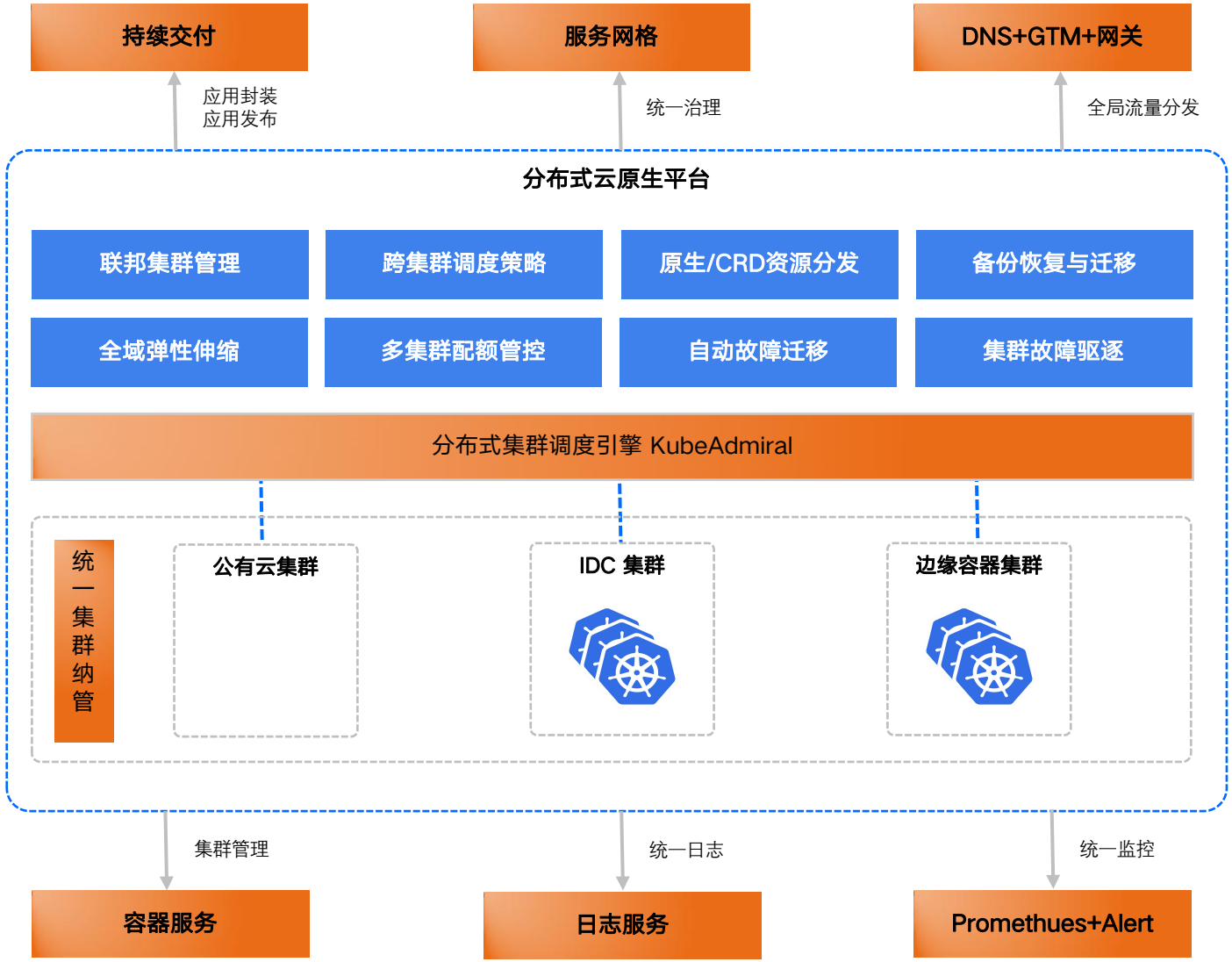
统一集群观测

- 多集群监控、日志数据统一收集与展示
- 多集群监控数据聚合
- 应用维度监控数据聚合
- 多集群故障诊断

统一流量管理

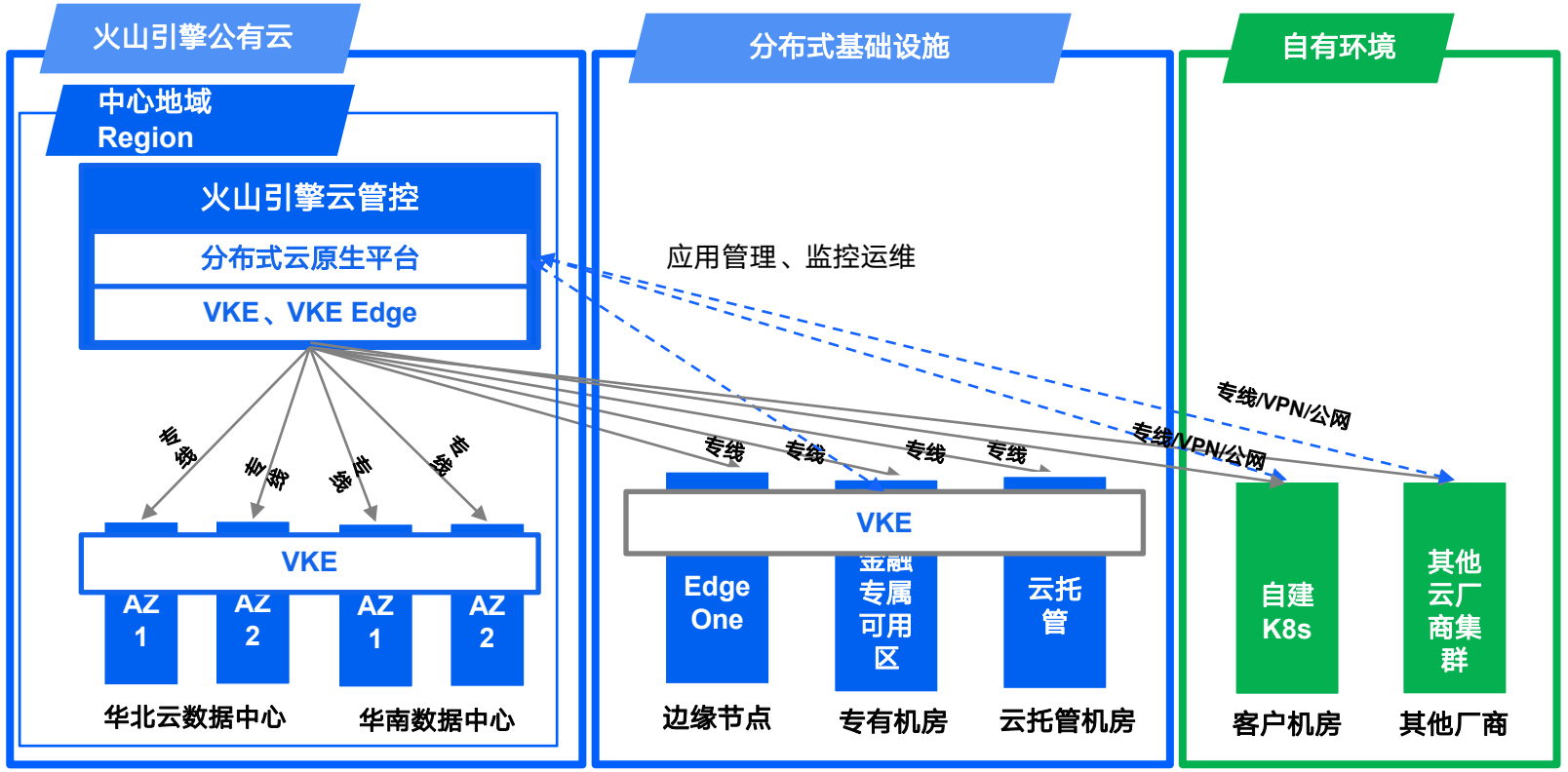
- 多集群统一流量调度
