SOFAStack

单元化应用服务 LHC

运维指南

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 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 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 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 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` *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}`

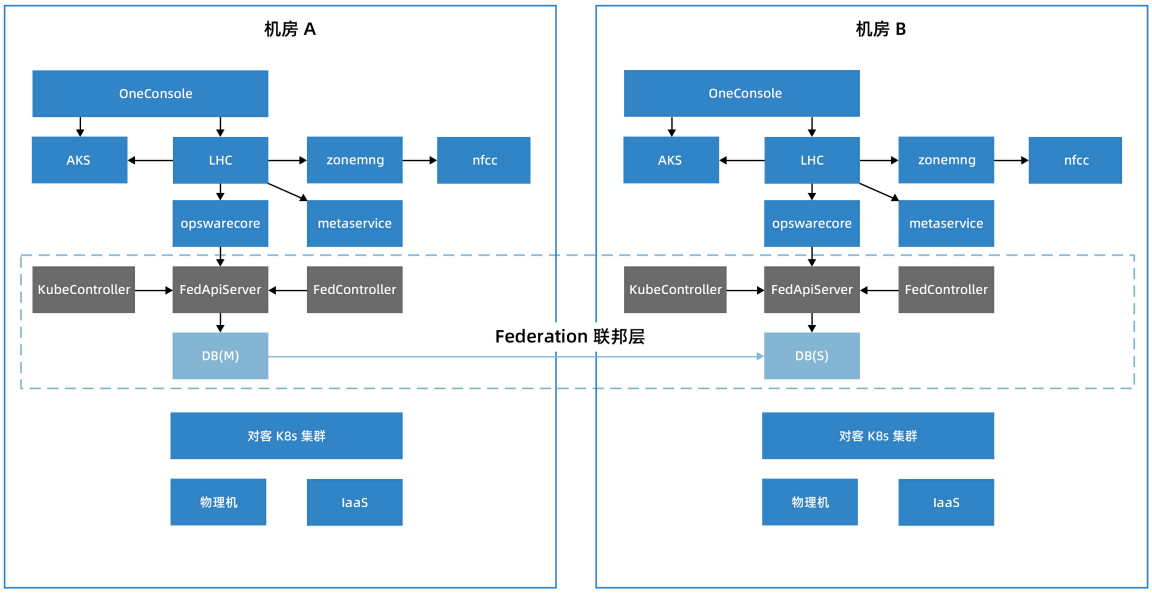
目录

1.运维概述

1.1. 产品架构

1.1.1. 系统架构

本节内容旨在帮助您在运维过程中了解 CAFE 的系统架构部署架构及其内部调用关系。



各组件说明如下：

- **产品控制台 OneConsole**

  OneConsole 用于为 LHC 提供用户访问的外部界面服务，是目前 LHC 提供给用户的唯一产品入口。通过控制台可以完成应用生命周期管理、应用发布部署、运维管控、镜像构建、元数据管理、单元化流量调拨等操作。

- **容器服务 AKS**

  容器服务 AKS 提供应用集群管理、镜像构建、镜像中心、原生工作负载的后台业务逻辑实现，提供 AddOn 组件在对客集群中的安装和升级。

- **单元化混合云服务 LHC**

  单元化混合云服务 LHC 提供应用服务生命周期管理、多集群发布部署、异地多活场景下的单元化流量管理、多地域高可用统一接入网关、同城和异地容灾等能力。

- **发布部署核心 opswarecore**

  发布部署核心 opswarecore 提供多集群运维管控原子能力。

- **流量管理 ldccore**

  流量管理 ldccore 提供单元化流量规则管理、容灾规则管理和灾难场景下的容灾规则推送（接入层、微服务层）的核心能力。

- **统一接入网络 nfcc**

从联邦视角提供多地域多机房的网关集群管理和统一接入规则的下发能力。

- **元数据管理 metaservice**

  核心领域模型管理系统，提供租户、工作空间组、工作空间、应用、地域、机房等元数据的增删改查能力。

- **联邦层 API 服务端 FedApiServer**

  FedApiServer 通过标准的 Kubernetes APIServer 接口为其他组件（如 FedController、LHC、opswarecore 等）提供读写联邦层数据的能力。

- **联邦层核心控制器 FedController**

  FedController 是运行在联邦层的 Kubernetes controller，其负责联邦层的多集群资源分发与状态回流逻辑。

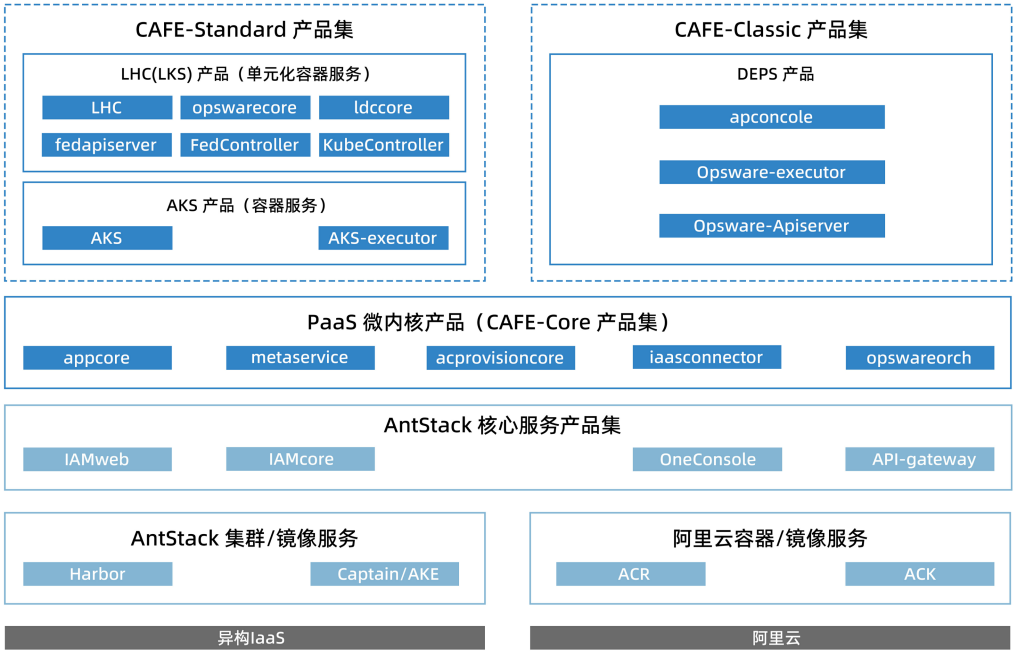
- **联邦层辅助控制器 KubeController**

  KubeController 是运行在联邦层的标准 Kubernetes Controller Manager，为联邦层提供命名空间生命周期管理与垃圾回收能力。

- **其他组件依赖**
  - 日志服务、监控、事件中心是容器产品的依赖组件，分别完成日志、监控、事件相关服务。
  - 产品 Open API 提供 openapi 网关入口。
  - 云游用于部署 CAFE 及依赖的相关组件，以及部署用户 SaaS 应用。
  - CAFE 产品依赖蚂蚁中间件注册中心服务完成自身组件的服务发现和消费。

1.1.2. 部署架构

下图展示了 CAFE 的整体部署架构：



各模块的说明如下：

- 阿里云底座下，使用阿里云 ACK 产品作为集群管理平台；AntStackPlus 底座下，使用 Captain 作为集群生命周期管理平台，提供 CAFE 集群导入能力。
- 云游用于部署 CAFE 及依赖的相关组件。
- CAFE 对客集群用于部署用户 SaaS 应用。
- CAFE 产品依赖中间件注册中心服务完成自身组件的服务发现和消费，同时会调用 CAFE 对客集群完成用户本身应用的发布部署和运维操作。
- LHC 会自动在纳管的用户集群内安装 CAFE AddOns 组件，来支持高阶的策略发布、灵活的负载均衡和统一接入流量管理、日志采集、镜像下载加速等。

1.2. 运维架构

在您了解了 CAFE 的产品架构后，本文档可以帮助您了解 CAFE 的运维架构。

运维大类	描述	运维工具
日常运维	包括人工巡检和监控	业务实时监控 RMS
停机维护	• 检查、确认容器、组件状态 • 停止、启动容器、组件 • 升级组件	云游运维平台
故障处理	主要处理 CAFE 的组件可用性和业务连续性故障。	云游运维平台和命令行

1.3. 技术支持渠道

服务热线

若有需要，可随时拨打热线电话：400-995-1887。

提交工单

若您有额外的产品功能需求或非紧急的技术咨询，可通过提交工单联系蚂蚁集团技术支持团队。

工单提交地址：https://user.cloud.alipay.com/#/tickets/submit。

2.高危操作

2.1. 运维操作的危险级别定义

所有运维操作按业务场景和操作的危险程度被分为三级：G1、G2、G3。

各级别定义如下：

- G1：L1|L2 可以根据文档完成操作，不需要做变更申请，该操作按文档执行完全安全，该操作不会对业务产生影响。
- G2：L1|L2 驻场人员可以根据文档指引，需要做变更申请，向产品侧咨询确认方可执行操作。该操不会对业务产生影响。
- G3：L1|L2 驻场人员可以根据文档指引，需要做变更申请，向产品侧以及客户侧咨询确认方可执行操作。该操作有可能对业务产生影响。

> 说明
>
> 原则上，驻场运维人员对任何变更操作都应该做变更申请，仅限于少数的应急场景中可以不做申请直接操作。

2.2. 运维操作和文件

本节将列出您在运维时需格外注意的高危运维操作和不能随意清理的高危文件。

高危操作

高危操作即上文提到的 G3 级别的运维操作，进行该类操作时请按照相应的指导进行。

迁移 hub 实例：因 hub 会存放用户镜像数据，迁移 hub 实例属于高危操作。

迁移 hub 实例需要根据具体的存储方案来执行：

- OSS 存储

  若存储为 OSS 存储，可以直接进行迁移，无具体风险。

- Minio 存储

  若存储为 Minio 存储，可以直接进行迁移，无具体风险。

- 本地存储

  若存储为本地存储，hub 将按照本地存储的拓扑进行部署，迁移时需要进行以下数据处理：

  > 说明
  >
  > hub 的本地存储的拓扑目前有两个应用，分别为 hub1 和 hub2。

  i. 登录云游运维平台管控，查看需要迁移的 hub 应用所在的宿主机，比如如果需要迁移的是 hub1，则查看 hub1 应用所在的宿主机。

  ii. 登录物理机，将物理机作为 `/home/t4/harbor/data` 的数据同步到新的宿主机上。

  iii. 将 hub1 应用迁移到上述新的宿主机上。

iv. 重新部署 hub2 应用。

v. 检查 hub 的功能的可用性，通过命令下载镜像，确定是否正常。

高危文件

高危文件是指不能被随意清理的文件，如需清理，需要与产品，客户进行确认。

应用	路径	用途
hub	容器内：/data/ 宿主机：/home/t4/harbor/data	存放用户镜像数据
hub2	容器内：/data/ 宿主机：/home/t4/harbor/data2	存放用户镜像数据

3.运维准备

产品运维过程中需要准备以下登录入口、账号、权限及工具。

名称	登录入口	权限
云游	yunyou.example.com。example 根域名，需要根据实际环境确认。	产品管理员
监控	rms.example.com。example 根域名，需要根据实际环境确认。	管理员

4.应急维护

4.1. 应急维护的定义和流程

4.1.1. 应急维护的定义

在出现重大故障时，最短时间内进行业务恢复的处理定义为应急处理操作。

所谓重大故障，是指发生突然、影响面广、涉及范围大、并可对网络的安全运行与服务质量造成严重后果的故障，如可用性故障。

4.1.2. 应急维护的流程和原则

如遇到下文覆盖的故障，建议按照文中指导进行故障排查和修复；若故障排查和修复遇到困难，可联系技术支持寻求帮助。如遇到下文未覆盖的故障，请直接联系技术支持。

4.2. 应用宕机

4.2.1. aks 应用宕机

【现象】CAFE 集群管理页面异常，报没有集群。`antcloud.aks.xx` OpenAPI 调用失败。

【可能的原因】

- Java 进程内存溢出导致应用宕机
- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录敏捷云游运维平台页面，找到 AKS 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，执行步骤 3。
2. 前往云游平台发起容器重启操作。
3. 通过 `ps -ef|grep java|grep -v xflush` ，检查容器 Java 进程是否存在。
   - 若 Java 进程不存在，执行步骤 4。
   - 若 Java 进程存在，执行步骤 5。
4. 查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件。若有此文件，保存此文件，参考步骤 2 重启服务。
5. 执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】检查页面是否可以正常访问。

4.2.2. aksexecutor 应用宕机

【现象】应用构建全部失败，一触发任务就直接报错。

【可能的原因】

- Java 进程内存溢出导致应用宕机
- 底层异常导致应用宕机

【解决方法】

1. 登录敏捷云游运维平台页面，找到 aksexecutor 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，执行步骤 3。
2. 前往云游平台发起容器重启操作。
3. 通过 `ps -ef|grep java|grep -v xflush` ，检查容器 Java 进程是否存在。
   - 若 Java 进程不存在，执行步骤 4。
   - 若 Java 进程存在，执行步骤 5。
4. 查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件。若有此文件，保存此文件，参考步骤 2 重启服务。
5. 执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】重新发起构建请求，看服务是否正常。

4.2.3. hub 应用宕机

【现象】镜像中心页面无法访问，客户端操作镜像中心无法上传下载。

【可能的原因】

- 内存溢出导致应用宕机
- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录云游平台页面，找到 hub 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，执行步骤 3。
2. 前往云游平台发起容器重启操作。
3. 通过云游运维平台 shell 功能登录容器，执行 `ps -ef | grep registry` 查看进程是否存在，可参考下图。

若不存在，通过运行

```
bash -c "unset REGISTRY_STORAGE_PROVIDER_NAME && /harbor/registry_entrypoint.sh
/etc/registry/config.yml" &
```

重启 registry 服务。

【验证方法】客户端上传镜像确定镜像服务是否正常。

4.2.4. metaservice 应用宕机

【现象】cafestandard 整个服务不可用。

【可能的原因】

- Java 进程内存溢出导致应用宕机
- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录云游运维平台页面，找到 metaservice 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，执行步骤 3。
2. 前往云游平台发起容器重启操作。
3. 通过 `ps -ef|grep java|grep -v xflush` ，检查容器 Java 进程是否存在。
   - 若 Java 进程不存在，执行步骤 4。
   - 若 Java 进程存在，执行步骤 5。
4. 查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件。若有此文件，保存此文件，参考步骤 2 重启服务。
5. 执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】检查页面是否可以正常访问。

4.2.5. lhc 应用宕机

【现象】容器服务应用服务或发布部署页面异常，应用服务或发布部署接口调用失败。

【可能的原因】

- Java 进程内存溢出导致应用宕机
- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录敏捷云游运维平台页面，找到 LHC 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，执行步骤 3。
2. 前往云游平台发起容器重启操作。
3. 通过 `ps -ef|grep java|grep -v xflush` ，检查容器 Java 进程是否存在。
   - 若 Java 进程不存在，执行步骤 4。
   - 若 Java 进程存在，执行步骤 5。
4. 查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件。若有此文件，保存此文件，参考步骤 2 重启服务。
5. 执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】检查发布部署页面是否可以正常访问。

4.2.6. opswarecore 应用宕机

【现象】容器服务应用服务/发布部署页面正常，但是发布部署时 pod 名称和预期不符，pod 长不出来。

【可能的原因】

- Java 进程内存溢出导致应用宕机
- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录敏捷云游运维平台页面，找到 opswarecore 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，执行步骤 3。
2. 前往云游平台发起容器重启操作。
3. 通过 `ps -ef|grep java|grep -v xflush` ，检查容器 Java 进程是否存在。
   - 若 Java 进程不存在，执行步骤 4。
   - 若 Java 进程存在，执行步骤 5。
4. 查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件。若有此文件，保存此文件，参考步骤 2 重启服务。
5. 执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】检查流量管理推送是否正常。

4.2.7. ldccore 应用宕机

【现象】流量管理-流量推送失败。

【可能的原因】

- Java 进程内存溢出导致应用宕机
- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录敏捷云游运维平台页面，找到 ldccore 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，执行步骤 3。
2. 前往云游平台发起容器重启操作。
3. 通过 `ps -ef|grep java|grep -v xflush` ，检查容器 Java 进程是否存在。
   - 若 Java 进程不存在，执行步骤 4。
   - 若 Java 进程存在，执行步骤 5。
4. 查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件。若有此文件，保存此文件，参考步骤 2 重启服务。
5. 执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】检查流量管理推送是否正常。

4.2.8. fedapiserver 应用宕机

【现象】LHC 应用服务、配置管理、网络等页面异常，且从浏览器控制台查看出错的 API 调用为 FederatedXXX 相关，并通过排查 LHC 的日志得出是调用 fedapiserver 报错。

【可能的原因】

- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录云游运维平台页面，找到 fedapiserver 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，说明不是宕机产生的问题，参考 Fed 问题排查 进行进一步排查。

> **说明**
>
> 当前容器内没有 bash，只能使用 sh 登录。

2. 在云游平台发起容器重启操作。

【验证方法】检查之前异常的页面是否恢复正常访问。

4.2.9. fedcontroller 应用宕机

【现象】LHC 发布部署故障 处理中推断为联邦层问题；命名空间、配置管理页面新建资源均卡在“同步中”的状态。

【可能的原因】

- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录云游运维平台页面，找到 fedcontroller 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，说明不是宕机产生的问题，参考 Fed 问题排查 进行进一步排查。

   > 说明
   >
   > 当前容器内没有 bash，只能使用 sh 登录。

2. 在云游平台发起容器重启操作。

【验证方法】检查之前异常的功能是否恢复正常。

4.2.10. kubecontroller 应用宕机

【现象】LHC 页面上删除命名空间卡在“删除中”。

【可能的原因】

- 组件的 bug
- 底层异常导致应用宕机

【解决方法】

1. 登录云游运维平台页面，找到 kubecontroller 应用的容器，检查容器状态是否健康。
   - 若容器状态异常，执行步骤 2。
   - 若容器可以登录，说明不是宕机产生的问题，参考 Fed 问题排查 进行进一步排查。

   > 说明
   >
   > 当前容器内没有 bash，只能使用 sh 登录。

2. 在云游平台发起容器重启操作。

【验证方法】检查之前异常的功能是否恢复正常。

4.3. 断电恢复

CAFÉ 涉及的组件通过容器方式部署，断电时原则上会自动连接。如果服务本身没有恢复，可以参考 应用宕机 章节的现场对应用进行重启操作。

4.4. 网络连接失败

健康检查失败可能由网络原因导致、调用失败，参考 故障处理 章节。

4.5. 数据丢失

Hub 存储用户数据，当 hub 中一台实例出现异常数据丢失时，参考高危操作中 hub 数据迁移 的操作进行对应的处理。

5.日常运维

5.1. 概述

CAFE 的日常运维主要包含监控和巡检两部分内容。

- **巡检**：人工拨测 URL 或端口等来判断 CAFE 的业务运转是否正常。
- **监控**：通过监控平台从客户机搜集日志并按一定方式汇聚出关键指标，以此来衡量系统运行状况。

5.2. 监控

登录实时监控平台页面，查找对应的应用，基于监控项查看应用监控信息。

监控项说明

以下监控配置在交付 CAFE 时已完成，无需用户手动配置。

应用	分类	日志路径	告警阈值	采集间隔	告警间隔
apaks	应用日志监控-错误日志	/home/admin/logs/apaks/common-error.log	最近 5 分钟内报错数超过 100 条	1 分钟	3 分钟
apaksexecutor	应用日志监控-错误日志	/home/admin/logs/apaksexecutor/common-error.log	最近 5 分钟内报错数超过 10 条	1 分钟	3 分钟
opswareorch	应用日志监控-错误日志	/home/admin/logs/opswareorch/common-error.log	最近 5 分钟内报错数超过 10 条	1 分钟	3 分钟
hub	应用日志监控-错误日志	/tmp/ui.log	最近 5 分钟内报错数超过 10 条	1 分钟	3 分钟
metaservice	应用日志监控-错误日志	/home/admin/logs/acmetaservice/common-error.log	最近 5 分钟内报错数超过 10 条	1 分钟	3 分钟

应用	分类	日志路径	告警阈值	采集间隔	告警间隔
lhc	应用日志监控-错误日志	/home/admin/logs/apildcconsole/common-error.log	最近 5 分钟内报错数超过 10 条	1 分钟	3 分钟
opswarecore	应用日志监控-错误日志	home/admin/logs/opswarecore/common-error.log	最近 5 分钟内报错数超过 10 条	1 分钟	3 分钟
ldccore	应用日志监控-错误日志	/home/admin/log/zonemng/common-error.log	最近 5 分钟内报错数超过 10 条	1 分钟	3 分钟

日志说明

日志是 CAFE 运维的重要方式之一，可以对日志进行监控以便及时发现运行问题和故障。

通过实时监控平台可以监控 CAFE 的日志，包括：

- apaks：/home/admin/logs/apaks
- apaksexecutor：/home/admin/logs/apaksexecutor
- hub：/tmp/
- metaservice：/home/admin/logs/acmetaservice
- lhc：/home/admin/logs/apldcconsole
- fedapiserver：/logs
- fedcontroller：/logs
- kubecontroller：/logs
- opswareorch：/home/admin/logs/opswareorch
- ldccore：/home/admin/log/zonemng

详细日志说明参考 日志参考。

设置监控项

登录实时监控系统，配置以下监控指标。

应用	分类	指标（关键字）	告警阈值
apaks apaksexecutor hub metaservice fedapiserver fedcontroller kubecontroller appcore acprovisioncore iaasconnector opswareorch ldccore	基础监控-cpu 使用率监控	cpu_usage	>90%
	基础监控-内存使用率监控	mem_usage	>90%
	基础监控-磁盘使用率监控	disk_usage	>90%
	基础监控-load1	load1	>4
apaks	基础监控-端口监控	2022/80/8080/8341/12200	不通
apaksexecutor	基础监控-端口监控	2022/8080/12200	不通
hub	基础监控-端口监控	80/5000	不通
metaservice	基础监控-端口监控	2022/8341/12200	不通
lhc	基础监控-端口监控	80/8080/2022/8341/12200	不通
fedapiserver	基础监控-端口监控	6443	不通
fedcontroller	基础监控-端口监控	8080	不通
kubecontroller	基础监控-端口监控	10252	不通
appcore	基础监控-端口监控	2022/80/12200	不通
acprovisioncore	基础监控-端口监控	2022/8341/12200	不通

应用	分类	指标（关键字）	告警阈值
iaasconnector	基础监控-端口监控	2022/8341/12200	不通
opswareorch	基础监控-端口监控	80/2022/8341/12200	不通
ldccore	基础监控-端口监控	80/2022/12200	不通

