

DaoCloud Enterprise

DaoCloud Enterprise Product Documentation

Table of contents

1. 数据服务与 Kubernetes 版本适配	3
1.1 Kubernetes 版本适配情况	4
2. 首次进入中间件数据服务	7
3. 数据服务（中间件）权限设计说明	12
4. 数据服务介绍	16
4.1 数据服务学习路径	16
5. Elasticsearch	17
5.1 Elasticsearch 索引服务 Release Notes	17
5.2 Intro	20
5.3 User guide	25
6. Kafka	45
6.1 Kafka 消息队列 Release Notes	45
6.2 Intro	47
6.3 User guide	53
7. Minio	72
7.1 MinIO 对象存储 Release Notes	72
7.2 Intro	74
7.3 User guide	81
8. Mysql	105
8.1 MySQL Release Notes	105
8.2 Faq	108
8.3 Intro	114
8.4 User guide	119
9. Rabbitmq	130
9.1 RabbitMQ Release Notes	130
9.2 Intro	132
9.3 User guide	139
10. Redis	161
10.1 Redis 缓存服务 Release Notes	161
10.2 Intro	163
10.3 User guide	165

1. 数据服务与 Kubernetes 版本适配

目前数据服务已正式发布了 RabbitMQ、Elasticsearch、MySQL、Redis、Kafka、MinIO 六个数据服中间件。

1.1 Kubernetes 版本适配情况

中间件	版本	功能	1.24	1.23	1.22	1.21	1.20	备注
RabbitMq	0.6.2-24-gb25cc385	operator	✓	✓	✓	✓	✓	
		创建实例	✓	✓	✓	✓	✓	
		编辑实例	✓	✓	✓	✓	✓	
		查询实例	✓	✓	✓	✓	✓	
		实例连接	✓	✓	✓	✓	✓	
		删除实例	✓	✓	✓	✓	✓	
ElasticSearch	0.3.4-16-g90e5ad9	operator	✓	✓	✓	✓	✓	
		创建实例	✓	✓	✓	✓	✓	
		编辑实例	✓	✓	✓	✓	✓	
		查询实例	✓	✓	✓	✓	✓	
		实例连接	✓	✓	✓	✓	✓	
		删除实例	✓	✓	✓	✓	✓	
MySQL	0.3.1-90-gb98cd99	operator	✓	✓	✓	✓	✗	policy/v1
		创建实例	✓	✓	✓	✓	✓	
		编辑实例	✓	✓	✓	✓	✓	
		查询实例	✓	✓	✓	✓	✓	
		实例连接	✓	✓	✓	✓	✓	
		删除实例	✓	✓	✓	✓	✓	
Redis	0.3.0-19-g706a21c	operator	✓	✓	✓	✓	✓	
		创建实例	✓	✓	✓	✓	✓	
		编辑实例	✓	✓	✓	✓	✓	
		查询实例	✓	✓	✓	✓	✓	
		实例连接	✓	✓	✓	✓	✓	
		删除实例	✓	✓	✓	✓	✓	
Kafka	0.1.8-16-g0387939	operator	✓	✓	✓	✓	✓	
		创建实例	✓	✓	✓	✓	✓	
		编辑实例	✓	✓	✓	✓	✓	
		查询实例	✓	✓	✓	✓	✓	
		实例连接	✓	✓	✓	✓	✓	
		删除实例	✓	✓	✓	✓	✓	
MinIO	0.1.5-15-g2adfc58	operator	✓	✓	✓	✓	✓	
		创建实例	✓	✓	✓	✓	✓	

中间件	版本	功能	1.24	1.23	1.22	1.21	1.20	备注
		编辑实例	✓	✓	✓	✓	✓	
		查询实例	✓	✓	✓	✓	✓	
		实例连接	✓	✓	✓	✓	✓	
		删除实例	✓	✓	✓	✓	✓	

2. 首次进入中间件数据服务

首次使用 DCE 5.0 的 [Elasticsearch 搜索服务](#)、[Kafka 消息队列](#)、[MinIO 对象存储](#)、[MySQL 数据库](#)、[RabbitMQ 消息队列](#)，或 [Redis 缓存服务组件](#)时，需要选择工作空间。有关工作空间的概念介绍，可参考[工作空间与层级](#)。

下面以 MinIO 为例介绍如何选择工作空间，其他数据服务组件同理。

2. 首次进入中间件数据服务

1. 在左侧导航栏中选择 中间件 -> MinIO 存储。

The screenshot shows the DaoCloud management console. On the left, there is a navigation sidebar with various links such as Overview, Dashboard, Workstation, Application Workstation, Container Management, Multi-Cloud Deployment, Image Registry, Microservices, Observability, Service Mesh, Data Services, and Intermediate Components. The 'Intermediate Components' link is highlighted with a red box. In the main content area, it says 'MinIO 对象存储' (MinIO Object Storage) and 'mcamel'. Below this is the MinIO logo and a brief description: 'MinIO 是一个基于 Apache License v2.0 开源协议的对象存储服务。' (MinIO is an open-source object storage service based on the Apache License v2.0). A blue button labeled '立即部署' (Deploy Now) is visible. The background of the main area has a large watermark of the MinIO logo.

2. 在弹窗中选择一个工作空间后，点击 确认。

The screenshot shows a 'Workspace Selection' dialog box overlaid on the main MinIO interface. The dialog has a search bar at the top and a table below it. The table lists workspaces with columns for ID, Workspace, and Name. The 'mcamel-ws' workspace is selected (radio button is checked). At the bottom right of the dialog is a blue 'Confirm' button.

ID	Workspace	Name
2	Default	
4	kpanda-ws-test001	
5	mcamel-ws	
6	frank-ws	
8	dsadasd	
9	pokemon	
10	skoala	

!!! note

如果未出现弹窗/想要切换工作空间，可手动点击红框中的切换图标选择新的工作空间。

3. 初次使用时，可以点击[立即部署](#)来创建 MinIO 实例。



3. 数据服务（中间件）权限设计说明

数据服务模块建立在 [工作空间](#) 之上，数据服务的各个中间件模块，对应工作空空间各个模块权限映射如下：

中间件模块	菜单对象	操作	Workspace Admin	Workspace Editor	Workspace Viewer
MySQL	MySQL 实例列表	查看列表	✓	✓	✓
		实例名称搜索	✓	✓	✓
		创建实例	✓	✓	✗
		更新实例配置	✓	✓	✗
		删除实例	✓	✗	✗
	MySQL 实例详情	实例概览	✓	✓	✓
		实例监控	✓	✓	✓
		查看实例配置参数	✓	✓	✓
		修改实例配置参数	✓	✓	✗
		查看实例访问密码	✓	✓	✗
	备份配置管理	查看实例备份列表	✓	✓	✓
		实例创建备份	✓	✓	✗
		实例修改自动备份任务	✓	✓	✗
		使用备份创建新实例	✓	✓	✗
		备份配置列表	✓	✗	✗
RabbitMQ	RabbitMQ 实例列表	创建备份配置	✓	✗	✗
		修改备份配置	✓	✗	✗
		删除备份配置	✓	✗	✗
		查看列表	✓	✓	✓
		实例名称搜索	✓	✓	✓
	RabbitMQ 实例详情	创建实例	✓	✓	✗
		更新实例配置	✓	✓	✗
		删除实例	✓	✗	✗
		实例概览	✓	✓	✓
		实例监控	✓	✓	✓
Elasticsearch	查看列表		✓	✓	✓

Elasticsearch 实例 列表		实例名称搜索	✓	✓	✓
	创建实例	✓	✓	✓	✗
	更新实例配置	✓	✓	✓	✗
	删除实例	✓	✗	✗	✗
Elasticsearch 实例 详情	实例概览	✓	✓	✓	✓
	实例监控	✓	✓	✓	✓
	查看实例配置参 数	✓	✓	✓	✓
	修改实例配置参 数	✓	✓	✓	✗
	查看实例访问密 码	✓	✓	✓	✗
Redis	Redis 实例列表	查看列表	✓	✓	✓
	实例名称搜索	✓	✓	✓	✓
	创建实例	✓	✓	✓	✗
	更新实例配置	✓	✓	✓	✗
	删除实例	✓	✗	✗	✗
Redis 实例详情	实例概览	✓	✓	✓	✓
	实例监控	✓	✓	✓	✓
	查看实例配置参 数	✓	✓	✓	✓
	修改实例配置参 数	✓	✓	✓	✗
	查看实例访问密 码	✓	✓	✓	✗
Kafka	Kafka 实例列表	查看列表	✓	✓	✓
	实例名称搜索	✓	✓	✓	✓
	创建实例	✓	✓	✓	✗
	更新实例配置	✓	✓	✓	✗
	删除实例	✓	✗	✗	✗
Kafka 实例详情	实例概览	✓	✓	✓	✓
	实例监控	✓	✓	✓	✓
	查看实例配置参 数	✓	✓	✓	✓
	修改实例配置参 数	✓	✓	✓	✗
		✓	✓	✓	✗

		查看实例访问密 码			
MinIO	MinIO 实例列表	查看列表	✓	✓	✓
		实例名称搜索	✓	✓	✓
		创建实例	✓	✓	✗
		更新实例配置	✓	✓	✗
MinIO	MinIO 实例详情	删除实例	✓	✗	✗
		实例概览	✓	✓	✓
		实例监控	✓	✓	✓
		查看实例配置参 数	✓	✓	✓
MinIO	MinIO 实例配置	修改实例配置参 数	✓	✓	✗
		查看实例访问密 码	✓	✓	✗

4. 数据服务介绍

DCE 5.0 针对实际应用场景，精选了一些经典的数据服务中间件，通过前后端开发，能够满足各类应用场景的开发和维护。

用户可以按需安装/启用以下数据服务中间件，即插即用：

- [Elasticsearch 搜索服务](#): 目前首选的全文搜索引擎
- [Kafka 消息队列](#): 常用于消息传输的数据管道
- [MinIO 对象存储](#): 一款非常热门的轻量对象存储方案
- [MySQL 数据库](#): 最流行的关系型数据库
- [RabbitMQ 消息队列](#): 常用于交易数据的传输管道
- [Redis 缓存服务](#): 一种内存数据库

4.1 数据服务学习路径

上述几个数据服务中间件的学习路径大致相同，此处以 RabbitMQ 为例，简单说明学习路径。

!!! info

在下方流程流程图中点击相应文字可以直接跳转到对应的操作指南页面。

```
graph TD
    B((B (选择工作空间))) --> C{C (部署/创建实例)}
    C --> D[更新/删除实例]
    C --> E[实例概览]
    C --> F[实例监控]
    C --> G[数据迁移]
    C --> H[卸载中间件]

    click B "https://docs.daocloud.io/middleware/rabbitmq/user-guide/login/"
    click C "https://docs.daocloud.io/middleware/rabbitmq/user-guide/create/"
    click D "https://docs.daocloud.io/middleware/rabbitmq/user-guide/update/"
    click E "https://docs.daocloud.io/middleware/rabbitmq/user-guide/view/"
    click F "https://docs.daocloud.io/middleware/rabbitmq/user-guide/insight/"
    click G "https://docs.daocloud.io/middleware/rabbitmq/user-guide/migrate/"
    click H "https://docs.daocloud.io/middleware/rabbitmq/quickstart/install/#_1"
```

[Elasticsearch](#){ .md-button .md-button--primary } [Kafka](#){ .md-button .md-button--primary } [MinIO](#){ .md-button .md-button--primary }
[MySQL](#){ .md-button .md-button--primary } [RabbitMQ](#){ .md-button .md-button--primary } [Redis](#){ .md-button .md-button--primary }

5. Elasticsearch

5.1 Elasticsearch 索引服务 Release Notes

本页列出 Elasticsearch 索引服务的 Release Notes，便于您了解各版本的演进路径和特性变化。

v0.5.0

发布日期：2023-02-23

API

- 新增 `mcamel-elasticsearch helm-docs` 模板文件。
- 新增 `mcamel-elasticsearch` 应用商店中的 Operator 只能安装在 `mcamel-system`。
- 新增 `mcamel-elasticsearch` 支持 cloud shell。
- 新增 `mcamel-elasticsearch` 支持导航栏单独注册。
- 新增 `mcamel-elasticsearch` 支持查看日志。
- 新增 `mcamel-elasticsearch` Operator 对接 chart-syncer。
- 新增 `mcamel-elasticsearch` 支持 LB。
- 修复 `mcamel-elasticsearch` 实例名太长导致自定义资源无法创建的问题。
- 修复 `mcamel-elasticsearch` 工作空间 Editor 用户无法查看实例密码。
- 修复 `mcamel-elasticsearch` 密码不能使用特殊字符的问题。
- 修复 `mcamel-elasticsearch` 超出索引导致 panic 的问题。
- 升级 `mcamel-elasticsearch` 升级离线镜像检测脚本。

文档

- 新增 日志查看操作说明，支持自定义查询、导出等功能。

v0.4.0

发布日期：2022-12-25

API

- 新增 `mcamel-elasticsearch` 获取集群已经分配的 NodePort 列表接口。
- 新增 `mcamel-elasticsearch` 增加状态详情。
- 新增 `mcamel-elasticsearch` 节点亲和性配置。
- 修复 `mcamel-elasticsearch` 修复 kb 不存在时候，删除会失败的 BUG。
- 修复 `mcamel-elasticsearch` 修复 es exporter 离线失效的问题。
- 修复 `mcamel-elasticsearch` 修复 es 创建成功后没有返回 ports 信息的 bug。
- 修复 `mcamel-elasticsearch` 查询实例列表和详情时，Kibana 的服务类型不符合预期。
- 优化 `mcamel-elasticsearch` 可以展示公共的 es，纳管之前是不可以删除的。
- 优化 `mcamel-elasticsearch` 增加健康状态返回。

v0.3.6

发布日期: 2022-11-28

- 改进 密码校验调整为 MCamel 中等密码强度
- 改进 角色可以升级
- 新增 新增 sc 扩容提示
- 新增 返回列表或者详情时的公共字段
- 新增 返回告警
- 新增 校验 Service 注释
- 修复 更新实例后, 集群使用了错误的镜像, 导致集群状态异常
- 修复 使用 NodePort 时, 更新实例报错
- 升级 中依赖的 eck operator 版本为 2.3.0
- 优化 在某些版本的 K8s 集群中, 默认 FD 不足, 无法启动的问题
- 优化 减小 elasticsearch 容器的运行权限

v0.3.4

发布日期: 2022-10-28

API

- 新增 同步 pod 状态到实例详情页
- 优化 workspace 界面逻辑调整
- 优化 不符合设计规范的样式调整
- 优化 password 获取逻辑调整
- 优化 cpu&内存请求量应该小于限制量逻辑调整
- 优化 实例版本不允许修改, 下拉框应该为文本
- 修复 更新实例服务设置, 确认无反应, 无法提交
- 新增 获取用户列表接口
- 新增 支持 arm 架构

v0.3.2

发布日期: 2022-9-25

API

- 新增 列表页增加分页功能
- 新增 增加修改配置的功能
- 新增 增加返回可修改配置项的功能
- 新增 更改创建实例的限制为集群级别, 原来为 namespace 级别
- 新增 增加监控地址的拼接功能
- 新增 增加可以修改版本号的功能
- 新增 修改底层 update 逻辑为 patch 逻辑
- 新增 将时间戳 api 字段统一调整为 int64
- 新增 单测覆盖率提升到 43%
- 新增 对接全局管理增加 workspace 接口

- 新增 对接 insight 通过 crd 注入 dashboard
- 新增 更新 release note 脚本，执行 release-process 规范

安装

- 新增 支持 helm 部署 eck-operator
- 新增 支持 helm 部署 mcamel-elasticsearch 服务

文档

- 新增 第一次文档网站发布
- 新增 功能说明
- 新增 产品优势
- 新增 什么是 Elasticsearch
- 新增 基本概念
- 新增 集群容量规划

5.2 Intro

产品优势

Elasticsearch 很快

由于 Elasticsearch 是在 Lucene 基础上构建而成的，所以在全文本搜索方面表现十分出色。 Elasticsearch 同时还是一个近实时的搜索平台，这意味着从文档索引操作到文档变为可搜索状态之间的延时很短，一般只有一秒。因此，Elasticsearch 非常适用于对时间有严苛要求的用例，例如安全分析和基础设施监测。

Elasticsearch 具有分布式的本质特征

Elasticsearch 中存储的文档分布在不同的容器中，这些容器称为分片，可以进行复制以提供数据冗余副本，以防发生硬件故障。 Elasticsearch 的分布式特性使得它可以扩展至数百台（甚至数千台）服务器，并处理 PB 量级的数据。

Elasticsearch 包含一系列广泛的功能

除了速度、可扩展性和弹性等优势以外，Elasticsearch 还有大量强大的内置功能（例如数据汇总和索引生命周期管理），可以方便用户更加高效地存储和搜索数据。

Elastic Stack 简化了数据采集、可视化和报告过程

通过与 Beats 和 Logstash 进行集成，用户能够在向 Elasticsearch 中索引数据之前轻松地处理数据。同时，Kibana 不仅可针对 Elasticsearch 数据提供实时可视化，同时还提供 UI 以便用户快速访问应用程序性能监测（APM）、日志和基础设施指标等数据。

Elasticsearch 支持多种编程语言

- Java
- JavaScript (Node.js)
- Go
- .NET (C#)
- PHP
- Perl
- Python
- Ruby

基本概念

本节列出有关 Elasticsearch 涉及的专有名词及术语。

- 节点 (node)

数据节点：存储索引数据的节点，主要对文档进行增删改查、聚合等操作。

专有主节点：对集群进行操作，例如创建或删除索引，跟踪哪些节点是集群的一部分，并决定哪些分片分配给相关的节点。稳定的主节点对集群的健康非常重要，默认情况下集群中的任一节点都可能被选为主节点。

协调节点：分担数据节点的 CPU 开销，从而提高处理性能和服务稳定性。

- 索引 (Index)

用于存储 Elasticsearch 的数据，是一个或多个分片分组在一起的逻辑空间。

- 分片 (Shard)

索引可以存储数据量超过 1 个节点硬件限制的数据。为满足这样的需求，Elasticsearch 提供了一个能力，将一个索引拆分为多个，称为 Shard。当您创建一个索引时，您可以根据实际情况指定 Shard 的数量。每个 Shard 托管在集群中的任意一个节点中，且每个 Shard 本身是一个独立的、全功能的“索引”。Shard 的数量只能在创建索引前指定，且在索引创建成功后无法修改。

- 副本 (replicas)

replicas 是索引的备份，Elasticsearch 可以设置多个副本。写操作会先在主分片上完成，然后分发到副本分片上。因为索引的主分片和副本分片都可以对外提供查询服务，所以副本能够提升系统的高可用性和搜索时的并发性能。但如果副本太多，也会增加写操作时数据同步的负担。

功能说明

Elasticsearch 以 JSON 文档的形式存储数据。每个文档都会在一组键（字段或属性的名称）与它们对应的值（字符串、数字、布尔值、日期、数值组、地理位置或其他类型的数据）之间建立联系。

Elasticsearch 使用的是一种名为倒排索引的数据结构，这一结构的设计可以允许十分快速地进行全文本搜索。倒排索引会列出在所有文档中出现的每个特有词汇，并且可以找到包含每个词汇的全部文档。

在索引过程中，Elasticsearch 会存储文档并构建倒排索引，这样用户便可以近实时地对文档数据进行搜索。索引过程是在索引 API 中启动的，通过此 API 您既可向特定索引中添加 JSON 文档，也可更改特定索引中的 JSON 文档。

Elasticsearch 支持的通用功能特性如下：

分类	特性	说明
分布式集群	集群部署集群监控	提供运维平台页面监控集群、节点、索引的运行状况
搜索管理	索引配置管理结构定义、索引重建	搜索管理平台提供配置功能
全文搜索	搜索功能排序功能统计分析功能	通过 RESTful API 方式提供
数据采集	ElasticSearch 数据导入 API Maxcompute 数据导入工具全量、增量采集方式	丰富的原生数据采集接口，集成 Maxcompute 数据导入工具
服务鉴权	服务级别的用户鉴权机制	统一的用户鉴权设置

在 DCE 5.0 中部署 Elasticsearch 后，还将支持以下特性：

- 支持 Elasticsearch 专有节点、热数据节点、冷数据节点、数据节点角色部署
- 集成 Kibana
- 基于 elasticsearch-exporter 暴露指标
- 基于 Grafana Operator 集成 Elasticsearch Dashboard，展示监控数据
- 使用 ServiceMonitor 对接 Prometheus 抓取指标
- 基于[工作空间 Workspace](#) 多租户化管理

适用场景

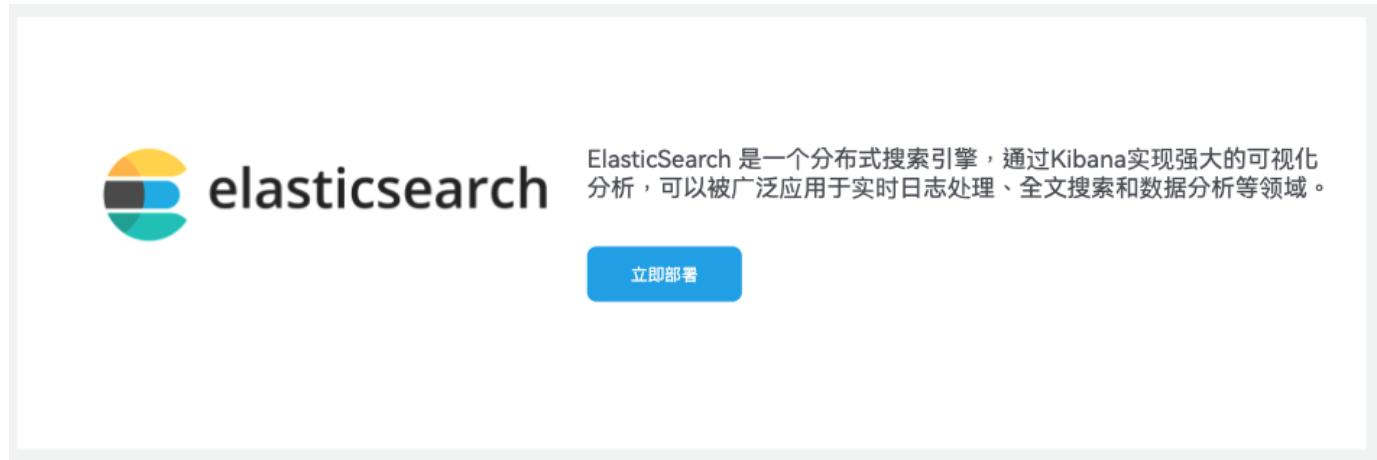
Elasticsearch 在速度和可扩展性方面都表现出色，而且还能够索引多种类型的内容，这意味着其可用于多种使用场景：

- 应用程序搜索
- 网站搜索
- 企业搜索
- 日志处理和分析
- 基础设施指标和容器监测
- 应用程序性能监测
- 地理空间数据分析和可视化
- 安全分析
- 业务分析

什么是 Elasticsearch

Elasticsearch（下文简称 Elastic）是目前全文搜索引擎的首选。它可以快速地存储、搜索和分析海量数据。Elastic 的底层是开源库 Lucene，但是 Lucene 无法直接用，必须自己写代码去调用它的接口。Elastic 是 Lucene 的封装，提供了 REST API 的操作接口，开箱即用。

DCE 5.0 内置的搜索服务基于 Elasticsearch，能够提供分布式搜索服务，为用户提供结构化、非结构化文本以及基于 AI 向量的多条件检索、统计、报表。完全兼容 Elasticsearch 原生接口。它可以帮助网站和 APP 搭建搜索框，提升用户的搜索体验；也可以用于搭建日志分析平台，助力企业实现数据驱动运维，数据驱动运营；它的向量检索能力可以帮助客户快速构建基于 AI 的图搜、推荐、语义搜索、人脸识别等丰富的应用。



ElasticSearch 的工作管理

原始数据会从多个来源（包括日志、系统指标和网络应用程序）输入到 Elasticsearch 中。数据采集旨在 Elasticsearch 中进行索引之前解析、标准化并充实这些原始数据的过程。这些数据在 Elasticsearch 中完成索引之后，用户便可针对数据运行复杂的查询，并使用聚合来检索自身数据的复杂汇总。在 Kibana 中，用户可以基于自己的数据创建强大的可视化。

Kibana 是什么？

Kibana 是一款适用于 Elasticsearch 的数据可视化和管理工具，可以提供实时的直方图、线形图、饼状图和地图。Kibana 同时还包括诸如 Canvas 和 Elastic Maps 等高级应用程序；其中 Canvas 允许用户基于自身数据创建定制的动态信息图表，而 Elastic Maps 则可用来对地理空间数据进行可视化。

创建 Elasticsearch 实例{ .md-button .md-button--primary }

5.3 User guide

节点信息

本节说明 Elasticsearch 集群的基本信息，包括：

- 实例运行状态、部署位置、创建时间、版本等信息
- 访问设置
- 资源配置
- Pod 列表

The screenshot shows the Daocloud platform interface for managing an Elasticsearch cluster. The main title is 'Daocloud'. The left sidebar has a '概览' tab selected, which is highlighted in grey. The main content area is titled '基本信息'.