- 支持东西/南北向
- 服务注册/服务发现

分布式云原生平台实践架构



核心能力-多集群管理

提供分布式集群全局管理能力，具备一致的集群管理体验，降低运维成本。

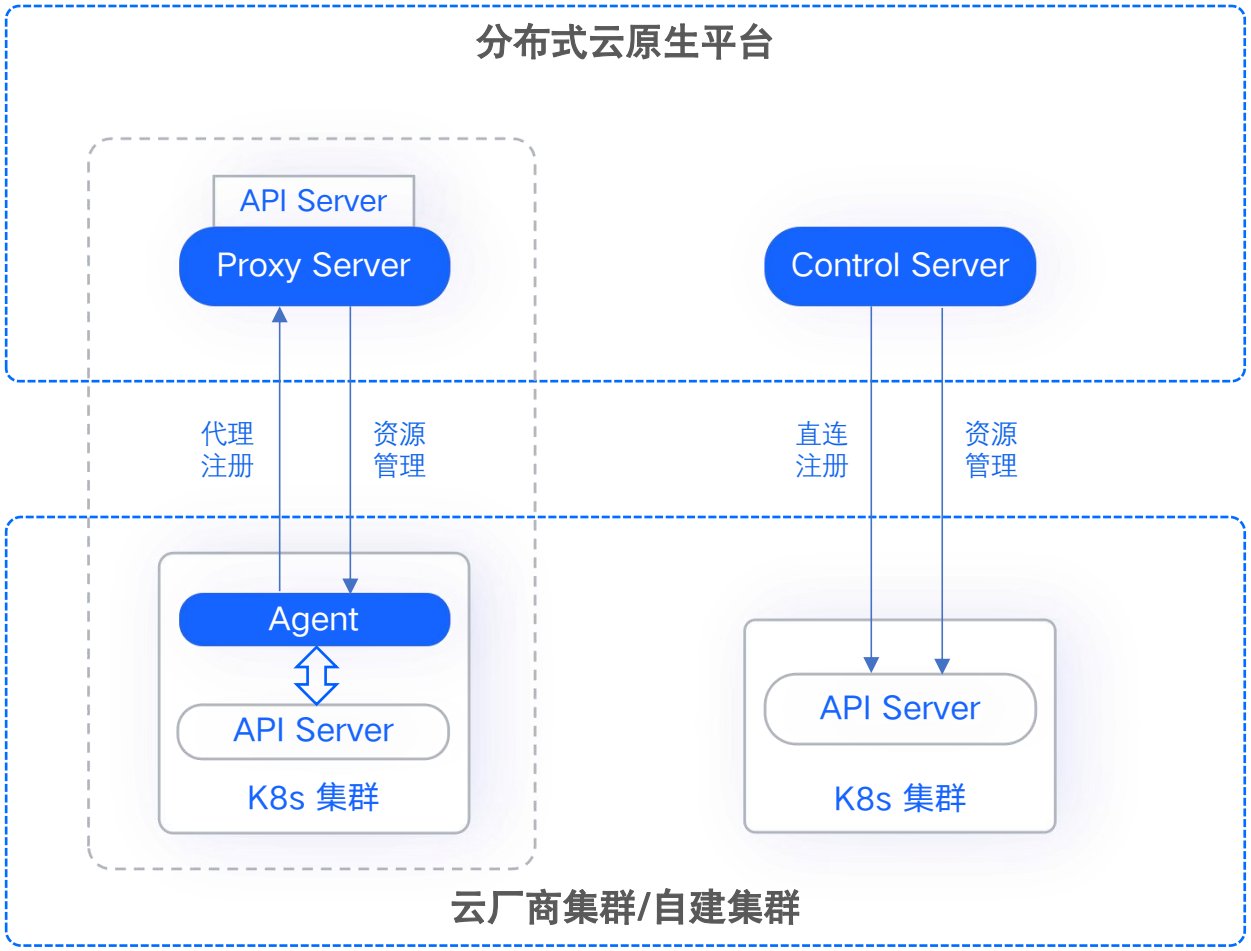


泛互联网 | 传统企业 | 运营商、广电 | IDC上云 | 金融、政府、企业 | 交通、工业、教育

- **多集群纳管**：连接并管理用户任何地域、任何基础设施上的 K8s 集群
- **统一资源管理**：使用容器服务控制台统一管理集群和应用，避免在不同云平台之间切换控制台。
- **统一安全治理**：对多云集群进行一致的 RBAC 授权管理。
- **统一可观测体系**：纳管集群监控、日志数据统一收集与展示。