CAFE 通过核心态监控的监控配置来完成对各系统组件的日常巡检，监控各组件是否正常工作。如果异常，则触发告警。

具体告警异常的影响和处理参考 告警处理 章节。

5.3. 巡检

CAFE 的例行巡检包含应用端口监听和服务组件巡检两部分。CAFE 不区分组件巡检和业务巡检，组件巡检即业务巡检。

监听端口

应用	分类	指标（关键字）	说明
apaks	基础监控-端口监控	2022/80/8080/8341/12200	检查 apaks 的 tcp 服务是否正常。
apaksexecutor	基础监控-端口监控	2022/8080/12200	检查 apaksexecutor 的 tcp 服务是否正常。
hub	基础监控-端口监控	80/5000	检查 hub 的 http 服务是否正常。
metaservice	基础监控-端口监控	2022/8341/12200	检查 metaservice 的 tcp 服务是否正常。
lhc	基础监控-端口监控	80/8080/2022/8341/12200	• 80 端口检查 lhc 的 http 服务是否正常。 • 其他端口检查 lhc 的 tcp 服务是否正常。
fedapiserver	基础监控-端口监控	6443	检查 fedapiserver 的 tcp 服务是否正常。

应用	分类	指标（关键字）	说明
fedcontroller	基础监控-端口监控	8080	检查 fedcontroller 的 tcp 服务是否正常。
kubecontroller	基础监控-端口监控	10252	检查 kubecontroller 的 tcp 服务是否正常。
appcore	基础监控-端口监控	2022/80/12200	检查 appcore 的 tcp 服务是否正常。
acprovisioncore	基础监控-端口监控	2022/8341/12200	检查 acprovisioncore 的 tcp 服务是否正常。
iaasconnector	基础监控-端口监控	2022/8341/12200	检查 iaasconnector 的 tcp 服务是否正常。
opswareorch	基础监控-端口监控	80/2022/8341/12200	检查 opswareorch 的 tcp 服务是否正常。
ldccore	基础监控-端口监控	80/2022/12200	检查 ldccore 的 tcp 服务是否正常。

组件巡检

CAFE 通过核心态监控的监控配置来完成对各系统组件的日常巡检，监控各组件是否正常工作。如果异常，则触发告警。

具体告警异常的影响和处理参考 告警处理 章节。

6.故障处理

6.1. Fed 问题排查

6.1.1. 排查前须知

准备工作

- 如果您尚不了解 Fed，建议先阅读下文的 背景知识，再查看具体排查方法。
- 拥有一个可以访问到联邦（Federation）集群的客户端配置（KubeConfig）。在专有云环境中，该 KubeConfig 配置位于中枢集群 LKS 产品所在命名空间的一个 ConfigMap 中，可以通过对 **中枢集群** 执行如下 kubectl 命令提取：

  ```
  kubectl --kubeconfig=<你的中枢集群客户端配置路径> get cm admin-kubeconfig -n <LKS产品所在的命名空间> -o jsonpath='{.data.admin\.kubeconfig}'
  ```

  获取到联邦集群的 KubeConfig 后，可以通过执行一个简单的 kubectl 查询命令来确认到联邦集群的访问链路是否通畅：

  ```
  kubectl --kubeconfig=<你的联邦集群客户端配置路径> get ns
  ```

  如果发现该命令执行报错或超时，首先确认执行 kubectl 所在机器到 KubeConfig 中的 server 地址是通的，即域名能够解析并且网络通。

  在排除网络问题后如果仍然有问题，则说明联邦集群的 ApiServer（fedapiserver）可能存在问题，请参考下文常见问题指南中的 联邦层 ApiServer 问题排查部分 进一步排查。

背景知识

什么是集群联邦（Federation）

集群联邦是 LHC 产品的一个底层组件，其主要作用为将用户的多个 K8s 集群抽象为一个联邦 K8s 集群，在这个联邦集群上，上层产品可以通过操作 Federated 类型的资源来同时向多个用户集群分发 K8s 资源，同时通过 Federated Status 类型的资源收集多个用户集群中 K8s 资源的实时状态。

以下是 Federation 所常用的英文名词及其对应的英文解释：

- 集群联邦层（Federation）：指整个部署的集群联邦全部组件及其功能。
- 联邦集群（Federation Cluster）：主要指集群联邦层的 Apiserver 所承载的 K8s 集群。
- Local 集群（Local Cluster）：指被集群联邦层所管辖的子集群，即客户实际在使用的 K8s 集群。
- Federated 资源类型（Federated Types）：在集群联邦层可以控制的资源类型。
- Federated Status 资源类型（Federated Status Types）：在集群联邦层可以控制的资源状态类型。
- 联邦层元数据集群（KubeFedCluster）：在集群联邦层中代表 Local 集群元数据信息的资源类型。

集群联邦包含哪些组件

应包含一个 MySQL 或 OB 或 etcd 数据库组件，多个互为副本的 Federation Apiserver，多个互为副本的 Federation Controller。

新版本（≥ LKS 1.22.0）中加入了多个互为副本的 Kube Controller，用于联邦集群命名空间的生命周期管理和自动垃圾回收。

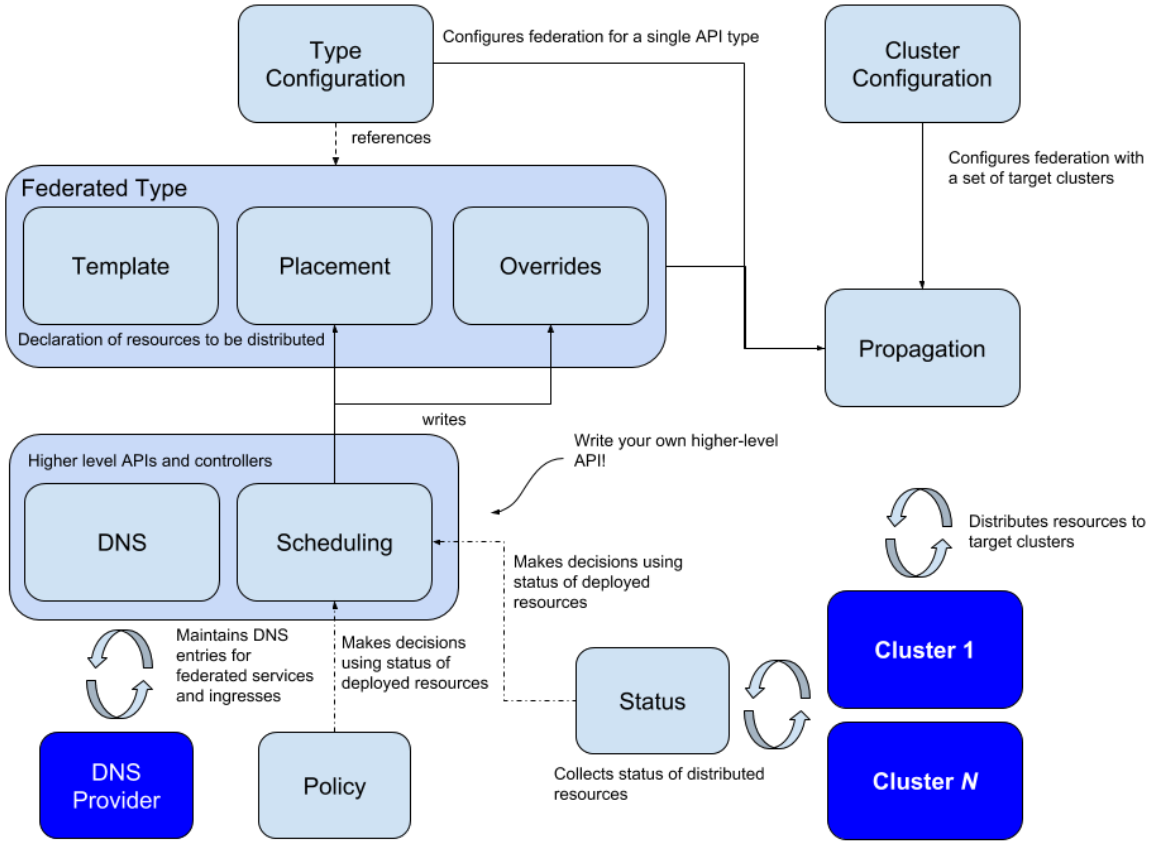
集群联邦的功能包含哪些

主要包含如下功能：

- 批量向 Local 集群下发 K8s 资源：通过创建 Federated 资源类型对象，指定向某些 Local 集群下发资源。
- k8s 资源状态回流：创建 Federated 资源类型对象之后，可以通过 Federated Status 资源类型获取该资源在各个 Local 集群中的实时状态。
- 集群健康状态探活：当 Local 集群状态出现不正常时，能及时发现并采取对应失败措施。

集群联邦大致如何下发资源

通过向联邦集群创建 Federated 资源类型对象（其类型与 Local 集群的 K8s 资源的类型一一对应，如 FederatedCafeDeployment 类型用于向 Local 集群下发 CafeDeployment 资源），在其 spec 字段的 template 和 overrides 两个字段中设置要向 Local 集群下发的具体内容。



其中 template 子字段是期望下发的对象通用部分，而 overrides 子字段是期望下发对象的内容覆盖部分，即每个 Local 集群的定制化部分。

同时在 Federated 资源类型对象的 placements 字段中我们来指定具体向哪些 Local 集群下发。

> 注意
>
> - 在 LHC 产品中，placements里指定的是 Cell 的名称，集群联邦层会向该 Cell 所在的 Local 集群下发资源。
> - 如需了解联邦层如何通过 Cell 查找到对应的 Local 集群，请参见 进阶问题排查。

被下发的资源名称强制添加"-cell-<Cell的名称>"作为资源后缀。

新版本（≥ LKS 1.22.0）中支持在其 spec 字段的 topologyType 字段中设置该资源采用的分发拓扑设置为 **cell** 则遵循上面描述的行为，将该资源以 Cell 的维度来分发。

设置为 **cluster** 则代表使用实际的 Local 集群维度来分发，此时 placements 里指定的就是代表该 Local 集群的联邦层元数据集群的名称，被下发的资源也不会添加资源后缀，目前命名空间（FederatedNamespace）就是以这种模式来进行分发的。

集群联邦大致如何回流资源状态

通过向所管辖的 Local 集群发起监视资源的长链接请求来实时获取资源的变化，并将其实时状态聚合记录在 Federated Status 资源类型对象中（其类型与 Local 集群的 k8s 资源的类型一一对应，如 FederatedCafeDeploymentStatus 类型用于记录 Local 集群的 CafeDeployment 资源的实时状态）。

当前联邦资源类型及对应产品层功能

当前产品在使用的联邦资源列表如下：

联邦资源类型	对应 Local K8s 资源类型	对应产品层功能
FederatedNamespace	Namespace	单元化应用服务 > 集群管理 > 命名空间
FederatedConfigMap	ConfigMap	单元化应用服务 > 配置管理 > 配置项
FederatedSecret	Secret	单元化应用服务 > 配置管理 > 保密字典
FederatedCafeDeployment	CafeDeployment	单元化应用服务的工作负载，发布时创建或更新
FederatedService	Service	单元化应用服务的负载均衡，发布时创建或更新
FederatedIngress	Ingress	单元化应用服务的统一接入，发布时创建或更新
FederatedL7Cluster	L7Cluster	单元化应用服务 > 网络 > 统一接入集群
FederatedUnifiedAccessInstance	UnifiedAccessInstance	单元化应用服务 > 网络 > 统一接入实例

联邦资源类型	对应 Local K8s 资源类型	对应产品层功能
FederatedLoadBalancer	LoadBalancer	单元化应用服务 > 网络 > 联邦负载均衡

6.1.2. 常见问题排查

6.1.2.1. 如何查看资源下发是否成功及错误信息

前置事项

- **确保该 Federated 资源不处于删除中。**

首先，确保该 Federated 资源不处于“删除中”，对于删除中的 Federated 资源是不会进行下发的。可通过执行下面的 kubectl 命令进行确认：

```
kubectl -n <Federated资源类型对象所在命名空间> get <Federated资源类型> <Federated资源类型名称> -o jsonpath='{.metadata.deletionTimestamp}'
```

例如：

```
kubectl -n abc-wsg-test get federatedcafedeployment test -o jsonpath='{.metadata.deletionTimestamp}'
```

如果该命令的输出不为空，说明该 Federated 资源已经处于“删除中”的状态，不会再进行下发操作。如果需要进一步排查该资源卡在删除中的原因，可参考 联邦资源卡在删除中 。

- **确保 fedcontroller 已处理该 Federated 资源。**

由于 k8s controller 是一个异步模型，从 Federated 资源的创建或更新到 fedcontroller 实际对其进行处理会有一定延迟（通常非常小）。可通过执行下面的 kubectl 命令进行确认：

```
kubectl -n <Federated资源类型对象所在命名空间> get <Federated资源类型> <Federated资源类型名称> -o jsonpath='{.metadata.generation}{"\n"}{.status.observedGeneration}'
```

例如：

```
kubectl -n abc-wsg-test get federatedcafedeployment test -o jsonpath='{.metadata.generation}{"\n"}{.status.observedGeneration}'
```

如果该命令输出了两个数字，且这两个数字相等，说明当前版本的 Federated 资源已经被 fedcontroller 处理过了，可以继续下面的排查。如果两个数字不相等或者只有一个数字，可尝试稍后重新检查，如果始终没有相等，说明 fedcontroller 可能出现了问题，可尝试对 fedcontroller 进行重启，如果重启后仍然不能解决请寻求进一步支持。

具体步骤

确保该 Federated 资源不处于“删除中”且已被 fedcontroller 处理过后，可通过以下几个步骤逐步排查其下发问题：

主要通过以下 3 个步骤查看：

1. 查看 Federated 资源类型对象的 status 字段，如果资源下发失败，其 status 中会记录失败的简要错误信息，其 kubectl 命令为：

   ```
   kubectl -n <Federated资源类型对象所在命名空间> get <Federated资源类型> <Federated资源类型名称>
   ```

   例如：

   ```
   kubectl -n abc-wsg-test get federatedcafedeployment test -o yaml
   ```

2. 查看 Federated 资源类型对象的事件，可以看到更详细的下发和错误相关信息，其 kubectl 命令为：

   > 注意
   >
   > 事件存留时间为 1 小时，如果因过期而没有看到错误相关事件，请看下一步继续排查。

   ```
   kubectl -n <Federated资源类型对象所在命名空间> describe <Federated资源类型> <Federated资源类型名称>
   ```

   例如：

   ```
   kubectl -n abc-wsg-test describe federatedcafedeployment test
   ```

3. 如果在上一步中仍然没有看到具体的错误事件，则需要通过 fedcontroller 的日志进行排查，可以通过中枢集群进入 fedcontroller 的容器（注意，fedcontroller 容器内没有 bash，只有 sh），其日志位于 `/logs` 目录下，可以通过在错误发生日期（注意，日志时间为容器内时间，可能有时差）的 INFO 日志中搜索相应关键字（如 Federated 资源类型对象的名称、应用服务的名称等）来查询详细的处理和错误信息。

   如果中枢集群有多个 fedcontroller 容器，则都尝试查询下，查询到一个即可，因为同一时间只有一个 fedcontroller 会进行实际的工作。

   > 注意
   >
   > 早期版本的 fedcontroller 内没有日志文件。因此如果找不到日志文件，只能通过在中枢集群使用 `kubectl logs` 命令来查看 fedcontroller 容器的标准输出日志。

6.1.2.2. 显示 ClusterNotReady 错误

【现象】

显示 ClusterNotReady 错误。

【解决方法】

1. 首先，请检查 Local 集群是否存活且正常工作。

   如果集群状态没有异常，通过以下命令，找到联邦层代表该部署单元所在 Local 集群的元数据集群：

   ```
   kubectl -n kube-federation-system get kubefedcluster | grep <工作空间组的名称（小写）> | grep <Cell的名称（小写）>
   ```

2. 再进一步通过以下命令查看其具体错误：

```
kubectl -n kube-federation-system get kubefedcluster <元数据集群的名称> -o yaml
```

其输出的 status 字段中会记录问题详情，关于状态异常故障排查信息，请参考 联邦层元数据集群（KubeFedCluster）状态异常。

6.1.2.3. 显示 CreationFailed/UpdateFailed 错误

【现象】

显示 CreationFailed/UpdateFailed 错误。

【解决方法】

1. 首先确认集群是否存活且正常工作。
2. 参考 如何查看资源下发是否成功及错误信息 中的第 2、3 步，在事件/日志中会具体显示创建失败的具体原因。

   如果仍然无法解决，请联系技术支持。

6.1.2.4. 显示 ApplyOverridesFailed 错误

【现象】

显示 ApplyOverridesFailed 错误。

【可能的原因】

Federated 资源对象内容不合法，大部分情况下原因和产品服务缺陷有关。

【解决方法】

参考 如何查看资源下发是否成功及错误信息 中的第 2 步，在事件中会具体显示具体哪些字段的不合法。

6.1.2.5. 显示 NamespaceNotFederated 错误

【现象】

显示 NamespaceNotFederated 错误。

【可能的原因】

可能是产品服务存在代码缺陷。

【解决方法】

请联系技术支持。

6.1.2.6. 未找到任何对应 Cell 的状态或错误信息

【现象】

未找到任何对应 Cell（部署单元）的状态或错误信息。

【解决方法】

请检查代表该部署单元所在 Local 集群的元数据集群是否存在。

> **说明**
>
> 对于不存在元数据集群的 placement，联邦层会自动忽略该 placement，因此也就不会有任何错误信息。

```
kubectl -n kube-federation-system get kubefedcluster | grep <工作空间组的名称（小写）> | grep <Cell的名称（小写）>
```

如果发现不存在，则说明该部署单元尚未绑定任何 Local 集群，因此无法下发资源。此时需要通过集群管理页的 **解绑/绑定** 功能将该部署单元和某一 Local 集群进行绑定。

6.1.2.7. 联邦资源类型的名称过长

【现象】

联邦资源类型的名称过长，通过 kubectl 命令需要输入过多的字符。

【解决方法】

新版本（≥ LKS 1.22.0）中所有的联邦资源类型都加入了类似 K8s 资源的缩写：

- 对于 Federated 资源类型对象，其规则为 `f+k8s资源的缩写`，如 FederatedCafeDeployment 可以缩写为 fcd，FederatedService 可以缩写为 fsvc 等。
- 对于 Federated Status 资源类型对象，其规则为 `f+k8s资源的缩写+s`，如 FederatedCafeDeploymentStatus 可以缩写为 fcds，FederatedServiceStatus 可以缩写为 fsvcs 等。

6.1.2.8. 联邦资源卡在删除中

【现象】

联邦资源卡在删除中。

【解决方案】

1. 首先，确认该联邦资源是否处于删除中的状态，可以通过查看该 Federated 资源类型对象的 metadata 字下是否有 deletionTimestamp 这个字段来判断。

   如果该字段不为空，说明该 Federated 资源类型对象处于删除中的状态。对于一个要删除的 Federated 资源类型对象，联邦层会在该工作空间组下的**所有 Local 集群**中检查并删除该资源（无论该 Federated 资源类型对象之前被分发到了哪些集群）。此外，对于删除中的联邦资源，联邦层不会再为其回流 Federated Status 资源类型对象。

2. 当发现联邦资源卡在了删除中的状态，请先确保该联邦资源所处工作空间组内的所有 Local 集群都存活并且正常工作，然后检查每个 Local 集群中由该联邦资源下发的实际 K8s 资源是否也卡在了删除中的状态。如果是，则需要进一步排查该 Local 资源删除卡住的原因。

3. 如果经检查所有 Local 集群中由该联邦资源下发的实际 k8s 资源都已被删除，但联邦资源本身还是卡在删除中，或发现存在部分 Local 集群中的资源并未处于删除中的状态，则需要进一步排查联邦层的问题，可以参考 如何查看资源下发是否成功及错误信息 中的第 2、3步进行排查。

6.1.2.9. 联邦层 ApiServer（fedapiserver）出现问题导致联邦层无法正常处理外部请求

【现象】

联邦层 ApiServer（fedapiserver）出现问题，导致联邦层无法正常处理外部请求。

联邦层 ApiServer 出现问题的通常表现为：

- 产品层（如 LHC）对联邦层的相关 API 调用（如查询 FederatedNamespace、FederatedConfigMap 等）报错或超时
- fedcontroller 日志中出现形如 `failed to tryAcquireOrRenew context deadline exceeded` 字样的报错

【解决方法】

1. 首先，通过中枢集群检查所有 fedapiserver 容器是否都处于 Running 且 Ready 的状态。
2. 进一步通过 fedapiserver 的日志排查错误原因。

   fedapiserver 的日志位于其容器内 `/logs` 目录下，可以通过查看错误发生时间的相应日志找到详细报错。

> **注意**
>
> - fedapiserver 容器内没有 bash，只有 sh。
> - 日志时间为容器内时间，可能有时差。

一般来说，fedapiserver 的异常绝大多数是由于其访问数据库异常导致，建议先从这个方向进行排查。其日志目录中名字带 `kine` 的日志（如有）会记录更多其访问数据库相关的细节。

> **说明**
>
> 早期版本的 fedapiserver 内没有日志文件。因此，如果找不到日志文件，只能通过在中枢集群使用 `kubectl logs` 命令来查看 fedapiserver 容器的标准输出日志。

6.1.2.10. 创建或导入集群时，联邦层集群元数据同步失败

【现象】

如果导入集群运维单页面上显示诸如 `Some KubeFedClusters of cluster xxx are not ready for now` 之类的错误，说明该新导入集群相关的元数据集群未通过联邦层集群健康检查。

【解决方法】

1. 检查 Local 集群是否存活且正常工作。

   如果集群状态没有异常，通过以下命令找到联邦层这个集群相关的元数据集群：

```
kubectl -n kube-federation-system get kubefedcluster -l cafe.sofastack.io/basicapiserver-cluster-name=<新导入集群的名称，即上面报错中 xxx 的内容>
```

2. 找到结果中 READY 列不为 True 的元数据集群，再进一步通过以下命令查看其具体问题：

```
kubectl -n kube-federation-system get kubefedcluster <元数据集群的名称> -o yaml
```

其输出的 status 字段中会记录问题详情，关于状态异常故障排查信息，请参考 联邦层元数据集群（KubeFedCluster）状态异常。

6.1.2.11. 联邦层元数据集群（KubeFedCluster）状态异常

【现象】

联邦层元数据集群（KubeFedCluster）状态异常，具体判断方式如下：

仅当 KubeFedCluster 的 status 中显示类似如下内容（即 condition 的 type 值为 `Ready`，status 值为 `True`）时，才说明其状态正常，否则均为异常。

```
status:
  conditions:
  - lastProbeTime: "2022-03-09T06:51:03Z"
    lastTransitionTime: "2022-03-08T07:56:55Z"
    message: /healthz responded with ok
    reason: ClusterReady
    status: "True"
    type: Ready
```

对于异常的 KubeFedCluster，其 status 会提供一些异常 condition 信息，具体如下：

- ClusterNotReachable
- ClusterNotReady
- APIGroupMissing
- ClusterConfigMalformed