基本信息

elastic-01	运行中	cluster01/namespace01	2022-04-22 14:00:03
elasticsearch实例名称	运行状态	部署位置	创建时间
6.8.10	3 副本	-	
版本	数据节点副本数	描述	

访问设置

NodePort 访问方式	elastic 用户名	CPU 限制量: 2 core 请求量: 2 core
192.168.12.32 集群IPv4	http://10.6.210.25:15762 kibana 控制台地址	内存 限制量: 4 G 请求量: 4 G
		磁盘 10 GB

资源配置

容器组列表

容器组名称	节点类型	运行状态	容器组 IP	重启次数	CPU 使用量/限制量	内存使用量/限制量	创建时间
pod-01	数据节点	运行中	192.168.21.100	0	0.15 core / 2 core	2.32GB / 4GB	2022-04-22 14:00:03
pod-02	数据节点	运行中	192.168.21.101	3	0.15 core / 2 core	2.32GB / 4GB	2022-04-22 14:00:03
pod-03	Kibana节点	运行中	192.168.21.102	3	0.15 core / 2 core	2.32GB / 4GB	2022-04-22 14:00:03
pod-03	冷数据节点	运行中	192.168.21.102	3	0.15 core / 2 core	2.32GB / 4GB	2022-04-22 14:00:03

Elasticsearch 集群规格和容量规划

存储容量规划

影响 Elasticsearch 服务存储容量的主要因数：

- 副本数量：副本有利于增加数据的可靠性，但同时会增加存储成本。默认和建议的副本数为 1。
- 内部任务开销：Elasticsearch 用于 segment 合并、ES Translog、日志，保留约 20% 的磁盘空间。
- 索引开销：通常比源数据大 10%，可以使用 `_cat/indices?v` API 和 `pri.store.size` 值计算确切的开销。
- 操作系统预留：默认情况下，Linux 将保留 5% 的磁盘空间，给 `root` 用户用于关键流程处理、系统恢复、防止磁盘碎片化问题。

公式

完整公式	简化版本
$\text{源数据} * (1 + \text{副本数量}) * (1 + \text{索引开销}) / (1 - \text{Linux 预留空间}) / (1 - \text{内部开销}) = \text{最小存储要求}$	$\text{源数据} * (1 + \text{副本数量}) * 1.45 = \text{最小存储要求}$

如果有 500G 数据存储并且需要一个副本，则最低存储要求更接近 $500 * 2 * 1.1 / 0.95 / 0.8 = 1.5T$.

集群配置

在生产环境部署推荐配置：尽量一个节点只承担一个角色。不同节点所需要的计算资源不一样。不同角色分离后，可以按需扩展互不影响。

- 集群最大节点数 = 单节点 CPU * 5
- 单节点磁盘最大容量
- 搜索类场景：单节点磁盘最大容量 = 单节点内存大小 (GB) * 10。
- 日志类等场景：单节点磁盘最大容量 = 单节点内存大小 (GB) * 50。

配置	最大节点数	单节点磁盘最大容量 (查询)	单节点磁盘最大容量 (日志)
4 核 16G	20	160 GB	800 GB
8 核 32G	40	320 GB	1.5 TB
16 核 64G	80	640 GB	2 TB

分片数量规划

适用场景：

- 日志类，写入频繁，查询较少，单个分片 30G 左右
- 搜索类，写入少，查询频繁，单个分片不超过 20G

每个 Elasticsearch 索引被分为多个分片，数据按哈希算法打散到不同的分片中。由于索引分片的数量影响读写性能和故障恢复速度，建议提前规划。

分片使用概要

- Elasticsearch 在 7.x 版本中，每个索引默认为 1 个主分片 和 1 个副本分片
- 在单节点上，7.x 版本最大分片数量为 1000
- 单个分片大小尽量保持在 10-50G 之间为最佳体验，一般推荐在 30G 左右
- 分片过大可能使 Elasticsearch 的故障恢复速度变慢
- 分片过小可能导致非常多的分片，因为每个分片会使用占用一些 CPU 和内存，从而导致读写性能和内存不足的问题。
- 当分片数量超过数据节点数量时，建议分片数量接近数据节点的整数倍，便于将分片均匀的分布到数据节点中。
- 对日志场景，建议启用 ILM 功能。在发现分片大小不合理时，通过该功能及时调整分片数量。

索引分片资源占用

每个索引和每个分片都需要一些内存和 CPU 资源。在大多数情况下，一小组大分片比许多小分片使用更少的资源。

段在分片的资源使用中起着重要作用。大多数分片包含几个段，用于存储其索引数据。 Elasticsearch 将段元数据保存在 JVM 堆内存中，以便可以快速检索它以进行搜索。 随着分片的增长，它的段被合并成更少、更大的段。这减少了段的数量，这意味着更少的元数据保存在堆内存中。

为了减少索引数量并避免造成过大且无序的映射，可以考虑在同一索引中存储类似结构的数据，而不要基于数据来源将数据分到不同的索引中。很重要的一点是在索引/分片的数量和每个单独索引的映射大小之间实现良好平衡。由于集群状态会加载到每个节点（包括主节点）上的堆内存中，而且堆内存大小与索引数量以及单个索引和分片中的字段数成正比关系，所以还需要同时监测主节点上的堆内存使用量并确保其大小适宜，这一点很重要。

分片过小会导致段过小，进而致使开销增加。您要尽量将分片的平均大小控制在至少几 GB 到几十 GB 之间。对时序型数据用例而言，分片大小通常介于 20GB 至 40GB 之间。

由于单个分片的开销取决于段数量和段大小，所以通过 `forcemerge` 操作强制将较小的段合并为较大的段能够减少开销并改善查询性能。理想状况下，应当在索引内再无数据写入时完成此操作。请注意：这是一个极其耗费资源的操作，所以应该在非高峰时段进行。

每个节点上可以存储的分片数量与可用的堆内存大小成正比关系，但是 Elasticsearch 并未强制规定固定限值。这里有一个很好的经验法则：确保对于节点上已配置的每个 GB，将分片数量保持在 20 以下。如果某个节点拥有 30GB 的堆内存，那其最多可有 600 个分片，但是在此限值范围内，您设置的分片数量越少，效果就越好。一般而言，这可以帮助集群保持良好的运行状态。

更多信息，请参考：

- [减少集群分片数](#)
- [我在 Elasticsearch 集群内应该设置多少个分片？](#)

分片计算公式

$(\text{元数据} + \text{增长空间}) * (1 + \text{索引开销}) / \text{所需的分片大小} = \text{主分片的大约数量}$

假设有 80GiB 的数据。希望将每个分片保持在 30GiB 左右。因此，您的分片数量应大约为 $80 * 1.1 / 30 = 3$

如何管理分片

使用索引生命周期管理 (ILM) 自动管理索引，管理策略如下：

- 根据索引大小，自动 rollover
- 根据索引创建时间，自动 rollover
- 根据文档数量，自动 rollover

索引生命周期执行策略，默认每 10 分钟执行一次，可以通过修改 `indices.lifecycle.poll_interval` 参数来控制检查频率。

创建 Elasticsearch 实例

在 Elasticsearch 实例列表中，执行以下操作创建一个新的实例。

1. 在右上角点击 新建实例。

The screenshot shows the Elasticsearch instance management interface. At the top, there's a search bar and a '新建实例' (Create Instance) button. Below the search bar, the instance name 'aaa' is listed with a red dot indicating it's '未就绪' (Not Ready). The instance details include:

- 部署位置: kpanda-global-cluster/default
- 版本: 7.16.3
- 副本数: 热数据节点: 2 副本 +3
- 资源配额: CPU 请求量 0.1 Core / 限制量 1 Core
内存 请求量 0.1 GB / 限制量 1 GB
磁盘 1 GB
- 状态: 正常/全部副本数 6 / 9

At the bottom, there are navigation buttons for page 1 of 1 and a dropdown for selecting 10 items.

2. 在创建 Elasticsearch 实例页面中，输入实例的基本信息后，点击 下一步。

The screenshot shows the 'Create Elasticsearch Instance' wizard, Step 1: Basic Information. The steps are numbered 1 to 4. The current step is 1. The basic information section includes:

- ElasticSearch 实例名称: es01
- 描述: (empty)
- 部署位置:
 - 集群: kpanda-global-cluster
 - 命名空间: skoala-sesame1

At the bottom right, there are '取消' (Cancel) and '下一步' (Next Step) buttons.

3. 选择一个版本，配置实例的以下规格后，点击 下一步。可以视情况选择启用/禁用数据节点、Kibana 节点、专用主节点和冷数据节点。

DaoCloud

创建 Elasticsearch 实例

基本信息 2 规格配置 3 服务设置 4 配置确认

选择版本

版本 * 7.16.3

规格配置

展开面板配置实例副本数及规格。

> 数据节点 启用

> Kibana 节点 启用

> 专用主节点

> 冷数据节点

取消 上一步 下一步

- 默认启用热数据节点，用于存放 Elasticsearch 搜索服务的日常活跃数据，默认 3 个副本，最少 1 个，最多 50 个。

数据节点 启用

数据节点 * 启用

副本数 * 3
1 副本仅适于 POC 模式。推荐 3 副本支持高可用。请配置副本数为奇数，否则有脑裂风险。

CPU 配额 * 请求量 0.1 Core 限制量 1 Core

内存配额 * 请求量 0.1 GB 限制量 1 GB

存储类 * sss

存储容量 * 1 GB

- 默认启用 Kibana 节点，用于存放 Elasticsearch 可视化数据的节点，默认为 1 个 Kibana 节点，不可增加或减少。

Kibana 节点 启用

Kibana 节点 * 启用

副本数 * 1

CPU 配额 * 请求量 0.1 Core 限制量 1 Core

内存配额 * 请求量 0.1 GB 限制量 1 GB

- 可选的专用主节点。这是为了一些专用目的而设定的节点，默认 3 个专用主节点，不可增加和减少。

专用主节点 启用

专用主节点 * 启用

副本数 * 3

CPU 配额 * 请求量 0.1 Core 限制量 1 Core

内存配额 * 请求量 0.1 GB 限制量 1 GB

存储类 * sss

存储容量 * 1 GB

- 可选的冷数据节点。这是存放一些 Elasticsearch 历史数据的节点，默认 3 个冷数据节点，最少 2 个，最多 50 个。

冷数据节点 启用

冷数据节点 启用

副本数 *	3	
CPU 配额 *	请求量 0.1 Core	限制量 1 Core
内存配额 *	请求量 0.1 GB	限制量 1 GB
存储类 *	sss	
存储容量 *	1 GB	

4. 设置访问类型（ClusterIP 或 NodePort）、用户名和密码后，点击下一步。

DaoCloud

[← 创建 Elasticsearch 实例](#)

基本信息 规格配置 服务设置 配置确认

服务设置

访问类型 * 集群内访问(ClusterIP) 节点端口(NodePort)

端口配置	协议	端口名称	服务端口	容器端口
HTTP	es	9200	9200	x
HTTP	kibana	5601	5601	x
+ 添加				
注释	key	value	+ 添加	

访问设置

用户名 * elastic

密码 *

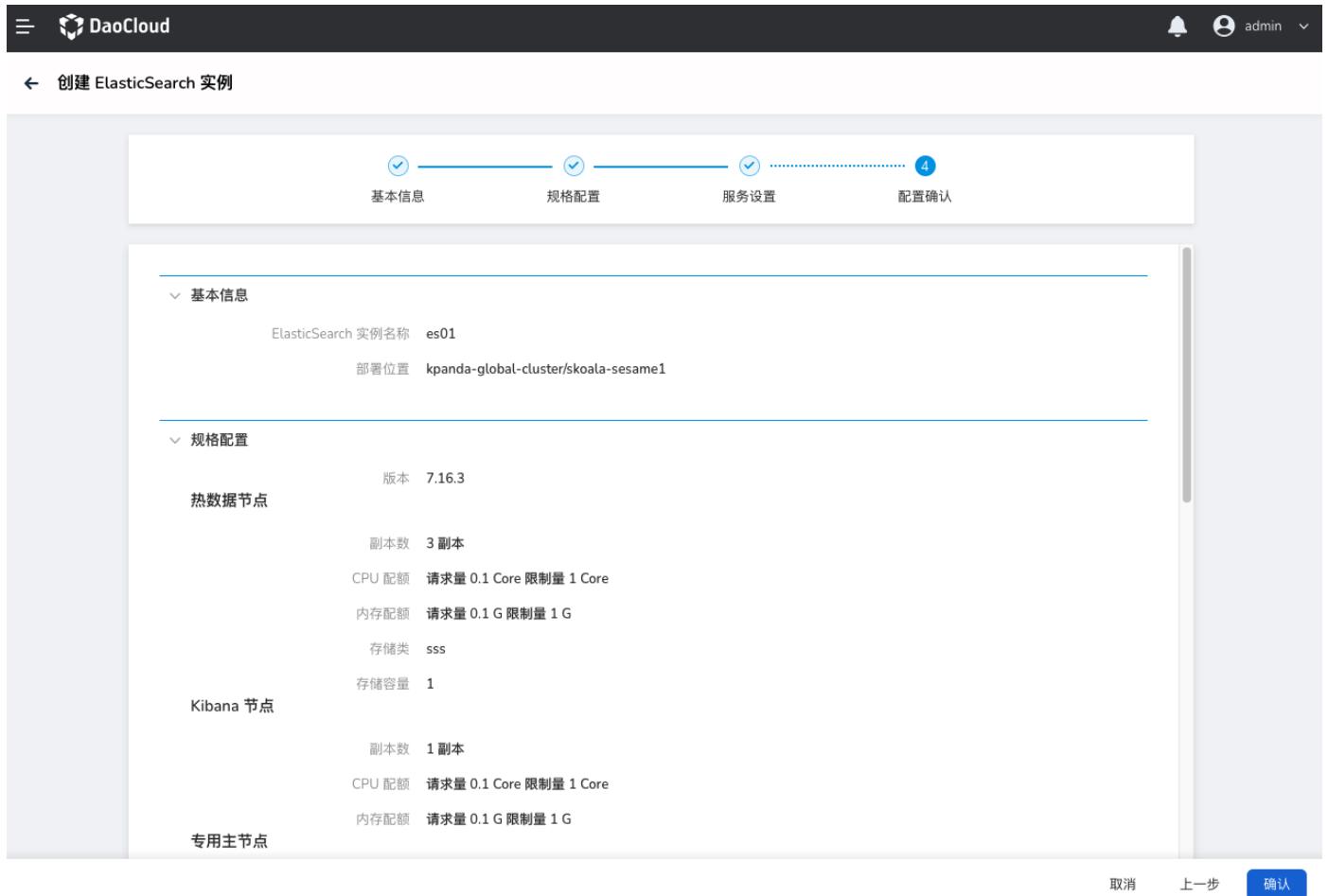
6~18个字符，必须同时包含英文大小写、数字、特殊符号([@\$%^&.,/?-=_+])。

确认密码 *

该用户名、密码用于登录 Elasticsearch 的用户认证和 Kibana 网页控制台，请谨慎设置。

取消 上一步 下一步

5. 确认上述基本信息、规格配置和服务设置无误后，点击确认。



6. 屏幕提示 创建实例成功。

aaa ● 未就绪

部署位置: kpanda-global-cluster/default
版本: 7.16.3
副本数: 热数据节点: 2 副本 +3
资源配额: CPU 请求量 0.1 Core / 限制量 1 Core
内存 请求量 0.1 GB / 限制量 1 GB
磁盘 1 GB
6 / 9 正常/全部副本数

es01 ● 未就绪

部署位置: kpanda-global-cluster/skoala-sesam...
版本: 7.16.3
副本数: 热数据节点: 3 副本 +3
资源配额: CPU 请求量 0.1 Core / 限制量 1 Core
内存 请求量 0.1 GB / 限制量 1 GB
磁盘 1 GB
0 / 10 正常/全部副本数

共 2 项 1 / 1 10 项

删除 Elasticsearch 实例

如果想要删除一个 Elasticsearch 实例，可以执行如下操作：

!!! warning

删除实例后，该实例相关的所有信息也会被全部删除，请谨慎操作。

- 在 Elasticsearch 实例列表中，点击右侧的 ... 按钮，在弹出菜单中选择删除实例。

The screenshot shows the Elasticsearch instance list. An instance named 'aaa' is selected, indicated by a red box around its row. On the right side of the instance card, there is a context menu with options: '更新实例' (Update Instance) and '删除实例' (Delete Instance), with the 'Delete Instance' option also highlighted with a red box.

- 在弹窗中输入该实例的名称，确认无误后，点击 删除 按钮。



- 自动返回实例列表，屏幕提示：删除实例 xxx 成功。

The screenshot shows the Elasticsearch instance list again. A success message '删除成功 es01 成功' (Delete instance es01 successfully) is displayed in the top right corner. The instance 'aaa' is still listed, and the status is now '未就绪' (Not Ready). The number of instances is shown as '6 / 9'.

更新 Elasticsearch 实例

如果想要更新或修改 Elasticsearch 实例的资源配置，可以按照本页说明操作。

- 在 Elasticsearch 实例列表中，点击右侧的 ... 按钮，在弹出菜单中选择 更新实例。

ElasticSearch Default

aaa ● 未就绪

部署位置 kpanda-global-cluster/default 资源配额 CPU 请求量 0.1 Core / 限制量 1 Core
版本 7.16.3 内存 请求量 0.1 GB / 限制量 1 GB
副本数 热数据节点: 2 副本 +3 磁盘 1 GB 正常/全部副本数

es01 ● 未就绪

部署位置 kpanda-global-cluster/skoala-sesam... 资源配额 CPU 请求量 0.1 Core / 限制量 1 Core
版本 7.16.3 内存 请求量 0.1 GB / 限制量 1 GB
副本数 热数据节点: 3 副本 +3 磁盘 1 GB 正常/全部副本数

共 2 项 1 / 1 10 项

- 修改基本信息后，点击下一步。此处暂时只支持修改描述信息。

更新实例 es01

1 ————— 2 ————— 3

基本信息

ElasticSearch 实例名称 es01

描述 这是一个 test 实例
描述信息不超过 256 个字符。

部署位置

集群 kpanda-global-cluster

命名空间 skoala-sesame1

取消 下一步

- 修改规格配置（包括热数据节点、Kibana 节点、专用主节点和冷数据节点）后，点击下一步。

更新实例 es01

X



选择版本

版本 * 7.16.3

规格配置

i 展开面板配置实例副本数及规格。

X

热数据节点 启用

热数据节点 * 启用

副本数 * 3

请求量	0.1	Core	限制量	1	Core
-----	-----	------	-----	---	------

请求量	0.1	GB	限制量	1	GB
-----	-----	----	-----	---	----

存储类 sss

存储容量 * 1	GB
----------	----

取消

上一步

下一步

4. 修改服务设置后，点击 确认。

更新实例 es01

X



访问类型 * 集群内访问(ClusterIP) * 节点端口(NodePort) *

端口配置	协议	端口名称	服务端口	容器端口	×				
HTTP	es	9200	9200		×				
HTTP	kibana	5601	5601		×				
+ 添加									
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 50%; padding: 5px;">注释</td> <td style="width: 50%; padding: 5px;">key</td> </tr> <tr> <td style="padding: 5px;"></td> <td style="padding: 5px;">value</td> </tr> </table>						注释	key		value
注释	key								
	value								
+ 添加									

取消

上一步

确认

5. 返回消息队列，屏幕右上角将显示消息：更新实例成功。

The screenshot shows the DaoCloud interface for managing Elasticsearch instances. At the top, there's a navigation bar with the DaoCloud logo and a search bar. Below it, the 'ElasticSearch' section is active, with a 'Default' cluster selected. Two instances are listed:

- aaa**: Status: 未就绪 (Not Ready). Deployment Location: kpanda-global-cluster/default. Version: 7.16.3. Replicas: 6 / 9 (热数据节点: 2 副本 +3). Resources: CPU Request 0.1 Core / Limit 1 Core, Memory Request 0.1 GB / Limit 1 GB, Disk 1 GB.
- es01**: Status: 未就绪 (Not Ready). Deployment Location: kpanda-global-cluster/skoala-sesam... (truncated). Version: 7.16.3. Replicas: 0 / 10 (热数据节点: 3 副本 +3). Resources: CPU Request 0.1 Core / Limit 1 Core, Memory Request 0.1 GB / Limit 1 GB, Disk 1 GB.

At the bottom, there's a pagination bar showing 1 / 1 item, a dropdown for selecting 10 items, and standard navigation icons.