核心能力-集群连接模式

分布式云原生平台纳管外部集群时，支持通过直连模式、代理模式进行注册接入



直连模式：通过目标集群提供的 KubeConfig 进行直接注册管理的方式。

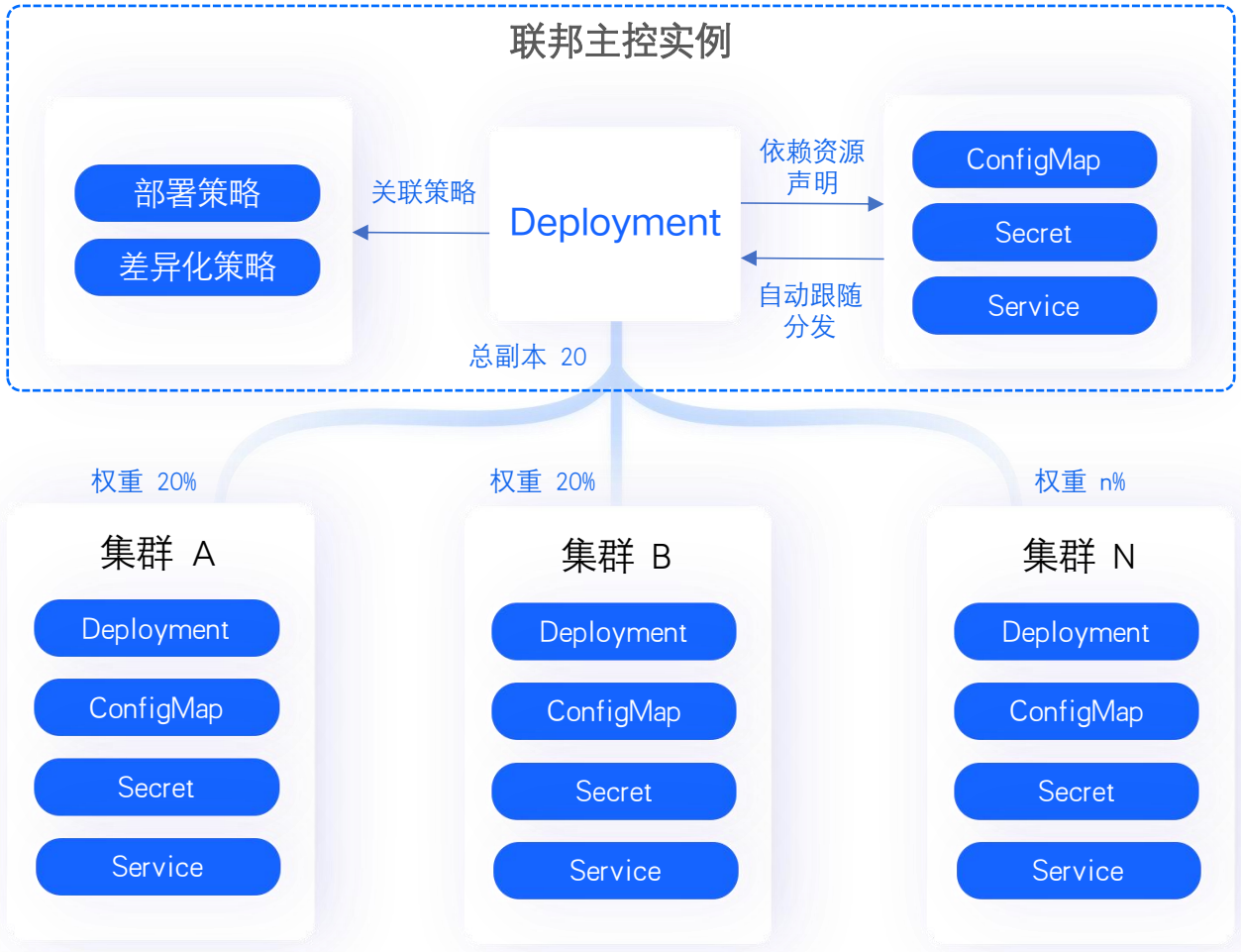
- 场景：控制面可以通过外网或同一私网中直接访问目标集群的 API Server 地址。如：私有云 DCP 公网直连注册公有云集群。
- 优势：注册简单、不消耗集群资源（可纳管空节点集群）
- 劣势：证书失效&到期替换问题、证书泄露风险

代理模式：控制面暴露代理服务地址，目标集群通过安装代理连接到控制面后进行统一管理的方式。

- 场景：当控制面无法直接访问目标集群，但反向可连通的场景。如：公有云 DCP，目标集群部署在 IDC 环境。
- 优势：无证书到期问题、注册安全（不泄露 KubeConfig）
- 劣势：需要保证集群有节点资源及带宽运行 Agent