【解决方法】

您可以根据具体异常信息采取操作，具体可参考下表。

状态异常信息	异常分析	解决方法

状态异常信息	异常分析	解决方法
ClusterNotReachable	当该 condition 为 `True` 时，说明 fedcontroller 无法连接到该集群的 ApiServer，fedcontroller 使用的连接地址记录在该 KubeFedCluster 的 spec 的 apiEndpoint 字段中。	确保从 fedcontroller 容器（位于中枢集群）内访问这一 apiEndpoint 地址的网络链路连通，且该 apiEndpoint 连接到的用户集群的 ApiServer 正常。
ClusterNotReady	当该 condition 为 `True` 时，说明该 KubeFedCluster 对应用户集群的 ApiServer 存在问题（健康检查失败）。	请联系集群提供方排查问题。
APIGroupMissing	当该 condition 为 `True` 时，说明该 KubeFedCluster 对应用户集群中未安装一些必须组件。	1. 登录 LHC 控制台，在左侧导航栏单击 **集群管理 > 集群**。 2. 单击对应用户集群卡片，进入 **集群详情** 页。 3. 单击 **组件管理** 页签，进入组件列表页面。 4. 检查相应组件是否正确安装。 ◦ 如果缺少 `apps.cafe.cloud.alipay.com`，请检查 cafeextcontroller 组件 ◦ 如果缺少 `nfcc.cafe.sofastack.io`，请检查 cloud-controller-manager 组件。
ClusterConfigMalformed	当该 condition 为 `True` 时，说明该 KubeFedCluster 的配置存在问题，可能是产品缺陷导致。	请直接联系相关技术支持人员协助解决。

6.1.3. 进阶问题排查

6.1.3.1. 集群联邦层如何查找 Cell 和元数据集群（KubeFedCluster）之间的对应关系

在集群联邦层，所有代表 Cell 的元数据集群（KubeFedCluster）的名字都是严格按一定规则生成的，其大致规则为 `<租户名称>-ws-<工作空间组名称>(在低于 LKS 1.18.0 的版本中为工作空间名称)-cell-<Cell 名称>` 。

同时，在所有的 Federated 资源类型对象上，会有 `cafe.sofastack.io/tenant` 和 `cafe.sofastack.io/workspace-group` 这两个分别代表该资源所处租户和工作空间组的 label。由此，联邦层通过这两个 label，就可以准确定位到该资源 placement 中指定的 Cell 所对应的具体元数据集群了。

在低于 LKS 1.18.0 的版本中，联邦层通过 meta-info 定位元数据集群：所有的 Federated 类型资源上都有强制的 meta-info 元数据注解 annotation，该注解涵盖了所有该 Fedeated 对象上所包含的 Cell 和 Cluster。其结构为以下三种元数据信息组合作为三元组的列表：

- 元数据 Cell 的名称
- 元数据 Cluster 的名称
- 元数据 Workspace 的名称

在联邦 Apiserver 服务中会强制检查 meta-info 的元数据注解是否和 Federated 资源类型对象的内容相匹配，例如是否所有的 placement 都在元数据注解中有对应的条目，以及 meta-info 是否存在重复畸形的内容等。

6.1.3.2. 什么是 Federated 资源类型对象的 meta-info 注解

新版本（>= LKS 1.22.0）中该注解已被完全废弃，联邦层不会再校验和使用该注解的信息。具体可参考 集群联邦层如何查找 Cell 和元数据集群（KubeFedCluster）之间的对应关系。

6.1.3.3. 什么是 Federated 对象类型上的 placement-mask 注解

该注解主要用于对 Federated 资源类型进行灰度下发，其内容为 placement 的名称列表通过逗号链接。

首先 placement-mask 的工作机制类似于子网掩码，在该 Federated 资源的 placement 基础上增加了灰度匹配的机制。即 placement-mask 出现在 Federated 资源上时，只有当该 placement 出现在 placement 列表中并且该 placement 出现在 placement-mask 列表中时，才会触发资源下发的流程。

反过来未被 placement-mask 匹配到的 placement 会处于暂时不生效的状态中，即联邦层不会去变更该 placement 下的资源。

6.1.3.4. 为什么名称带 -dirty 后缀

部分元数据集群（KubeFedCluster）的名称，以及从 Local 集群看所下发的部分资源名称带 `-dirty` 后缀，说明该资源所属的租户名称、工作空间组名称、Cell 名称中的一项或多项存在大写字符。

集群联邦层在设计上要求所有的元数据按小写英文字符构成名称，但由于复杂的历史兼容问题，同时也提供了对大写的元数据名称的兼容性。

该兼容性是通过将原本大写的英文名称后缀添加 `-dirty-编码表` 构成。

总而言之，名称中出现 `-dirty` 后缀是保证不会影响集群联邦的工作流程的，只是在提醒元数据存在不规范的情况。

6.1.3.5. 如何让联邦层下发的 Local 集群的资源脱离联邦层的管控

主要有以下两种方法，请根据实际情况选择策略：

- **在不需要删除 Federated 类型对象的情况下：**

  在原本被集群联邦层管控的 Local 资源上会存在 Label：kubefed.io/managed: "true"，可以通过将该 Label 的值改为 false 来达到使该 Local 资源脱离联邦层管控的效果。

- **在需要删除 Federated 类型对象的情况下：**

  在 Federated 类型资源上打上 Label：kubefed.io/orphan:true。之后再将该 Federated 类型对象删除，即可保留其此前下发的所有 Local 资源。

6.1.4. 已知问题清单

6.1.4.1. 使用 MySQL 作为 fedapiserver 数据库时，fedapiserver 无法启动

【现象】

使用 MySQL 作为 fedapiserver 数据库时，fedapiserver 无法启动，日志提示：`Error: building kine: Error 1071: Specified key was too long; max key length is 767 bytes` 。

【可能存在此问题的版本】

≤ LKS 1.15.x

【产生原因】

MySQL 5.x 版本中 InnoDB 的索引 key 默认有长度限制，可能因默认字符集字符宽度过大而报错。

【解决方法】

1. 修改 fedapiserver 的 StatefulSet，将其副本数调至 0，防止因自动重启 Pod 影响后续操作。
2. 进入 fedapiserver 的数据库，执行如下 SQL 语句：

```
DROP TABLE kine;
CREATE TABLE kine
  (
    id BIGINT AUTO_INCREMENT,
    name VARCHAR(630) CHARACTER SET ascii,
    created INTEGER,
    deleted INTEGER,
    create_revision BIGINT,
    prev_revision BIGINT,
    lease INTEGER,
    value MEDIUMBLOB,
    old_value MEDIUMBLOB,
    PRIMARY KEY (id)
  );
```

3. 将 fedapiserver 的 StatefulSet 副本数设回原来的值。
4. 重试云游发布单。

6.1.4.2. 使用 OB 作为 fedapiserver 数据库时，fedapiserver 无法启动

【现象】

使用 OB 作为 fedapiserver 数据库时，fedapiserver 无法启动，日志提示：`Error 1235: Not supported feature or function` 。

【可能存在此问题的版本】

全部版本

【产生原因】

fedapiserver 对 OB 版本有严格要求，要求 OB 版本 ≥ 2.2.76。

【解决方法】

可通过如下方式之一处理：

- 将 OB 升级到 2.2.76 及以上版本。
- 使用非 OB 数据库（如原生 MySQL）作为 fedapiserver 的数据库。

6.1.4.3. Fed 不识别新导入的集群

【现象】

LHC 的工作空间进行集群替换（解绑旧集群并导入新集群）后，Fed 不识别新导入的集群，而是继续试图在旧集群进行资源管理，导致发布部署或资源删除异常。

【可能存在此问题的版本】

< LKS 1.22.0

【产生原因】

Fed 的元数据与产品层未完全打通，工作空间层面集群的变化会产生脏数据。

【解决方法】

可通过如下几种方法来解决此问题：

方法 1

最简单的办法是创建一个新的与原来不同名的工作空间来绑定新集群，不同的工作空间的数据不会互相影响。

方法 2

如果希望保留当前工作空间，可以将该工作空间里原来的部署单元（Cell）全部删除掉，然后创建与原先名称不同的 Cell，往新的 Cell 做发布不会受旧 Cell 的脏数据所影响。

方法 3

如果既希望保留当前工作空间，又希望保持现有部署单元规划，则需要手动清理 Fed 层的脏数据。具体步骤如下：

1. 执行以下步骤前，尽量先不要解绑或释放原来的旧集群，先在 LHC 页面上删除所有命名空间下的应用服务、配置项与统一接入集群和实例等。
2. 在中枢集群执行

   ```
   kubectl get cm admin-kubeconfig -n {lks产品的命名空间} -o jsonpath='{.data.admin\.kubeconfig}'
   ```

   得到 Fed 层的 kubeconfig。

   > **说明**
   >
   > 执行该命令前，需要将 `{lks产品的命名空间}` 替换成环境实际值。

3. 使用上一步得到的 kubeconfig 执行如下命令：

   > **注意**
   >
   > 以下命令会删除该工作空间内所有的 K8s 资源。

```
# 1.删除当前工作空间内所有 fed 资源
# 如果参数包含命名空间名，说明对该工作空间的每一个命名空间都要执行一遍此命令，将大括号的内容替换成实际值（不要保留大括号）
# 如果原来的老集群已经不可用，会导致删除卡住，这时候需要开一个新窗口，手动 get 出所有资源，然后依次编辑每个资源，删除 Finalizers 里的 kubefed.io/sync-controller 这一行
# 如果原来的老集群元数据已经被清理，可能会导致无法编辑资源，这时候为该资源添加一个 key 为 test.only/skip-validation 的 annotation 即可（value 可以为空）
kubectl delete federatedcafedeployment -n {命名空间名} --all
kubectl delete federatedconfigmap -n {命名空间名} --all
kubectl delete federatedsecret -n {命名空间名} --all
kubectl delete federatedservice -n {命名空间名} --all
kubectl delete federatedingress -n {命名空间名} --all
kubectl delete federatedl7cluster -l cafe.sofastack.io/workspace-group={工作空间名（大小写敏感）}
kubectl delete federatedunifiedaccessinstance -l cafe.sofastack.io/workspace-group={工作空间名（大小写敏感）}

# 2.删除当前工作空间内所有集群数据
# 将大括号的内容替换成实际值（不要保留大括号），执行后应该会显示一些数据被删除，如果没有输出说明可能填错了值
kubectl delete kubefedcluster -n kube-federation-system -l cafe.sofastack.io/workspace-group={工作空间名（大小写敏感）}
```

4. 如果之前有创建过自定义命名空间，导入新集群后需要删除并重新创建下自定义命名空间。

6.1.4.4. 联邦层部分资源状态不一致

【现象】

FederatedXXXStatus 资源内收集到的 local 资源的版本（generation/resource version）或状态（topology status）与实际 local 资源不符。

【可能存在此问题的版本】

< LKS 1.24.0

【产生原因】

存在代码 bug，导致 fedapiserver 可能会偶发性地丢 etcd 事件，进而导致联邦层部分资源状态不一致。

【解决方法】

先重启所有 fedapiserver，再重启所有 fedcontroller。

6.2. 告警处理

6.2.1. aks 异常告警

80 端口异常告警

【现象】80 端口异常告警。

【可能的原因】nginx 服务异常。

【对系统的影响】容器服务不可用。

【解决方法】

1. 登录云游运维平台页面，找到 aks 应用的容器，检查容器状态是否健康。
2. 若容器状态异常，可直接通过云游运维平台发起容器重启操作（危险等级 G2）。
3. 若容器可以登录，通过 `ps -ef|grep nginx` 查看 nginx 进程是否存在，保存容器内 `/home/admin/logs/nginx` 内的日志文件。
4. 将进程是否存在的情况、日志内容给到技术支持分析，然后参考步骤 2 重启容器服务。

【验证方法】访问容器服务控制台，看是否服务正常。

8080/8341/12200 端口异常告警

【现象】8080/8341/12200 端口异常告警。

【可能的原因】

- 进程夯住
- nginx 服务异常

【对系统的影响】容器服务不可用。

【解决方法】

1. 登录云游运维平台页面，找到 aks 应用的容器，检查容器状态是否健康。
2. 若容器状态异常，可直接通过云游运维平台发起容器重启操作（危险等级 G2）。
3. 若容器可以登录，执行如下排查操作：

排查项	具体步骤
检查容器 Java 进程是否存在	运行 `ps -ef|grep java|grep -v xflush`，检查容器 Java 进程是否存在。 ○ 若 Java 进程不存在，执行以下操作： a. 查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件。若有此文件，则保存此文件。 b. 参考步骤 2 重启服务。 ○ 若 Java 进程存在，执行以下操作： a. 执行 `jmap -heap <pid>` 命令，查看各内存区域的设置。 b. 执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查。 c. 参考步骤 2 重启服务。

排查项	具体步骤
查看 nginx 进程是否存在	i. 通过 `ps -ef|grep nginx` 查看 nginx 进程是否存在，保存容器内 `/home/admin/logs/nginx` 内的日志文件。 ii. 将进程是否存在的情况、日志内容给到技术支持分析，然后参考步骤 2 重启容器服务。

【验证方法】访问容器服务控制台，看是否服务正常。

CPU 使用率异常告警

【现象】CPU 使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 执行 `top` 命令，查看组件的 CPU 使用率。
   - 如果 CPU 使用率不高，确定是否监控误报。
   - 如果 CPU 使用率高，需要根据调用量做进一步排查。
3. 执行 `netstat -tnlp | grep -E "80|8080" | wc -l` 命令，检查组件的调用情况。
   - 如果调用量偏高，并且符合业务状态，需要对实例进行扩容。
   - 如果调用量不高，需要执行下面的步骤排查 CPU 使用率高的原因。
     a. 执行命令 `top -Hp <组件进程号>`，找到导致 CPU 或内存使用率高的进程，将其转换成十六进制。
     b. 进入 `jstack ##id` 路径中，打开 ps.txt 文件，根据线程十六进制 ID 定位具体的进程类。
     c. 如果有 Full GC，查看 gc.log 或者执行 `jstat -gcutil [pid]` 命令查看对应的 GC 日志，并将 top/jstack/full GC 等做好记录，交由技术支持人员处理。

内存使用率异常告警

【现象】内存使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 进入目标容器，执行 `top` 命令，查看组件的内存使用率。
   - 如果内存使用率不高，继续后续磁盘、JVM 的排查。
   - 如果内存使用率高，需要进一步排查是否因为 JVM 导致。执行 `jmap` 命令，观察内存使用情况。
3. 执行 `jstat -gcutil [pid]` 命令，查看内存使用情况。

磁盘使用率异常告警

【现象】磁盘使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足
- 磁盘空间不足或没有定期清理

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 执行 `df -lh` 命令，查看各目录的磁盘使用情况，找出消耗过高、过快的目录。
3. 如果 logs 目录下的磁盘使用率过高，可以先进行磁盘清理，若其他目录下的磁盘使用率过高，将具体目录信息记录下来，交给技术支持人员分析处理。

JVM 使用率异常告警

【现象】JVM 使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 进入目标容器，执行 `jmap -heap <pid>` 命令，查看内存区域的设置。
3. 登录目标容器，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名。每分钟执行一次，连续执行 5 次。
4. 保存收集到的 JVM 相关数据、信息，联系技术支持人员处理。

6.2.2. aksexecutor 异常告警

8080/8341/12200 端口异常告警

【现象】8080/8341/12200 端口异常告警。

【可能的原因】进程夯住。

【对系统的影响】容器构建服务不可用。

【解决方法】

1. 登录云游运维平台页面，找到 aksexecutor 应用的容器，检查容器状态是否健康。
2. 若容器状态异常，可直接通过云游运维平台发起容器重启操作。
3. 若容器可以登录，通过 `ps -ef|grep java|grep -v xflush` ，检查容器 Java 进程是否存在。
4. 若 Java 进程不存在则查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件，若有此文件，则保存此文件，然后参考步骤 2 重启服务。
5. 若 Java 进程存在，可以执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】访问容器服务控制台，看是否服务正常。

CPU使用率异常告警

【现象】CPU 使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 执行 `top` 命令，查看组件的 CPU 使用率。
   - 如果 CPU 使用率不高，结束，确定是否监控误报。
   - 如果 CPU 使用率高，需要根据调用量做进一步排查。
3. 执行 `netstat -tnlp | grep -E "80|8080" | wc -l` 命令，检查组件的调用情况。
   - 如果调用量偏高，并且符合业务状态，需要对实例进行扩容。
   - 如果调用量不高，需要执行下面的步骤排查 CPU 使用率高的原因。
     a. 执行命令 `top -Hp <组件进程号>` ，找到导致 CPU 或内存使用率高的进程，将其转换成十六进制。
     b. 进入 `jstack ##id` 路径中，打开 ps.txt 文件，根据线程十六进制 ID 定位具体的进程类。
     c. 如果有 Full GC，查看 gc.log 或者执行 `jstat -gcutil [pid]` 命令查看对应的 GC 日志。并将 top/jstack /full GC 等做好记录，交由技术支持人员处理。

内存使用率异常告警

【现象】内存使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 进入目标容器，执行 `top` 命令，查看组件的内存使用率。
    i. 如果内存使用率不高，继续后续磁盘、JVM 的排查。
    ii. 如果内存使用率高，需要进一步排查是否因为 JVM 导致。
    iii. 执行 jmap 命令，观察内存使用情况。
3. 执行 `jstat -gcutil [pid]` 命令，查看内存使用情况。

磁盘使用率异常告警

【现象】磁盘使用率异常告警

【可能的原因】

- 并发访问量高
- 应用实例不足
- 磁盘空间不足或没有定期清理

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 执行 `df -lh` 命令，查看各目录的磁盘使用情况，找出消耗过高、过快的目录。
3. 如果 logs 目录下的磁盘使用率过高，可以先进行磁盘清理，若其他目录下的磁盘使用率过高，将具体目录信息记录下来，交给技术支持人员分析处理。

JVM 使用率异常告警

【现象】JVM 使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。

2. 进入目标容器，执行 `jmap -heap <pid>` 命令，查看各内存区域的设置。
3. 登录目标容器，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名。每分钟执行一次，连续执行 5 次。
4. 保存收集到的 JVM 相关数据、信息，联系技术支持人员处理。

6.2.3. hub 告警

5000 端口告警

【现象】 5000 端口告警。

【可能的原因】 registry 服务异常。

【对系统的影响】 用户可以发起构建服务，但是上传、下载镜像失败。

【解决方法】

1. 登录云游运维平台页面，找到 hub 应用的容器，检查容器状态是否健康。
   - 若容器状态正常，执行步骤 2。
   - 若容器状态不正常，执行步骤 3。
2. 通过云游运维平台 shell 功能登录容器，执行 `ps -ef | grep registry` 查看进程是否存在。

   若不存在可以通过运行

   ```
   bash -c "unset REGISTRY_STORAGE_PROVIDER_NAME && /harbor/registry_entrypoint.sh
   /etc/registry/config.yml" &
   ```

   重启 registry 服务。
3. 通过云游运维平台发起容器重启操作。

【验证方法】 上传镜像，确定功能是否正常。

80 端口告警

【现象】 80 端口告警。

【可能的原因】 nginx 服务异常。

【对系统的影响】 该 harbor 实例无法正常提供服务。

【解决方法】

1. 登录云游运维平台页面，找到 hub 应用的容器，检查容器状态是否健康。
   - 若容器状态正常，执行步骤 2。
   - 若容器状态不正常，执行步骤 3。
2. 通过云游运维平台 shell 功能登录容器，执行 `ps -ef | grep nginx` 查看进程是否存在。

   若不存在可以通过运行 `nginx -s reload` 重启 nginx 服务。
3. 通过云游运维平台发起容器重启操作。

【验证方法】

1. 访问 harbor 管理端确认是否正常。

2. 确认上传、下载镜像是否正常。

8082 端口告警

【现象】8082 端口告警。

【可能的原因】harbor ui 服务异常。

【对系统的影响】

- 无法访问 harbor 管理端。
- docker pull/push 鉴权不可用。

【解决方法】

1. 登录云游运维平台页面，找到 hub 应用的容器，检查容器状态是否健康。
   - 若容器状态正常，执行步骤 2。
   - 若容器状态不正常，执行步骤 3。
2. 通过云游运维平台 shell 功能登录容器，执行 `ps -ef | grep ui` 查看进程是否存在。

   若不存在可以通过运行 `/harbor/ui_start.sh &` 重启 harbor ui 服务。
3. 通过云游运维平台发起容器重启操作。

【验证方法】

1. 访问 harbor 管理端确认是否正常。
2. docker logout，docker login 确认是否正常。
3. 确认上传、下载镜像是否正常。

6.2.4. metaservice 异常告警

8341/8888/12200 端口异常告警

【现象】8341/8888/12200 端口异常告警。

【可能的原因】进程夯住

【对系统的影响】

- 容器服务不可用
- CAFE 页面无法登录

【解决方法】

1. 登录云游运维平台页面，找到 metaservice 应用的容器，检查容器状态是否健康。
2. 若容器状态异常，可直接通过云游运维平台发起容器重启操作（危险等级 G2）。
3. 若容器可以登录，通过 `ps -ef|grep java|grep -v xflush` ， 检查容器 Java 进程是否存在。
   - 若 Java 进程不存在，则查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件，若有此文件，则保存此文件，然后参考第2步骤重启服务。

- 若 Java 进程存在，可以执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】访问容器服务控制台，看是否服务正常。

CPU 使用率异常告警

【现象】CPU 使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 执行 `top` 命令，查看组件的 CPU 使用率。
   i. 如果 CPU 使用率不高，结束，确定是否监控误报。
   ii. 如果 CPU 使用率高，需要根据调用量做进一步排查。
3. 执行 `netstat -tnlp | grep -E "80|8080" | wc -l` 命令，检查组件的调用情况。
   i. 如果调用量偏高，并且符合业务状态，需要对实例进行扩容。
   ii. 如果调用量不高，需要执行下面的步骤排查CPU使用率高的原因。
      a. 执行命令 `top -Hp <组件进程号>`，找到导致 CPU 或内存使用率高的进程，将其转换成十六进制。
      b. 进入 `jstack ##id` 路径中，打开 ps.txt 文件，根据线程十六进制 ID 定位具体的进程类。
      c. 如果有 GC，查看 gc.log 或者执行 `jstat -gcutil [pid]` 命令查看对应的 GC 日志。并将 top/jstack /full GC 等做好记录，交由技术支持人员处理。

内存使用率异常告警

【现象】内存使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 进入目标容器，执行 `top` 命令，查看组件的内存使用率。

- 如果内存使用率不高，继续后续磁盘、JVM 的排查。
- 如果内存使用率高，需要进一步排查是否因为 JVM 导致。

3. 执行 `jmap` 命令，观察内存使用情况。
4. 执行 `jstat -gcutil [pid]` 命令，查看内存使用情况。

磁盘使用率异常告警

【现象】磁盘使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足
- 磁盘空间不足或没有定期清理

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 执行 `df -lh` 命令，查看各目录的磁盘使用情况，找出消耗过高、过快的目录。
3. 如果 logs 目录下的磁盘使用率过高，可以先进行磁盘清理，若其他目录下的磁盘使用率过高，将具体目录信息记录下来，交给技术支持人员分析处理。