查看 Elasticsearch 日志

操作步骤

通过访问每个 Elasticsearch 的实例详情，页面；可以支持查看 Elasticsearch 的日志。

- 在 Elasticsearch 实例列表中，选择想要查看的日志，点击 实例名称 进入到实例详情页面。

The screenshot shows the DaoCloud interface for managing Elasticsearch instances. In the center, the details for the instance 'mcamel-common-es-cluster-masters' are displayed. Key information includes:

- 部署位置:** kpanda-global-cluster/mc...
- 版本:** 7.16.3
- 副本数:** 数据节点: 3 副本
- 访问地址:** mcamel-common-es-cluster-masters-es-http.mcamel-system.svc:9200
- 状态:** GREEN (2 / 3)

- 在实例的左侧菜单栏，会发现有一个日志查看的菜单栏选项。

The screenshot shows the DaoCloud interface for managing Elasticsearch instances. The left sidebar highlights the '日志查看' (Log View) option. The main panel displays the following information:

- 基本信息:**
 - 实例名称: mcamel-common-es-cluster-masters
 - 运行状态: 未就绪
 - 部署位置: kpanda-global-cluster/mc...
 - 创建时间: 2023-02-26 14:17
- 健康状态:** GREEN (集群健康状态)
- 访问设置:**
 - 访问方式: NodePort
 - 用户名: elastic
 - Kibana 控制台地址: https://10.233... (显示为 *****)
- 资源配额:**
 - CPU 使用率: 20% (已用 1.17 Core, 总计 6 Core)
 - 内存使用率: 78% (已用 9.367 GB, 总计 12 GB)
 - 磁盘使用率: 5% (已用 1.758 GB, 总计 35 GB)
- 监控告警 (最近 10 条):** 表格视图，列有规则名称、告警级别、资源类型、资源名称、描述、发生时间、持续时间。

- 点击 日志查看 即可进入到日志查看页面（Insight 日志查看）。

日志查看说明

在日志查看页面，我们可以很方便的进行日志查看，常用操作说明如下：

- 支持 自定义日志时间范围，在日志页面右上角，可以方便地切换查看日志的时间范围（可查看的日志范围以 可观测系统设置内保存的日志时长为准）
- 支持 关键字检索日志，左侧检索区域支持查看更多的日志信息
- 支持 日志量分布查看，中上区域柱状图，可以查看在时间范围内的日志数量分布
- 支持 查看日志的上下文，点击右侧 上下文 图标即可
- 支持 导出日志

时间	日志
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.268Z","log.logger":"elasticsearch-controller","message":"Ending reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters","took":0.466213453}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.242Z","log.logger":"zen2","message":"Ensuring no voting exclusions are set","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"} 5
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.196Z","log.logger":"transport","message":"Skipping pod because it has no IP yet","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","pod_name":"mcamel-common-es-cluster-masters-es-masters-2"}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:54.802Z","log.logger":"elasticsearch-controller","message":"Starting reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"}

实例监控

本页说明如何从图表面板快速获取 Elasticsearch 有关的信息以及一些注意点和小提示。

- 在 Elasticsearch 实例列表中点击一个实例名称。

ElasticSearch Default

搜索 新建实例

aaa ● 未就绪

部署位置: kpanda-global-cluster/default 资源配额: CPU 请求量 0.1 Core / 限制量 1 Core 6 / 9
版本: 7.16.3 内存 请求量 0.1 GB / 限制量 1 GB
副本数: 热数据节点: 2 副本 +3 磁盘 1 GB 正常/全部副本数

共 1 项 1 / 1 10 项

- 在左侧导航栏点击 实例监控 , 进入实例监控大屏。

DaoCloud admin

aaa Default

ElasticSearch 实例: aaa / 实例监控

概览 实例监控

Last 30 minutes

KPI

Cluster health	Trip...	CPU usage Avg.	JVM memory us...	No...	Dat...	Pen...	Open file descrip...
Green	0	5.17%	101%	6	4	0	2.55 k

Shards

Active primary s...	Active shards	Initializing shards	Relocating shards	Delayed shards	Unassigned shard...
10	18	0	0	0	0

> JVM Garbage Collection (2 panels)

> Translog (2 panels)

> Breakers (2 panels)

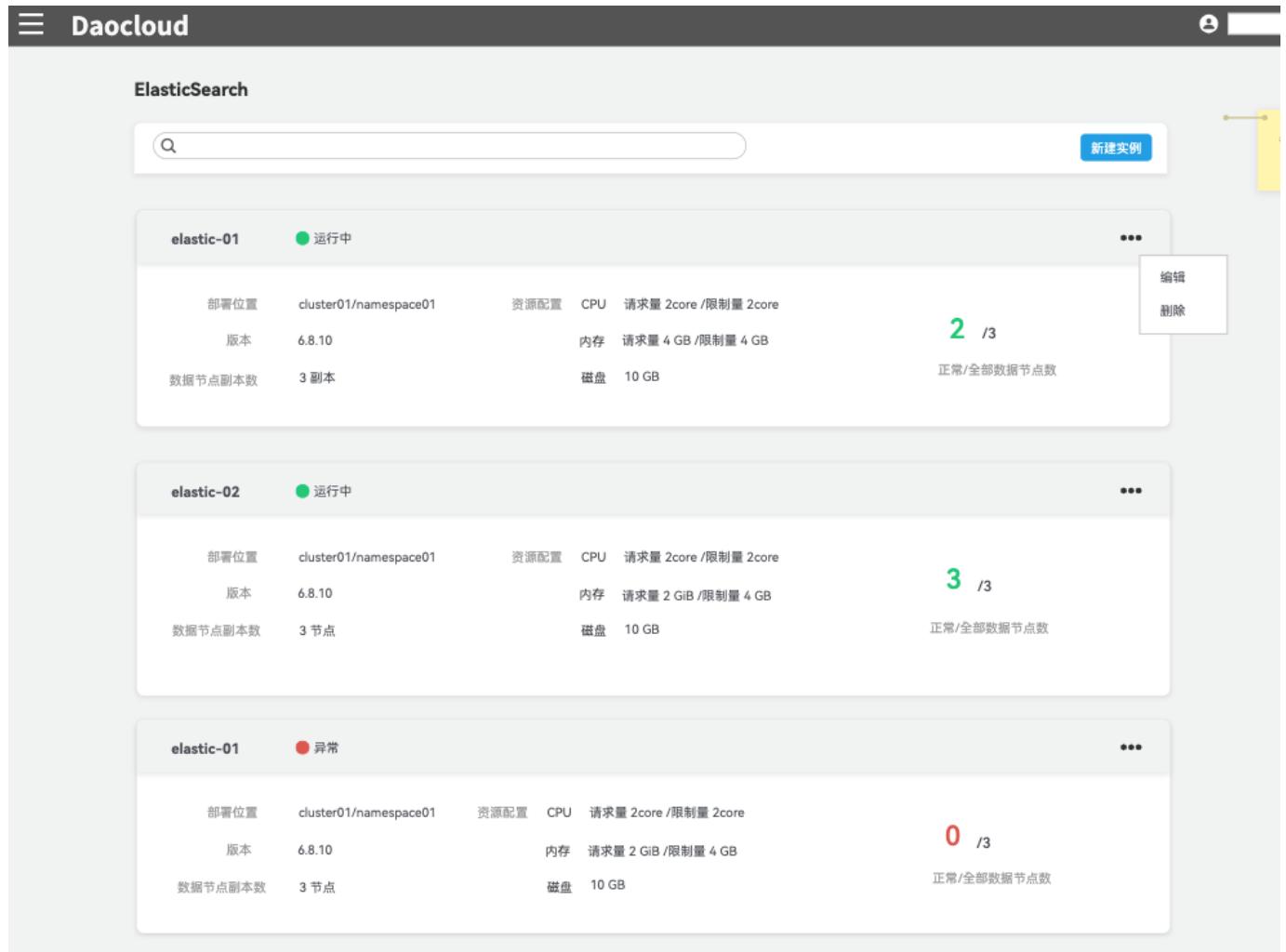
CPU and Memory

Load average

CPU usage

查看 Elasticsearch

本页说明 Elasticsearch 列表。



The screenshot shows the Daocloud interface for managing Elasticsearch instances. At the top, there's a search bar and a '新建实例' (Create New Instance) button. Below the search bar, there are three cards representing different Elasticsearch nodes:

- elastic-01** (Green Dot, Running):
 - 部署位置: cluster01/namespace01
 - 版本: 6.8.10
 - 数据节点副本数: 3 副本
 - 资源配置: CPU 请求量 2core /限制量 2core
内存 请求量 4 GB /限制量 4 GB
磁盘 10 GB
 - 状态: 正常/全部数据节点数
- elastic-02** (Green Dot, Running):
 - 部署位置: cluster01/namespace01
 - 版本: 6.8.10
 - 数据节点副本数: 3 节点
 - 资源配置: CPU 请求量 2core /限制量 2core
内存 请求量 2 GiB /限制量 4 GB
磁盘 10 GB
 - 状态: 正常/全部数据节点数
- elastic-01** (Red Dot, Abnormal):
 - 部署位置: cluster01/namespace01
 - 版本: 6.8.10
 - 数据节点副本数: 3 节点
 - 资源配置: CPU 请求量 2core /限制量 2core
内存 请求量 2 GiB /限制量 4 GB
磁盘 10 GB
 - 状态: 正常/全部数据节点数

6. Kafka

6.1 Kafka 消息队列 Release Notes

本页列出 Kafka 消息队列的 Release Notes，便于您了解各版本的演进路径和特性变化。

v0.3.0

发布日期：2023-02-23

API

- 新增 `mcamel-kafka helm-docs` 模板文件。
- 新增 `mcamel-kafka` 应用商店中的 Operator 只能安装在 `mcamel-system`。
- 新增 `mcamel-kafka` 支持 cloud shell。
- 新增 `mcamel-kafka` 支持导航栏单独注册。
- 新增 `mcamel-kafka` 支持查看日志。
- 新增 `mcamel-kafka` Operator 对接 chart-syncer。
- 修复 `mcamel-kafka` 实例名太长导致自定义资源无法创建的问题。
- 修复 `mcamel-kafka` 工作空间 Editor 用户无法查看实例密码。
- 升级 `mcamel-kafka` 升级离线镜像检测脚本。

文档

- 新增 日志查看操作说明，支持自定义查询、导出等功能。

v0.2.0

发布日期：2022-12-25

API

- 新增 `mcamel-kafka` NodePort 端口冲突提前检测。
- 新增 `mcamel-kafka` 节点亲和性配置。
- 优化 `mcamel-kafka manager` 去掉 probe，防止 kafka 没准备好不能打开 manager。
- 优化 `mcamel-kafka zooEntrance` 重新打包镜像地址为 1.0.0。

v0.1.6

发布日期：2022-11-28

API

- 改进 完善优化复制功能
- 改进 实例详情 - 访问设置，移除集群 IPv4
- 改进 中间件密码校验难度调整
- 新增 对接告警能力
- 新增 新增判断 sc 是否支持扩容并提前提示功能

- 优化 优化安装环境检测的提示逻辑 & 调整其样式
- 优化 中间件样式走查优化
- 修复 离线镜像有数字和大写无法被扫描到

v0.1.4

发布日期: 2022-11-08

API

- 修复 更新时无法校验到正确字段, 如 managerPass
- 改进 密码校验调整为 MCamel 低等密码强度
- 新增 返回是否可以更新 sc 容量的校验
- 新增 返回列表或者详情时的公共字段
- 新增 返回告警
- 新增 校验 Service 注释
- 修复 operator 用名字选择
- 修复 服务地址展示错误
- 修复 Kafka 使用 NodePort 时, 创建失败

v0.1.2

发布日期: 2022-10-28

- 新增 同步 Pod 状态到实例详情页
- 优化 workspace 界面逻辑调整
- 优化 不符合设计规范的样式调整
- 优化 password 获取逻辑调整
- 优化 cpu&内存请求量应该小于限制量逻辑调整

v0.1.1

发布日期: 2022-9-25

- 新增 支持 kafka 列表查询, 状态查询, 创建, 删除和修改
- 新增 支持 kafka-manager 对 kafka 进行管理
- 新增 支持 kafka 的指标监控, 查看监控图表
- 新增 支持 ghippo 权限联动
- 新增 `mcamel-elasticsearch` 获取用户列表接口
- 优化 更新 release note 脚本, 执行 release-process 规范

6.2 Intro

Kafka 常见术语

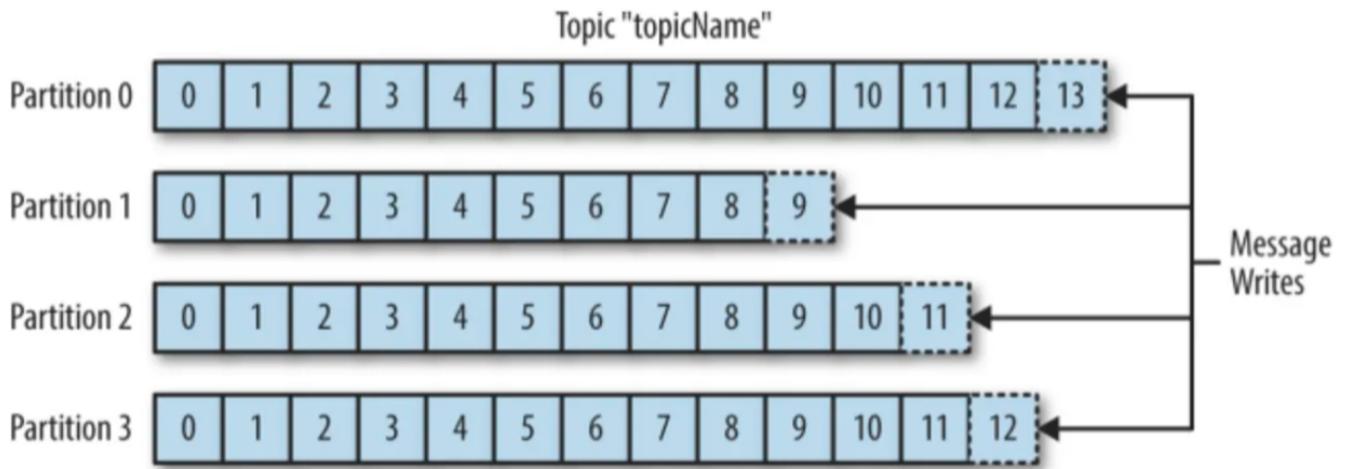
- 消息和批次

Kafka 的基本数据单元称为 message（消息），为减少网络开销，提高效率，多个消息会被放入同一批次（Batch）中后再写入。

- 主题和分区

Kafka 的消息通过 Topic（主题）进行分类，一个主题可以分为若干个 Partition（分区），一个分区就是一个提交日志（commit log）。消息以追加的方式写入分区，然后以先入先出的顺序读取。Kafka 通过分区来实现数据的冗余和伸缩性，分区可以分布在不同的服务器上，这意味着一个 Topic 可以横跨多个服务器，以提供比单个服务器更强大的性能。

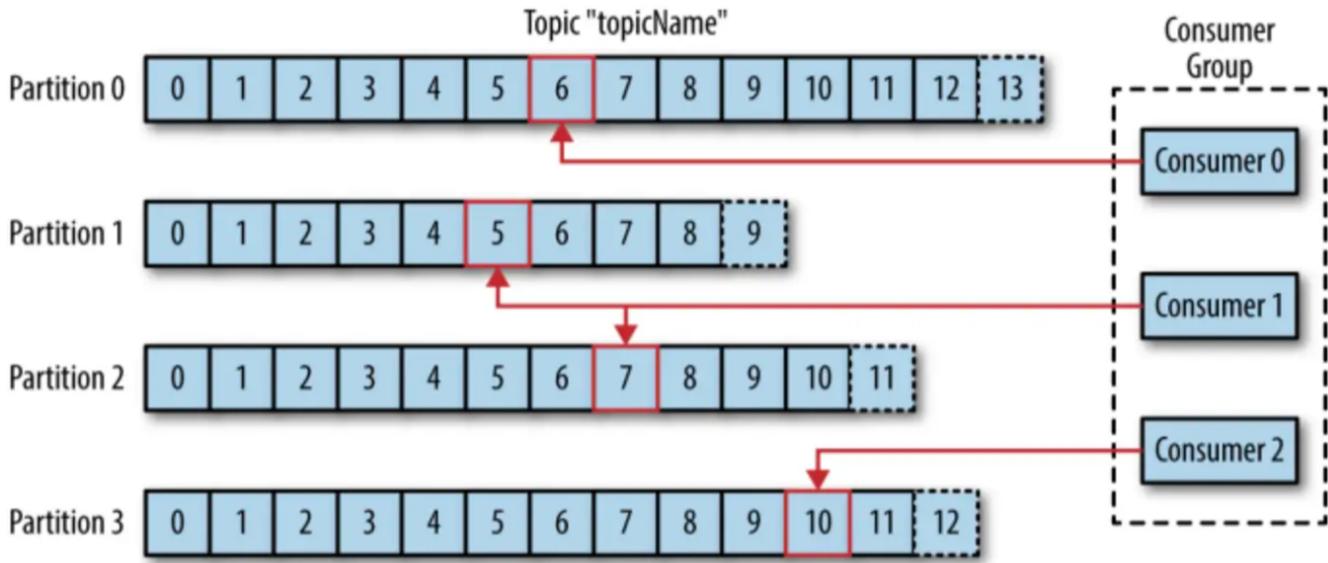
由于一个 Topic 包含多个分区，因此无法在整个 Topic 范围内保证消息的顺序性，但可以保证消息在单个分区内的顺序性。



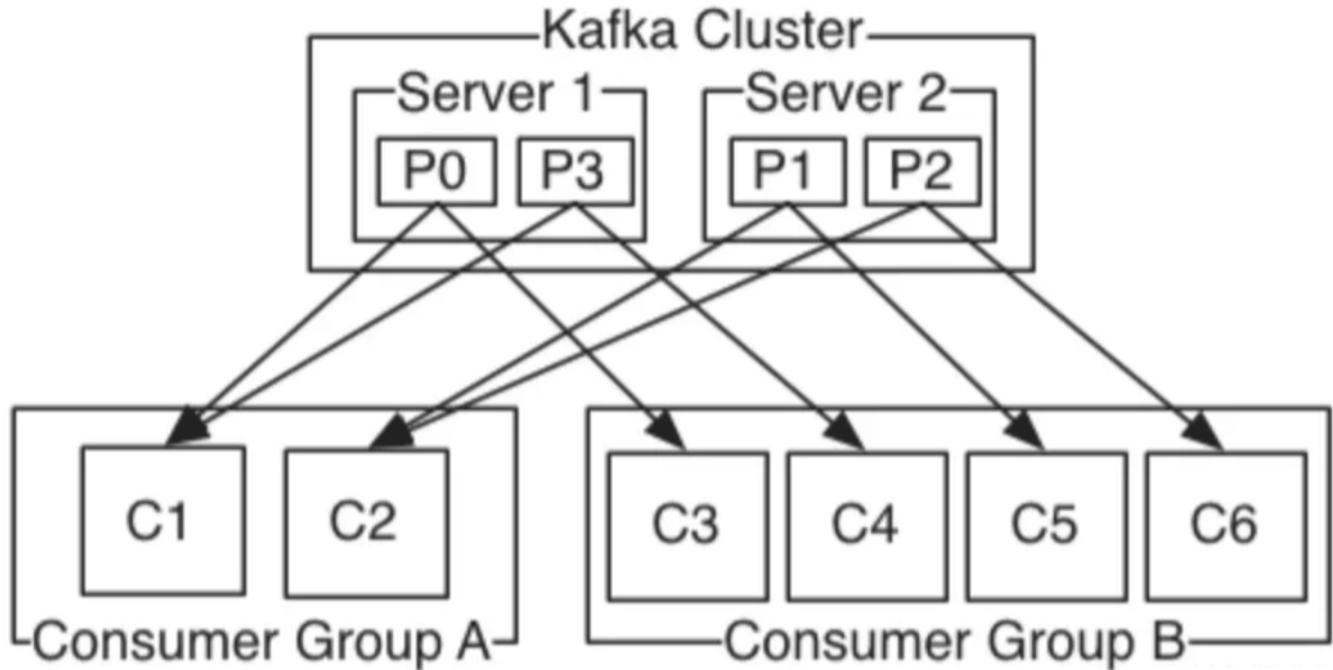
- 生产者和消费者

生产者负责创建消息。一般情况下，生产者在把消息均衡地分布到在主题的所有分区上，而并不关心消息会被写到哪个分区。如果我们想要把消息写到指定的分区，可以通过自定义分区器来实现。

消费者是消费者群组的一部分，消费者负责消费消息。消费者可以订阅一个或者多个主题，并按照消息生成的顺序来读取它们。消费者通过检查消息的偏移量（offset）来区分读取过的消息。偏移量是一个不断递增的数值，在创建消息时，Kafka 会把它添加到其中，在给定的分区里，每个消息的偏移量都是唯一的。消费者把每个分区最后读取的偏移量保存在 Zookeeper 或 Kafka 上，如果消费者关闭或者重启，它还可以重新获取该偏移量，以保证读取状态不会丢失。



一个分区只能被同一个群组中的一个消费者读取，但可以被不同群组中的多个消费者共同读取。多个群组中的消费者共同读取同一个主题时，彼此之间互不影响。

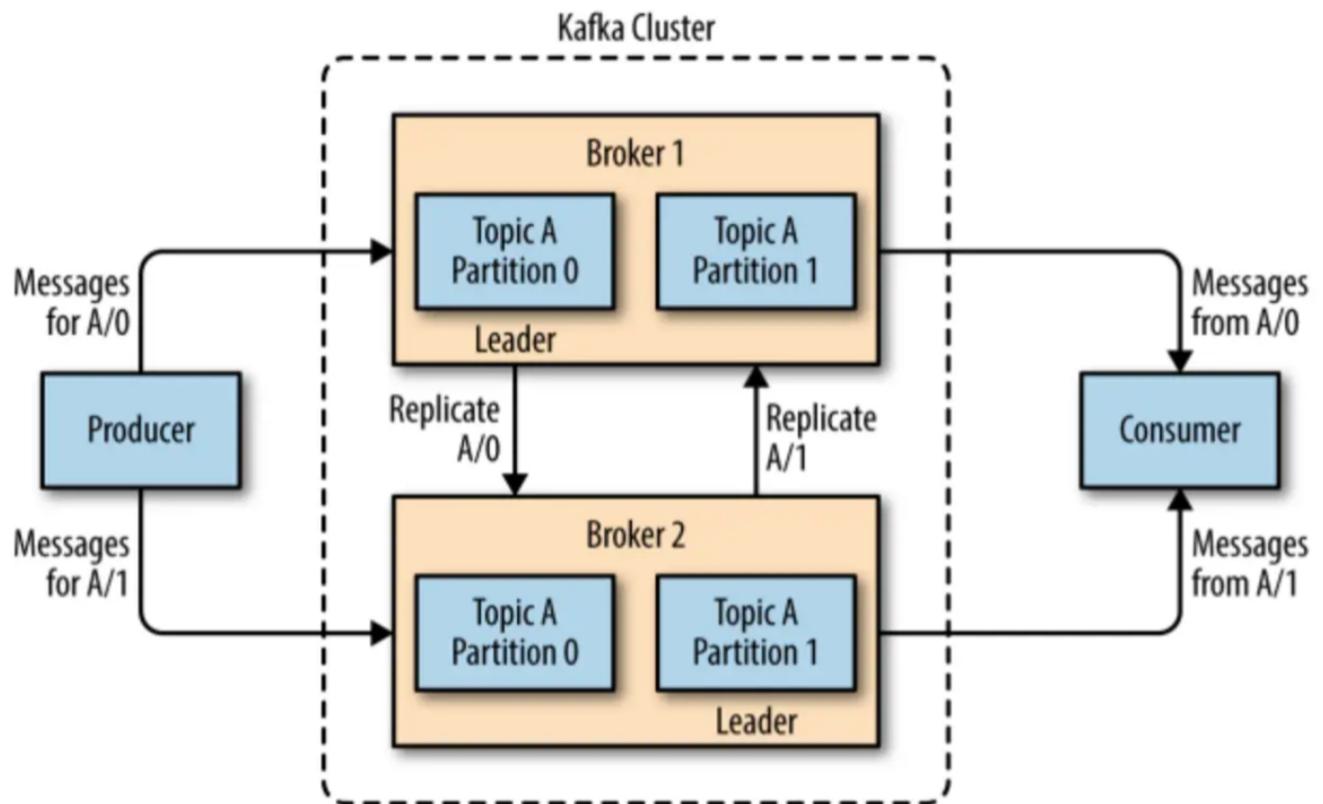


- Broker 和 Cluster

一个独立的 Kafka 服务器被称为 Broker。Broker 接收来自生产者的消息，为消息设置偏移量，并提交消息到磁盘保存。Broker 为消费者提供服务，对读取分区的请求做出响应，返回已经提交到磁盘的消息。