核心能力-应用跨集群分发

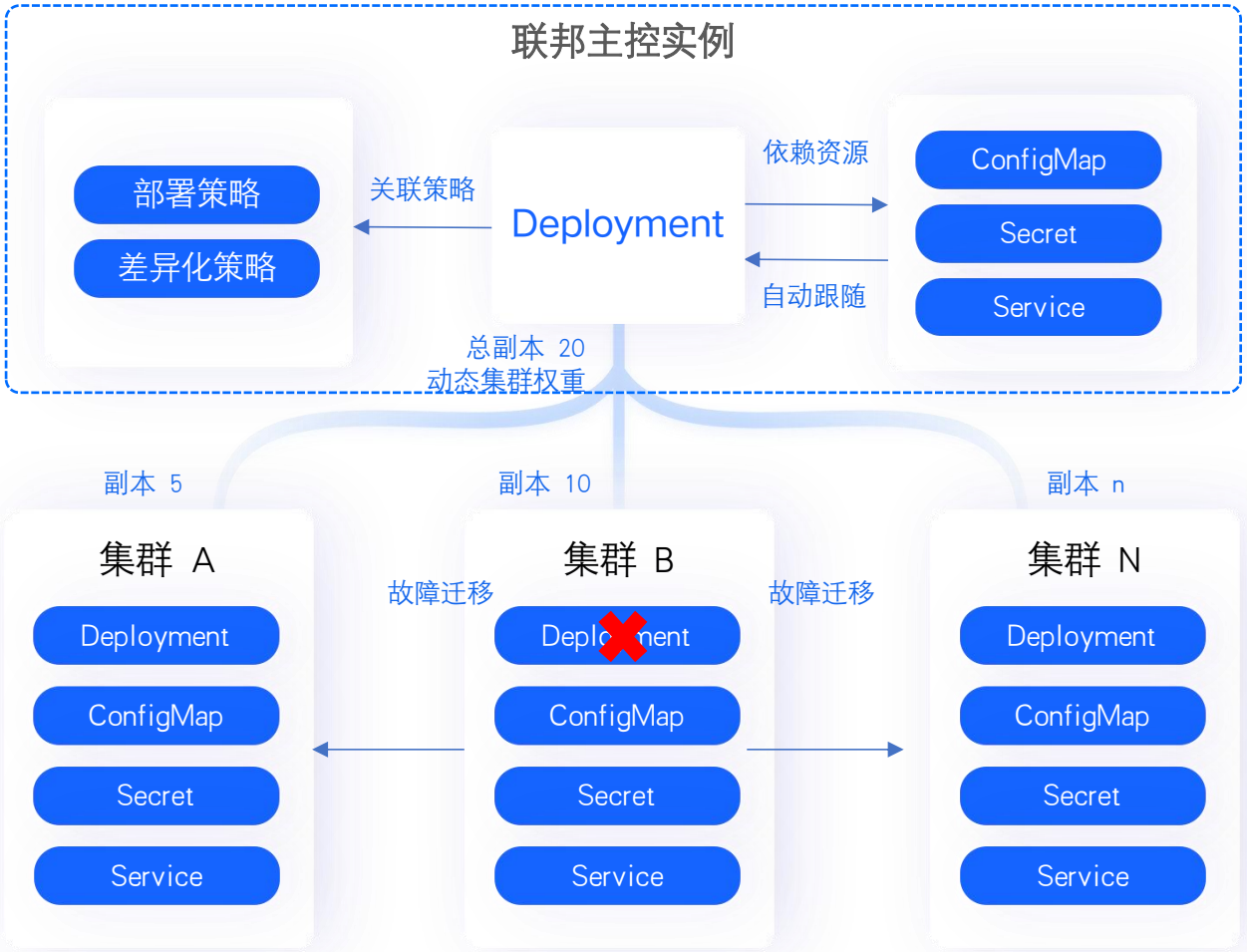
将多集群构建成为**集群联邦**形成统一资源池，通过统一算力分发入口及多集群分发策略，可高效地将应用分发到目标集群，实现了**降低操作复杂度、减少运维成本**，同时可大大**提高集群部署率和资源利用率**。



- **统一应用分发**：通过统一界面入口或 kubeconfig 将应用分发到多个集群中，而无需切换集群进行多次操作。
- **兼容K8s生态**：支持原生 K8s 及 CRD 资源、Helm 等应用定义。
- **存量应用无缝接管**：无需改造应用及不影响现有业务运行的情况下转变为多集群联邦应用，降低使用门槛和迁移成本。
- **灵活的分发策略**：支持指定集群名称、标签、污点容忍调度；支持依赖资源自动跟随分发；支持分发到不同集群时的差异化覆写配置。

核心能力-多集群应用故障迁移

为了提高业务的高可用性，工作负载可能会被部署在多个集群中，当集群发生故障或应用在该集群中无法正常部署时，需要进行**自动/手动故障迁移**，进而**保证用户业务的可用性与连续性**。