JVM 使用率异常告警

【现象】JVM 使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 进入目标容器，执行 `jmap -heap <pid>` 命令，查看各内存区域的设置。
3. 登录目标容器，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名。每分钟执行一次，连续执行 5 次。
4. 保存收集到的 JVM 相关数据、信息，联系技术支持人员处理。

6.2.5. lhc 异常告警

8080/8341/12200 端口异常告警

【现象】8080/8341/12200 端口异常告警。

【可能的原因】进程夯住

【对系统的影响】负载均衡页面无法展示和创建。

【解决方法】

1. 登录云游运维平台页面，找到 LHC 应用的容器，检查容器状态是否健康。
2. 若容器状态异常，可直接通过云游运维平台发起容器重启操作。
3. 若容器可以登录，通过 `ps -ef|grep java|grep -v xflush` ，检查容器 Java 进程是否存在。
   - 若 Java 进程不存在，则查看 `/home/admin/logs` 下是否有 .Hprof 结尾的文件，若有此文件，则保存此文件，然后参考步骤 2 重启服务。
   - 若 Java 进程存在，可以执行 `jmap -heap <pid>` 命令，查看各内存区域的设置，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率 和类全名，保存收集到的 JVM 相关数据、信息，联系技术支持人员排查，然后参考步骤 2 重启服务。

【验证方法】访问容器服务控制台，看是否服务正常。

CPU 使用率异常告警

【现象】CPU 使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 执行 top 命令，查看组件的 CPU 使用率。
   - 如果 CPU 使用率不高，结束，确定是否监控误报。
   - 如果 CPU 使用率高，需要根据调用量做进一步排查。
3. 执行 `netstat -tnlp | grep -E "80|8080" | wc -l` 命令，检查组件的调用情况。
   - 如果调用量偏高，并且符合业务状态，需要对实例进行扩容。
   - 如果调用量不高，需要执行下面的步骤排查 CPU 使用率高的原因。
     a. 执行命令 `top -Hp <组件进程号>` ，找到导致 CPU 或内存使用率高的进程，将其转换成十六进制。
     b. 进入 `jstack ##id` 路径中，打开 ps.txt 文件，根据线程十六进制 ID 定位具体的进程类。
     c. 如果有 Full GC，查看 gc.log 或者执行 `jstat -gcutil [pid]` 命令查看对应的 GC 日志。并将 top/jstack /full GC 等做好记录，交由技术支持人员处理。

内存使用率异常告警

【现象】内存使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 进入目标容器，执行 `top` 命令，查看组件的内存使用率。
   - 如果内存使用率不高，继续后续磁盘、JVM 的排查。
   - 如果内存使用率高，需要进一步排查是否因为 JVM 导致。
3. 执行 `jmap` 命令，观察内存使用情况。
4. 执行 `jstat -gcutil [pid]` 命令，查看内存使用情况。

磁盘使用率异常告警

【现象】磁盘使用率异常告警

【可能的原因】

- 并发访问量高
- 应用实例不足
- 磁盘空间不足或没有定期清理

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 执行 `df -lh` 命令，查看各目录的磁盘使用情况，找出消耗过高、过快的目录。
3. 如果 logs 目录下的磁盘使用率过高，可以先进行磁盘清理，若其他目录下的磁盘使用率过高，则记录具体目录信息，提供给技术支持人员作分析处理。

磁盘使用率异常告警

【现象】JVM 使用率异常告警。

【可能的原因】

- 并发访问量高
- 应用实例不足

【对系统的影响】可能导致容器服务页面操作响应较慢，影响服务性能。

【解决方法】

1. 登录云游运维平台页面，找到应用所在容器。
2. 进入目标容器，执行 `jmap -heap <pid>` 命令，查看各内存区域的设置。
3. 登录目标容器，执行 `jmap-histo:live <pid>` 命令，查看每个 class 的实例数目、内存使用率和类全名。每分钟执行一次，连续执行 5 次。

4. 保存收集到的 JVM 相关数据、信息，联系技术支持人员处理。

6.3. 组件可用性故障

6.3.1. apaks 节点异常

【现象】

- 单节点异常，不影响服务，监控中可收到 apaks 相关端口异常告警。
- 全部节点异常，容器服务无法使用。

【可能的原因】

- 组件的 bug
- 严重的误操作
- 主机宕机
- 容器进程异常
- 网络问题

【对系统的影响】

- 单个节点异常，不影响服务。
- 全部节点异常，则容器服务无法使用。

【解决方法】参考端口异常告警，处理容器监控异常。

【验证方法】

- 登录 CAFÉ 页面，查看容器服务是否恢复。
- 登录监控页面，查看告警是否恢复。

6.3.2. aksexecutor 节点异常

【现象】

- 单节点异常，不影响服务，监控中可收到 aksexecutor 相关端口异常告警。
- 全部节点异常，则容器构建服务无法使用。

【可能的原因】

- 组件的 bug
- 严重的误操作
- 主机宕机
- 容器进程异常
- 网络问题

【对系统的影响】

- 单个节点异常，不影响服务。
- 全部节点异常，则容器服务无法使用。

【解决方法】

参考端口异常告警，处理容器监控异常。

【验证方法】

- 登录 CAFÉ 页面，查看容器构建服务是否恢复。
- 登录监控页面，查看告警是否恢复。

6.3.3. hub 节点异常

【现象】

- 单节点异常，不影响服务，监控中可收到 hub 相关端口异常告警。
- 全部节点异常，则容器构建服务无法使用。

【可能的原因】

- 组件的 bug
- 严重的误操作
- 主机宕机
- 容器进程异常
- 网络问题

【对系统的影响】

- 单个节点异常，不影响服务。
- 全部节点异常，则镜像服务无法使用。

【解决方法】参考端口异常告警，处理监控异常。

【验证方法】

- 登录 CAFE 页面，参考镜像中心是否可以访问，通过客户端进行镜像上传下载操作，查看是否恢复。
- 登录监控页面，查看告警是否恢复。

6.3.4. metaservice 节点异常

【现象】

- 单节点异常，不影响服务，监控中可收到 metaservice 相关端口异常告警。
- 全部节点异常，则容器构建服务无法使用。

【可能的原因】

- 组件的 bug
- 严重的误操作
- 主机宕机
- 容器进程异常
- 网络问题

【对系统的影响】

- 单个节点异常，不影响服务。
- 全部节点异常，则 CAFE 整个服务无法使用。

【解决方法】参考端口异常告警，处理容器监控异常。

【验证方法】

- 登录 CAFÉ 页面，参考服务是否恢复。
- 登录监控页面，查看告警是否恢复。

6.3.5. lhc 节点异常

【现象】

- 单节点异常，不影响服务，监控中可收到 lhc 相关端口异常告警。
- 全部节点异常，则容器发布发布部署服务无法使用。

【可能的原因】

- 组件的 bug
- 严重的误操作
- 主机宕机
- 容器进程异常
- 网络问题

【对系统的影响】

- 单个节点异常，不影响服务。
- 全部节点异常，则负载均衡服务无法使用。

【解决方法】参考端口异常告警，处理容器监控异常。

【验证方法】

- 登录 CAFÉ 页面，参考负载均衡服务是否恢复。
- 登录监控页面，查看告警是否恢复。

6.4. 发布部署故障

6.4.1. 发布单冲突

【现象】发布单出现冲突。

【解决方法】

1. 前往 LHC 应用的 lhcconsole 数据库执行对应操作。

   如果单击 **开始发布** 时提示存在冲突的发布单，而冲突的发布单又无法取消，则需要手动删除 LHC 中 lks_res_deploy_mutex 表内记录。

```
-- 先查询是否存在记录
SELECT * FROM `lks_res_deploy_mutex` WHERE`ref_res_type`='MULTI_CONTAINER_SERVICE' AND `ref_resource`='应用服务名';

-- 删除存在的记录
DELETE FROM `lks_res_deploy_mutex` WHERE`ref_res_type`='MULTI_CONTAINER_SERVICE' AND `ref_resource`='应用服务名';
```

2. （可选）执行以下命令：

> 说明
>
> 步骤 2 为可选步骤，用于订正发布单状态。理论上，无需关注发布单状态。

```
-- 查询一下发布单，核对信息是否正确
SELECT * FROM `ldc_plan` WHERE `time_series_id`='发布单id';

-- 更新发布单状态
UPDATE ldc_plan SET state='CANCELED' WHERE time_series_id='';

-- 查询应用服务发布单详情
SELECT * FROM `ldc_service` WHERE `plan_id`=(SELECT `id` FROM `ldc_plan`WHERE `time_ser
ies_id` = 'shuang_202103261537288405');

-- 更新应用服务发布单状态
UPDATE `ldc_service` SET state='CANCELED' WHERE `plan_id`=(SELECT `id`FROM `ldc_plan` W
HERE `time_series_id` = 'shuang_202103261537288405');
```

6.4.2. 发布单失败

6.4.2.1. 初始化失败—页面没有错误信息

【现象】初始化失败，且页面无任何错误提示信息。

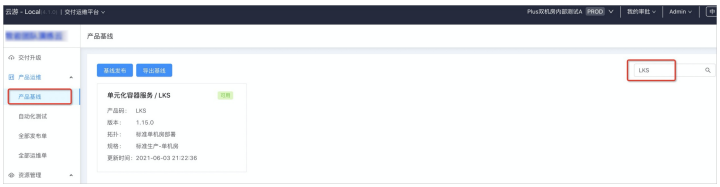
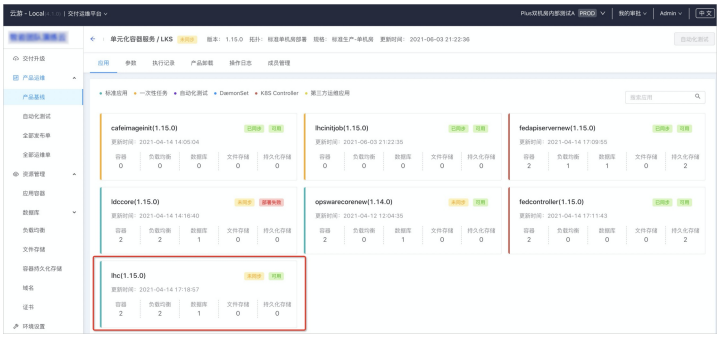
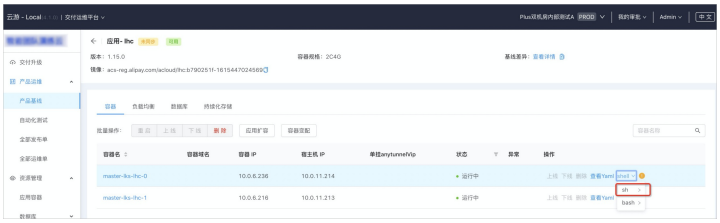
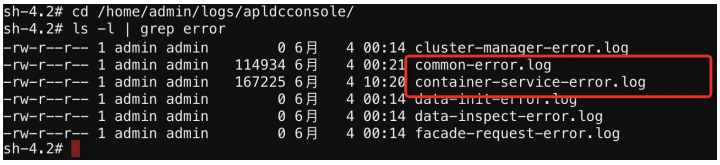
【解决方法】

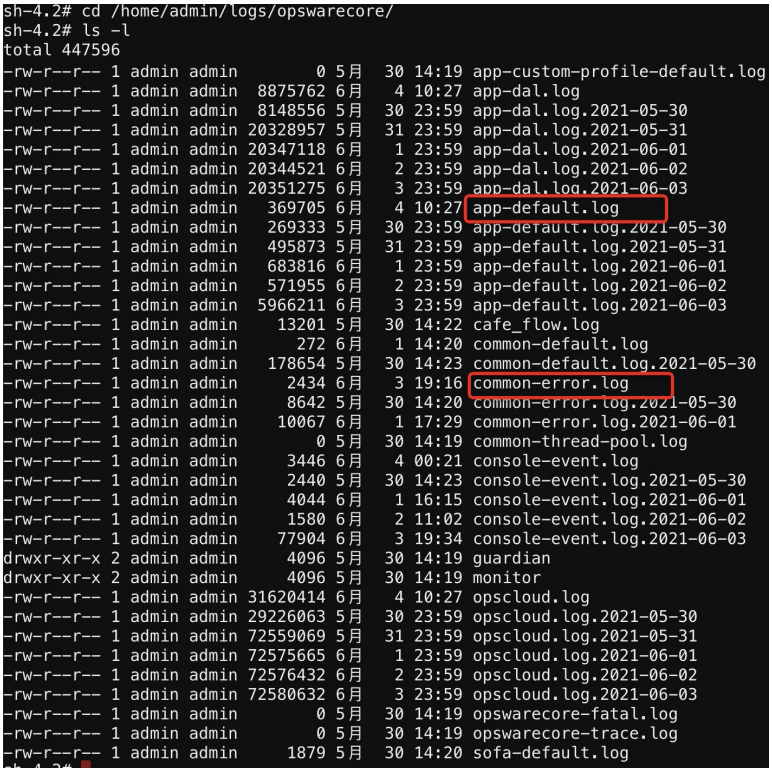
前往 lhc、opswarecore、opswareorch 这几个应用内查看错误日志。具体步骤如下表所示：

> 说明
>
> 如果还有疑问，建议将错误日志提供给相关开发人员以供排查。

操作项	说明

操作项	说明

如何查找 lhc 日志	1. 登录云游 Local。
	2. 在左侧导航栏单击 **产品基线**，在右侧产品码搜索框中输入 **LKS**。
	 
	3. 进入产品详情页面，找到 lhc 这个应用。
	 
	4. 找到 lhc 的容器，单击 **操作** 列中的 **sh** 进入 webshell:
	 
	5. 进入容器后，使用 `cd` 命令跳转到 `/home/admin/logs/apldcconsole/` 目录，查看 common-error.log 或 container-service-error.log 是否有错误日志。
	 
	`sh-4.2# cd /home/admin/logs/apldcconsole/`
	`sh-4.2# ls -l | grep error`
	`-rw-r--r-- 1 admin admin 0 6月 4 00:14 cluster-manager-error.log`
	`-rw-r--r-- 1 admin admin 114934 6月 4 00:21 common-error.log`
	`-rw-r--r-- 1 admin admin 167225 6月 4 10:20 container-service-error.log`
	`-rw-r--r-- 1 admin admin 0 6月 4 00:14 data-init-error.log`
	`-rw-r--r-- 1 admin admin 0 6月 4 00:14 data-inspect-error.log`
	`-rw-r--r-- 1 admin admin 0 6月 4 00:14 facade-request-error.log`
	`sh-4.2#`

操作项	说明
如何查找 opswarecorenew 日志	参考上述 lhc 日志查找方法，找到 opswarecorenew 这个应用，进入容器，重点看 common-error.log 或 app-default.log 这两个日志： 

操作项	说明
如何查找 opswareorch 日志	1. 登录云游 Local。 2. 在左侧导航栏单击 **产品基线**，在右侧产品码搜索框中输入 **APPPLATFORM**。 3. 找到 opswareorch 应用，进入容器，查看 `/home/admin/logs/acopswareorchestration` 目录的 common-error.log。

```
[root@ys-appplatform-opswareorch-0 /home/admin/logs/acopswareorchestration]
# pwd
/home/admin/logs/acopswareorchestration

[root@ys-appplatform-opswareorch-0 /home/admin/logs/acopswareorchestration]
# ls -l
total 123212
-rw-r--r-- 1 admin admin    57854 2月  22 16:43 action-handler-default.log
-rw-r--r-- 1 admin admin        0 2月  11 17:17 action-handler-default.log.2022-02-11
-rw-r--r-- 1 admin admin    42510 2月  16 17:54 action-handler-default.log.2022-02-16
-rw-r--r-- 1 admin admin   149096 2月  18 19:26 action-handler-default.log.2022-02-18
-rw-r--r-- 1 admin admin    19848 2月  21 20:07 action-handler-default.log.2022-02-21
-rw-r--r-- 1 admin admin  8928214 2月  22 17:39 activity-default.log
-rw-r--r-- 1 admin admin  2970147 2月  11 23:59 activity-default.log.2022-02-11
-rw-r--r-- 1 admin admin 10632147 2月  12 23:59 activity-default.log.2022-02-12
-rw-r--r-- 1 admin admin 10632147 2月  13 23:59 activity-default.log.2022-02-13
-rw-r--r-- 1 admin admin 10630916 2月  14 23:59 activity-default.log.2022-02-14
-rw-r--r-- 1 admin admin 10632147 2月  15 23:59 activity-default.log.2022-02-15
-rw-r--r-- 1 admin admin 11391360 2月  16 23:59 activity-default.log.2022-02-16
-rw-r--r-- 1 admin admin 10649421 2月  17 23:59 activity-default.log.2022-02-17
-rw-r--r-- 1 admin admin 12059745 2月  18 23:59 activity-default.log.2022-02-18
-rw-r--r-- 1 admin admin 10648188 2月  19 23:59 activity-default.log.2022-02-19
-rw-r--r-- 1 admin admin 10649421 2月  20 23:59 activity-default.log.2022-02-20
-rw-r--r-- 1 admin admin 11254129 2月  21 23:59 activity-default.log.2022-02-21
-rw-r--r-- 1 admin admin   252160 2月  22 16:43 activity-event.log
-rw-r--r-- 1 admin admin        0 2月  11 17:17 activity-event.log.2022-02-11
-rw-r--r-- 1 admin admin   174786 2月  16 17:56 activity-event.log.2022-02-16
-rw-r--r-- 1 admin admin   308001 2月  18 19:26 activity-event.log.2022-02-18
-rw-r--r-- 1 admin admin   145015 2月  21 20:07 activity-event.log.2022-02-21
-rw-r--r-- 1 admin admin   469049 2月  22 17:38 app-default.log
-rw-r--r-- 1 admin admin    45178 2月  11 23:59 app-default.log.2022-02-11
-rw-r--r-- 1 admin admin   161280 2月  12 23:58 app-default.log.2022-02-12
-rw-r--r-- 1 admin admin   161280 2月  13 23:58 app-default.log.2022-02-13
-rw-r--r-- 1 admin admin   161280 2月  14 23:58 app-default.log.2022-02-14
-rw-r--r-- 1 admin admin   161280 2月  15 23:58 app-default.log.2022-02-15
-rw-r--r-- 1 admin admin   438305 2月  16 23:58 app-default.log.2022-02-16
-rw-r--r-- 1 admin admin   161280 2月  17 23:58 app-default.log.2022-02-17
-rw-r--r-- 1 admin admin   701606 2月  18 23:58 app-default.log.2022-02-18
-rw-r--r-- 1 admin admin   161280 2月  19 23:58 app-default.log.2022-02-19
-rw-r--r-- 1 admin admin   161280 2月  20 23:58 app-default.log.2022-02-20
-rw-r--r-- 1 admin admin   336939 2月  21 23:59 app-default.log.2022-02-21
-rw-r--r-- 1 admin admin     3939 2月  22 16:42 common-default.log
-rw-r--r-- 1 admin admin    57030 2月  11 17:17 common-default.log.2022-02-11
-rw-r--r-- 1 admin admin     2902 2月  16 17:54 common-default.log.2022-02-16
-rw-r--r-- 1 admin admin     4203 2月  18 17:29 common-default.log.2022-02-18
-rw-r--r-- 1 admin admin     1382 2月  21 17:24 common-default.log.2022-02-21
-rw-r--r-- 1 admin admin     5586 2月  22 15:35 common-error.log
-rw-r--r-- 1 admin admin        0 2月  11 17:17 common-error.log.2022-02-11
-rw-r--r-- 1 admin admin    16782 2月  16 17:56 common-error.log.2022-02-16
-rw-r--r-- 1 admin admin   123615 2月  22 16:43 orch-event.log
-rw-r--r-- 1 admin admin     2716 2月  11 17:17 orch-event.log.2022-02-11
-rw-r--r-- 1 admin admin    90151 2月  16 17:56 orch-event.log.2022-02-16
-rw-r--r-- 1 admin admin   192413 2月  18 19:26 orch-event.log.2022-02-18
-rw-r--r-- 1 admin admin    61916 2月  21 20:07 orch-event.log.2022-02-21
-rw-r--r-- 1 admin admin        0 2月  11 17:17 routing-inspection-default.log
-rw-r--r-- 1 admin admin        0 2月  11 17:17 routing-inspection-error.log
-rw-r--r-- 1 admin admin     7853 2月  11 17:17 sofa-default.log
drwxr-xr-x 2 admin admin     4096 2月  22 11:07 zdal
```

操作项	说明

6.4.2.2. 发布失败—Pod 已创建成功

【现象】发布失败，Pod 已成功创建。

【解决方法】

需要使用 fed 的 kubeconfig 查看 federatedcafedeployment、federatedcafedeploymentstatuses 这两个对象是否已经生成。关于如何查找 fed kubeconfig，可参考 如何查找 fed kubeconfig。

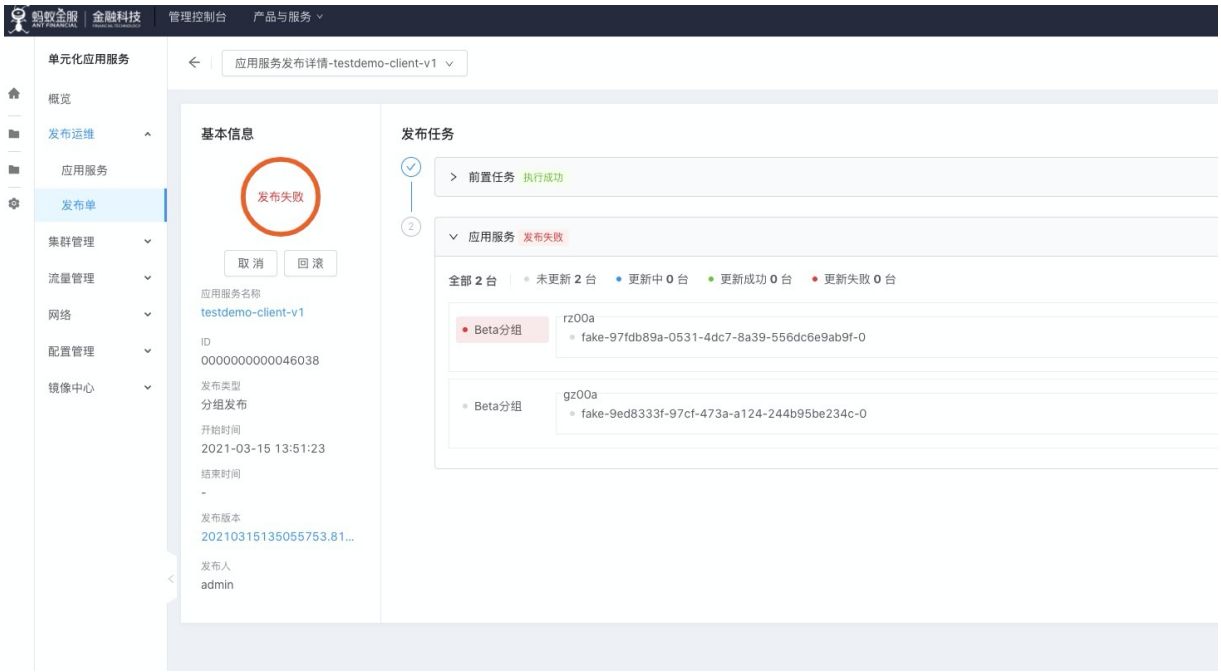
如果 federatedcafedeploymentstatuses 未生成，需重启 fedcontroller。该问题一般出现在新搭建的环境中。