Broker 是集群 (Cluster) 的组成部分。每个集群都会选举出一个 Broker 作为集群控制器 (Controller)，集群控制器负责管理工作，包括将分区分配给 Broker 和监控 Broker。

在集群中，一个分区 (Partition) 从属一个 Broker，该 Broker 被称为分区的首领 (Leader)。一个分区可以分配给多个 Broker，这个时候会发生分区复制。这种复制机制为分区提供了消息冗余，如果有一个 Broker 失效，其他 Broker 可以接管领导权。



使用场景

相比支持广播、事务消息、消息路由、死信队列、优先级队列等且广泛应用于秒杀、流控、系统解耦等场景的 RabbitMQ，Kafka 消息队列适用于构建实时数据管道、流式数据处理、第三方解耦、流量削峰去谷等场景，具有大规模、高可靠、高并发访问、可扩展且完全托管的特点。

与 RabbitMQ 对比

正所谓没有最好的技术，只有最合适的技术。每个消息中间件服务都有自己的优劣，以下对 RabbitMQ 和 Kafka 做一个简单的对比。

	Kafka	RabbitMQ
性能	单节点 QPS 达百万级，吞吐量高	单节点 QPS 为万级，吞吐量低
可靠性	多副本机制，数据可靠性高	多副本机制，数据可靠性高
功能	持久化 事务消息 单分区级别的顺序性	持久化 优先级队列 延迟队列 死信队列 事务消息
消费模式	消息过滤 客户端主动拉取 消息回溯 广播消费	客户端主动拉取和服务端推送 广播消费
客户端支持	只支持 Kafka 自定义协议 采用 Scala 和 Java 编写 支持 SSL/SASL 认证和读写权限控制	支持 MQTT、STOMP 等多种协议 采用 Erlang 编写 支持 SSL/SASL 认证和读写权限控制
服务可用性	采用集群部署，分区与多副本的设计，使用单代理宕机对服务无影响，且支持消息容量的线性提升	支持集群部署，集群代理数量有多种规格
其他	消息堆积 流量控制：支持 client 和 user 级别，通过主动设置可将流控作用于生产者或消费者	消息追踪 消息堆积 多租户 流量控制：流控基于 Credit-based 算法，是内部被动触发的保护机制，作用于生产者层面

总之，Kafka 采用拉取（Pull）方式消费消息，吞吐量相对更高，适用于海量数据收集与传递场景，例如日志采集和集中分析。

而 RabbitMQ 基于 Erlang 语言开发，不利于做二次开发和维护，适用于对路由、负载均衡、数据一致性、稳定性和可靠性要求很高，对性能和吞吐量要求没那么高的场景。

典型场景

Kafka 作为一款热门的消息队列中间件，具备高效可靠的消息异步传递机制，主要用于不同系统间的数据交流和传递，在企业解决方案、金融支付、电信、电子商务、社交、即时通信、视频、物联网、车联网等众多领域都有广泛应用。

1. 异步通信

将业务中属于非核心或不重要的流程部分使用消息异步通知的方式发给目标系统，这样主业务流程无需同步等待其他系统的处理结果，从而达到系统快速响应的目的。

如网站的用户注册场景，在用户注册成功后，还需要发送注册邮件与注册短信，这两个流程使用 Kafka 消息服务通知邮件发送系统与短信发送系统，从而提升注册流程的响应速度。

2. 错峰流控与流量削峰

在电子商务或大型网站中，上下游系统处理能力存在差异，处理能力高的上游系统的突发流量可能会对处理能力低的某些下游系统造成冲击，需要提高系统可用性的同时降低系统实现的复杂性。电商大促销等流量洪流突然来袭时，可以通过队列服务堆积缓存订单等信息，在下游系统有能力处理消息的时候再处理，避免下游订阅系统因突发流量崩溃。消息队列提供亿级消息堆积能力，3 天的默认保留时长，消息消费系统可以错峰进行消息处理。

另外，在商品秒杀、抢购等流量短时间内暴增场景中，为了防止后端应用被压垮，可在前后端系统间使用 Kafka 消息队列传递请求。

3. 日志同步

在大型业务系统设计中，为了快速定位问题，全链路追踪日志，以及故障及时预警监控，通常需要将各系统应用的日志集中分析处理。

Kafka 设计初衷就是为了应对大量日志传输场景，应用通过可靠异步方式将日志消息同步到消息服务，再通过其他组件对日志做实时或离线分析，也可用于关键日志信息收集进行应用监控。

日志同步主要有三个关键部分：日志采集客户端，Kafka 消息队列以及后端的日志处理应用。

什么是 Kafka

Kafka 模块是一款基于开源软件 Kafka 提供的分布式消息队列服务。 DaoCloud 为其开发了简单易用的图形化界面，向用户提供计算、存储和带宽资源独占的 Kafka 专享实例。

Kafka 是一个拥有高吞吐、可持久化、可水平扩展，支持流式数据处理等多种特性的分布式消息流处理中间件，采用分布式消息发布与订阅机制，在日志收集、流式数据传输、在线/离线系统分析、实时监控等领域有广泛的应用。

目前支持的特性如下：

- 创建/更新/删除 Kafka 实例
- 可靠性保证机制
- 支持自定义参数配置
- 支持集群高可用
- 多语言客户端支持
- 简单易用的图形界面



[创建 Kafka 实例](#) { .md-button .md-button--primary }

6.3 User guide

配置参数

Kafka 内置了参数配置 UI 界面。

- 在消息队列页面中，点击某个名称。

Kafka 消息服务 Default

部署位置	资源配额	CPU	请求量 1 Core / 限制...	Kafka 副本数	Zookeeper 副本数
mcamel-test/default	内存	请求量 1 GB / 限制量 ...	0 / 3	0 / 3	
Kafka 版本 3.1.0	磁盘	10 GB			
访问地址 kafka01.default-kafka-servi...					

部署位置	资源配额	CPU	请求量 1 Core / 限制...	Kafka 副本数	Zookeeper 副本数
mcamel-test/default	内存	请求量 1 GB / 限制量 ...	0 / 3	0 / 3	
Kafka 版本 3.1.0	磁盘	1 GB			
访问地址 test-kafka-123.default-kafk...					

共 2 项 1 / 1 10 项

- 在左侧导航栏，点击 配置参数。

DaoCloud

kafka01 Default

概览 实例监控 配置参数

Kafka 实例: kafka01 / 配置参数

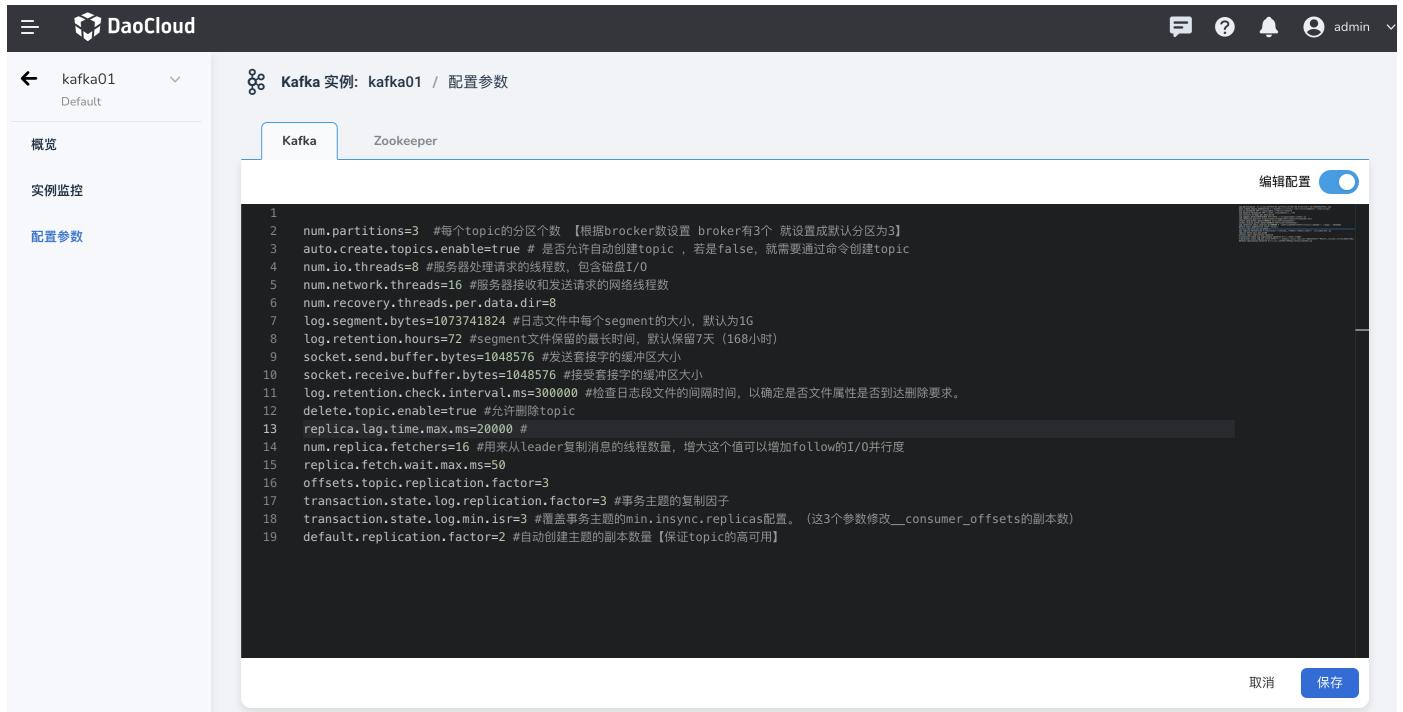
Kafka Zookeeper 编辑配置

```

1 num.partitions=3 #每个topic的分区个数 【根据broker数设置 broker有3个 就设置默认分区为3】
2 auto.create.topics.enable=true #是否允许自动创建topic ,若是false,就需要通过命令创建topic
3 num.io.threads= #服务器处理请求的线程数,包含磁盘I/O
4 num.network.threads=16 #服务器接收和发送请求的网络线程数
5 num.recovery.threads.per.data.dir=8
6 log.segment.bytes=1073741824 #日志文件中每个segment的大小,默认为1G
7 log.retention.hours=72 #segment文件保留的最长时间,默认保留7天(168小时)
8 socket.send.buffer.bytes=1048576 #发送套接字的缓冲区大小
9 socket.receive.buffer.bytes=1048576 #接受套接字的缓冲区大小
10 log.retention.check.interval.ms=300000 #检查日志段文件的间隔时间,以确定是否文件属性是否到达删除要求。
11 delete.topic.enable=true #允许删除topic
12 replica.lag.time.max.ms=20000 #
13 replica.fetch.wait.max.ms=50
14 num.replica.fetchers=16 #用来从leader复制消息的线程数量,增大这个值可以增加follow的I/O并行度
15 offsets.topic.replication.factor=3
16 transaction.state.log.replication.factor=3 #事务主题的复制因子
17 transaction.state.log.min.insync.replicas=3 #覆盖事务主题的min.insync.replicas配置。(这3个参数修改__consumer_offsets的副本数)
18 default.replication.factor=2 #自动创建主题的副本数量【保证topic的高可用】
19

```

- 点击 编辑配置 滑块开关，可以很方便地配置 Kafka 和 Zookeeper 的各项参数。



Kafka 实例: kafka01 / 配置参数

Kafka Zookeeper

编辑配置

```
1 num.partitions=3 #每个topic的分区个数 【根据brocker数设置 broker有3个 就设置成默认分区为3】
2 auto.create.topics.enable=true #是否允许自动创建topic ,若是false,就需要通过命令创建topic
3 num.io.threads=8 #服务器处理请求的线程数,包含磁盘I/O
4 num.network.threads=16 #服务器接收和发送请求的网络线程数
5 num.recovery.threads.per.data.dir=8
6 log.segment.bytes=1073741824 #日志文件中每个segment的大小,默认为1G
7 log.retention.hours=72 #segment文件保留的最长的时间,默认保留7天 (168小时)
8 socket.send.buffer.bytes=1048576 #发送套接字的缓冲区大小
9 socket.receive.buffer.bytes=1048576 #接受套接字的缓冲区大小
10 log.retention.check.interval.ms=30000 #检查日志段文件的间隔时间,以确定是否文件属性是否到达删除要求。
11 delete.topic.enable=true #允许删除topic
12 replica.lag.time.max.ms=20000 #
13 replica.fetchers=16 #用来从leader复制消息的线程数量,增大这个值可以增加follow的I/O并行度
14 replica.fetch.wait.max.ms=50
15 offsets.topic.replication.factor=3
16 transaction.state.log.replication.factor=3 #事务主题的复制因子
17 transaction.state.log.min.isr=3 #覆盖事务主题的min.insync.replicas配置。 (这3个参数修改__consumer_offsets的副本数)
18 default.replication.factor=2 #自动创建主题的副本数量【保证topic的高可用】
19
```

取消 保存

4. 点击 保存，参数将立即生效。

创建 Kafka

在 Kafka 消息队列中，执行以下操作创建 Kafka 实例。

1. 在右上角点击 新建实例 。

2. 在创建 Kafka 实例页面中，设置基本信息后，点击下一步。

3. 配置规格后，点击下一步。

- 版本：Kafka 的版本号，当前仅支持 Kafka 3.1.0。
- 副本数：支持 1、3、5、7 副本数。
- 资源配额：根据实际情况选择规则。
- 存储卷：选择 Kafka 实例的存储卷和储存空间总量。

选择版本: 版本 * 3.1.0

规格配置

- Kafka 副本数 * 3 副本数建议设置为大于 3 个, 保证高可用。
- Kafka CPU 配额 * 请求量 1 Core 限制量 1 Core
- Kafka 内存配额 * 请求量 1 GB 限制量 1 GB
- Zookeeper 副本数 * 3 副本数建议设置为大于 3 个, 保证高可用。
- Zookeeper CPU 配额 * 请求量 1 Core 限制量 1 Core

取消 上一步 下一步

4. 服务设置后, 点击下一步。

- 访问方式: 可以选择集群内访问还是 Nodeport 访问。
- 服务设置: 设置连接 Kafka 实例的用户名、密码。

访问类型 * 集群内访问(ClusterIP) 节点端口(NodePort)

端口配置

服务端口	9092
容器内部服务端口, 无需变更	

注释 + 添加

访问设置

用户名 * adminyhf
4~64个字符之间, 必须以字母开头, 区分大小写, 可以包含字母、数字、中划线或者下划线, 不能包含其他特殊字符。

取消 上一步 下一步

5. 确认实例配置信息无误, 点击确认 完成创建。

创建 Kafka 实例

基本信息

Kafka 实例名称: kafka01
部署位置: mcamel-test/default

规格配置

版本: 3.1.0
Kafka 副本数: 3 副本
Zookeeper 副本数: 3 副本
Kafka CPU 配额: 请求量 1 Core / 限制量 1 Core
Kafka 内存配额: 请求量 1 GB / 限制量 1 GB
Zookeeper CPU 配额: 请求量 1 Core / 限制量 1 Core
Zookeeper 内存配额: 请求量 1 GB / 限制量 1 GB

取消 上一步 确认

6. 在实例列表页查看实例是否创建成功。刚创建的实例状态为未就绪，等几分钟后该状态变为运行中。

创建实例成功

Kafka 消息服务 Default

kafka01 ● 未就绪

部署位置: mcamel-test/default 资源配额: CPU 请求量 1 Core / 限制... 0 / 3
Kafka 版本: 3.1.0 内存 请求量 1 GB / 限制量 ... 0 / 3
访问地址: kafka01.default-kafka-servi... 磁盘 10 GB Kafka 副本数 Zookeeper 副本数

test-kafka-123 ● 未就绪

部署位置: mcamel-test/default 资源配额: CPU 请求量 1 Core / 限制... 0 / 3
Kafka 版本: 3.1.0 内存 请求量 1 GB / 限制量 ... 0 / 3
访问地址: test-kafka-123.default-kafk... 磁盘 1 GB Kafka 副本数 Zookeeper 副本数

共 2 项 1 / 1 10 项

删除 Kafka

如果想要删除一个消息队列，可以执行如下操作：

- 在消息队列中，点击右侧的 按钮，在弹出菜单中选择 **删除实例**。

kafka01 ● 未就绪

部署位置: mcamel-test/default 资源配额: CPU 请求量 1 Core / 限制... 0 / 3
Kafka 版本: 3.1.0 内存 请求量 1 GB / 限制量 ...
访问地址: kafka01.default-kafka-servi... 磁盘 10 GB Kafka 副本数 Zookeeper 副本数

更新实例

删除实例 (highlighted with a red box)

test-kafka-123 ● 未就绪

部署位置: mcamel-test/default 资源配额: CPU 请求量 1 Core / 限制... 0 / 3 0 / 3
Kafka 版本: 3.1.0 内存 请求量 1 GB / 限制量 ...
访问地址: test-kafka-123.default-kafk... 磁盘 1 GB Kafka 副本数 Zookeeper 副本数

共 2 项 1 / 1 10 项

- 在弹窗中输入该消息队列的名称，确认无误后，点击 **删除** 按钮。

确认删除实例 kafka01 吗?

确认删除实例 kafka01 吗? 删除后对应的数据将会全部丢失, 请谨慎操作。

请输入 kafka01

kafka01

删除 取消

kafka01 ● 未就绪

部署位置: mcam... 资源配额: CPU 请求量 1 Core / 限制... 0 / 3
Kafka 版本: 3.1.0 内存 请求量 1 GB / 限制量 ...
访问地址: kafka01.default-kafka-servi... 磁盘 10 GB Kafka 副本数 Zookeeper 副本数

test-kafka-123 ● 未就绪

部署位置: mcamel-test/default 资源配额: CPU 请求量 1 Core / 限制... 0 / 3 0 / 3
Kafka 版本: 3.1.0 内存 请求量 1 GB / 限制量 ...
访问地址: test-kafka-123.default-kafk... 磁盘 1 GB Kafka 副本数 Zookeeper 副本数

共 2 项 1 / 1 10 项

!!! warning

删除实例后，该实例相关的所有消息也会被全部删除，请谨慎操作。

实例监控

Kafka 内置了 Prometheus 和 Grafana 监控模块。

- 在消息队列页面中，点击某个名称。

The screenshot shows a list of Kafka instances. The first instance, **kafka01**, is highlighted with a red border. Its details are displayed below:

- 部署位置: mcamel-test/default
- Kafka 版本: 3.1.0
- 访问地址: kafka01.default-kafka-servi...
- 资源配额: CPU 请求量 1 Core / 限制... (0 / 3)
- 内存: 请求量 1 GB / 限制量 ... (0 / 3)
- 磁盘: 10 GB
- Kafka 副本数: 0 / 3
- Zookeeper 副本数: 0 / 3

The second instance, **test-kafka-123**, is also shown with similar metrics.

At the bottom, there is a pagination bar indicating "共 2 项" (2 items) and a dropdown for "10 项" (10 items).

- 在左侧导航栏，点击 实例监控，可以接入监控模块。

The screenshot shows the **Kafka 实例: kafka01 / 实例监控** page. On the left, there is a sidebar with navigation links: 概览, 实例监控 (selected), and 配置参数.

The main area displays the **Kafka Overview** with several key metrics:

指标	状态
Brokers Online	N/A
Active Contr...	N/A
Unclean Lea...	N/A
Online Replic...	N/A
Under Replic...	N/A
Partitions at ...	0
Partitions un...	0
Offline Partit...	N/A

Below this, there is a section titled **Kafka** containing various time-series charts:

- Memory Usage:** Y-axis ranges from -1 B to 1 B. The chart shows "No data" for the period 09:30 to 10:00.
- CPU Usage:** Y-axis ranges from -1 to 1. The chart shows "No data" for the period 09:30 to 10:00.
- Available Disk Space:** Y-axis ranges from -1 B to 1 B. The chart shows "No data" for the period 09:30 to 10:00.
- Open File Descriptors:** Y-axis ranges from 0 to 1. The chart shows "No data" for the period 09:30 to 10:00.
- JVM Memory Used:** Y-axis ranges from 1 B to -1 B. The chart shows "No data" for the period 09:30 to 10:00.
- JVM GC Time:** Y-axis ranges from 1 ms to -1 ms. The chart shows "No data" for the period 09:30 to 10:00.
- JVM GC Count:** Y-axis ranges from 1 to -1. The chart shows "No data" for the period 09:30 to 10:00.
- JVM Thread Count:** Y-axis ranges from 1 to -1. The chart shows "No data" for the period 09:30 to 10:00.

查看 Kafka 日志

操作步骤

通过访问每个 Kafka 的实例详情，页面可以支持查看 Kafka 的日志。

- 在 Kafka 实例列表中，选择想要查看的日志，点击 实例名称 进入到实例详情页面。

The screenshot shows the Kafka instance list interface. At the top, there's a search bar and a 'New Instance' button. Below the header, a table lists instances. One instance, 'test-aaa', is highlighted with a red box. To its right, there are resource details: CPU 1 Core, Memory 1 GB, and Disk 10 GB. To the right of the table, two status indicators show '3 / 3' for both Kafka and Zookeeper副本数 (replicas). At the bottom, there's a pagination bar showing 1/1 item.

- 在实例的左侧菜单栏，会发现有一个日志查看的菜单栏选项。

The screenshot shows the Kafka instance detail page for 'test-aaa'. On the left, a sidebar has several options: 'Logs' (highlighted with a red box), 'Metrics', 'Configuration Parameters', and 'Logs' again. The main content area displays basic information about the instance, including its status (Running), deployment location (mcamel-test/mcamel-system), creation time (2023-02-08 14:50), version (3.1.0), and access address (test-aaa-kafka-service.mcamel-sys...). It also shows resource allocation (CPU 1 Core, Memory 1 GB, Disk 10 GB) and monitoring settings (NodePort 10.6.51.101:30181, kafka user). At the bottom, there's a section for monitoring alerts.