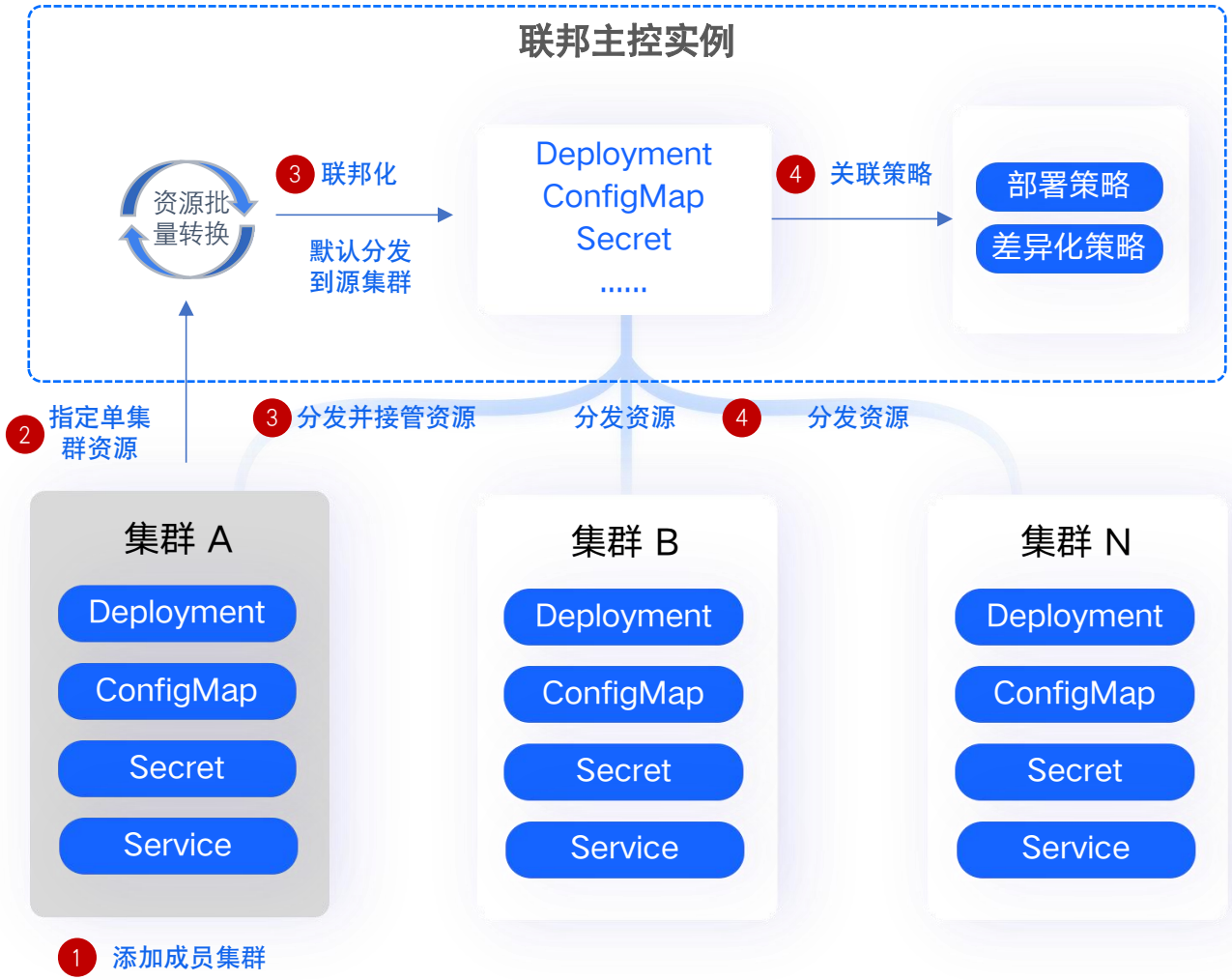


- **集群故障迁移**：当集群发生故障(不健康或失联)，或是不希望在某个集群上继续运行工作负载（如集群下线、升级）时，支持手动进行集群应用驱逐，被驱逐的工作负载将被调度至其他健康的集群中。

- **应用故障迁移**：当应用实例调度到目标成员集群后无法正常启动(Cond=Unschedulable)，联邦控制面会触发重调度并将实例迁移到其他健康集群中。