6.4.2.3. 发布失败—Pod 创建失败

如果过了很长时间才出现发布失败

【现象】如果过了很长时间，才出现发布失败。

【可能的原因】当发布失败时，Pod 名称仍以 fake 开头，可能是 fed 层有问题。



【解决方法】

使用 fed 的 kubeconfig 查看 federatedcafedeployment、federatedcafedeploymentstatuses 这两个对象是否已经生成。

- 如果没有生成，联系对应开发人员进行排查。
- 如果已经生成，使用 describe 查看是否报错。

> **说明**
>
> 关于如何查找 fed kubeconfig，可参考 如何查找 fed kubeconfig。

```
# 先获取fed kubeconfig文件，替换 namespace 和 应用服务名
kubectl --kubeconfig fed.kubeconfig -n namespace getfederatedcafedeployment | grep
应用服务名
kubectl --kubeconfig fed.kubeconfig -n namespace get
federatedcafedeploymentstatuses | grep 应用服务名
```

如果 describe federatedcafedeployment 看到有如下报错，则说明该 cell 对应的集群有问题。

```
            Requests:
              Cpu:                  100m
              Memory:               100Mi
            Termination Message Policy:  FallbackToLogsOnError
          Dns Config:
            Searches:
          Dns Policy:    ClusterFirstWithHostNet
          Host Network:  false
          Readiness Gates:
            Condition Type:  cafe.sofastack.io/service-ready
          Restart Policy:    Always
          Volumes:
      Topology:
        Unit Replicas:
        Unit Type:  None
        Values:
          Cell
  Topology Type:  cell
Status:
  Conditions:
    Last Transition Time:  2021-03-24T05:51:41Z
    Last Update Time:      2021-03-24T05:51:41Z
    Reason:                CheckClusters
    Status:                False
    Type:                  Propagation
  Observed Generation:     3
  Topologies:
    Name:    rz00a
    Status:  ClusterNotReady
Events:
  Type     Reason                 Age                 From                                Message
  ----     ------                 ----                ----                                -------
  Warning  EnsureFinalizerError   25m                 federatedcafedeployment-controller  Failed to ensure finalizer: Operation cannot be fulfilled on federatedcafedeployments.application.cafe.sofastack.io "ss
your changes to the latest version and try again
  Normal   Propagating            25m                 opswarecore                         In propagating for task tfhwywof-lhcpretest0223-default/pipeline-0000000000694058/pipeline-stage-0000000000708033-1/dep
source application.cafe.sofastack.io/v1/FederatedCafeDeployment/tfhwywof-lhcpretest0223-default/sss
  Normal   Propagating            25m                 opswarecore                         In propagating for task tfhwywof-lhcpretest0223-default/pipeline-0000000000694058/pipeline-stage-0000000000708033-1/dep
d resource is null for application.cafe.sofastack.io/v1/FederatedCafeDeployment/tfhwywof-lhcpretest0223-default/sss
  Normal   Propagating            25m                 opswarecore                         In propagating for task tfhwywof-lhcpretest0223-default/pipeline-0000000000694058/pipeline-stage-0000000000708033-1/dep
d for federated resource application.cafe.sofastack.io/v1/FederatedCafeDeployment/tfhwywof-lhcpretest0223-default/sss with status ClusterNotReady
  Normal   FieldManagerPruned     10m                 opswarecore                         Successfully pruned field manager
  Normal   WaitingFedStatus       10m                 opswarecore                         waiting for creation of fed status for task tfhwywof-lhcpretest0223-default/pipeline-0000000000694058/pipeline-stage-00
  Warning  ClusterNotReady        10m (x6 over 25m)   federatedcafedeployment-controller  Cluster not ready
```

如果是新搭建的环境，有 federatedcafedeployment 但没有 federatedcafedeploymentstatuses，可以重启，并尝试重新发布。如果仍然存在问题，联系对应开发人员。

如果立即出现发布失败

【现象】立即出现发布失败。单击 **重试** 也是立即出现发布失败。

【可能的原因】这种情况一般都是应用层报错。

【解决方法】

前往 opswarecorenew 查看 common-error.log。

如果报错出现 `missesmeta-info for status of cell-placement rz00x` ，可能是这个应用服务之前用过的 cell 被删除。

```
2021-03-18 21:47:14,605 [0b7c742b16160752344686003850 0 - /// - ] ERROR task.WorkTask - [0b7c742b16160752344686003850] failed to patch workload GroupVersionKind[group=application.cafe.sofastack.i
;version=v1;kind=FederatedCafeDeployment;]/NamespacedName[namespace=antcloud-test0301-default;name=service;] for task antcloud-test0301-default/pipeline-0000000000062007/pipeline-stage-000000000006
007-1/deploy-task-mixedcells as V1Status[apiVersion=v1;code=500;details=null;kind=Status;message=[validatefederatedresources] misses meta-info for status of cell-placement rz00b;metadata=V1ListMeta
_continue=null;remainingItemCount=null;resourceVersion=null;selfLink=null;];reason=null;status=Failure;]
2021-03-18 21:47:14,605 [0b7c742b16160752344686003850 0 - /// - ] ERROR task.WorkTask - [0b7c742b16160752344686003850] failed to execute work task for antcloud-test0301-default/pipeline-000000000
062007/pipeline-stage-0000000000062007-1/deploy-task-mixedcells when operating kubernetes
com.alipay.cloud.opscore.core.service.pipelinev2.task.exception.KubernetesOpsException: [validatefederatedresources] misses meta-info for status of cell-placement rz00b
        at com.alipay.cloud.opscore.core.service.pipelinev2.task.WorkTask.applyWorkload(WorkTask.java:286)
        at com.alipay.cloud.opscore.core.service.pipelinev2.task.WorkTask.applyWorkloadInFirstStage(WorkTask.java:219)
        at com.alipay.cloud.opscore.core.service.pipelinev2.task.WorkTask.execute(WorkTask.java:137)
        at java.util.concurrent.FutureTask.run(FutureTask.java:266)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:748)
```

6.4.2.4. 发布失败—Pod 发布失败

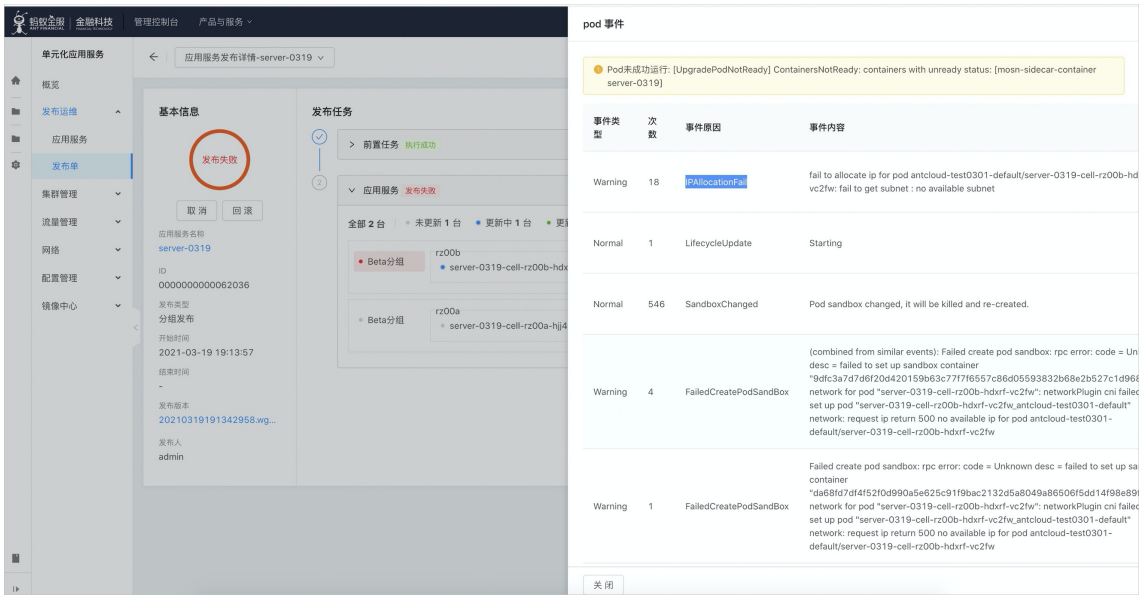
【现象】发布失败—Pod 发布失败。

【解决方法】

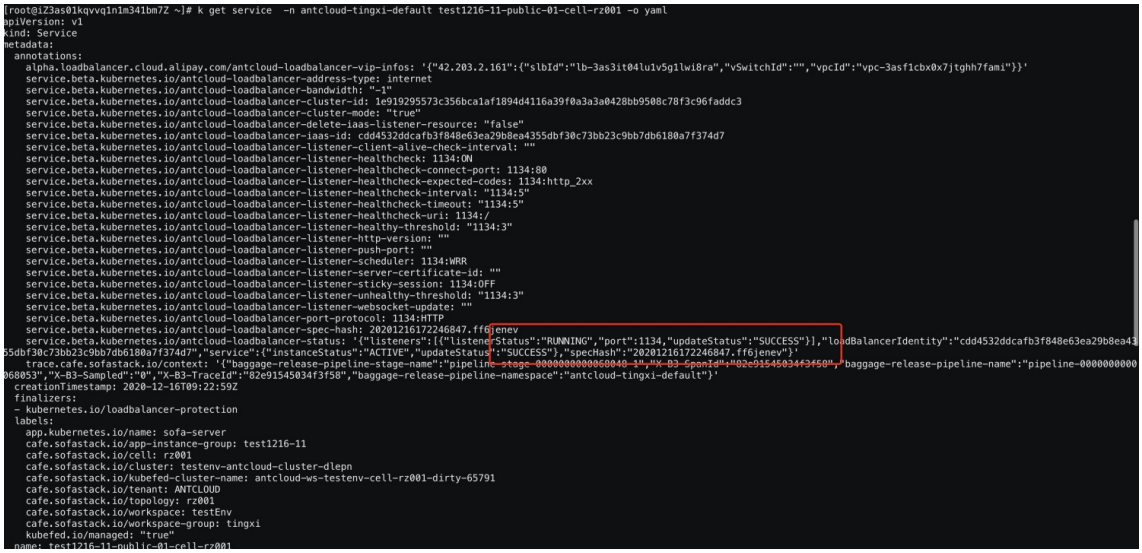
1. 排查 Pod。

   查看 Pod 的事件。

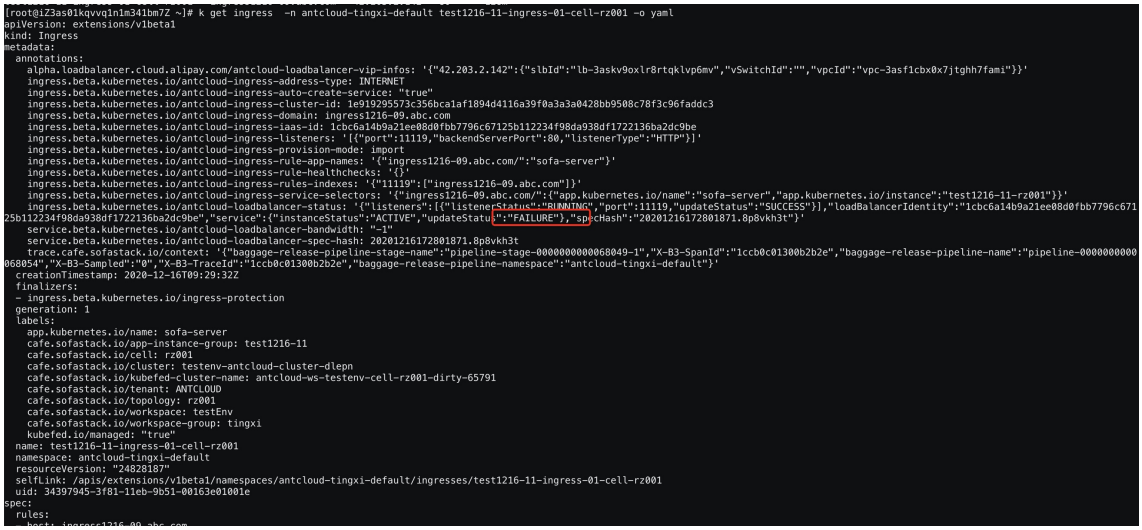
- 如果事件报错信息为 `IPAllocationFail:fail to allocate ip for pod...`（如下图所示），则表示 IP 不够用，可以删除掉一些应用服务，释放出一些 IP。



- 如果事件报错信息为 `[UpgradePodNotAvailable]available condition not match`，查看 Ingress 和 Service 的 YAML。可以将获取到的报错信息提交给对应开发人员以供进一步排查。



```
[root@iZ3as01kqvvq1n1m341bm7Z ~]# k get service  -n antcloud-tingxi-default test1216-11-public-01-cell-rz001 -o yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
    alpha.loadbalancer.cloud.alipay.com/antcloud-loadbalancer-vip-infos: '{"42.203.2.161":{"slbId":"lb-3as3it04lu1v5g1lwi8ra","vSwitchId":"","vpcId":"vpc-3asf1cbx0x7jtghh7fami"}}'
    service.beta.kubernetes.io/antcloud-loadbalancer-address-type: internet
    service.beta.kubernetes.io/antcloud-loadbalancer-bandwidth: "-1"
    service.beta.kubernetes.io/antcloud-loadbalancer-cluster-id: 1e919295573c356bca1af1894d4116a39f0a3a3a0428bb9508c78f3c96faddc3
    service.beta.kubernetes.io/antcloud-loadbalancer-cluster-mode: "true"
    service.beta.kubernetes.io/antcloud-loadbalancer-delete-iaas-listener-resource: "false"
    service.beta.kubernetes.io/antcloud-loadbalancer-iaas-id: cdd4532ddcafb3f848e63ea29b8ea4355dbf30c73bb23c9bb7db6180a7f374d7
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-client-alive-check-interval: ""
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-healthcheck: 1134:ON
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-healthcheck-connect-port: 1134:80
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-healthcheck-expected-codes: 1134:http_2xx
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-healthcheck-interval: "1134:5"
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-healthcheck-timeout: "1134:5"
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-healthcheck-uri: 1134:/
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-healthy-threshold: "1134:3"
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-http-version: ""
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-push-port: ""
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-scheduler: 1134:WRR
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-server-certificate-id: ""
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-sticky-session: 1134:OFF
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-unhealthy-threshold: "1134:3"
    service.beta.kubernetes.io/antcloud-loadbalancer-listener-websocket-update: ""
    service.beta.kubernetes.io/antcloud-loadbalancer-port-protocol: 1134:HTTP
    service.beta.kubernetes.io/antcloud-loadbalancer-spec-hash: 20201216172246847.ff6jenev
    service.beta.kubernetes.io/antcloud-loadbalancer-status: '{"listeners":[{"listenerStatus":"RUNNING","port":1134,"updateStatus":"SUCCESS"}],"loadBalancerIdentity":"cdd4532ddcafb3f848e63ea29b8ea43
55dbf30c73bb23c9bb7db6180a7f374d7","service":{"instanceStatus":"ACTIVE","updateStatus":"SUCCESS"},"specHash":"20201216172246847.ff6jenev"}'
    trace.cafe.sofastack.io/context: '{"baggage-release-pipeline-stage-name":"pipeline-stage-0000000000068048-1","X-B3-SpanId":"402e01545034f358","baggage-release-pipeline-name":"pipeline-0000000000
068053","X-B3-Sampled":"0","X-B3-TraceId":"82e91545034f3f58","baggage-release-pipeline-namespace":"antcloud-tingxi-default"}'
  creationTimestamp: 2020-12-16T09:22:59Z
  finalizers:
  - kubernetes.io/loadbalancer-protection
  labels:
    app.kubernetes.io/name: sofa-server
    cafe.sofastack.io/app-instance-group: test1216-11
    cafe.sofastack.io/cell: rz001
    cafe.sofastack.io/cluster: testenv-antcloud-cluster-dlepn
    cafe.sofastack.io/kubefed-cluster-name: antcloud-ws-testenv-cell-rz001-dirty-65791
    cafe.sofastack.io/tenant: ANTCLOUD
    cafe.sofastack.io/topology: rz001
    cafe.sofastack.io/workspace: testEnv
    cafe.sofastack.io/workspace-group: tingxi
    kubefed.io/managed: "true"
  name: test1216-11-public-01-cell-rz001
```

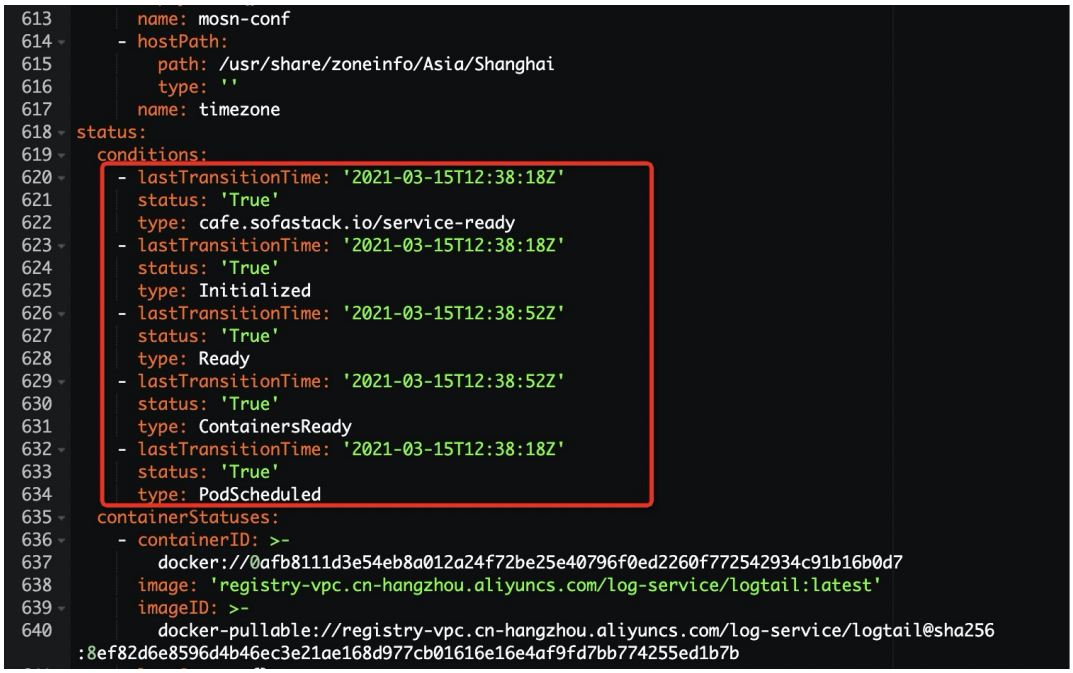


```
[root@iZ3as01kqvvq1n1m341bm7Z ~]# k get ingress  -n antcloud-tingxi-default test1216-11-ingress-01-cell-rz001 -o yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    alpha.loadbalancer.cloud.alipay.com/antcloud-loadbalancer-vip-infos: '{"42.203.2.142":{"slbId":"lb-3askv9oxlr8rtqklvp6mv","vSwitchId":"","vpcId":"vpc-3asf1cbx0x7jtghh7fami"}}'
    ingress.beta.kubernetes.io/antcloud-ingress-address-type: INTERNET
    ingress.beta.kubernetes.io/antcloud-ingress-auto-create-service: "true"
    ingress.beta.kubernetes.io/antcloud-ingress-cluster-id: 1e919295573c356bca1af1894d4116a39f0a3a3a0428bb9508c78f3c96faddc3
    ingress.beta.kubernetes.io/antcloud-ingress-domain: ingress1216-09.abc.com
    ingress.beta.kubernetes.io/antcloud-ingress-iaas-id: 1cbc6a14b9a21ee08d0fbb7796c67125b112234f98da938df1722136ba2dc9be
    ingress.beta.kubernetes.io/antcloud-ingress-listeners: '[{"port":11119,"backendServerPort":80,"listenerType":"HTTP"}]'
    ingress.beta.kubernetes.io/antcloud-ingress-provision-mode: import
    ingress.beta.kubernetes.io/antcloud-ingress-rule-app-names: '{"ingress1216-09.abc.com/":"sofa-server"}'
    ingress.beta.kubernetes.io/antcloud-ingress-rule-healthchecks: '{}'
    ingress.beta.kubernetes.io/antcloud-ingress-rules-indexes: '{"11119":["ingress1216-09.abc.com"]}'
    ingress.beta.kubernetes.io/antcloud-ingress-service-selectors: '{"ingress1216-09.abc.com/":{"app.kubernetes.io/name":"sofa-server","app.kubernetes.io/instance":"test1216-11-rz001"}}'
    ingress.beta.kubernetes.io/antcloud-loadbalancer-status: '{"listeners":[{"listenerStatus":"RUNNING","port":11119,"updateStatus":"SUCCESS"}],"loadBalancerIdentity":"1cbc6a14b9a21ee08d0fbb7796c671
25b112234f98da938df1722136ba2dc9be","service":{"instanceStatus":"ACTIVE","updateStatus":"FAILURE"},"specHash":"20201216172801871.8p8vkh3t"}'
    service.beta.kubernetes.io/antcloud-loadbalancer-bandwidth: "-1"
    service.beta.kubernetes.io/antcloud-loadbalancer-spec-hash: 20201216172801871.8p8vkh3t
    trace.cafe.sofastack.io/context: '{"baggage-release-pipeline-stage-name":"pipeline-stage-0000000000068049-1","X-B3-SpanId":"1ccb0c01300b2b2e","baggage-release-pipeline-name":"pipeline-0000000000
068054","X-B3-Sampled":"0","X-B3-TraceId":"1ccb0c01300b2b2e","baggage-release-pipeline-namespace":"antcloud-tingxi-default"}'
  creationTimestamp: 2020-12-16T09:29:32Z
  finalizers:
  - ingress.beta.kubernetes.io/ingress-protection
  generation: 1
  labels:
    app.kubernetes.io/name: sofa-server
    cafe.sofastack.io/app-instance-group: test1216-11
    cafe.sofastack.io/cell: rz001
    cafe.sofastack.io/cluster: testenv-antcloud-cluster-dlepn
    cafe.sofastack.io/kubefed-cluster-name: antcloud-ws-testenv-cell-rz001-dirty-65791
    cafe.sofastack.io/tenant: ANTCLOUD
    cafe.sofastack.io/topology: rz001
    cafe.sofastack.io/workspace: testEnv
    cafe.sofastack.io/workspace-group: tingxi
    kubefed.io/managed: "true"
  name: test1216-11-ingress-01-cell-rz001
  namespace: antcloud-tingxi-default
  resourceVersion: "24828187"
  selfLink: /apis/extensions/v1beta1/namespaces/antcloud-tingxi-default/ingresses/test1216-11-ingress-01-cell-rz001
  uid: 34397945-3f81-11eb-9b51-00163e01001e
spec:
  rules:
```