- 点击 日志查看 即可进入到日志查看页面（Insight 日志查看）。

日志查看说明

在日志查看页面，我们可以很方便的进行日志查看，常用操作说明如下：

- 支持 自定义日志时间范围，在日志页面右上角，可以方便地切换查看日志的时间范围（可查看的日志范围以 可观测系统设置内保存的日志时长为准）
- 支持 关键字检索日志，左侧检索区域支持查看更多的日志信息
- 支持 日志量分布查看，中上区域柱状图，可以查看在时间范围内的日志数量分布
- 支持 查看日志的上下文，点击右侧 上下文 图标即可
- 支持 导出日志

The screenshot shows the DaoCloud observability platform's log search feature. The left sidebar has a tree structure with '可观测性' (Observability) as the root, followed by '概览', '仪表盘', '资源监控', '场景监控', '数据查询', '指标查询', '日志查询' (selected), '链路查询', '告警中心', '采集管理', and '系统管理'. The main area is titled '日志查询' (Log Search). It includes a search bar with placeholder '请输入日志内容' (Enter log content) (2), a dropdown for '集群' (Cluster) set to 'kpanda-global-cluster', and a histogram showing the distribution of log entries over time from 00:34 to 00:48. Below the histogram is a table of logs (日志) (4) with columns '时间' (Time) and '日志' (Log). The first log entry is highlighted with a red box and shows a detailed log message from 'elastic-operator-0 manager'. A context menu (5) is open over this log entry, containing options like '查看上下文' (View Context), '复制' (Copy), and '下载' (Download).

时间	日志
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.268Z","log.logger":"elasticsearch-controller","message":"Ending reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters","took":0.466213453}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.242Z","log.logger":"zen2","message":"Ensuring no voting exclusions are set","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"} 5
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.196Z","log.logger":"transport","message":"Skipping pod because it has no IP yet","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","pod_name":"mcamel-common-es-cluster-masters-es-masters-2"}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:54.802Z","log.logger":"elasticsearch-controller","message":"Starting reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"}

更新 Kafka

如果想要更新或修改 Kafka 的资源配置，可以按照本页说明操作。

- 在消息队列中，点击右侧的 **...** 按钮，在弹出菜单中选择 **更新实例**。

kafka01 ● 未就绪

部署位置 mcamel-test/default 资源配额 CPU 请求量 1 Core / 限制... 0 / 3

Kafka 版本 3.1.0 内存 请求量 1 GB / 限制量 ... Kafka 副本数

访问地址 kafka01.default-kafka-servi... 磁盘 10 GB Zookeeper 副本数

test-kafka-123 ● 未就绪

部署位置 mcamel-test/default 资源配额 CPU 请求量 1 Core / 限制... 0 / 3

Kafka 版本 3.1.0 内存 请求量 1 GB / 限制量 ... Kafka 副本数

访问地址 test-kafka-123.default-kafk... 磁盘 1 GB Zookeeper 副本数

共 2 项 1 / 1 10 项

- 修改基本信息后，点击 **下一步**。

更新实例 kafka01

1 基本信息 2 规格配置 3 服务设置

基本信息

Kafka 实例名称 kafka01

描述

描述信息不超过 256 个字符。

部署位置

集群 mcamel-test

命名空间 default

取消 下一步

- 修改规格配置后，点击 **下一步**。



4. 修改服务设置后，点击确认。



5. 返回消息队列，屏幕右上角将显示消息：| 更新实例成功。

DaoCloud

Kafka 消息服务 Default

更新实例成功

搜索 新建实例

kafka01 未就绪

部署位置	mcamel-test/default	资源配额	CPU 请求量 1 Core / 限制...	0 / 3	内存 请求量 1 GB / 限制量 ...	0 / 3
Kafka 版本	3.1.0					
访问地址	kafka01.default-kafka-servi...		磁盘 10 GB		Kafka 副本数	Zookeeper 副本数

test-kafka-123 未就绪

部署位置	mcamel-test/default	资源配额	CPU 请求量 1 Core / 限制...	0 / 3	内存 请求量 1 GB / 限制量 ...	0 / 3
Kafka 版本	3.1.0					
访问地址	test-kafka-123.default-kafk...		磁盘 1 GB		Kafka 副本数	Zookeeper 副本数

共 2 项 1 / 1 10 项

部署位置: mcamel-test/default 资源配额: CPU 请求量 1 Core / 限制... 0 / 3 内存 请求量 1 GB / 限制量 ... 0 / 3 Kafka 版本: 3.1.0 访问地址: kafka01.default-kafka-servi... 磁盘: 10 GB Kafka 副本数 Zookeeper 副本数

部署位置: mcamel-test/default 资源配额: CPU 请求量 1 Core / 限制... 0 / 3 内存 请求量 1 GB / 限制量 ... 0 / 3 Kafka 版本: 3.1.0 访问地址: test-kafka-123.default-kafk... 磁盘: 1 GB Kafka 副本数 Zookeeper 副本数

查看消息队列

本节说明如何查看 Kafka 消息队列。

1. 在消息队列页面中，点击某个名称。

The screenshot shows a list of Kafka instances. The first instance, 'kafka01', is highlighted with a red border. Its details are displayed below:

部署位置	mcamel-test/default	资源配额	CPU 请求量 1 Core / 限制... 内存 请求量 1 GB / 限制量 ... 磁盘 10 GB	0 / 3	0 / 3
Kafka 版本	3.1.0			Kafka 副本数	Zookeeper 副本数
访问地址	kafka01.default-kafka-servi...				

The second instance, 'test-kafka-123', is also shown with similar details.

At the bottom, there is a pagination bar showing '共 2 项' (2 items), a page number '1 / 1', and a dropdown for '10 项' (10 items).

2. 进入消息队列概览，查看基本信息、访问设置、资源配置和 Pod 列表等信息。

The screenshot shows the detailed view for the 'kafka01' instance. It includes sections for basic information, access settings, resource allocation, and pod lists.

基本信息

实例名称	kafka01	运行状态	● 未就绪	部署位置	mcamel-test/default	创建时间	2022-10-10 09:56
版本	3.1.0	访问方式	集群内访问(ClusterIP)	访问地址	kafka01.default-kafka-service.svc:9...		

访问设置

NodePort	adminyhf	用户名	adminyhf
访问方式	10.103.27.31	访问地址	http://10.6.51.101:32148
集群 IPv4			

资源配置

CPU	限制量 1 Core 请求量 1 Core	内存	限制量 1 GB 请求量 1 GB	磁盘	存储量 10 GB
-----	--------------------------	----	----------------------	----	-----------

容器组列表

容器组	节点类型	运行状态	容器组 IP	重启次数	CPU 使用量/限...	内存使用量/限...	创建时间
kafka01-mana...	Manager	● 运行中	10.244.23.96	6	0 Core / 0.2 Core	0 GB / 0.2 GB	2022-10-10 0...

At the bottom, there is a pagination bar showing '共 1 项' (1 item), a page number '1 / 1', and a dropdown for '10 项' (10 items).

7. Minio

7.1 MinIO 对象存储 Release Notes

本页列出 MinIO 对象存储的 Release Notes，便于您了解各版本的演进路径和特性变化。

v0.3.0

发布日期：2023-02-23

API

- 新增 `mcamel-minio helm-docs` 模板文件。
- 新增 `mcamel-minio` 应用商店中的 Operator 只能安装在 `mcamel-system`。
- 新增 `mcamel-minio` 支持 cloud shell。
- 新增 `mcamel-minio` 支持导航栏单独注册。
- 新增 `mcamel-minio` 支持查看日志。
- 新增 `mcamel-minio` Operator 对接 chart-syncer。
- 修复 `mcamel-minio` 实例名太长导致自定义资源无法创建的问题。
- 修复 `mcamel-minio` 工作空间 Editor 用户无法查看实例密码。
- 升级 `mcamel-minio` 升级离线镜像检测脚本。

文档

- 新增 日志查看操作说明，支持自定义查询、导出等功能。

v0.2.0

发布日期：2022-12-25

API

- 新增 `mcamel-minio` NodePort 端口冲突提前检测。
- 新增 `mcamel-minio` 节点亲和性配置。
- 修复 `mcamel-minio` 修复单实例的时候，状态显示异常的问题。
- 修复 `mcamel-minio` 创建实例时未校验名字。

UI

- 优化 `mcamel-minio` 取消认证信息输入框。

v0.1.4

发布日期: 2022-11-28

- 改进 更新获取前端的接口，增加 sc 列表是否可以扩容
- 改进 密码校验调整为 MCamel 中等密码强度
- 新增 创建 MinIO 集群时配置 Bucket
- 新增 返回列表或者详情时的公共字段
- 新增 返回告警列表
- 新增 校验 Service 注释
- 修复 创建 MinIO 时，密码校验从 between 调整为 length
- 改进 完善优化复制功能
- 改进 实例详情 - 访问设置，移除 集群 IPv4
- 改进 中间件密码校验难度调整
- 新增 minio 支持创建的时候内置 BUCKET 创建
- 新增 对接告警能力
- 新增 新增判断 sc 是否支持扩容并提前提示功能
- 优化 优化安装环境检测的提示逻辑&调整其样式
- 优化 中间件样式走查优化

v0.1.2

发布日期: 2022-11-08

- 新增 获取用户列表接口
- 新增 minio 实例创建
- 新增 minio 实例的修改
- 新增 minio 实例的删除
- 新增 minio 实例的配置修改
- 新增 minio 实例支持 nodeport 的 svc
- 新增 minio 实例的监控数据导出
- 新增 minio 实例的监控查看
- 新增 多租户全局管理的对接
- 新增 mcamel-minio-ui 的创建/列表/修改/删除/查看
- 新增 APIServer/UI 支持 mtls
- 修复 单例模式下，只有一个 pod，修复 grafana 无法获取数据问题
- 优化 完善了计算器功能

7.2 Intro

MinIO 常见术语

本页说明 MinIO 相关概念。

概念	含义
Object 对象	存储的基本对象，例如文件、图片等
Bucket 桶	用于存储 Object 的逻辑空间，相互之间隔离，类似于系统中的顶层文件夹
Drive 驱动磁盘	即存储数据的磁盘，所有的对象数据都存储在 Drive 中
Set 磁盘集	即一组 Drive 的集合，分布式部署根据集群规模自动划分一个或多个 Set

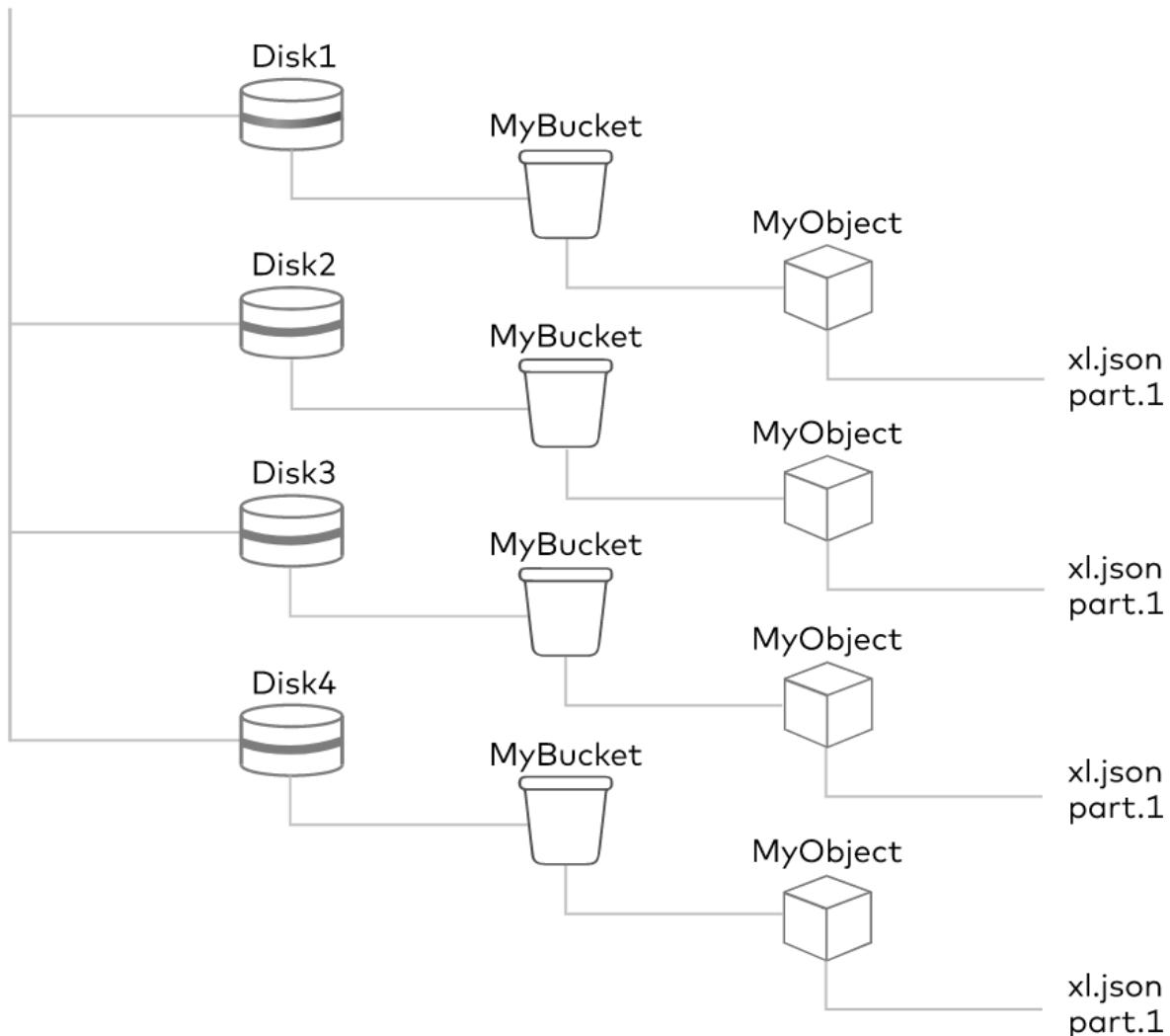
- 一个 Object 存储在一个 Set
- 一个集群划分为多个 Set
- 一个 Set 中的 Drive 尽可能分布在不同的节点上
- 一个 Set 包含的 Drive 数量是固定的，默认由系统根据集群规模自动计算

更多概念的阐述如下所述。

擦除码 Erasure Code

MinIO 使用按对象的嵌入式擦除编码保护数据，该编码以汇编代码编写，可提供最高的性能。MinIO 使用 Reed-Solomon 代码将对象划分为 $n/2$ 个数据块和 $n/2$ 个奇偶校验块，也可以将对象配置为任何所需的冗余级别。这意味着在 12 个驱动器设置中，将一个对象分片为 6 个数据和 6 个奇偶校验块。即使丢失了多达 5 个 ($(n / 2) - 1$) 个驱动器（无论是奇偶校验还是数据），仍然可以从其余驱动器可靠地重建数据。MinIO 的实现可确保即使丢失或无法使用多个设备，也可以读取对象或写入新的对象。最后，MinIO 的擦除代码位于对象级别，并且可以一次修复一个对象。

export-xl



Bitrot 保护

无声的数据损坏或 Bitrot 是磁盘驱动器面临的严重问题，导致数据在用户不知情的情况下损坏。原因多种多样：驱动器老化，电流尖峰，磁盘固件错误，虚假写入，读/写方向错误，驱动程序错误，意外覆盖等。但结果是一样的：数据泄漏。

MinIO 对高速哈希算法的优化实现可确保它永远不会读取损坏的数据，可以实时捕获和修复损坏的对象。通过在 READ 上计算哈希值，并在 WRITE 上从应用程序、整个网络以及内存/驱动器的哈希值来确保端到端的完整性。该实现旨在提高速度，并且可以在 Intel CPU 的单个内核上实现超过每秒 10 GB 的哈希速度。

OBJECT ERASURE-CODED OVER 16 DRIVES
Tolerates up to any 8 disk failures

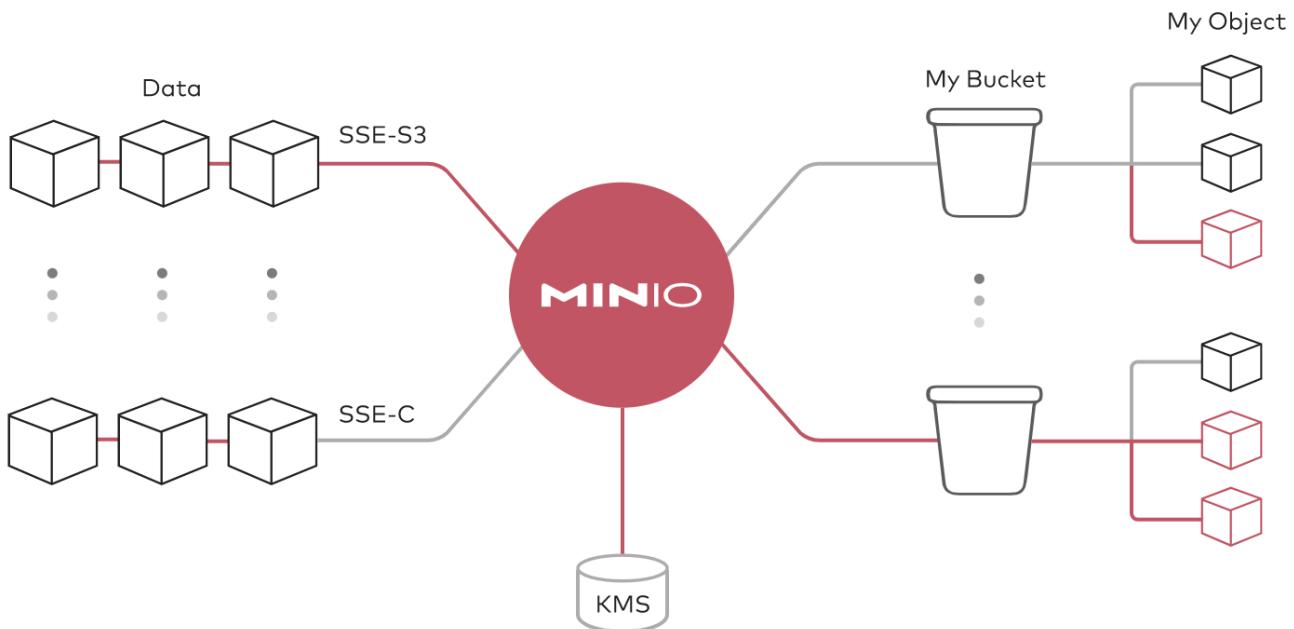


加密

加密活跃数据与保护静态数据不同。MinIO 支持多种复杂的服务器端加密方案，以保护所有位置的存储数据。MinIO 的方法可确保机密性、完整性和真实性，而性能开销却可以忽略不计。使用 AES-256-GCM、ChaCha20-Poly1305 和 AES-CBC 支持服务器端和客户端加密。加密的对象使用 AEAD 服务器端加密进行了防篡改。此外，MinIO 与所有常用的密钥管理解决方案（例如 HashiCorp Vault）兼容并经过测试。

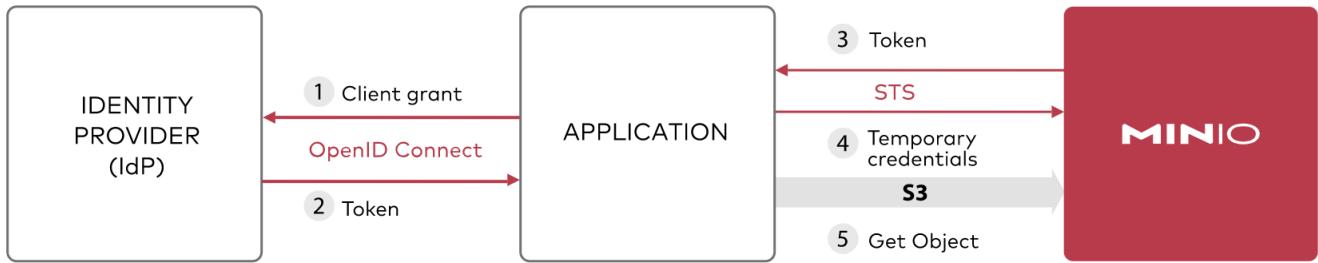
MinIO 使用密钥管理系统（KMS）支持 SSE-S3。如果客户端请求 SSE-S3 或启用了自动加密，则 MinIO 服务器会使用唯一的对象密钥对每个对象进行加密，该对象密钥受 KMS 管理的主密钥保护。由于开销极低，因此可以为每个应用程序和实例打开自动加密。

启用 WORM 后，MinIO 会禁用所有可能会使对象数据和元数据发生变异的 API。这意味着一旦写入数据就可以防止篡改。这对于许多不同的法规要求具有实际应用。



身份认证和管理

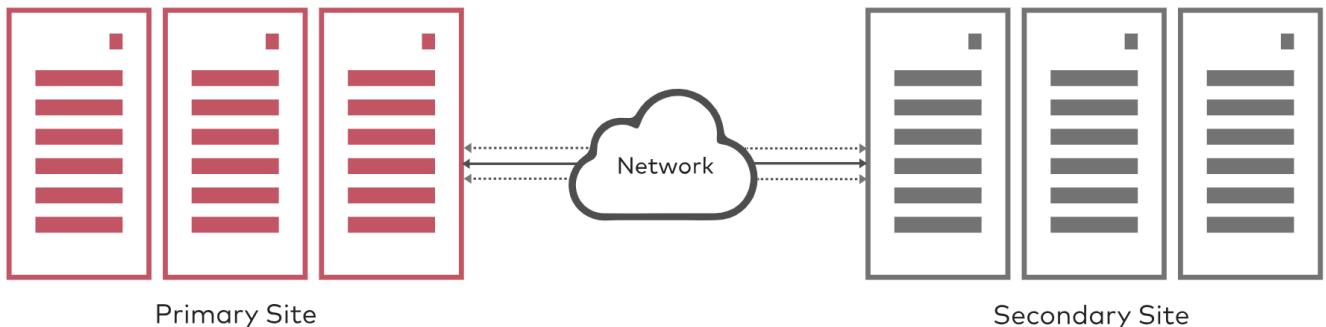
MinIO 支持身份管理中最先进的标准，并与 OpenID connect 兼容提供商以及主要的外部 IDP 供应商集成。这意味着访问是集中的，密码是临时的和轮换的，而不是存储在配置文件和数据库中。此外，访问策略是细粒度的且高度可配置的，这意味着支持多租户和多实例部署变得简单。



连续复制

传统复制方法的挑战在于它们无法有效扩展到几百 TB。话虽如此，每个人都需要一种复制策略来支持灾难恢复，并且该策略需要跨越地域、数据中心和云。MinIO 的连续复制旨在用于大规模的跨数据中心部署。通过利用 Lambda 计算通知和对象元数据，它可以高效、快速地计算增量。

Lambda 通知确保与传统的批处理模式相反，更改可以立即传播。连续复制意味着即使发生高动态数据集，如果发生故障，数据丢失也将保持在最低水平。最后，就像 MinIO 所做的一样，连续复制是多厂商的，这意味着您的备份位置可以是从 NAS 到公共云的任何位置。