核心能力-单集群资源联邦化

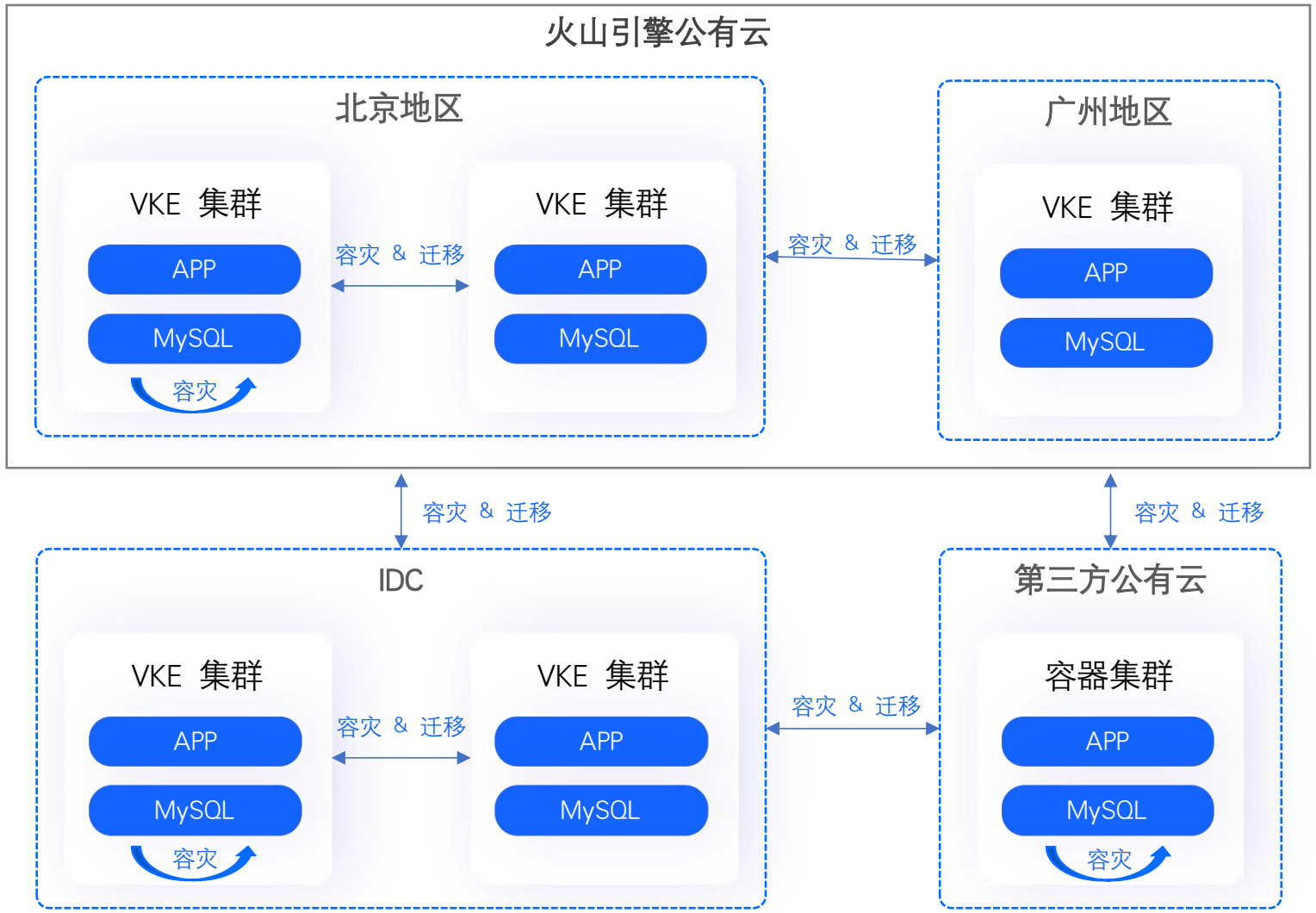
凭借多集群调度引擎对存量单集群资源的无缝接管能力，可以做到无需改造应用及不影响现有业务运行的情况下转变为多集群联邦应用



资源联邦化步骤

- **步骤 1**：将待联邦化资源所在容器集群（以下称源集群）添加到联邦主控实例中，作为联邦集群成员集群。
- **步骤 2**：指定源集群中一个或多个待联邦化的资源，支持指定 K8s 原生和 CRD 资源。
- **步骤 3**：平台自动将单集群资源转化为联邦资源定义，并关联默认部署策略将资源分发到源集群，对单集群资源进行接管（anno:kubeadmiral.io/conflict-resolution：adopt）。
- **步骤 4**：已经联邦化的资源可更新部署策略、差异化策略，实现多集群的资源分发/资源迁移。