- 如果事件无报错信息，查看 Pod 状态。

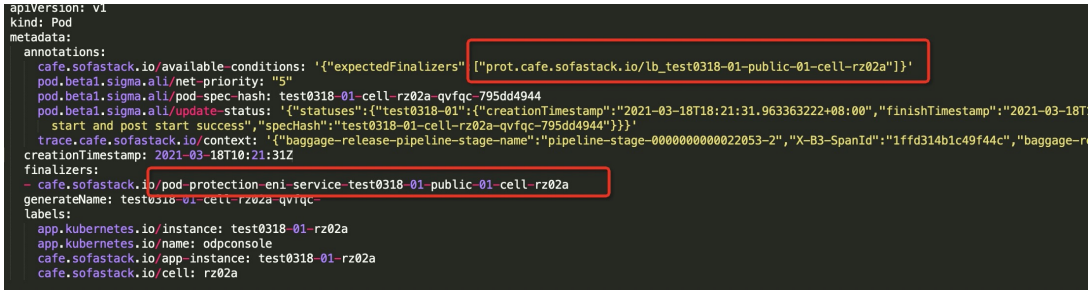
a. 查看 Pod 的 YAML，判断 Pod 是否已 ready。具体判断方法如下：

四个参数 status（见下图）须均为 **True** 才视为 ready。



b. 如果 Pod 已 ready，检查 Service 或 Ingress 是否有问题。

如果是添加了 Service 或 Ingress 导致发布失败，需要看一下 Pod YAML 内的 finalizers 是否与 expectedFinalizers 一致。下图是一个 finalizers 错误示例。



2. 若执行完上述操作后无任何进展，查看 cafedeployment。

获取到 cafed 的 YAML 和 Pod 的 YAML，然后提供给对应开发人员以供进一步排查。

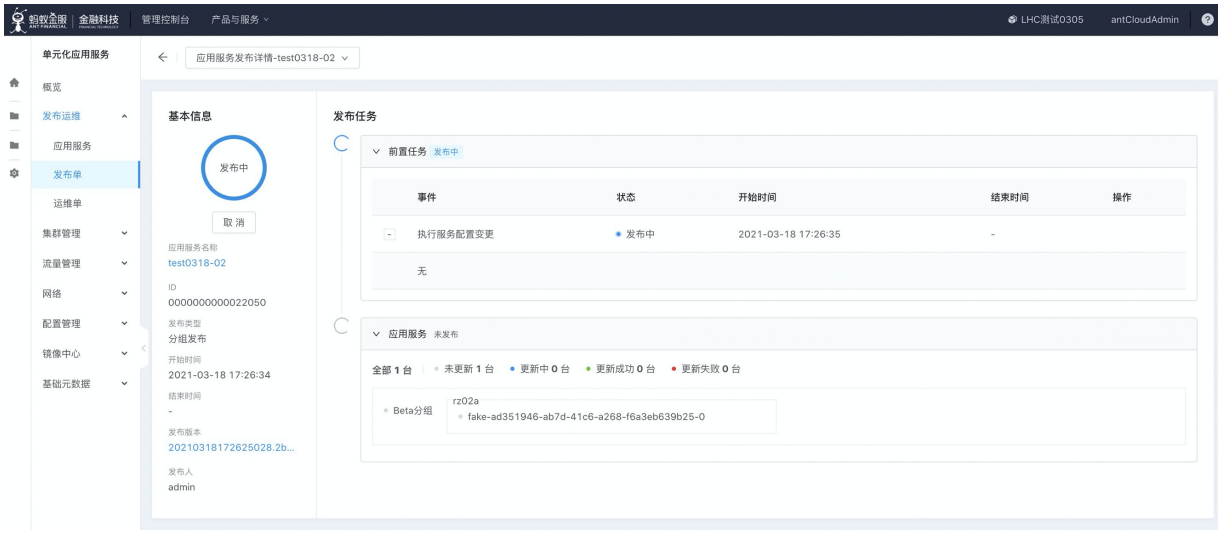
```
# 需要使用业务集群的kubeconfig
kubectl --kubeconfig local.kubeconfig -n namespace get cafedeployment |grep 应用服务名
kubectl --kubeconfig local.kubeconfig -n namespace get inplaceset | grep 应用服务名
```

下图是正常的 cafedeployment 状态。

```
    - rz00a
status:
  availableReplicas: 1
  conditions:
    - last_transition_time: '2021-03-17T15:29:02Z'
      status: 'True'
      type: UnitProvision
    - last_transition_time: '2021-03-19T12:12:05Z'
      status: 'True'
      type: UnitRelease
  currentRevision: test0317jf-cell-rz00a-f88f6f985
  fullyLabeledReplicas: 1
  observedGeneration: 32
  releaseStatus:
    currentBatchType: Normal
    currentPartitions:
      rz00a: 1
    lastTransitionTime: '2021-03-19T12:12:05Z'
    progress: Completed
    updateRevision: test0317jf-cell-rz00a-f88f6f985
  replicas: 1
  scheduledReplicas: 1
  selector: >-
    app.kubernetes.io/instance=test0317jf-rz00a,app.kubernetes.io/name=zxz0710-test001,cafe.sofastack.
io/workspace=lhcpretest0223,cafe.sofastack.io/workspace-group=lhcpretest0223
  unitPodUpgradeDetail:
    unitPodUpgradeDetails:
      rz00a:
        - identity: '0'
          ip: 10.3.40.46
          isReady: true
          isUpdatedRevision: true
          name: test0317jf-cell-rz00a-lh9k9-kgkd5
          uid: 005006a3-c9d6-4a3a-9fd1-3bb11fec27a5
          upgradeProgress: ServiceAvailable
  unitReplicas:
    rz00a: 1
  updatedAvailableReplicas: 1
  updatedReadyReplicas: 1
  updatedReplicas: 1
```

6.4.2.5. 发布失败—前置任务执行服务配置变更失败

【现象】发布失败—前置任务执行服务配置变更失败。



【解决方法】

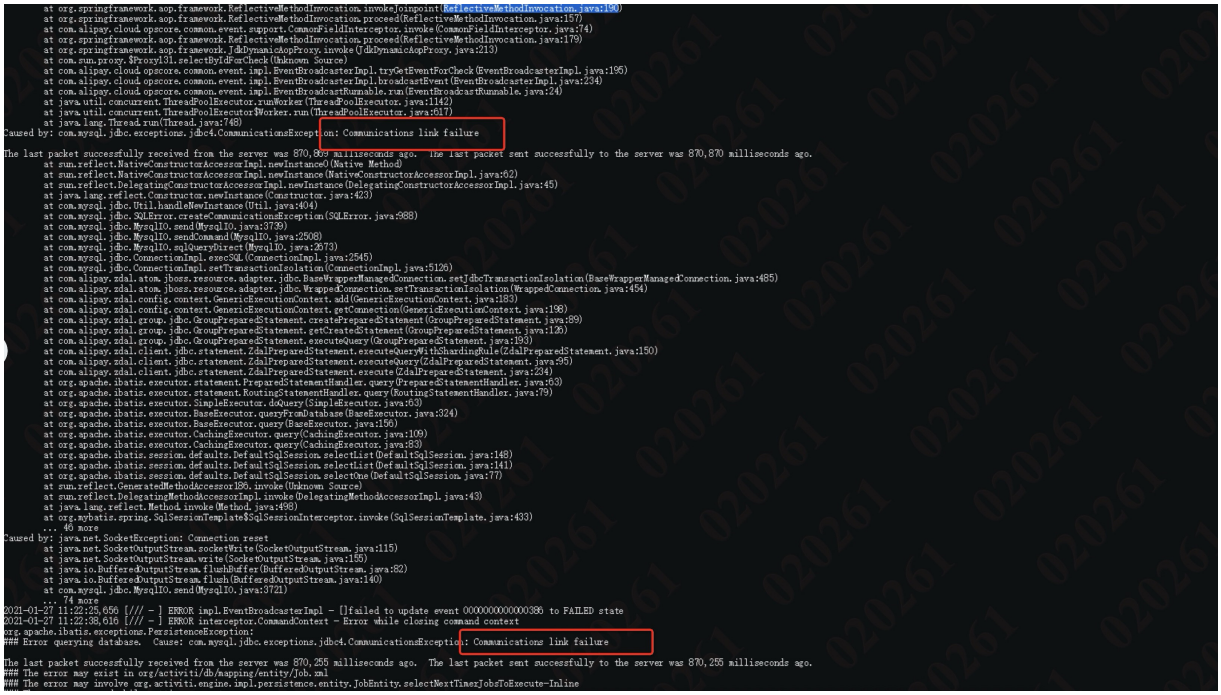
如果发布一直卡在执行服务配置变更处，且无任何日志，需要使用 fed kubeconfig 查看 federatedservicestatus、federatedingressstatus。

> **说明**
>
> 关于如何查找 fed kubeconfig，可参考 如何查找 fed kubeconfig。

```
kubectl --kubeconfig fed.kubeconfig -n namespace getfederatedservicestatus | grep
应用服务名
kubectl --kubeconfig fed.kubeconfig -n namespace getfederatedingressstatus | grep
应用服务名
```

6.4.3. orch 日志报错

【现象】当发布单执行失败时，在 acopswareorchestration 或 opswareorch 应用中发现日志报错“Communications link failure”，如下图所示。



【可能的原因】大概率是当前环境的网络不稳定，导致数据库连接池中的连接被强制断开。

【解决方法】

1. 尝试重启一下 orch。

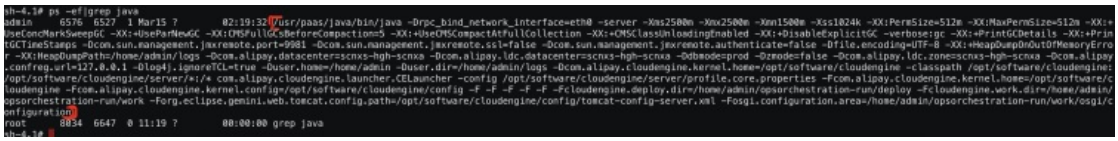
   如果重启后仍常出现此问题，请执行下述步骤。

2. 前往 acopswareorchestration 或 opswareorch 容器中更改配置。

```
cd /home/admin/conf/zdal/acopswareorchestration_1.0_acopswareorchestration_ds

# 备份文件 app.acopswareorchestration_host.json
cp app.acopswareorchestration_host.json app.acopswareorchestration_host.json.bak

# 执行命令修改配置
echo '{
  "connectionProperties":{
    "connectTimeout":"500",
    "socketTimeout":"10000",
    "allowMultiQueries":"true"
  },
  "blockingTimeoutMillis":30000,
  "queryTimeout":0,
  "backgroundValidation":true,
  "prefill":true,
  "configType":"connProps",
  "allowMinPoolTimeout":true,
  "logicDbNameSet":[],
  "testOnBorrow":false,
  "testConnectionOnCheckin":true,
  "backgroundValidationMinutes":10,
  "idleTimeoutMinutes":30,
  "maxConn":80,
  "minConn":10,
  "idleConnectionTestPeriod":60,
  "dbkeyName":"acopswareorchestration_host",
  "validationQueryTimeout":1
}' > app.acopswareorchestration_host.json

# 重启java进程，不能通过删除pod重启
```

3. 重启 Java 进程。

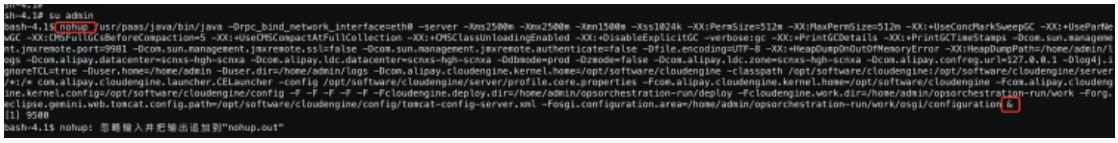
   i. 通过 `ps -ef|grep java` 查询 Java 进程信息，从中获取启动命令（下图中红色括号标出来的部分即为启动命令）。

   ii. 运行 `kill` 命令终止 Java 进程。

   iii. 先切换到 admin 用户，使用 `nohup` 启动 Java：`nohup /usr/paas/java... &` 。

iv. 检查应用是否启动成功。

```
curl localhost:9500/checkService | grep passed:true
```

下图代表成功：

```
bash-4.1$ curl localhost:9500/checkService | grep passed:true
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0passed:true
appname:acopswareorchestration,passed:true
100  5905  100  5905    0     0  1738k      0 --:--:-- --:--:-- --:--:-- 2883k
```

如果仍提示报同样的错误信息，需要更改 acopswareorchestration_host.json 中的 testOnBorrow:true 配置。

6.4.4. orch 内存泄漏分析

【现象】

opswareorch 应用健康报警 CPU 高，并且一直触发 fgc。

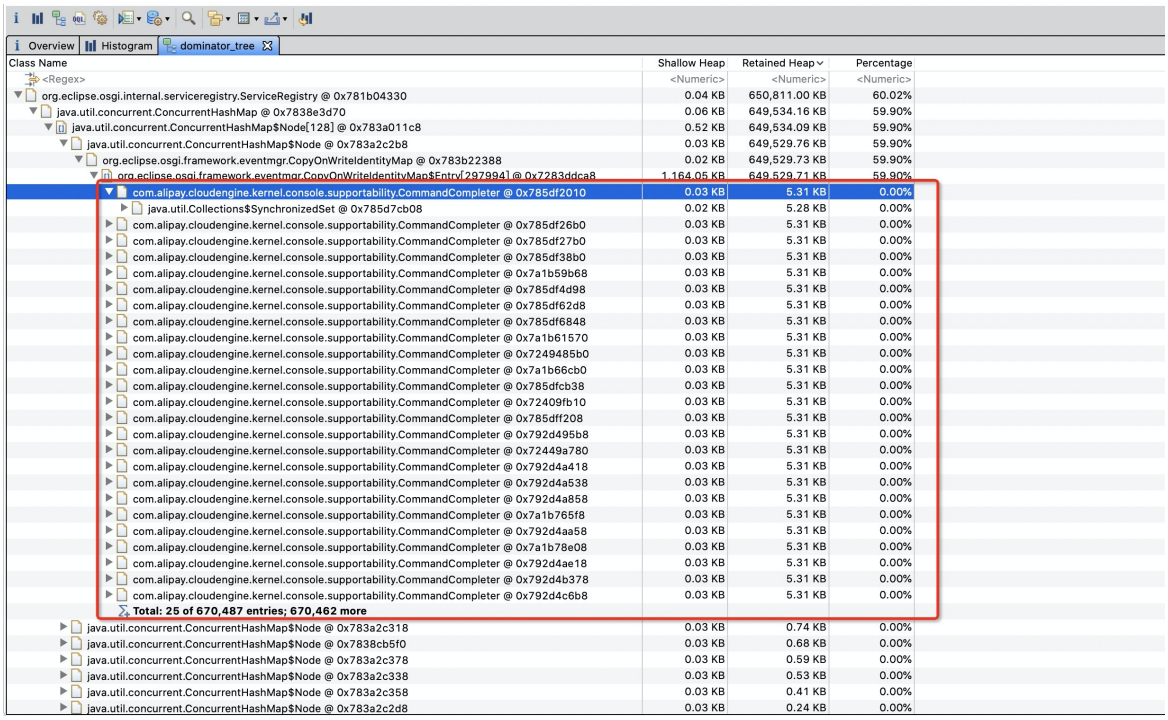
使用 `jstat -gcutil pid` 命令看到 OLD 区使用率已达 100%。

```
sh-4.1# su admin
bash-4.1$ ps -fe | grep java
admin     6248  6217 10 Sep18 ?        10-07:58:25 /usr/paas/java/bin/java -Drpc_bind_network_
-XX:CMSFullGCsBeforeCompaction=5 -XX:+UseCMSCompactAtFullCollection -XX:+CMSClassUnloadingEnab
.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false -Dfile.encoding=UTF-8 -
ode=false -Dcom.alipay.ldc.zone=defaultA -Dcom.alipay.confreg.url=127.0.0.1 -Dlog4j.ignoreTCL=
e/cloudengine:/opt/software/cloudengine/server/*:/* com.alipay.cloudengine.launcher.CELauncher
loudengine.kernel.config=/opt/software/cloudengine/config -F -F -F -F -F -Fcloudengine.deploy.
path=/opt/software/cloudengine/config/tomcat-config-server.xml -Fosgi.configuration.area=/hom
admin    58182 58177  0 19:23 pts/8    00:00:00 grep java
bash-4.1$ jstat -gcutil 6248
  S0     S1     E      O      M     CCS    YGC     YGCT    FGC    FGCT     GCT
  0.00   0.00  70.73 100.00  94.37  89.55   5104 1765.702 140698 235520.055 237285.757
bash-4.1$
```

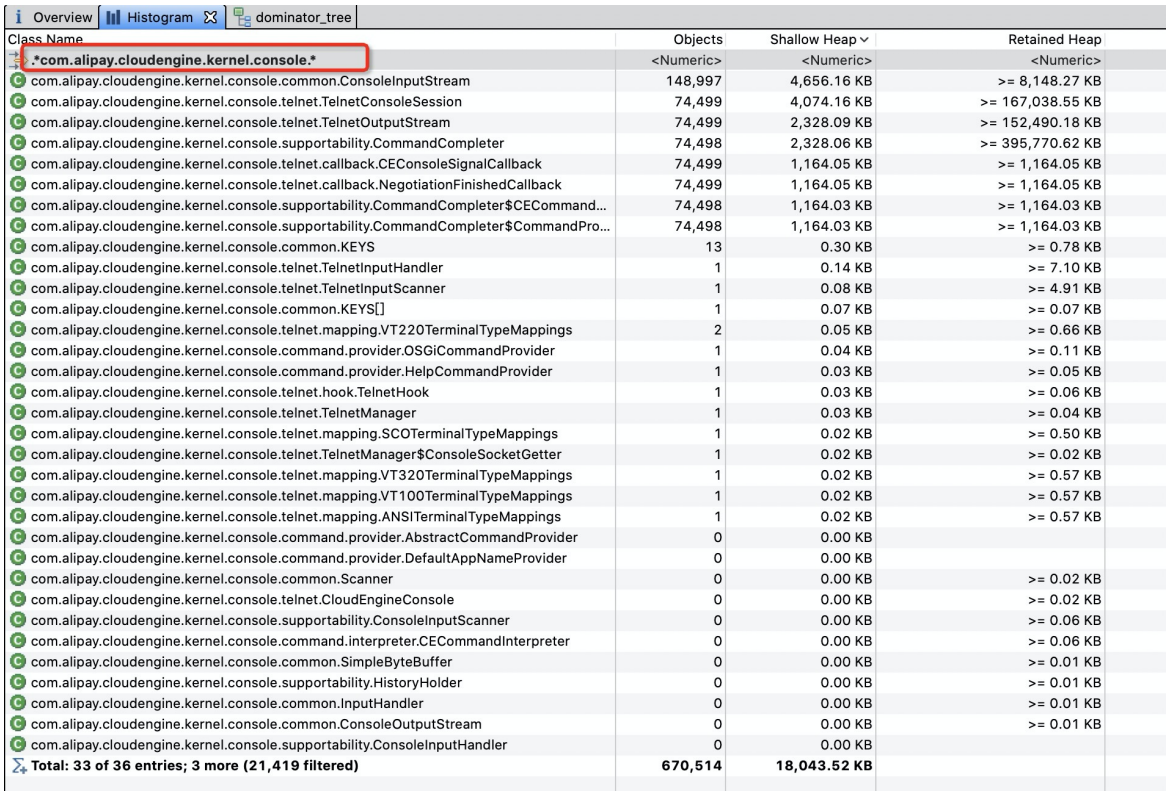
【可能的原因】

Telnet 1234 端口造成内存泄漏。具体分析如下：

1. 使用 `jmap -dump:live,format=b,file=./heapdump.bin pid` 命令 dump jvm 内存。使用 MAT 工具分析结果如下：

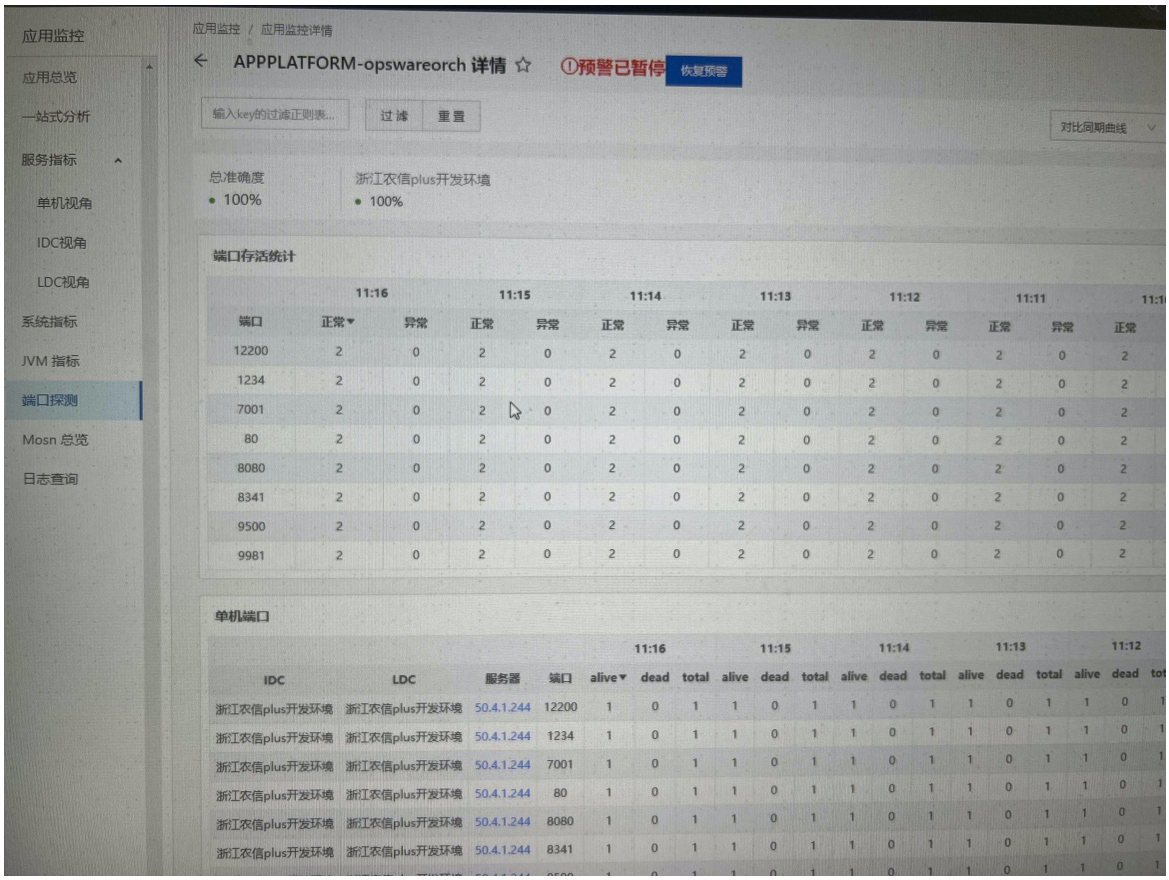


内存大量 `com.alipay.cloudengine.kernel.console` 相关的包，查看 ce 源代码，发现均隶属于 `ce-kernel-console-4.17.4.jar` 包下，这是一个 Cloudengine Telnet 工具。