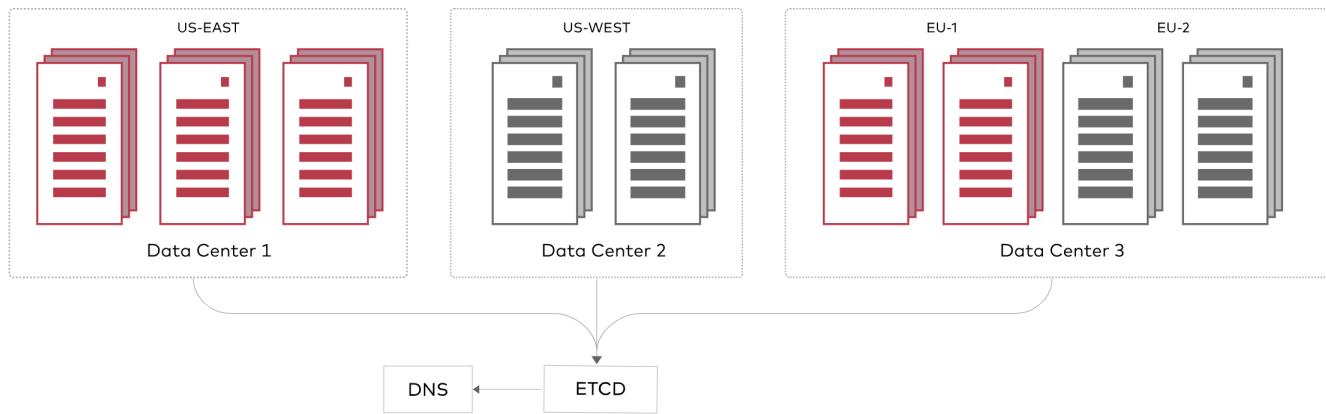
```
mc mirror --watch --recursive dc1/ dc2/
```

全局一致性

现代企业到处都有数据。MinIO 允许将这些各种实例组合在一起以形成统一的全局命名空间。具体来说，最多可以将 32 个 MinIO 服务器组合成一个分布式模式集，并且可以将多个分布式模式集组合成一个 MinIO 服务器联合。每个 MinIO Server Federation 都提供统一的管理员和命名空间。

MinIO Federation Server 支持无限数量的分布式模式集。

这种方法的影响在于，对象存储可以为大型的、地理上分散的企业进行大规模扩展，同时保留从以下位置容纳各种应用程序（S3 Select、MinSQL、Spark、Hive、Presto、TensorFlow、H2O）的能力。具有单一控制台。

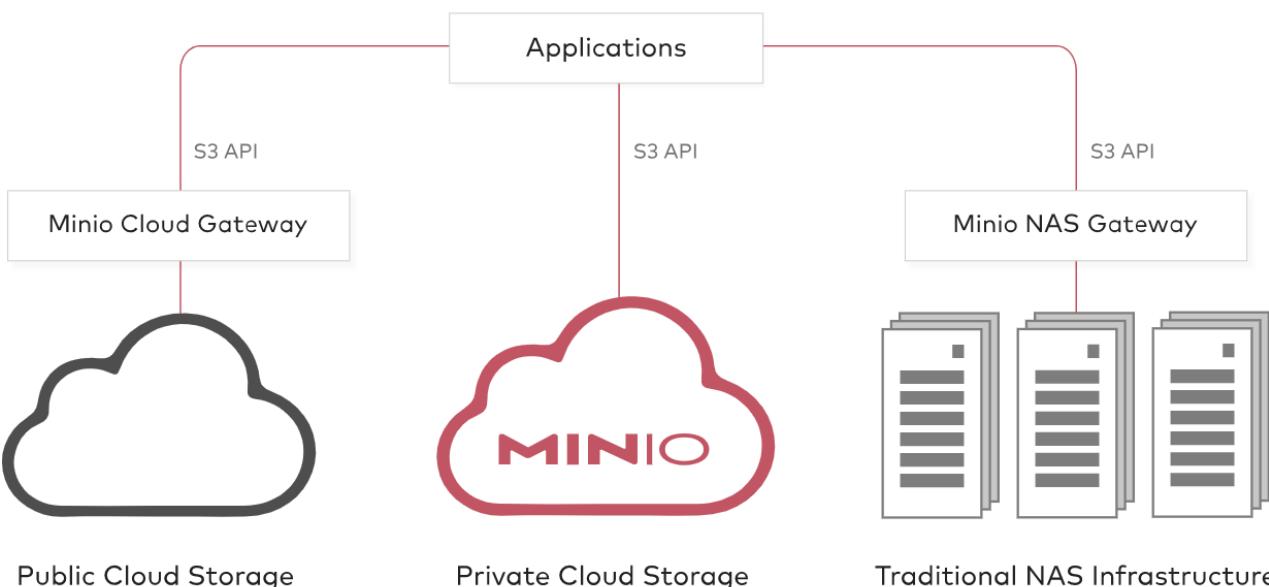


多云网关

所有企业都在采用多云策略，这也包括私有云。因此，您的裸机虚拟化容器和公共云服务（包括 Google、Microsoft 和阿里等非 S3 提供商）必须看起来完全相同。尽管现代应用程序具有高度的可移植性，但为这些应用程序提供支持的数据却并非如此。

MinIO 应对的主要挑战是：无论数据位于何处都都能使数据可用。MinIO 在裸机、网络连接存储和每个公共云上运行。更重要的是，MinIO 通过 Amazon S3 API 从应用程序和管理角度确保您对数据的看法完全相同。

MinIO 可以走得更远，使您现有的存储基础架构与 Amazon S3 兼容。其影响是深远的。现在，组织可以从文件到块真正统一其数据基础架构，所有这些都显示为可通过 Amazon S3 API 访问的对象，而无需迁移。



使用场景

MinIO 是一个基于 Apache License v2.0 开源协议的对象存储服务。它兼容 AWS S3 云存储服务接口，非常适合于存储大容量非结构化的数据，例如图片、视频、日志文件、备份数据和容器/虚拟机镜像等，而一个对象文件可以是任意大小，从 KB 到最大 TB 不等。

常见的使用场景有：

- 网盘：海量文件
- 社交网站：海量图片
- 电商网站：海量商品图片
- 视频网站：海量视频文件

每个 MinIO 集群都是分布式 MinIO 服务器的集合，每个节点一个进程。MinIO 作为单个进程在用户空间中运行，并使用轻量级的协同例程来实现高并发。将驱动器分组到擦除集（默认情况下，每组 16 个驱动器），然后使用确定性哈希算法将对象放置在这些擦除集上。

MinIO 专为大规模、多数据中心云存储服务而设计。每个租户都运行自己的 MinIO 集群，该集群与其他租户完全隔离，从而使租户能够免受升级、更新和安全事件的任何干扰。每个租户通过联合跨地理区域的集群来独立扩展。

什么是 MinIO

MinIO 是一款热门、轻量、开源的对象存储方案，完美兼容 AWS S3 协议，友好支持 K8s。MinIO 是专为 AI 等云原生工作负载而设计的，可以为高性能的云原生数据机器学习、大数据分析、海量存储的基础架构等提供数据工作负载。

DCE 的 MinIO 模块是基于开源 MinIO 构建的对象存储中间件，即插即用。DaoCloud 为其开发了简单易用的图形化界面，向用户提供计算、存储和带宽资源独占的 MinIO 专享实例。

MinIO 之所以广受欢迎，是因为其具有以下特点：

1. 高性能。

MinIO 是全球领先的对象存储服务先锋，目前在全世界有数百万的用户。在标准硬件上，读/写速度高达每秒几百 GB。

2. 可扩展性。

MinIO 借鉴了 Web 缩放器，为对象存储带来了简单的扩缩模型。在部署 MinIO 时，扩展从单个群集开始。

3. 云原生支持。

MinIO 是在过去几年内从 0 开始打造的一款存储方案，符合一切云原生计算的架构和构建流程，并且包含最新的云计算技术和理念。

4. 纯开源。

MinIO 基于 Apache V2 license 100% 开放源代码。这就意味着 MinIO 的客户能够自动、无限制、自由免费地使用和集成 MinIO、自由创新和创造、自由修改和完善、自由再次发行新版本和组合软件。

5. 兼容 S3 存储。

AWS 的 S3 API（接口协议）是在全球范围内达到共识的对象存储协议，是全世界内大家都认可的标准。

6. 极简。

极简主义是 MinIO 的指导性设计原则。简单性减少了出错的机会，提高了正常运行时间，提供了可靠性，同时简单性又是性能的基础。

7. 支持多云。

可以创建数以百万计的实例部署在私有云、公有云和边缘计算环境。

MinIO 为云原生而设计，可以作为轻量级容器运行，由外部编排服务（如 Kubernetes）管理。整个服务器约为几十 MB 的静态二进制文件，即使在高负载下也可以高效利用 CPU 和内存资源，使得企业可以在共享硬件上共同托管大量租户。



7.3 User guide

配置参数

MinIO 内置了参数配置 UI 界面。

- 在 MinIO 实例列表中，找到想要配置参数的 MinIO 实例，点击实例名称。

The screenshot shows the MinIO Object Storage interface. At the top, there's a navigation bar with the title 'MinIO 对象存储' and a search bar. Below the navigation bar, a list of instances is shown. One instance, 'wy-single', is highlighted with a red border and labeled '运行中' (Running). The instance details are displayed below: Deployment Location: mcamel-test/mcamel-system, Version: 4.0.15, Access Address: minio-svc.mcamel-system.svc:80. Resource Allocation: CPU Request 1 Core / Limit 1 Core, Memory Request 1 GB / Limit 1 GB, Disk 1 GB. Status: 正常/全部副本数 1/1. At the bottom, there's a pagination bar showing '共 1 项' (1 item total) and a dropdown for '10 项'.

- 在左侧导航栏，点击 配置参数。

The screenshot shows the DaoCloud interface. The left sidebar has a navigation tree with 'wy-single' selected under 'mcamel-ws'. The main content area is titled 'MinIO 实例: wy-single / 配置参数'. It contains a code editor window with the following configuration script:

```
1 export MINIO_BROWSER="on"
2 export MINIO_ROOT_USER="123"
3 export MINIO_ROOT_PASSWORD="12345678"
4 export MINIO_STORAGE_CLASS_STANDARD="EC:8"
5
6
```

A toggle switch labeled '编辑配置' (Edit Configuration) is visible at the top right of the code editor.

- 打开 编辑配置，对 MinIO 的各项参数进行添加、删除、修改。

The screenshot shows the DaoCloud platform interface. At the top, there's a navigation bar with the DaoCloud logo, user information (admin), and a dropdown menu. Below the navigation bar, the main content area has a title "MinIO 实例: wy-single / 配置参数". On the left, there's a sidebar with links: "概览", "实例监控", and "配置参数". The main content area contains a code editor with the following configuration script:

```
1 export MINIO_BROWSER="on"
2 export MINIO_ROOT_USER="123"
3 export MINIO_ROOT_PASSWORD="12345678"
4 export MINIO_STORAGE_CLASS_STANDARD="EC:8"
5
6
```

A red box highlights the "编辑配置" (Edit Configuration) button at the top right of the code editor. At the bottom right of the code editor, there are "取消" (Cancel) and "保存" (Save) buttons.

4. 点击 保存，修改内容将立即生效。

创建 MinIO 实例

- 从左侧导航栏选择 Minio 存储。

The screenshot shows the DaoCloud management interface. On the left, there's a navigation sidebar with sections like 容器, 多云编排, 镜像仓库, 多云服务平台, 可观测性, 微服务引擎, 服务网格, 数据服务, 中间件, 管理, 数字原生能力开放平台, and 全局管理. Under '中间件', 'Minio 存储' is highlighted with a red box. The main dashboard area has several cards: 'CPU 用量' (CPU Usage) with a line graph; '内存用量' (Memory Usage) with a line graph; '健康状态' (Health Status) which says '不健康' (Unhealthy); '告警' (Alerts) with counts for 紧急 (Emergency), 警告 (Warning), and 提示 (Notice); and a log section with recent entries. Below these is a '功能一览' (Function Overview) section with links to various services.

- 可以点击列表右上角的 新建实例 按钮。

如果是首次使用，需要先选择工作空间，然后点击立即部署 创建 MinIO 实例。

This screenshot shows the 'MinIO 对象存储' creation page. At the top, it says 'MinIO 对象存储' and 'Default'. There's a search bar and a '新建实例' (Create Instance) button with a red box around it. Below that, there's a table for the 'minio-test' instance, showing details like 部署位置 (Deployment Location), 资源配额 (Resource Quota), and 访问地址 (Access Address). At the bottom, it says '共 1 项' (1 item) and '1 / 1'.

- 参考下方信息填写实例基本信息，然后点击下一步。

!!! note

- 实例名称、所在集群/命名空间在实例创建之后不可修改
- 注意查看输入框下方的填写要求，输入符合要求的内容
- 如未通过安装环境检测，可根据提示安装相关插件后重新创建实例，或勾选‘不需要监控’后直接进行后续操作

DaoCloud

创建 MinIO 实例

1 基本信息 2 规格配置 3 服务设置 4 配置确认

基本信息

MinIO 实例名称 * min
2~40 个字符，只能包含小写字母、数字及分隔符("-")，且必须以小写字母开头、字母或数字结尾。

描述

描述信息不超过 256 个字符。

部署位置

集群 * lrf-test

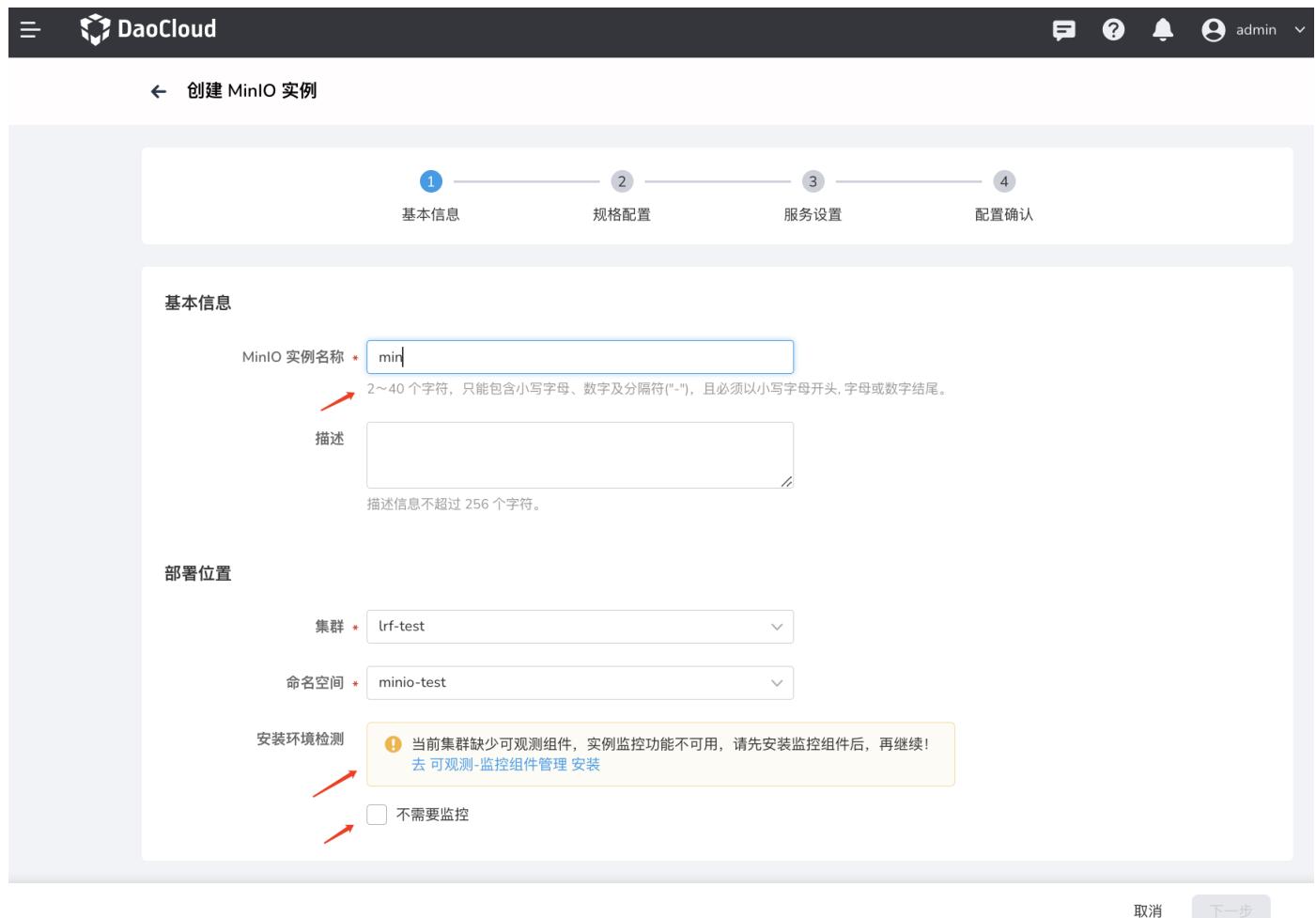
命名空间 * minio-test

安装环境检测

当前集群缺少可观测组件，实例监控功能不可用，请先安装监控组件后，再继续！
[去可观测-监控组件管理 安装](#)

不需要监控

取消 下一步



4. 参考下方信息填写配置规格，然后点击下一步。

- 部署模式在实例创建之后不可更改
- 生产模式下建议采用高可用部署模式
- 高可用模式下需要至少 4 个副本
- 存储类：所选的存储类应有足够的可用资源，否则会因资源不足导致实例创建失败
- 存储容量：每个磁盘具有多少容量。实例创建之后不可调低
- 每副本磁盘数：为每个副本提供多少个次盘。实例创建之后不可调低

The screenshot shows the 'Create MinIO Instance' wizard on the DaoCloud platform. The current step is 'Step 2: Configuration'. The steps are: 1. Basic Information (checkmark), 2. Configuration (highlighted), 3. Service Settings, 4. Configuration Confirmation.

Version: RELEASE.2022-10-02T19-29-29Z

Deployment Mode: High Availability Single Node
Warning: Current deployment mode does not support upgrading副本数 (replica count). Please reasonably plan the replica count.

Replica Count: 4
Note: The replica count must be at least 4 to ensure high availability.

CPU Allocation: Requested: 1 Core, Limit: 1 Core

Memory Allocation: Requested: 1 GB, Limit: 1 GB

Storage Class: standard
Warning: Current storage class does not support capacity upgrade. Please reasonably plan storage capacity.

Storage Capacity: 1 GB
Note: Set the value for each disk's capacity. It cannot be reduced.

Number of Disks per Replica: 4
Note: Set the number of disks each replica挂载 (mounts). Changes will affect available storage capacity. It cannot be reduced.

Buttons at the bottom: 取消 (Cancel), 上一步 (Previous Step), 下一步 (Next Step)

5. 参考下方信息填写服务设置，点击下一步。

- 集群内访问：只能在同一集群内部访问服务
- 节点端口：通过节点的 IP 和静态端口访问服务，支持从集群外部访问服务
- 负载均衡器：使用云服务提供商的负载均衡器使得服务可以公开访问
- 负载均衡器/外部流量策略：规定服务将外部流量路由到节点本地还是集群范围内的断点
- Cluster：流量可以转发到集群中其他节点上的 Pod
- Local：流量只能转发到本节点上的 Pod
- 控制台账号：访问此新建实例时需要用到的用户名、密码

[← 创建 MinIO 实例](#)

服务设置

访问类型 • 集群内访问(ClusterIP) 节点端口(NodePort) 负载均衡器(Loadbalancer)

负载均衡器配置

外部流量策略: Cluster

负载均衡 IP 地址:

端口配置

服务端口: 80
容器内部服务端口: 无需变更

注释: + 添加

访问工具

控制台账号 *

用户名: minio

密码: 随机生成
8~20 个字符，使用数字、大小写字母、特殊符号!@#\$%^&*，至少 2 种；不能包含空格。

确认密码:

访问类型 • 节点端口(NodePort)

取消 上一步 下一步

??? note “点击查看高级配置说明”

```

- Bucket 名称：在此实例下新建一个存储桶，设置新建存储桶的名称
- 调度策略/条件：设置 Pod 调度的节点亲和性，可参考 Kubernetes 官方文档[节点亲和性] (https://kubernetes.io/zh-cn/docs/concepts/scheduling-eviction/assign-pod-node/#affinity-and-anti-affinity)
    - 尽量满足：先尝试调度到满足规则的节点。如果找不到匹配的节点，也会执行 Pod 调度
    - 必须满足：只有找到满足规则的节点时才执行 Pod 调度
- 调度策略/权重：为满足每条调度策略的节点设置权重，优选使用权重高的策略。取值范围 1 到 100
- 调度策略/选择器

- In: 节点必须包含所选的标签，并且该标签的取值必须 **属于** 某个取值集合。多个值用 `;` 分隔
- NotIn: 节点必须包含所选的标签，并且该标签的取值必须 **不属于** 某个取值集合。多个值用 `;` 分隔
- Exists: 节点包含某个标签即可，不关注标签的具体取值
- DoesNotExist: 节点不包含某个标签，不关注标签的具体取值
- Gt: 节点必须包含某个标签，并且标签的取值必须大于某个整数
- Lt: 节点必须包含某个标签，并且标签的取值必须小于某个整数

```

! [访问模式] (.../images/create09.png)

6. 确认实例配置信息无误，点击确认完成创建。

基本信息

Kafka 实例名称: kafka01
部署位置: mcamel-test/default

规格配置

版本: 3.1.0
Kafka 副本数: 3 副本
Zookeeper 副本数: 3 副本
Kafka CPU 配额: 请求量 1 Core / 限制量 1 Core
Kafka 内存配额: 请求量 1 GB / 限制量 1 GB
Zookeeper CPU 配额: 请求量 1 Core / 限制量 1 Core
Zookeeper 内存配额: 请求量 1 GB / 限制量 1 GB

取消 上一步 确认

7. 返回实例列表页查看实例是否创建成功。

新创建的实例状态为 **未就绪**，等所有相关容器成功启动之后状态变为 **运行中**。

实例名	状态	部署位置	资源配额	CPU	内存	Kafka 副本数	Zookeeper 副本数
kafka01	未就绪	mcamel-test/default	请求量 1 Core / 限制量 1 Core 内存 请求量 1 GB / 限制量 1 GB 磁盘 10 GB	0 / 3	0 / 3	0 / 3	0 / 3
test-kafka-123	未就绪	mcamel-test/default	请求量 1 Core / 限制量 1 Core 内存 请求量 1 GB / 限制量 1 GB 磁盘 1 GB	0 / 3	0 / 3	0 / 3	0 / 3

共 2 项 < 1 / 1 > 10 项

删除 MinIO

如果想要删除一个实例列表，可以执行如下操作：

- 在实例列表中，点击右侧的 **...** 按钮，在弹出菜单中选择 **删除实例**。

The screenshot shows the MinIO Object Storage interface. At the top, there's a search bar and a '新建实例' (Create New Instance) button. Below the search bar, a table lists an instance named 'wy-single'. The instance status is '运行中' (Running). The table includes columns for '部署位置' (Deployment Location), '版本' (Version), and '访问地址' (Access Address). To the right of the instance details, there's a context menu with options: '更新实例' (Update Instance), '删除实例' (Delete Instance), and '正常/全部副本数' (Normal/All Replicas). The '删除实例' option is highlighted with a red border. At the bottom of the table, it says '共 1 项' (1 item) and has a page navigation section with '1 / 1' and a dropdown for '10 项'.