核心能力-集群备份恢复迁移



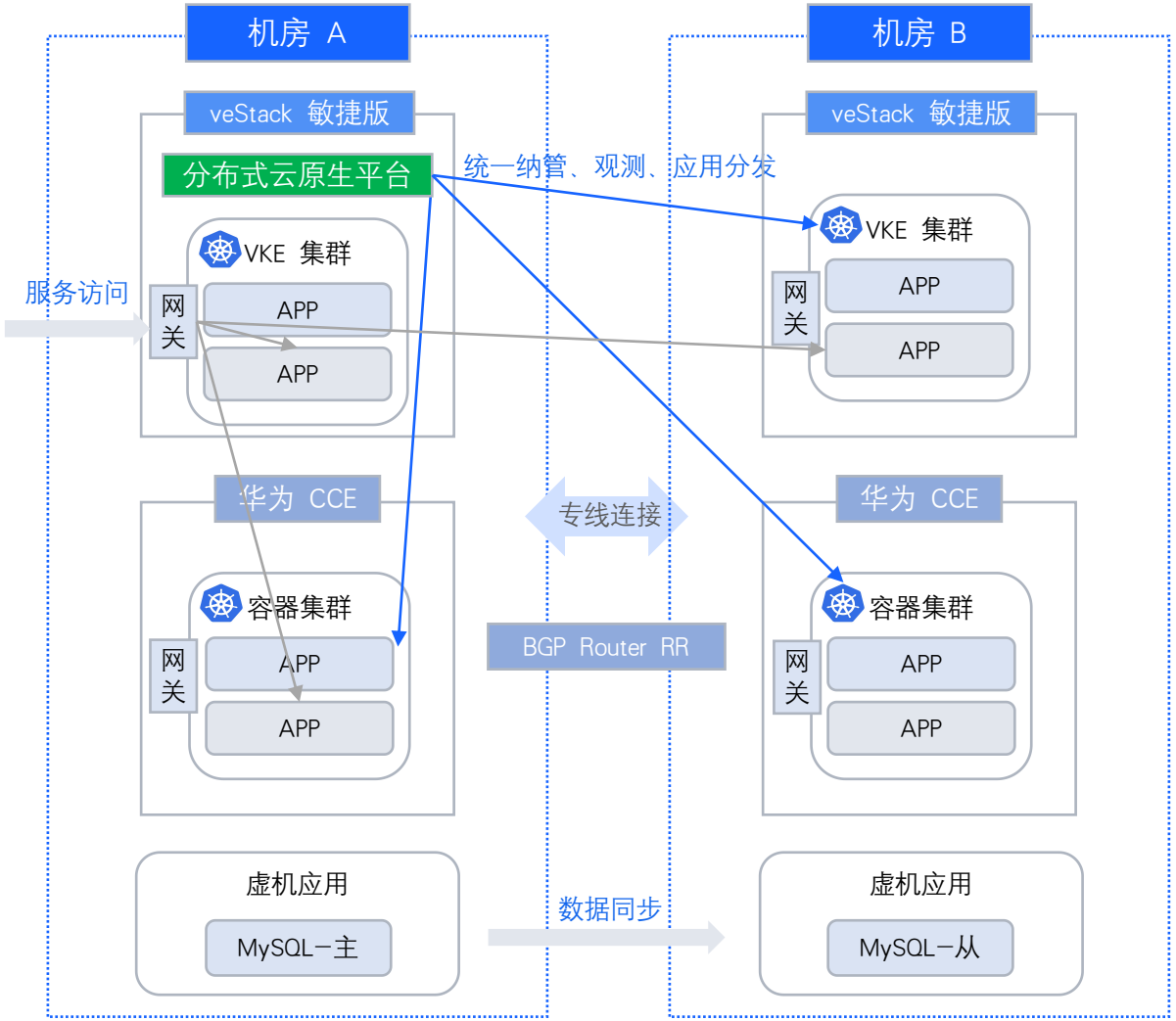
应用场景

- 单集群内应用容灾
- 跨集群/地域/云厂商容灾与迁移
- 混合云集群间容灾与迁移

核心特性

- **应用数据备份与恢复**： 将应用元数据、持久化卷数据(定期)备份到对象存储仓库中，待必要时进行快速恢复
- **恢复资源更新适配**：源备份与恢复数据的命名空间、存储类等资源的重映射。

客户案例-某银行客户



客户痛点

- K8S集群分布在多家私有云上，运维管理成本高
- 希望应用可以跨集群/机房容灾

解决方案

- 多云集群统一纳管、观测
- 联邦应用跨集群分发
- 联邦集群统一资源分发入口
- 容器应用数据备份恢复、迁移
- API 网关实现多集群服务发现、负载均衡、集群或应用故障时自动切流

客户价值

- 应用多集群容灾，提高业务可用性、连续性
- 多集群统一管理，提高资源管理、运维效率
- 多集群动态权重调度，提高集群部署率、资源利用率

第四部分

分布式云原生多云多集群管理未来展望

未来演进方向

在线应用

- 精准调度，使用率调度
- 联邦HPA
- 监控聚合
- 多集群故障诊断
- 应用拓扑

离线计算

- GPU卡调度
- 成本/成功率调度
- Argo workflow深入适配
- Kubeflow/volcano等生态适配

多云FinOps

多云安全

KubeAdmiral开源

KubeAdmiral 已经开源：https://github.com/kubewharf/kubeadmiral

之后将吸取社区经验与反馈，并不断更新和完善KubeAdmiral的功能，以更好地回馈社区。

扫码关注字节跳动云原生

Thanks