2. 查看 RMS 监控配置，发现会一直探活 1234 端口，如下图所示。

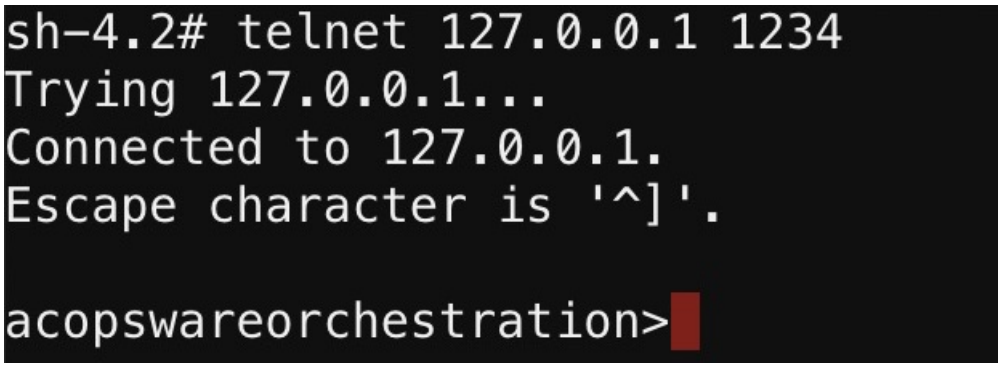
> **说明**
>
> 大致操作路径：RMS 控制台 > 应用监控 > 端口探测。

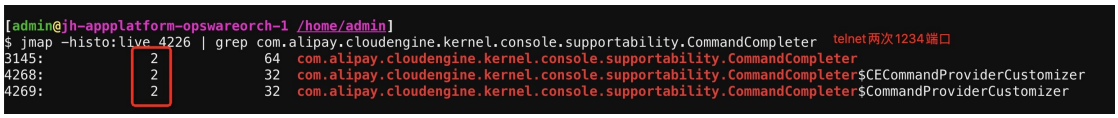


3. 执行如下验证操作。

i. 运行以下命令。

```
# 4226 改成 java 进程 id
jmap -histo:live 4226 | grep com.alipay.cloudengine.kernel.console.supportability.C
ommandCompleter
```





ii. 每使用 Telnet 测试一次 1234 端口，便会新增一个 CommandCompleter 对象，且 gc 无法回收，与预期效果一样。

由此可证是 `telnet 127.0.0.1 1234` 引起的内存泄漏。

【解决方法】

Telnet 1234 端口会造成内存泄漏，因为这只是一个后端端口，业务上无需使用，因此只需将 RMS 中的端口探活关掉即可。

6.5. 镜像构建失败

6.5.1. 镜像构建失败，日志一直处于加载中

【现象】镜像构建失败，日志一直处于加载中。

【解决方法】

1. 在 AKS 的数据库 cloud_config 这张表中查询构建集群。

```
select cloud_env_config from cloud_config;
buildConfig 对应的 masterClusterName字段
```

2. 前往构建集群，在 **工作负载** > **Pods** 查看是够有对应的构建 Pod 创建出来。
   - 如果构建 Pod 的状态是 pending，则查看 Pod 事件。

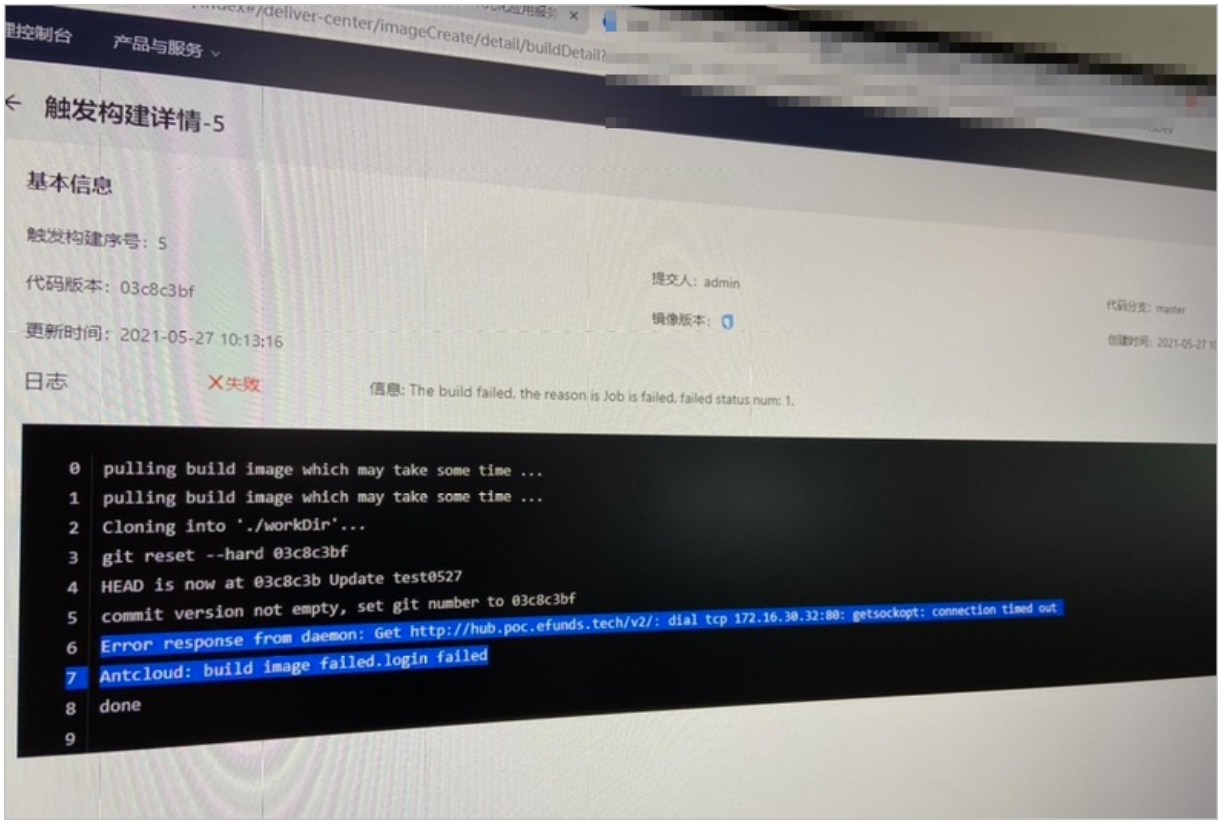
如果提示镜像拉取失败的错误，按错误提示前往 Harbor 镜像中心 push 对应的镜像。

- 如果构建 Pod 的状态是其他状态，则查看 Pod 事件，然后进行对应处理即可。

  如果未找到对应 Pod，则查看 aks-executor 系统是否运行正常。

6.5.2. 镜像构建失败，构建日志显示镜像无法拉取

【现象】镜像构建失败，构建日志显示镜像无法拉取，如下图所示。



【解决方法】

1. 确认 AntStak Plus 底座下 IP 地址是容器 IP 地址或 F5 SLB 地址，云游渲染的 slb 地址为 clusterIp，无法跨集群访问。
2. 在 AKS 的数据库 cloud_config 这张表中，将 `cloud_env_config` 字段的 Json 数据镜像地址字段 `registryAddress` 修改成期望的 IP，其他值保持不变。
3. 备份 cloud_config 数据。
4. 更新 cloud_config 表。

```
update cloud_config set cloud_env_config='' where id='';
```

6.5.3. 镜像构建失败，获取日志超时

【现象】镜像构建失败，获取日志超时。

【解决方法】需确保构建集群和部署 AKS 的中枢 VPC 已实现网络互通。

6.5.4. 镜像构建失败，可以看到构建日志

【现象】镜像构建失败，可以看到构建日志。

【可能的原因】一般是业务 Dockerfile 或业务代码编写方面存在问题。

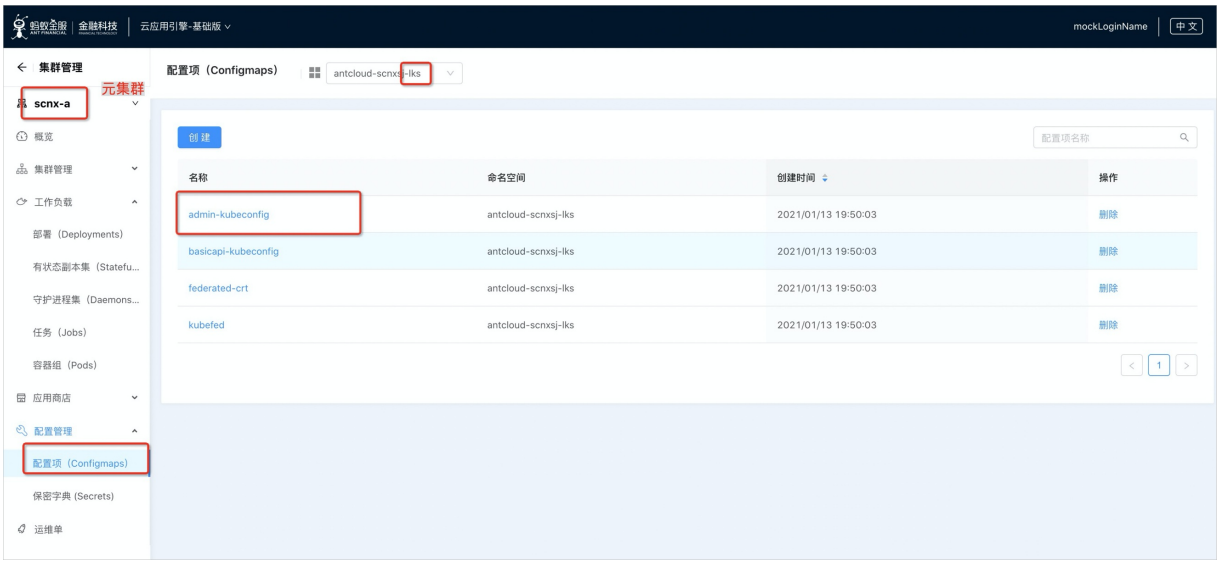
【解决方法】根据日志提示，修改 Dockerfile 或业务代码即可。

6.6. 常见故障排查操作

如何查找 fed kubeconfig

在 Caption 的元集群中，找到 lks 命名空间下的 configmap，admin-kubeconfig 即为 fed 的 kubeconfig。

> 说明
>
> 如果当前无 Captain，请联系对应技术支持人员。



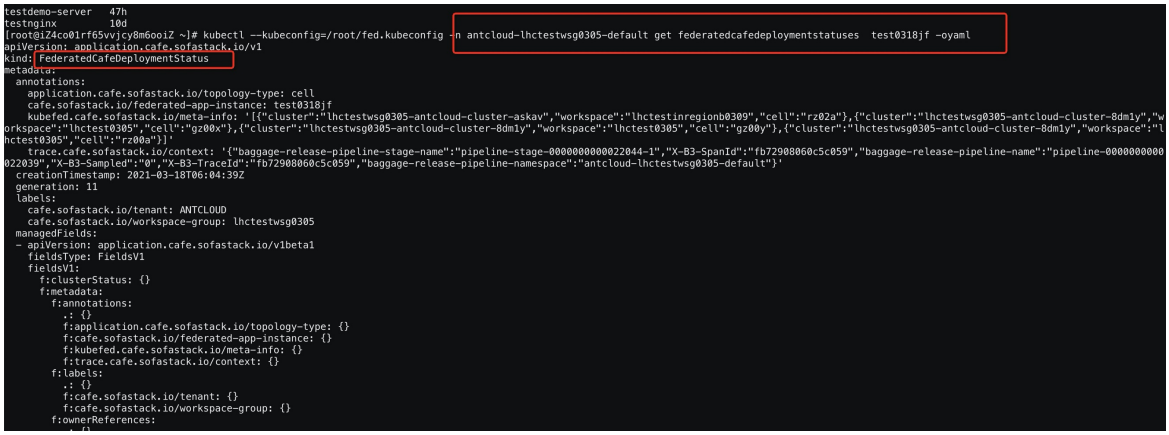
如何查找 fed 资源对象

1. 找到 fed kubeconfig 文件。

   > 说明
   >
   > 关于如何查找 fed kubeconfig，可参考 如何查找 fed kubeconfig。

2. 执行以下命令，获取 fed 资源对象。

```
# 先获取fed kubeconfig文件，替换 namespace 和 应用服务名
kubectl --kubeconfig fed.kubeconfig -n namespace get federatedcafedeploymentstatuses | grep 应用服务名

# 获取 fed cafed
kubectl --kubeconfig fed.kubeconfig -n namespace get federatedcafedeployment | grep 应用服务名

# 获取 fed service 和 ingress
kubectl --kubeconfig fed.kubeconfig -n namespace get federatedservices | grep 应用服务名
kubectl --kubeconfig fed.kubeconfig -n namespace get federatedingresses | grep 应用服务名

# 获取 fed service 和 ingress status
kubectl --kubeconfig fed.kubeconfig -n namespace get federatedservicestatus | grep 应用服务名
kubectl --kubeconfig fed.kubeconfig -n namespace get federatedingressstatus | grep 应用服务名
```



如何查找 cafedeployment

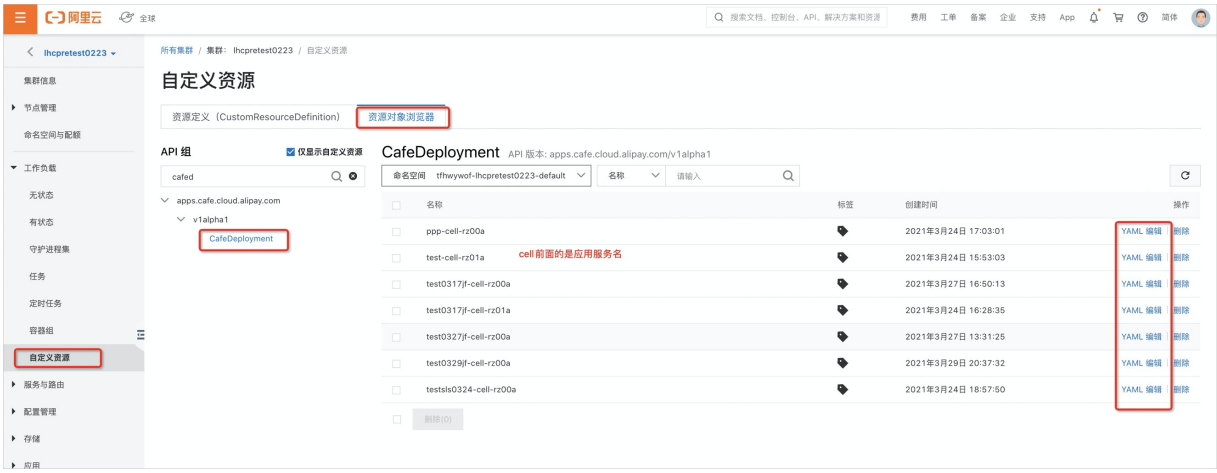
1. 获取业务集群的 kubeconfig 文件。

   因为 cafedeployment 是部署在业务集群中，所以需要获取 kubeconfig 文件。

2. 执行以下命令，获取 cafedeployment。

```
kubectl --kubeconfig local.kubeconfig -n namespace get cafedeployment | grep 应用服务名
kubectl --kubeconfig local.kubeconfig -n namespace get inplaceset | grep 应用服务名
```

如果是阿里云 ACK 集群，可以直接在阿里云 ACK 容器服务控制台查看 cafedeployment，如下图所示。

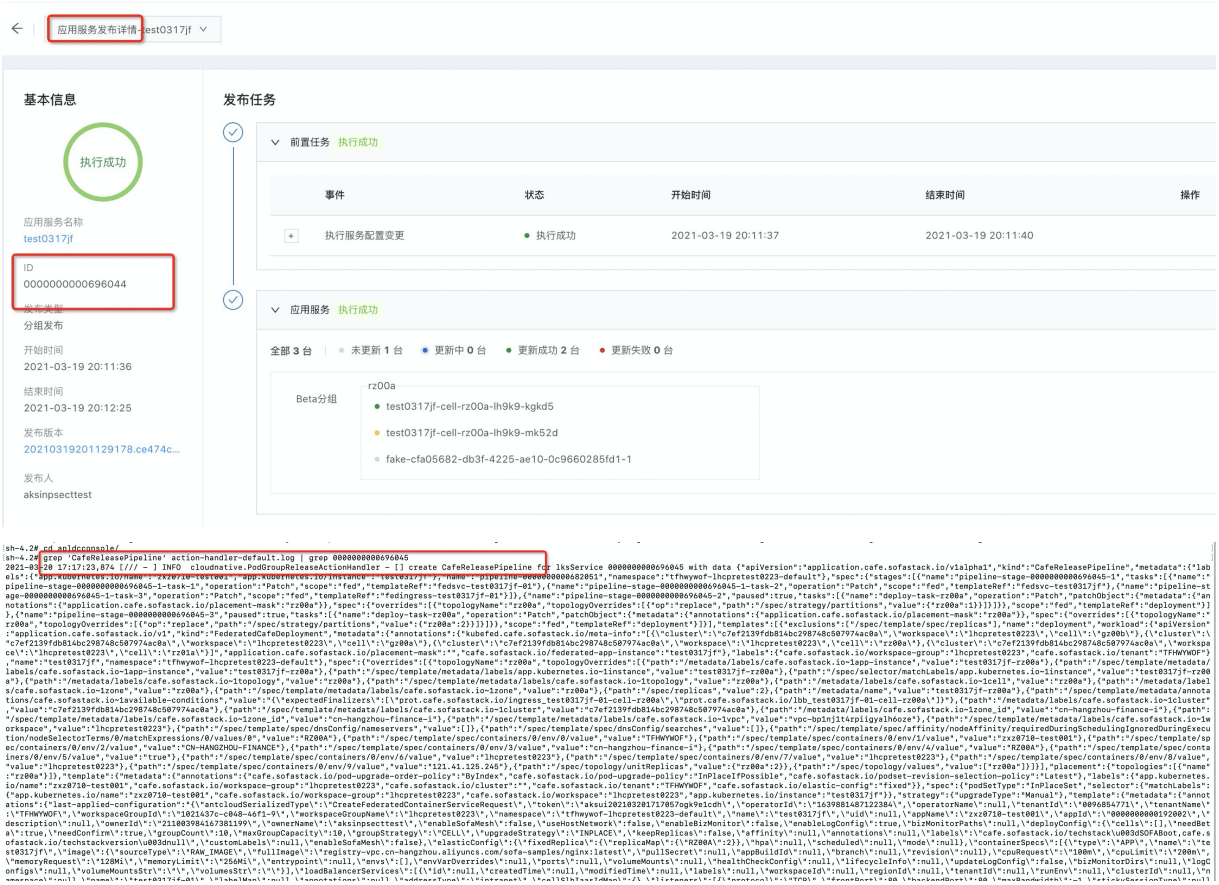


如何查找 releasePipeline

releasePipeline 描述了单个应用服务发布单的详细信息，还包含向 fed 层下发的 template 信息。

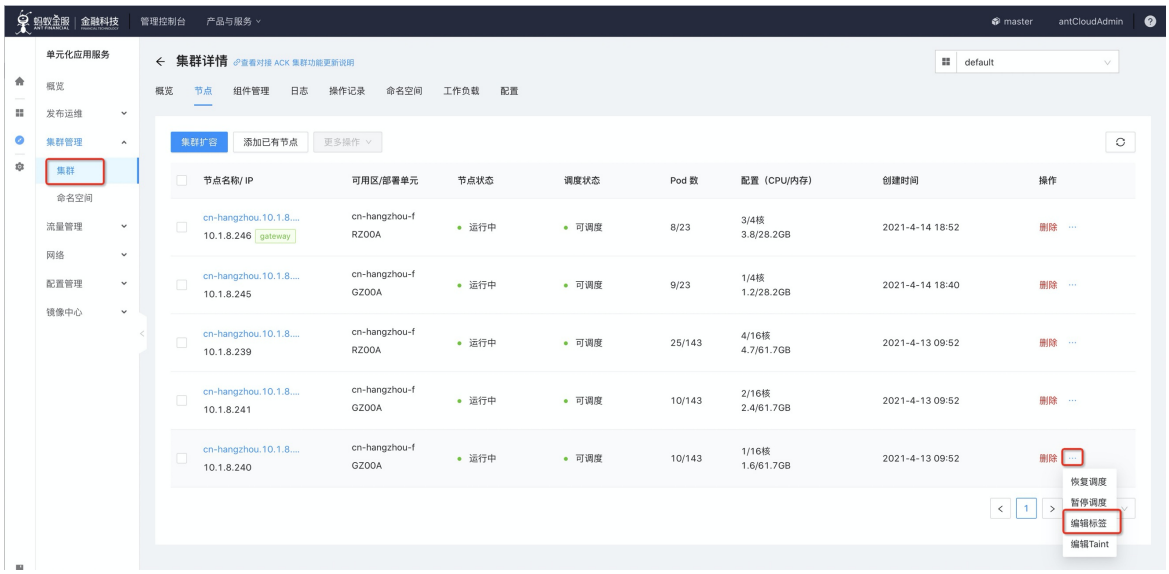
```
# 在 lhc 里面 /home/admin/logs/apldcconsole 目录执行下面的命令

# 发布单的 releasePipeline
grep 'CafeReleasePipeline' action-handler-default.log | grep ${service_id}

# 运维单的 releasePipeline
grep 'createCafeRebootPipeline' action-handler-default.log | grep ${service_id}
```