- 在弹窗中输入该实例列表的名称，确认无误后，点击 **删除** 按钮。

!!! warning

删除实例后，该实例相关的所有消息也会被全部删除，请谨慎操作。

确认删除实例 wy-single 吗？

×

确认删除实例 wy-single 吗？删除后对应的数据将会全部丢失，请谨慎操作。

请输入wy-single

wy-single

删除

取消

实例监控

MinIO 内置了 Prometheus 和 Grafana 监控模块。

- 在实例列表页面中，找到想要查看监控信息的实例，点击该实例的名称。

部署位置 mcamel-test/mcamel-system
资源配额 CPU 请求量 1 Core / 限制量 1 Core
版本 4.0.15 内存 请求量 1 GB / 限制量 1 GB
访问地址 minio-svc.mcamel-system.svc:80 磁盘 1 GB
正常/全部副本数 1 / 1

共 1 项 1 / 1 10 项

- 在左侧导航栏点击 实例监控。

MinIO 实例: wy-single / 实例监控

警告 监控组件异常或未启用，点击 可观测性-采集管理 去修复

wy-single
mcamel-ws
概览
实例监控
配置参数

!!! note
如果提示监控组件异常，请按提示[安装 insight-agent 插件] (../../../../insight/user-guide/quickstart/install-agent.md)。

- 查看实例的监控信息。点击红框里的信息符号可查看每个指标的含义说明。

MinIO 实例: test-minio-2 / 实例监控

MinIO 实例: mcamel-system / Minio Overview

Instance 10.244.0.217:9000 Interval 5m
Last 30 minutes 10s

Uptime Total disks Disks
4 hours 1 0

Disk Usage
Disk Usage
Total CPU
HTTP Requests Duration
Data Transmitted
GO Memory Stats
Uptime

查看 MinIO 日志

通过访问每个 MinIO 的实例详情页面，可以支持查看 MinIO 的日志。

- 在 MinIO 实例列表中，找到想要查看日志的实例，点击实例名称进入详情页面。

MinIO 对象存储

test-minio-2 ● 运行中

部署位置: test-mcamel-zlh/mcamel-system
版本: RELEASE.2022-10-02T19-29-2...
访问地址: minio-svc.mcamel-system.sv... +1

资源配额: CPU 请求量 1 Core 限制量 1 ...
内存 请求量 1 GB 限制量 1 ...
磁盘 1 GB

1 / 1 正常/全部副本数

共 1 项 1 / 1 10 项

- 在左侧菜单栏点击 日志查看。

MinIO 实例: test-minio-2 / 概览

基本信息

test-minio-2	● 运行中	test-mcamel-zlh/mcamel-syste...	2023-03-22 11:12
实例名称	运行状态	部署位置	创建时间
RELEASE.2022-10-02T19-29-2...	节点端口(NodePort)	cas	minio-svc.mcamel-system.s... +1
版本	访问方式	描述	访问地址

访问设置 资源配额

NodePort	minio	CPU	限制量 1 C... 请求量 1 C...
访问方式	用户名	内存	限制量 1 GB 请求量 1 GB
http://10.6.216.5:31601	*****	磁盘	存储量 1 GB
访问地址	登录密码		

监控告警 (最近 10 条) 查看更多 创建告警规则

规则名称	告警级别	资源类型	资源名称	描述	发生时间	持续时间
------	------	------	------	----	------	------

- 根据需要调整日志查询的时间范围和刷新周期，具体可参考[日志查询](#)。

日志查询

筛选

搜索: 请输入日志内容

集群: test-mcamel-zlh

类型: 工作负载

命名空间: mcamel-system

负载类型: 有状态负载

负载名称: test-minio-2-pool-0

容器组名称: 所有容器组

容器名称: 所有容器

最近 6 小时

日志

时间	日志
2023-03-22 11:13	test-minio-2-pool-0-0 minio
2023-03-22 11:13	test-minio-2-pool-0-0 minio You are running an older version of MinIO released 5 months ago
2023-03-22 11:13	test-minio-2-pool-0-0 minio Update: Run 'mc admin update'
2023-03-22 11:13	test-minio-2-pool-0-0 minio
2023-03-22 11:13	test-minio-2-pool-0-0 minio
2023-03-22 11:13	test-minio-2-pool-0-0 minio Console: http://10.244.0.217:9090 http://127.0.0.1:9090
2023-03-22 11:13	test-minio-2-pool-0-0 minio
2023-03-22 11:13	test-minio-2-pool-0-0 minio Documentation: https://min.io/docs/minio/linux/index.html
2023-03-22 11:13	test-minio-2-pool-0-0 minio MinIO Object Storage Server
2023-03-22 11:13	test-minio-2-pool-0-0 minio Copyright: 2015-2022 MinIO, Inc.

共 35 项

!!! note “常用操作”

- * 自定义日志查询时间范围：在日志页面右上角可以切换日志的查询时间范围
- * 通过关键字检索日志：在左侧“搜索框”下输入关键字即可查询带有特定内容的日志
- * 查看日志的时间分布情况：通过柱状图查看在某一时间范围内的日志数量
- * 查看日志的上下文：在日志最右侧点击“查看上下文”即可
- * 导出日志：在柱状图下方点击“下载”即可

![image] (./images/log03.png)

更新 MinIO 实例

如果想要更新或修改 MinIO 的资源配置，可以按照本页说明操作。

- 在实例列表中，点击右侧的 **...** 按钮，在弹出菜单中选择 **更新实例**。

MinIO 对象存储 mcamel-ws

wy-single ● 运行中

部署位置: mcamel-test/mcamel-system 资源配额: CPU 请求量 1 Core / 限制量 1 Core
版本: 4.0.15 内存 请求量 1 GB / 限制量 1 GB
访问地址: minio-svc.mcamel-system.svc:80 磁盘 1 GB 正常/全部副本数

共 1 项 1 / 1 10 项

- 修改基本信息，然后点击 **下一步**。

- 仅支持修改描述信息
- 实例名称、部署位置不可修改

更新实例 wy-single

1 基本信息 2 规格配置 3 服务设置

基本信息

MinIO 实例名称: wy-single

描述:

描述信息不超过 256 个字符。

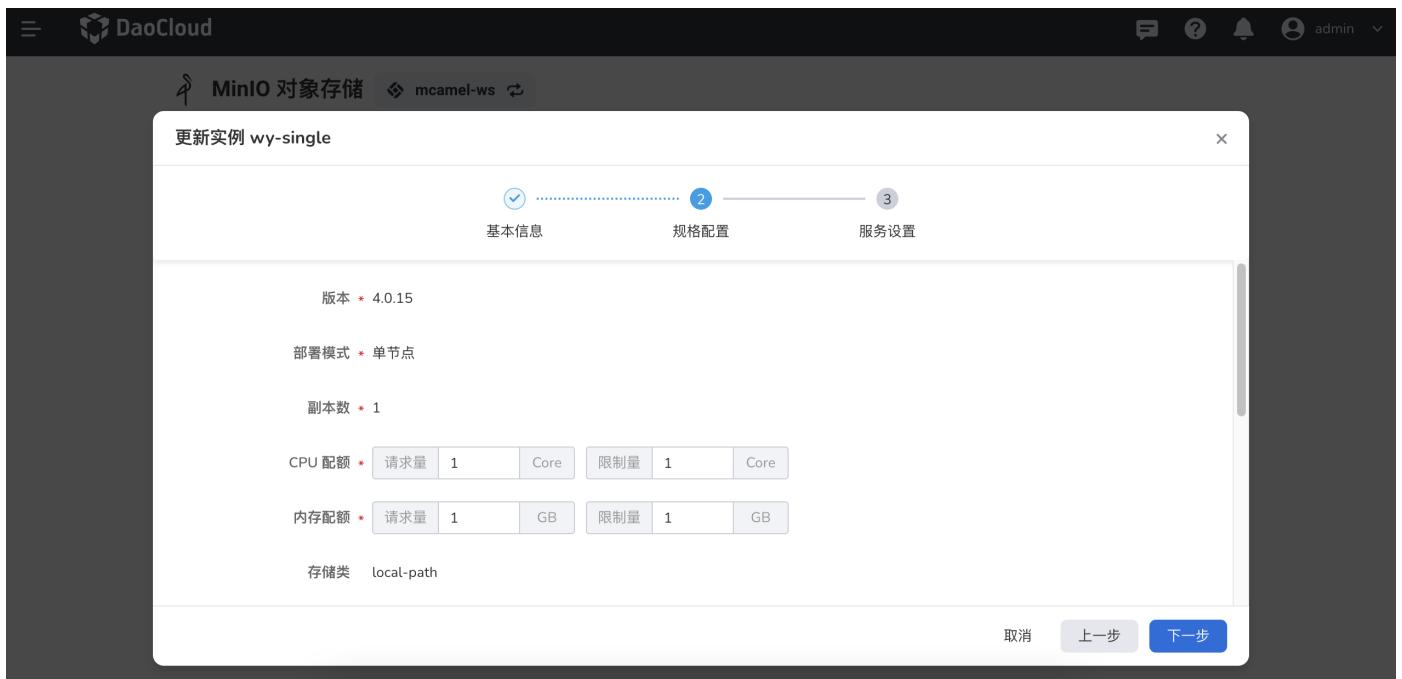
部署位置

集群: mcamel-test

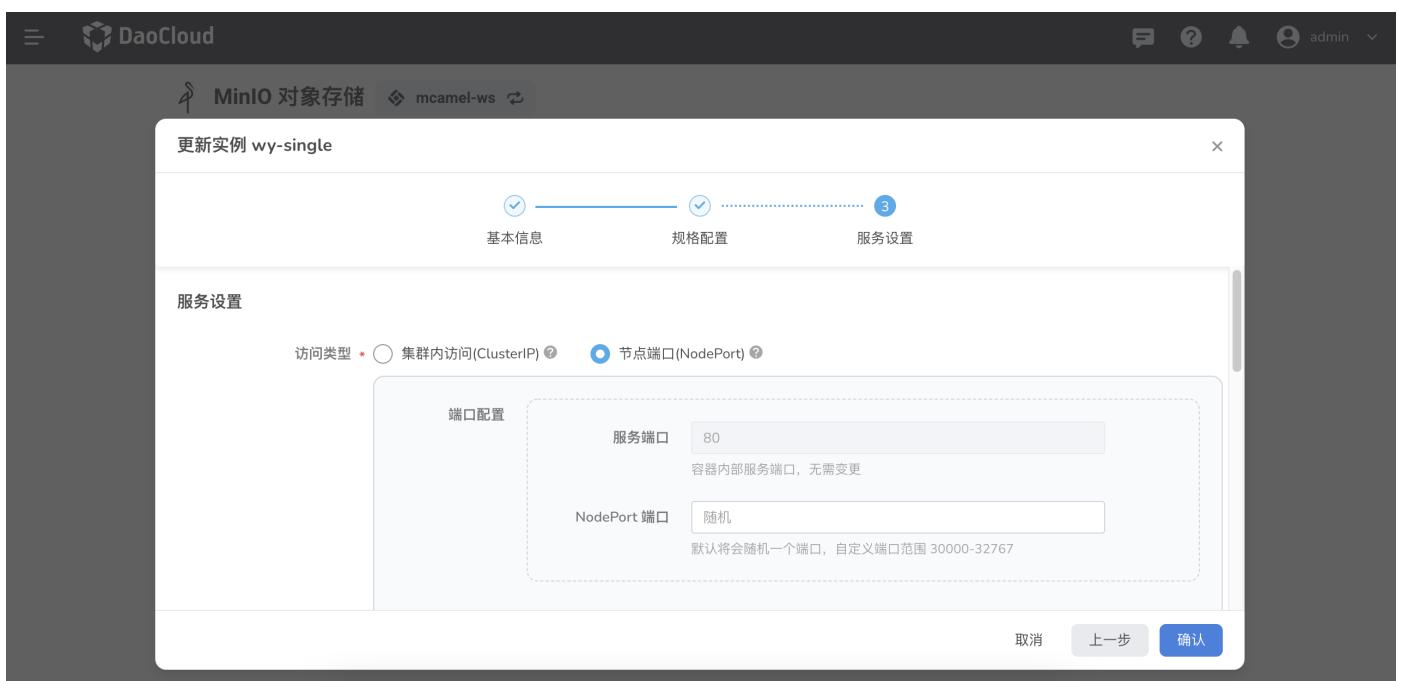
取消 下一步

- 修改规格配置，然后点击 **下一步**。

- 仅支持修改：CPU 配额和内存配额
- 不可修改：版本、部署模式、存储类、容量



4. 修改服务设置，然后点击确认。



5. 返回实例列表，屏幕右上角将显示消息：「更新实例成功」。

The screenshot shows the DaoCloud platform interface for managing MinIO objects. At the top, there's a navigation bar with the DaoCloud logo and a search bar. Below it, a header bar displays "MinIO 对象存储" and the service name "mcamel-ws". A green success message "更新实例成功" is visible in the top right corner. The main content area shows a single instance named "wy-single" which is "运行中" (Running). It provides detailed information about the deployment: location "mcamel-test/mcamel-system", version "4.0.15", and address "minio-svc.mcamel-system.svc:80". Resource allocation includes CPU (1 Core / 1 Core), memory (1 GB / 1 GB), and disk (1 GB). The status is "正常/全部副本数" (Normal/All replicas). Navigation controls at the bottom include a search bar, a page number "1 / 1", and a dropdown for "10 项".

MinIO 的身份管理

DCE 5.0 提供的 MinIO 服务自带网页控制台（Web Console）。了解 MinIO 的身份管理（identity management）有助于快速了解如何在 MinIO 内安全有效地管理子账号。

本文简单介绍 MinIO 的身份管理规则，更多详细说明可参考 [MinIO 的官方文档](#)。

用户

默认情况下，MinIO 使用内置的 IDentity Provider (IDP) 来完成身份管理。除了 IDP，还支持第三方 [OIDC](#) 和 [LDAP](#) 的方式。

用户由一对 username 和 password 组成。在 MinIO 的语境中，username 又被称为 access key（注意与后面 service account 层级的 access key 区分开来），password 又称为 secret key。

ROOT 用户

在启动 MinIO 时，可以通过环境变量的方式设置 MinIO 集群中 root 用户的账号密码，分别是以下两个变量：

- MINIO_ROOT_USER
- MINIO_ROOT_PASSWORD

root 用户拥有所有资源的所有操作权限。

注意：如果要变更 root 用户，需要重启 MinIO 集群中所有的节点。

普通用户

支持通过三种方式创建普通用户：

- Web Console，在 UI 界面中通过表单进行创建
- mc，使用 CLI 命令行创建
- Operator CR，使用 CR 进行创建

Console 创建

1. 在 DCE 5.0 的 MinIO 实例详情页面，点击访问地址，使用右侧的用户名和密码即可登录该实例的 Console 控制台。

The screenshot shows the 'MinIO 实例: minio-test / 概览' (Overview) page. On the left sidebar, '概览' (Overview) is selected. In the main area, under '基本信息' (Basic Information), the instance name is 'minio-test', status is '运行中' (Running), deployment location is 'lrf-test/minio-test', creation time is '2023-03-13 13:49', and the '访问地址' (Access Address) is 'http://172.18.0.2:31541'. Below this, there are '访问设置' (Access Settings) and '资源配额' (Resource Quota) sections. A red arrow points from the '访问地址' field to the '用户名' (Username) and '登录密码' (Login Password) fields.

2. 登录 Console 控制台之后根据下图所示，创建用户。

mc 创建

需要事先安装 `mc` 命令，并配置连接到 MinIO 实例

创建用户：

`ALIAS` 指 MinIO 实例的别名

```
mc admin user add ALIAS ACCESSKEY SECRETKEY
```

授予权限：

`USERNAME` 指 MinIO 用户的用户名，即 `ACCESSKEY`

```
mc admin policy set ALIAS readwrite user=USERNAME
```

operator CR 创建

如果是通过 cr 安装 MinIO，也可以通过 `users` 字段来指定普通用户的 secret：

```
type TenantSpec struct {
    ...
    ...
    ...
    // *Optional* +
    //
    // An array of https://kubernetes.io/docs/concepts/configuration/secret/[Kubernetes opaque secrets] to use for generating MinIO users during tenant provisioning. +
    //
    // Each element in the array is an object consisting of a key-value pair `name: <string>`, where the `<string>` references an opaque Kubernetes secret. +
    //
    // Each referenced Kubernetes secret must include the following fields: +
    //
    // * `CONSOLE_ACCESS_KEY` - The "Username" for the MinIO user +
    //
    // * `CONSOLE_SECRET_KEY` - The "Password" for the MinIO user +
    //
    // The Operator creates each user with the `consoleAdmin` policy by default. You can change the assigned policy after the Tenant starts. +
    // +optional
    Users []*corev1.LocalObjectReference `json:"users,omitempty"`
    ...
    ...
}
```

服务账户

服务账户（Service Account）通常使用用户登录 console 或者通过 `mc` 命令对 MinIO 进行管理操作。但如果应用程序需要访问 MinIO，则通常使用 Service Account（这是比较正式的叫法，某些上下文中也称之为 access key）。

一个用户可以创建多个 Service Account。

注意：无法通过 Service Account 登录 MinIO console，这也是它与用户最大的不同之处。

Console 创建

mc 命令创建

```
mc [GLOBALFLAGS] admin user svcacct add \
    [--access-key] \
    [--secret-key] \
    [--policy] \
    ALIAS
USER
```

有关 MinIO 用户的详细说明，可参考 [User Management](#)

用户组

用户组，顾名思义即多个用户形成的集合。通过用户组合结合授权策略可以批量管理一组用户的权限。通过授权策略可以为用户组分配资源权限，该组中的用户会继承用户组的资源权限。

MinIO 用户的权限分为两部分：用户原本具有的权限 + 从所在用户组继承而来的权限。在 MinIO 语境中，用户仅具有其明确被授予或从用户组继承而来的授权。如果用户没有明确获得（被直接授予或继承）某一资源的权限，则无法访问该资源。

有关 MinIO 用户组的详细说明，可参考 [Group Management](#)

授权策略

MinIO 使用基于策略的访问控制 (PBAC) 来管理用户对哪些资源具有哪些权限。每条策略通过规定一些动作或条件来限制用户和用户组具有的权限。

内置策略

MinIO 内置了四种策略可以直接分配给用户或用户组。为用户/用户组授权时需要使用 `mc admin policy set` 命令，具体可参考 [mc admin policy](#)

- `readonly`: 对 MinIO 副本中的所有存储桶和存储对象具有 只读 权限
- `readwrite`: 对 MinIO 副本中的所有存储桶和存储对象具有 读写 的权限
- `diagnostics`: 对 MinIO 副本具有 诊断 权限
- `writeonly`: 对 MinIO 副本中的所有存储桶和存储对象具有 只写 权限

策略文件示例

MinIO 授权策略文件的模式和[亚马逊云 IAM Policy](#) 相同。

```
{
  "Version" : "2012-10-17",
  "Statement" : [
    {
      "Effect" : "Allow",
      "Action" : [ "s3:<ActionName>", ... ],
      "Resource" : "arn:aws:s3:::*",
      "Condition" : { ... }
    },
    {
      "Effect" : "Deny",
      "Action" : [ "s3:<ActionName>", ... ],
      "Resource" : "arn:aws:s3:::mybucket"
    }
  ]
}
```

```
        "Resource" : "arn:aws:s3::::*",
        "Condition" : { ... }
    ]
}
```

有关 MinIO 授权策略的详细说明，可参考 [Policy Management](#)

查看实例详情

本节说明如何查看 MinIO 实例列表。

- 在实例列表页面中，找到想要查看详情的实例，点击实例名称。

- 查看基本信息、访问设置、资源配额和 Pod 列表等信息。

在当前页面除了查看基本信息外，还支持下列操作：

- 开启云端控制台通过命令行对实例进行相关操作
- 更新或删除实例
- 点击访问地址可以在 Web 界面中打开控制台并使用右侧的用户名和密码进行登录
- 可以查看或创建告警规则
- 点击容器组名称可以跳转到容器管理模块查看该容器组的详细情况

8. Mysql

8.1 MySQL Release Notes

本页列出 MySQL 数据库的 Release Notes，便于您了解各版本的演进路径和特性变化。

v0.6.0

发布日期：2023-02-23

API

- 新增 `mcamel-mysql helm-docs` 模板文件。
- 新增 `mcamel-mysql` 应用商店中的 Operator 只能安装在 `mcamel-system`。
- 新增 `mcamel-mysql` 支持 cloud shell。
- 新增 `mcamel-mysql` 支持导航栏单独注册。
- 新增 `mcamel-mysql` 支持查看日志。
- 新增 `mcamel-mysql` 更新 Operator 版本。
- 新增 `mcamel-mysql` 展示 common MySQL 在实例列表中。
- 新增 `mcamel-mysql` 支持 MySQL8.0.29。
- 新增 `mcamel-mysql` 支持 LB。
- 新增 `mcamel-mysql` 支持 Operator 对接 chart-syncer。
- 新增 `mcamel-mysql` Operator 的 finalizers 权限以支持 openshift。
- 修复 `mcamel-mysql` 实例名太长导致自定义资源无法创建的问题。
- 修复 `mcamel-mysql` 工作空间 Editor 用户无法查看实例密码。
- 修复 `mcamel-mysql` 配置文件里 `expire-logs-days` 参数定义重复。
- 修复 `mcamel-mysql` 8.0 环境下 binlog 过期时间不符合预期。
- 修复 `mcamel-mysql` 备份集列表会展示出同名集群旧的备份集。
- 升级 `mcamel-mysql` 升级离线镜像检测脚本。

UI

- 新增 增加 MySQL 主从复制延迟情况展示

文档

- 新增 日志查看操作说明，支持自定义查询、导出等功能。

v0.5.0

发布日期：2022-12-25

API

- 新增 NodePort 端口冲突提前检测。
- 新增 节点亲和性配置。
- 新增 创建备份配置时，可校验 Bucket。

- 修复 arm 环境中无法展示默认配置。
- 修复 创建实例时 name 校验与前端不一致。
- 修复 重新接入变更名字的集群后，配置管理地址展示错误。
- 修复 保存自动备份配置失败。
- 修复 自动备份集无法展示的问题。
- 优化 创建托管 MinIO 类型的备份配置时，可填写 AccessKey/SecretKey/Bucket。

v0.4

发布日期: 2022-11-08

API

- 新增 增加 MySQL 生命周期管理接口功能
- 新增 增加 MySQL 详情接口功能
- 新增 基于 grafana crd 对接 insight
- 新增 增加与 ghippo 服务对接接口
- 新增 增加与 kpanda 对接接口
- 新增 单测覆盖率提升到 30%
- 新增 新增备份恢复功能
- 新增 新增备份配置接口
- 新增 实例列表接口新增备份恢复来源字段
- 新增 获取用户列表接口
- 新增 mysql-operator chart 参数，来指定 metric exporter 镜像
- 新增 支持 arm64 架构
- 新增 添加 arm64 operator 镜像打包
- 新增 支持密码脱敏
- 新增 支持服务暴露为 nodeport
- 新增 支持 mtls
- 修复 修复备份列表接口模糊搜索无效
- 修复 修复依赖漏洞
- 修复 备份 Job 被删除后，无法展示备份任务列表
- 优化 将时间戳 api 字段统一调整为 int64
- 改进 当镜像有大写和数字时不能被抓取到的问题