上述命令中用到的 `service_id` 可以在应用服务发布单详情页查看，如下图所示。

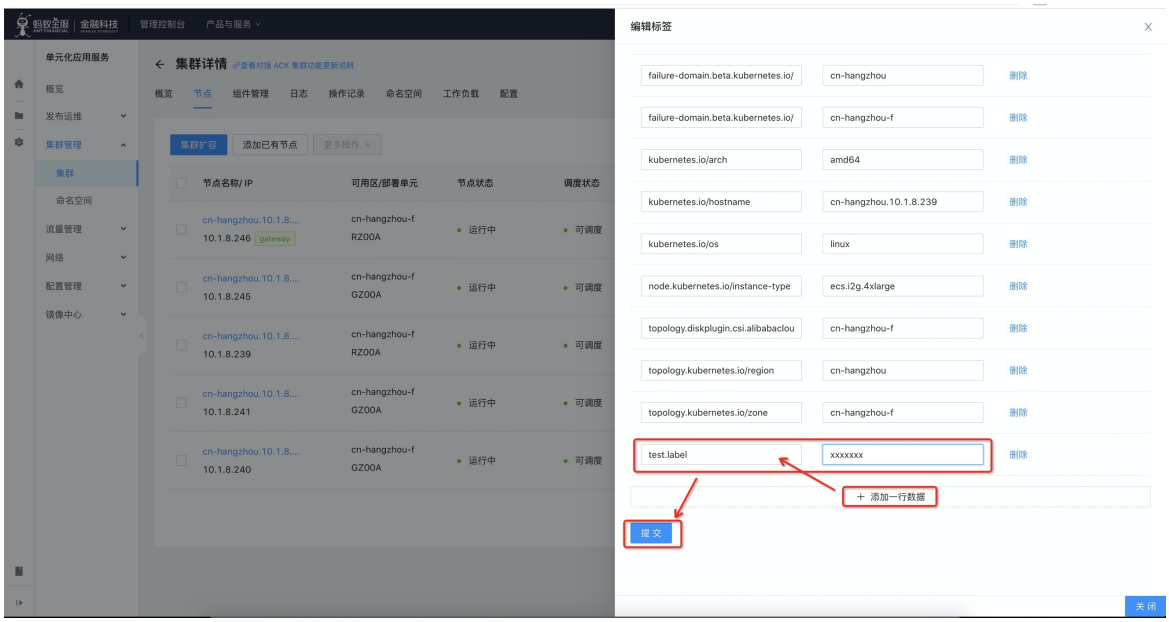


如何为集群 node 打标

1. 登录单元化应用服务控制台。
2. 单击 **集群管理** > **集群**，进入集群管理页面。
3. 找到需要打标的集群节点，单击操作列下的更多图标 > **编辑标签**。



4. 在 **编辑标签** 页增加标签。

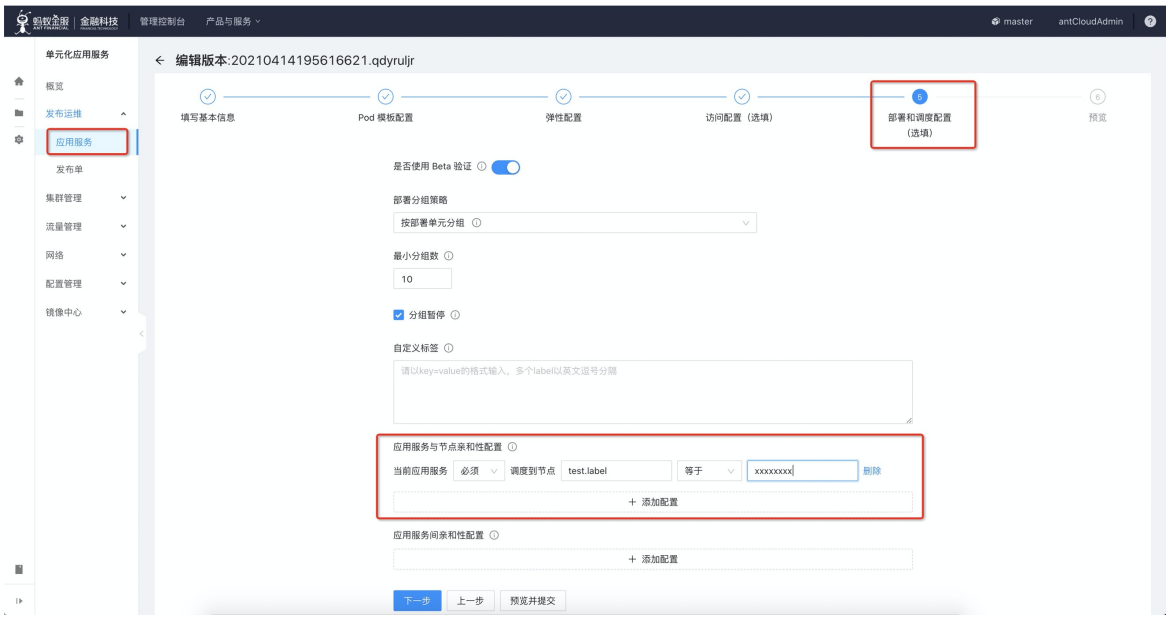


key、value 规则如下。

- **key**：由字母、数字、中划线（-）、下划线（_）、小数点（.）组成，且必须以字母或数字开头和结尾，不得超过 63 个字符。
- **value**：可以为空字符串或由字母、数字、中划线（-）、下划线（_）、小数点（.）组成，且必须以字母或数字开头和结尾，不得超过 63个字符。

如何使应用 Pod 落在指定标签 node 上

1. 在创建或编辑应用服务的 **部署和调度配置** 这一步，设置 **应用服务与节点亲和性配置**。



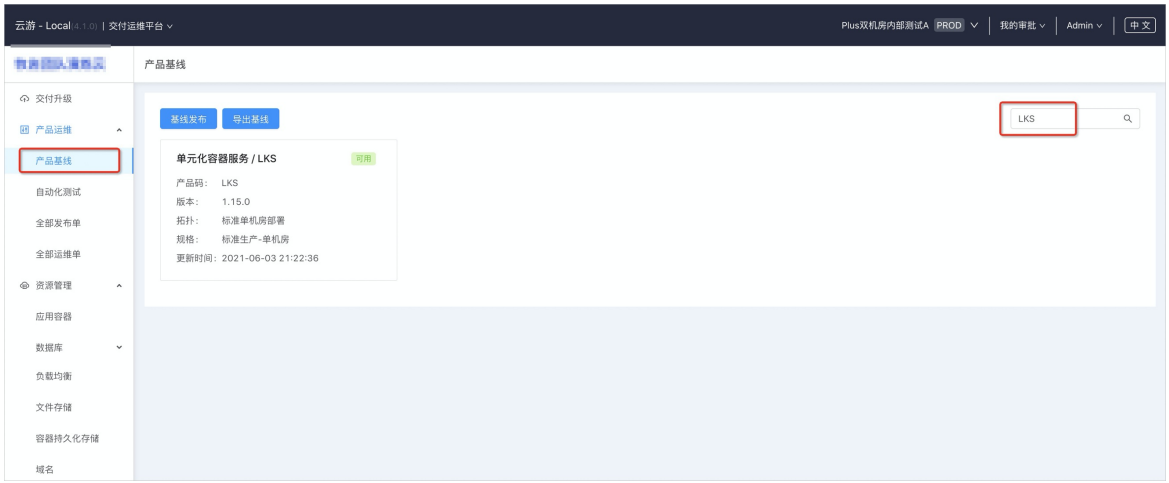
2. 提交之后，重新发布应用服务。

   应用 Pod 便会落在指定标签的 node 上。

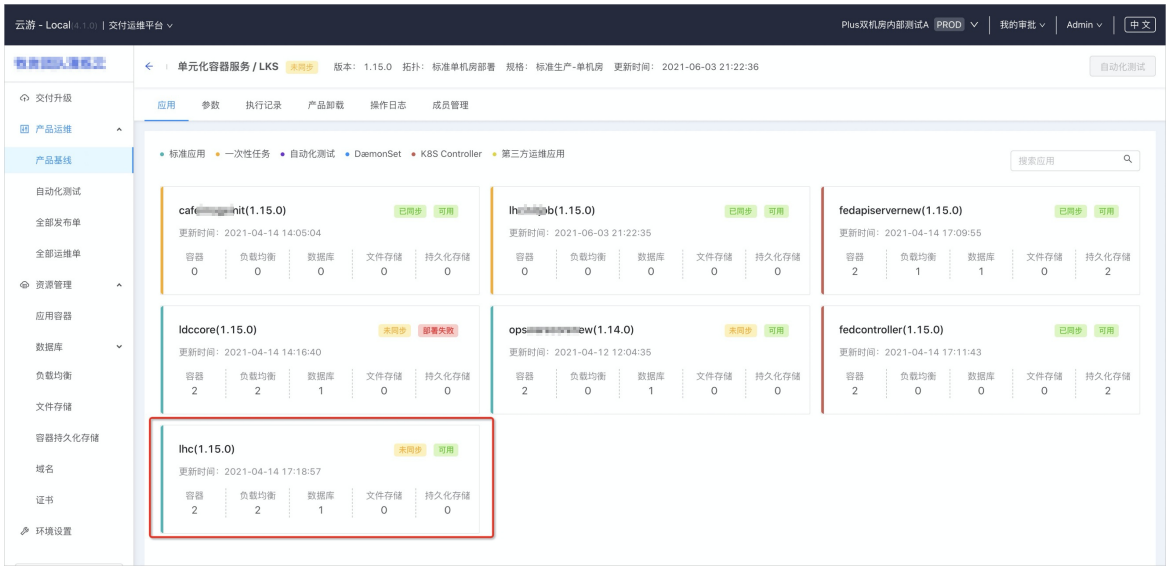
> 注意
>
> - 亲和性需要 Pod 重新创建才会生效。因此，在原地升级时，Pod 不会重新调度让亲和性生效
> - `cafe.sofastack.io/cell` 为系统保留调度标签，不允许使用。

如何查找 LHC 日志

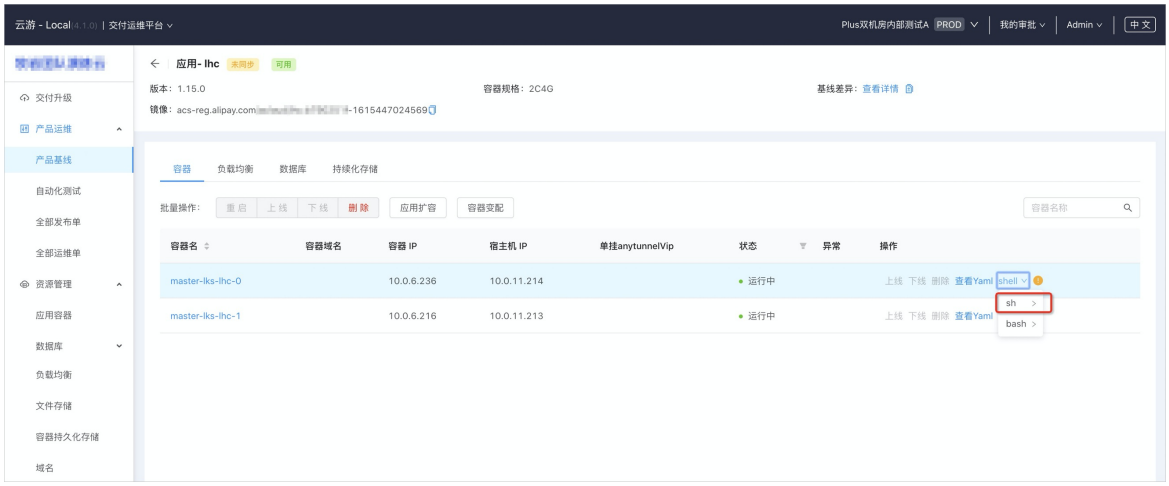
1. 登录云游 Local。
2. 在左侧导航栏上，单击 **产品基线**，在搜索框中输入要查找的产品码 **LKS**。



3. 进入产品详情页面，找到 lhc 应用，如下图所示。



4. 找到 lhc 容器，单击 **操作** 列下的 **shell > sh** 进入 webshell。



5. 进入容器之后，通过 cd 跳转到 `/home/admin/logs/apldcconsole/` 目录，查看 common-error.log 或 container-service-error.log 是否有错误日志。

```
sh-4.2# cd /home/admin/logs/apldcconsole/
sh-4.2# ls -l | grep error
-rw-r--r-- 1 admin admin        0 6月   4 00:14 cluster-manager-error.log
-rw-r--r-- 1 admin admin   114934 6月   4 00:21 common-error.log
-rw-r--r-- 1 admin admin   167225 6月   4 10:20 container-service-error.log
-rw-r--r-- 1 admin admin        0 6月   4 00:14 data-init-error.log
-rw-r--r-- 1 admin admin        0 6月   4 00:14 data-inspect-error.log
-rw-r--r-- 1 admin admin        0 6月   4 00:14 facade-request-error.log
sh-4.2#
```

如何查找 opswarecorenew 日志

方法与查找 LHC 日志的类似，可参考 如何查找 LHC 日志。

找到 opswarecorenew 这个应用，进入容器，重点查看 common-error.log 或 app-default.log 这两个日志。

单元化工作空间中多集群容器引擎使用消息功能正常，但容器应用服务使用消息功能异常

【现象】

在 SOFAStack 单元化工作空间中，多集群容器引擎使用消息功能正常，但容器应用服务使用消息功能异常。

【可能的原因】

同时使用了容器应用服务和多集群容器引擎产品。

【解决方法】

在 SOFAStack 单元化工作空间中，容器应用服务和多集群容器引擎产品不可同时使用。即，在单元化工作空间中不要执行任何容器应用服务相关的操作。

7.扩容和缩容

操作场景

参考监控和告警章节，当 CPU、内存、磁盘出现异常告警，超过水位后，可进行扩容操作。

操作入口

如需扩缩容，前往云游 Local，可参考下图。

操作说明

应用类别		扩容说明
无状态应用	apaks	支持直接扩容
	apaksexecutor	支持直接扩容
	metaservice	支持直接扩容
	basicapiserver	支持直接扩容
hub	输出非本地存储拓扑 说明 **本地或非本地存储**：一般指镜像存储设置，参考配置中 storage-type 的设置，如果为 S3 或 OSS，即为非本地存储。	支持直接扩容
	输出为本地存储拓扑	暂不支持扩容操作。 如需扩容，联系技术支持人员。

8.配置参考

目前各应用不支持通过参数修改 JVM 配置来修改系统性能，您可通过云游运维平台进行组件的扩容操作。

9.日志参考

9.1. 概述

在运维过程中，您可以通过日志了解 LHC 各组件的状态或定位故障。

CAFE 包括如下组件日志：

- apaks
- aksexecutor
- hub
- metaservice
- lhc
- fedapiserver
- fedcontroller
- kubecontroller
- opswarecorenew
- opswareorch
- ldccore

CAFE 会根据既定策略将这些日志进行归档和清理。

9.2. 日志名称

9.2.1. apaks 日志文件名和说明

apaks 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
app-default.log	/home/admin/logs/apaks	运维日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

文件名	日志文件路径	说明	归档策略
common-error.log	/home/admin/logs/apaks	业务异常日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
sofa-default.log	/home/admin/logs/sofa-runtime	应用启动日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-client-digest.log	/home/admin/logs/tracelog	rpc 调用日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-server-digest.log	/home/admin/logs/tracelog	rpc 响应日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
access.log	/home/admin/logs/nginx/cronolog/{yyyy}/{MM}/	Nginx 请求日志	• 每日一个新文件 • 新文件的名称中会加日期前缀：{yyyy-MM-dd}- access.log • 最多保留 7 天的日志文件

文件名	日志文件路径	说明	归档策略
error.log	/home/admin/logs/nginx/cronolog/{yyyy}/{MM}/	Nignx 异常日志	• 每日一个新文件 • 新文件的名称中会加日期前缀：{yyyy-MM-dd}- error.log • 最多保留 7 天的日志文件

9.2.2. aksexecutor 日志文件名和说明

aksexecutor 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
app-default.log	/home/admin/logs/apakse xecutor/app-default.log	运维日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
common-error.log	/home/admin/logs/apakse xecutor	业务异常日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
sofa-default.log	/home/admin/logs/sofa-runtime	应用启动日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-client-digest.log	/home/admin/logs/tracelo g	rpc 调用日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

文件名	日志文件路径	说明	归档策略
rpc-server-digest.log	/home/admin/logs/tracelog	rpc 响应日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

9.2.3. hub 日志文件名和说明

hub 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
ui.log	/tmp	运维日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
registry.log	/tmp	业务异常日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
nginx.log	/tmp	Nginx 请求日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
error.log	/tmp /nginx/cronolog/{yyyy}/{MM}/	Nginx 异常日志	• 每日一个新文件 • 新文件的名称中会加日期前缀：{yyyy-MM-dd}- error.log • 最多保留 7 天的日志文件

9.2.4. metaservice 日志文件名和说明

metaservice 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
app-default.log	/home/admin/logs/acmetaservice	运维日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
common-error.log	/home/admin/logs/acmetaservice	业务异常日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
sofa-default.log	/home/admin/logs/sofa-runtime	应用启动日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-client-digest.log	/home/admin/logs/tracelog	rpc 调用日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-server-digest.log	/home/admin/logs/tracelog	rpc 响应日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

文件名	日志文件路径	说明	归档策略
access.log	/home/admin/logs/nginx/cronolog/{yyyy}/{MM}/	Nginx 请求日志	• 每日一个新文件 • 新文件的名称中会加日期前缀：{yyyy-MM-dd}- access.log • 最多保留 7 天的日志文件
error.log	/home/admin/logs/nginx/cronolog/{yyyy}/{MM}/	Nginx 异常日志	• 每日一个新文件 • 新文件的名称中会加日期前缀：{yyyy-MM-dd}- error.log • 最多保留 7 天的日志文件

9.2.5. lhc 日志文件名和说明

lhc 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
app-default.log	/home/admin/logs/apldcconsole	运维日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
common-error.log	/home/admin/logs/apldcconsole	业务异常日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

文件名	日志文件路径	说明	归档策略
sofa-default.log	/home/admin/logs/sofa-runtime	应用启动日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-client-digest.log	/home/admin/logs/tracelog	rpc 调用日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-server-digest.log	/home/admin/logs/tracelog	rpc 响应日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
access.log	/home/admin/logs/nginx/cronolog/{yyyy}/{MM}/	Nginx 请求日志	• 每日一个新文件 • 新文件的名称中会加日期前缀：{yyyy-MM-dd}- access.log • 最多保留 7 天的日志文件
error.log	/home/admin/logs/nginx/cronolog/{yyyy}/{MM}/	Nignx 异常日志	• 每日一个新文件 • 新文件的名称中会加日期前缀：{yyyy-MM-dd}- error.log • 最多保留 7 天的日志文件

9.2.6. fedapiserver 日志文件名和说明

fedapiserver 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
kubefed-apiserver.log.INFO	/logs	fedapiserver 组件的全量日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.{yyyy-MM-dd-00} • 最多保留 7 天的日志文件
kubefed-apiserver.log.WARNING	/logs	仅包含 fedapiserver 组件警告和错误的日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.{yyyy-MM-dd-00} • 最多保留 7 天的日志文件
kubefed-apiserver.log.ERROR	/logs	仅包含 fedapiserver 组件错误的日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.{yyyy-MM-dd-00} • 最多保留 7 天的日志文件
kine.log	/logs	kine 组件（负责处理 fedapiserver 的数据库请求）的全量日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.{yyyy-MM-dd-00} • 最多保留 7 天的日志文件

9.2.7. fedcontroller 日志文件名和说明

fedcontroller 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略

文件名	日志文件路径	说明	归档策略
kubefed-controller-manager.log.INFO	/logs	fedcontroller 组件的全量日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.{yyyy-MM-dd-00} • 最多保留 7 天的日志文件
kubefed-controller-manager.log.WARNING	/logs	仅包含 fedcontroller 组件警告和错误的日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.{yyyy-MM-dd-00} • 最多保留 7 天的日志文件
kubefed-controller-manager.log.ERROR	/logs	仅包含 fedcontroller 组件错误的日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.{yyyy-MM-dd-00} • 最多保留 7 天的日志文件

9.2.8. kubecontroller 日志文件名和说明

kubecontroller 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
kube-controller-manager.log	/logs	kubecontroller 组件的全量日志	最多保留 5GB 大小的最近日志数据

9.2.9. opswarecorenew 日志文件名和说明

opswarecorenew 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
app-default.log	/home/admin/logs/opswarecore	运维日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
common-error.log	/home/admin/logs/opswarecore	业务异常日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
sofa-default.log	/home/admin/logs/sofa-runtime	应用启动日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-client-digest.log	/home/admin/logs/tracelog	rpc 调用日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-server-digest.log	/home/admin/logs/tracelog	rpc 响应日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

9.2.10. opswareorch 日志文件名和说明

opswareorch 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
app-default.log	/home/admin/logs/acopswareorchestration	运维日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
common-error.log	/home/admin/logs/acopswareorchestration	业务异常日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-client-digest.log	/home/admin/logs/tracelog	rpc 调用日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-server-digest.log	/home/admin/logs/tracelog	rpc 响应日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

9.2.11. ldccore 日志文件名和说明

ldccore 日志文件名和说明如下表所示。

文件名	日志文件路径	说明	归档策略
zonemng-default.log	/home/admin/logs/zonemng/	运维日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

文件名	日志文件路径	说明	归档策略
common-error.log	/home/admin/log/zonemng/	业务异常日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-client-digest.log	/home/admin/logs/tracelog	rpc 调用日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件
rpc-server-digest.log	/home/admin/logs/tracelog	rpc 响应日志	• 每日一个新文件 • 旧文件的名称中会加日期后缀：名称.log.{yyyy-MM-dd} • 最多保留 7 天的日志文件

10.错误码参考

下表中列出 CAFE 常见错误码。

应用	错误码	常见原因	问题描述	处理方法
apaks	WORKSPACE_NOT_FOUND	参数不对	未找到指定的工作空间	核对工作空间名称是否正确并完整
apaks	APP_NOT_FOUND	参数不正确	未找到指定名字的应用。	核对应用名称是否正确并完整并确认目标租户是否正确
apaks	CONTAINER_SERVICE_NOT_EXISTS	参数不正确	指定的容器服务实例不存在	确认输入的服务实例是否存在以及名称拼写是否正确
apaks	CONTAINER_SERVICE_EXISTS	重名	指定名称的 container_service 已经存在	核对名称的正确性
apaks	MAIN_CONTAINER_OVERFLOW	参数不正常	主容器超过一个	确定容器配置
apaks	MAIN_CONTAINER_NOT_FOUND	填写的配置不正常	没有主容器配置	查看容器配置
lhc	DUPLICATED_NAME	参数不正确	应用服务编码重复	更改应用服务编码
lhc	CLUSTER_NOT_READY	系统配置不正确	cell 绑定的集群不可用	检查集群状态是否正常
lhc	CELL_NOT_BIND	系统配置不正确	cell 未绑定集群	检查是否为集群 node 打 cell 标
lhc	EXCEEDED_QUOTA	系统配置不正确	超出集群配额限制	检查命名空间的配额限制

应用	错误码	常见原因	问题描述	处理方法
lhc	APP_NOT_FOUND	参数不正确	未找到指定名字的应用	核对应用名称是否正确并完整 并确认目标租户是否正确
lhc	124	发布单冲突	发布单冲突	取消或完成冲突的发布

11.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	-	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： • 同一个应用在每个单元中拥有独立使用的资源。 • 同一个应用的业务在不同单元中按水平方向拆分。 • 不同单元处理的业务分片不重叠。
单元化架构	-	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。 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。 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。 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。