文档

- 新增 第一次文档网站发布
- 新增 基本概念
- 新增 Concepts
- 新增 首次使用 MySQL
- 新增 删除 MySQL 实例

v0.3

发布日期: 2022-10-18

API

- 新增 增加 MySQL 生命周期管理接口功能
- 新增 增加 MySQL 详情接口功能
- 新增 基于 grafana crd 对接 insight
- 新增 增加与 ghippo 服务对接接口
- 新增 增加与 kpanda 对接接口
- 新增 单测覆盖率提升到 30%
- 新增 新增备份恢复功能
- 新增 新增备份配置接口
- 新增 实例列表接口新增备份恢复来源字段
- 修复 修复备份列表接口模糊搜索无效
- 优化 将时间戳 api 字段统一调整为 int64

文档

- 新增 第一次文档网站发布
- 新增 基本概念
- 新增 Concepts
- 新增 首次使用 MySQL
- 新增 删除 MySQL 实例

8.2 Faq

MySQL 排障手册

本文将持续统计和梳理常见的 MySQL 异常故障以及修复方式，若遇到使用问题，请优先查看此排障手册。

如果您发现遇到的问题，未包含在本手册，可以快速跳转到页面底部，提交您的问题。

mysql Pod 出现 (2/4) running 的情况

一般出现此类问题，很可能是因为 MySQL 实例使用的磁盘用量达到了 100%，您可以在 master 节点上运行以下命令检测磁盘用量。

```
kubectl get pod -n mcamel-system | grep cluster-mysql | awk '{print $1}' | xargs -I {} kubectl exec {} -n mcamel-system -c sidecar -- df -h | grep /var/lib/mysql
```

输出类似于：

```
/dev/drbd43001      50G   30G   21G  60% /var/lib/mysql
/dev/drbd43005      80G   29G   52G  36% /var/lib/mysql
```

如果发现某个 pvc 满了进行 pvc 扩容即可。

```
kubectl edit pvc data-mcamel-common-mysql-cluster-mysql-0 -n mcamel-system # 修改request大小即可
```

mysql pod 一直有一个 pod 出现 (0/4) Init:Error 的状态

遇到此类问题时，我们优先查看 master 节点的 (sidecar) 日志信息。

```
kubectl get pod -n mcamel-system -lhealthy,role | grep cluster-mysql | grep master | awk '{print $1}' | xargs -I {} kubectl logs -f {} -n mcamel-system -c sidecar
```

输出类似于：

```
2023-02-09T05:38:56.208445-00:00 0 [Note] [MY-011825] [Xtrabackup] perl binary not found. Skipping the version check
2023-02-09T05:38:56.208521-00:00 0 [Note] [MY-011825] [Xtrabackup] Connecting to MySQL server host: 127.0.0.1, user: sys_replication, password: set, port: not set, socket: not set
2023-02-09T05:38:56.212595-00:00 0 [Note] [MY-011825] [Xtrabackup] Using server version 8.0.29
2023-02-09T05:38:56.217325-00:00 0 [Note] [MY-011825] [Xtrabackup] Executing LOCK INSTANCE FOR BACKUP ...
2023-02-09T05:38:56.219880-00:00 0 [ERROR] [MY-011825] [Xtrabackup] Found tables with row versions due to INSTANT ADD/DROP columns
2023-02-09T05:38:56.219931-00:00 0 [ERROR] [MY-011825] [Xtrabackup] This feature is not stable and will cause backup corruption.
2023-02-09T05:38:56.219940-00:00 0 [ERROR] [MY-011825] [Xtrabackup] Please check https://docs.percona.com/percona-xtrabackup/8.0/em/instant.html for more details.
2023-02-09T05:38:56.219945-00:00 0 [ERROR] [MY-011825] [Xtrabackup] Tables found:
2023-02-09T05:38:56.219951-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/USER_SESSION
2023-02-09T05:38:56.219956-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/AUTHENTICATION_EXECUTION
2023-02-09T05:38:56.219960-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/AUTHENTICATION_FLOW
2023-02-09T05:38:56.219968-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/AUTHENTICATOR_CONFIG
2023-02-09T05:38:56.219984-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/CLIENT_SESSION
2023-02-09T05:38:56.219991-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/IDENTITY_PROVIDER
2023-02-09T05:38:56.219998-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/PROTOCOL_MAPPER
2023-02-09T05:38:56.220005-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/RESOURCE_SERVER_SCOPE
2023-02-09T05:38:56.220012-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/REQUIRED_ACTION_PROVIDER
2023-02-09T05:38:56.220018-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/COMPONENT
2023-02-09T05:38:56.220027-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/RESOURCE_SERVER
2023-02-09T05:38:56.220036-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/CREIDENTIAL
2023-02-09T05:38:56.220043-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/FED_USER_CREDENTIAL
2023-02-09T05:38:56.220049-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/MIGRATION_MODEL
2023-02-09T05:38:56.220054-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/REALM
2023-02-09T05:38:56.220062-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/CLIENT
2023-02-09T05:38:56.220069-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/REALM_ATTRIBUTE
2023-02-09T05:38:56.220075-00:00 0 [ERROR] [MY-011825] [Xtrabackup] keycloak/OFFLINE_USER_SESSION
2023-02-09T05:38:56.220084-00:00 0 [ERROR] [MY-011825] [Xtrabackup] Please run OPTIMIZE TABLE or ALTER TABLE ALGORITHM=COPY on all listed tables to fix this issue.
E0209 05:38:56.223635      1 deleg.go:144] sidecar "msg"="failed waiting for xtrabackup to finish" "error"="exit status 1"
```

登录 master 节点的 mysql，执行 alter 表结构：

```
[root@master-01 ~]$ kubectl get pod -n mcamel-system -lhealthy,role | grep cluster-mysql | grep master
mcamel-common-mysql-cluster-mysql-0
```

```
#获取密码
[root@master-01 ~]$ kubectl get secret -n mcamel-system mcamel-common-mysql-cluster-secret -o=jsonpath='{.data.ROOT_PASSWORD}' | base64 -d
```

```
[root@master-01 ~]$ kubectl exec -it mcamel-common-mysql-cluster-mysql-0 -n mcamel-system -c mysql -- /bin/bash
# 注意: 修改表结构需要root权限登录
~$ bash: mysql -uroot -p
```

SQL 语句如下：

```
use keycloak;
ALTER TABLE USER_SESSION ALGORITHM=COPY;
ALTER TABLE AUTHENTICATION_EXECUTION ALGORITHM=COPY;
ALTER TABLE AUTHENTICATION_FLOW ALGORITHM=COPY;
ALTER TABLE AUTHENTICATOR_CONFIG ALGORITHM=COPY;
ALTER TABLE CLIENT_SESSION ALGORITHM=COPY;
ALTER TABLE IDENTITY_PROVIDER ALGORITHM=COPY;
ALTER TABLE PROTOCOL_MAPPER ALGORITHM=COPY;
ALTER TABLE RESOURCE_SERVER_SCOPE ALGORITHM=COPY;
ALTER TABLE REQUIRED_ACTION_PROVIDER ALGORITHM=COPY;
ALTER TABLE COMPONENT ALGORITHM=COPY;
ALTER TABLE RESOURCE_SERVER ALGORITHM=COPY;
ALTER TABLE CREDENTIAL ALGORITHM=COPY;
ALTER TABLE FED_USER_CREDENTIAL ALGORITHM=COPY;
ALTER TABLE MIGRATION_MODEL ALGORITHM=COPY;
ALTER TABLE REALM ALGORITHM=COPY;
ALTER TABLE CLIENT ALGORITHM=COPY;
ALTER TABLE REALM_ATTRIBUTE ALGORITHM=COPY;
ALTER TABLE OFFLINE_USER_SESSION ALGORITHM=COPY
```

mysql 不健康

我们可以通过以下命令快速的查看 当前集群上所有 MySQL 的健康状态

```
[root@master-01 ~]$ kubectl get mysql -A
NAMESPACE      NAME           READY   REPLICAS   AGE
ghippo-system  test          True    1          3d
mcamel-system  mcamel-common-mysql-cluster   False   2          62d
```

如果，发现某个 mysql 的 ready 为 False (这里为 True 的判断是 延迟小于 30s 同步)，然后我们快速查看 从库 MySQL 的日志

```
[root@master-01 ~]$ kubectl get pod -n mcamel-system -lhealthy,role | grep cluster-mysql | grep replica | awk '{print $1}' | xargs -I {} kubectl logs {} -n mcamel-system -c mysql | grep ERROR
```

从库没有错误日志

如果上面的命令执行后，没有任何错误 ERROR 信息，这说明只是因为主从同步的延迟过大，

```
# 寻找到从节点的pod
[root@master-01 ~]$ kubectl get pod -n mcamel-system -lhealthy,role | grep cluster-mysql | grep replica | awk '{print $1}'
mcamel-common-mysql-cluster-mysql-1

# 设置binlog参数
[root@master-01 ~]$ kubectl exec mcamel-common-mysql-cluster-mysql-1 -n mcamel-system -c mysql -- mysql --defaults-file=/etc/mysql/client.conf -NB -e 'set global sync_binlog=10086;'

# 进入mysql的容器
[root@master-01 ~]$ kubectl exec -it mcamel-common-mysql-cluster-mysql-1 -n mcamel-system -c mysql -- mysql --defaults-file=/etc/mysql/client.conf
```

获取从库状态 Seconds_Behind_Master 为主从延迟，如果是 0~30，表示没有主从延迟了；表示从库追上了（主库）

```
mysql> show slave status\G; #获取从库状态 Seconds_Behind_Master 为主从延迟,如果是 0~30, 表示没有主从延迟了。表示从库追上了（主库）
***** 1. row *****
Slave_IO_State: Waiting for source to send event
Master_Host: mcamel-common-mysql-cluster-mysql-0.mysql.mcamel-system
Master_User: sys_rePLICATION
Master_Port: 3306
Connect_Retry: 1
Master_Log_File: mysql-bin.000304
Read_Master_Log_Pos: 83592007
Relay_Log_File: mcamel-common-mysql-cluster-mysql-1-relay-bin.000002
Relay_Log_Pos: 83564355
Relay_Master_Log_File: mysql-bin.000304
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB:
Replicate_Ignore_DB:
Replicate_Do_Table:
Replicate_Ignore_Table:
Replicate_Wild_Do_Table:
Replicate_Wild_Ignore_Table:
Last_Error:
Skip_Counter: 0
```

```

Exec_Master_Log_Pos: 83564299
Relay_Log_Space: 83592303
Until_Condition: None
Until_Log_File:
Until_Log_Pos: 0
Master_SSL_Allowed: No
Master_SSL_CA_File:
Master_SSL_CA_Path:
Master_SSL_Cert:
Master_SSL_Cipher:
Master_SSL_Key:
Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
Last_IO_Error: 0
Last_IO_Error:
Last_SQL_Error: 0
Last_SQL_Error:
Replicate_Ignore_Server_Ids:
Master_Server_Id: 100
Master_UUID: e17dae09-8da0-11ed-9104-c2f9484728fd
Master_Info_File: mysql.slave_master_info
SQL_Delay: 0
SQL_Remaining_Delay: NULL
Slave_SQL_Running_State: Replica has read all relay log; waiting for more updates
Master_Retry_Count: 86400
Master_Bind:
Last_IO_Error_Timestamp:
Last_SQL_Error_Timestamp:
Master_SSL_Crl:
Master_SSL_Crlpath:
Retrieved_Gtid_Set: e17dae09-8da0-11ed-9104-c2f9484728fd:21614244-21621569
Executed_Gtid_Set: 4bc2107c-819a-11ed-bf23-22be07e4eaff:1-342297,
7cc717ea-7c1b-11ed-b59d-c2ba3f807d12:1-619197,
a5ab763a-7c1b-11ed-b5ca-522707642ace:1-178069,
a6045297-8743-11ed-8712-8e52c3ace534:1-4073131,
a95cf9df-84d7-11ed-8362-5e8a1c335253:1-9493942,
b5175b1b-a2ac-11ed-b0c6-d6fbe05d7579:1-3754703,
c4dc2b14-9ed9-11ed-ac61-36da81109699:1-945884,
e17dae09-8da0-11ed-9104-c2f9484728fd:1-21621569
Auto_Position: 1
Replicate_Rewrite_DB:
Channel_Name:
Master_TLS_Version:
Master_public_key_path:
Get_master_public_key: 0
NetworkNamespace:
1 row in set, 1 warning (0.00 sec)

```

等从库追上以后 (Seconds_Behind_Master 小于 30s)，再设置成 sync_binlog=1

```
kubectl exec mcamel-common-mysql-cluster-mysql-1 -n mcamel-system -c mysql -- mysql --defaults-file=/etc/mysql/client.conf -NB -e 'set global sync_binlog=1';
```

如果此时依然不见缓解，可以看看是不是从库的 mysql 的宿主机的负载或者 IO 太高。把它降低。

```
[root@master-01 ~]$ uptime
11:18 up 1 day, 17:49, 2 users, load averages: 9.33 7.08 6.28
```

关注下 load averages，正常情况下，不应该超过 3 个数值都不应长期超过 10；如果 超过 >30 以上，请合理调配下该节点的 POD 和磁盘

从库出现复制错误

如果我们在查看 Pod 日志时，发现了从库复制错误，这可能有多种情况，注意根据出现的错误内容，需要分别进行修复

出现了 purged binlog 错误

```
[root@demo-alpha-master-01 /]$ kubectl get pod -n mcamel-system -Lhealthy,role | grep cluster-mysql | grep replica | awk '{print $1}' | xargs -I {} kubectl logs {} -n mcamel-system -c mysql | grep ERROR
2023-02-08T18:43:21.991730Z 116 [ERROR] [MY-010557] [Repl] Error reading packet from server for channel ''': Cannot replicate because the master purged required binary logs. Replicate the missing transactions from elsewhere, or provision a new slave from backup. Consider increasing the master's binary log expiration period. The GTID sets and the missing purged transactions are too long to print in this message. For more information, please see the master's error log or the manual for GTID_SUBTRACT (server_errno=1236)
2023-02-08T18:43:21.991777Z 116 [ERROR] [MY-013114] [Repl] Slave I/O for channel ''': Got fatal error 1236 from master when reading data from binary log: 'Cannot replicate because the master purged required binary logs. Replicate the missing transactions from elsewhere, or provision a new slave from backup. Consider increasing the master's binary log expiration period. The GTID sets and the missing purged transactions are too long to print in this message. For more information, please see the master's error log or the manual for GTID_SUBTRACT', Error_code: MY-013114
```

注意关键字 purged binlog，如果发现了此类错误，基本上，我们需要进行 从库的重建处理

```
# 寻找到从节点的pod
[root@master-01 ~]$ kubectl get pod -n mcamel-system -Lhealthy,role | grep cluster-mysql | grep replica | awk '{print $1}'
mcamel-common-mysql-cluster-mysql-1

# 寻找从节点的pvc
[root@master-01 /]$ kubectl get pvc -n mcamel-system | grep mcamel-common-mysql-cluster-mysql-1
data-mcamel-common-mysql-cluster-mysql-1           Bound   pvc-5840569e-834f-4236-a5c6-878e41c55c85   50Gi      RWO
```

```
local-path          33d

# 删除从节点的pvc
[root@master-01 /]$ kubectl delete pvc data-mcamel-common-mysql-cluster-mysql-1 -n mcamel-system
persistentvolumeclaim "data-mcamel-common-mysql-cluster-mysql-1" deleted

# 删除从库的pod
[root@master-01 /]$ kubectl delete pod mcamel-common-mysql-cluster-mysql-1 -n mcamel-system
pod "mcamel-common-mysql-cluster-mysql-1" deleted
```

主键冲突错误

```
[root@demo-alpha-master-01 /]$ kubectl get pod -n mcamel-system -lhealthy,role | grep cluster-mysql | grep replica | awk '{print $1}' | xargs -I {} kubectl logs {} -n mcamel-system -c mysql | grep ERROR
2023-02-08T18:43:21.991Z 0116 [ERROR] [MY-010557] [Repl] Could not execute Write_rows event on table dr_browser_db.dr_user_info; Duplicate entry '24' for key 'PRIMARY', Error_code:1062; handler error HA_ERR_FOUND_DUPP_KEY; the event's master logmysql-bin.000010, end_log_pos 5295916
```

我们通过错误日志看到了类型如下内容：

```
Duplicate entry '24' for key 'PRIMARY', Error_code:1062; handler error HA_ERR_FOUND_DUPP_KEY;
```

这说明出现了主键冲突，或者主键不存在的错误；此时，我们可以以幂等模式恢复：

```
# 寻找到从节点的pod
[root@master-01 ~]$ kubectl get pod -n mcamel-system -lhealthy,role | grep cluster-mysql | grep replica | awk '{print $1}'
mcamel-common-mysql-cluster-mysql-1

# 设置mysql 幂等模式
[root@master-01 ~]$ kubectl exec mcamel-common-mysql-cluster-mysql-1 -n mcamel-system -c mysql -- mysql --defaults-file=/etc/mysql/client.conf -NB -e 'stop slave;set global slave_exec_mode="IDEMPOTENT";set global sync_binlog=10086;start slave;'
```

也可以插入空事务的形式跳过错误：

```
mysql> stop slave;
mysql> SET @SESSION.GTID_NEXT= 'xxxxx:105220'; /* 具体数值，在日志里面提到 */
mysql> BEGIN;
mysql> COMMIT;
mysql> SET SESSION GTID_NEXT = AUTOMATIC;
mysql> START SLAVE;
```

执行完成以上操作后，我们可以观察下从库重建的进度，当主从没有延迟了之后，

```
# 进入mysql的容器
[root@master-01 ~]$ kubectl exec -it mcamel-common-mysql-cluster-mysql-1 -n mcamel-system -c mysql -- mysql --defaults-file=/etc/mysql/client.conf
```

查看从库的状态 `Seconds_Behind_Master` 为主从延迟，如果是 0~30，表示没有主从延迟了。表示从库追上了（主库）

```
mysql> show slave status\G;
```

等从库追上以后（`Seconds_Behind_Master` 小于 30s），执行下方指令，设定 MySQL 严格模式：

```
[root@master-01 ~]$ kubectl exec mcamel-common-mysql-cluster-mysql-1 -n mcamel-system -c mysql -- mysql --defaults-file=/etc/mysql/client.conf -NB -e 'stop slave;set global slave_exec_mode="STRICT";set global sync_binlog=10086;start slave;'
```

两个都是 REPLIC (从库)

```
[root@aster-01 ~]$ kubectl get pod -n mcamel-system -lhealthy,role|grep mysql
mcamel-common-mysql-cluster-mysql-0           4/4     Running   5 (16h ago)   16h    no      replica
mcamel-common-mysql-cluster-mysql-1           4/4     Running   6 (16h ago)   16h    no      replica
mysql-operator-0                            2/2     Running   1 (16h ago)   16h    no      replica
```

通过上方的命令，我们发现这 2 个 MySQL 的 Pod，都是 `replica` 角色，此时我们修正其中一个为 `master`：

```
[root@master-01 ~]$ kubectl exec -it mcamel-common-mysql-cluster-mysql-0 -n mcamel-system -c mysql -- mysql --defaults-file=/etc/mysql/client.conf
```

分别进入到 MySQL 的 Pod 内，查看 `slave` 的状态信息，为空的一个就是原来的 `master`：

```
-- mysql-0
mysql> show slave status\G;
empty set, 1 warning (0.00 sec)

-- mysql-1
mysql> show slave status\G;
***** 1. row *****
Slave_IO_State: Waiting for source to send event
Master_Host: mcamel-common-mysql-cluster-mysql-0.mysql.mcamel-system
Master_User: sys_replication
```

```

        Master_Port: 3306
        Connect_Retry: 1
        Master_Log_File: mysql-bin.000004
        Read_Master_Log_Pos: 38164242
        Relay_Log_File: mcamel-common-mysql-cluster-mysql-1-relay-bin.000002
        Relay_Log_Pos: 38164418
        Relay_Master_Log_File: mysql-bin.000004
        Slave_IO_Running: Yes
        Slave_SQL_Running: Yes
        Replicate_Do_DB:
        Replicate_Ignore_DB:
        Replicate_Do_Table:
        Replicate_Ignore_Table:
        Replicate_Wild_Do_Table:
        Replicate_Wild_Ignore_Table:
        Last_Error: 0
        Last_Error:
        Skip_Counter: 0
        Exec_Master_Log_Pos: 38164242
        Relay_Log_Space: 38164658
        Until_Condition: None
        Until_Log_File:
        Until_Log_Pos: 0
        Master_SSL_Allowed: No
        Master_SSL_CA_File:
        Master_SSL_CA_Path:
        Master_SSL_Cert:
        Master_SSL_Cipher:
        Master_SSL_Key:
        Seconds_Behind_Master: 0
        Master_SSL_Verify_Server_Cert: No
        Last_IO_Error: 0
        Last_IO_Error:
        Last_SQL_Error: 0
        Last_SQL_Error:
        Replicate_Ignore_Server_Ids:
        Master_Server_Id: 100
        Master_UUID: c16da70b-ad12-11ed-8084-0a580a810256
        Master_Info_File: mysql.slave_master_info
        SQL_Delay: 0
        SQL_Remaining_Delay: NULL
        Slave_SQL_Running_State: Replica has read all relay log; waiting for more updates
        Master_Retry_Count: 86400
        Master_Bind:
        Last_IO_Error_Timestamp:
        Last_SQL_Error_Timestamp:
        Master_SSL_Crl:
        Master_SSL_Crlpath:
        Retrieved_Gtid_Set: c16da70b-ad12-11ed-8084-0a580a810256:537-59096
        Executed_Gtid_Set: c16da70b-ad12-11ed-8084-0a580a810256:1-59096
        Auto_Position: 1
        Replicate_Rewrite_DB:
        Channel_Name:
        Master_TLS_Version:
        Master_public_key_path:
        Get_master_public_key: 0
        NetworkNamespace:
1 row in set, 1 warning (0.01 sec)

```

针对 master 的 mysql shell 执行:

```
mysql > stop slave;reset slave;
```

此时再手动 edit master 的 pod: `role replica => master ,healthy no => yes`; 针对 slave 的 mysql shell 执行:

```
mysql > start slave;
```

如果主从没有建立联系, 在 slave 的 mysql shell 执行:

```
-- 注意替换下 {master-host-pod-index}
mysql > change master to master_host='mcamel-common-mysql-cluster-mysql-{master-host-pod-index}.mysql.mcamel-
system',master_port=3306,master_user='root',master_password='{password}',master_auto_position=1,MASTER_HEARTBEAT_PERIOD=2,MASTER_CONNECT_RETRY=1,
MASTER_RETRY_COUNT=86400;
```

Mysql pod 节点是 (3/4) running 状态, describe 出 unhealthy 状态

```
17/1 Running 1 (45m ago) 110
] bash-4.2# kubectl get pod -n mcamel-system | grep mysql
] mcamel-common-mysql-cluster-mysql-0 3/4 Running 9 (4h2m ago) 9d
] mcamel-common-mysql-cluster-mysql-1 4/4 Running 11 (45m ago) 11d
] mcamel-common-mysql-cluster-mysql-2 2/2 Running 2 (45h ago) 9d
] mcamel-common-mysql-cluster-mysql-3 2/2 Running 4 (45h ago) 36d
] mcamel-common-mysql-operator-0 2/2 Running 28 (4h2m ago) 9d
] bash-4.2#
```

使用 `kubectl describe` 上图中框起来的 pod, 发现异常提示: Warning Unhealthy 4m50s (x7194 over 3h58m) kubelet Readiness probe failed:

此时需要手工进行修复, 这是目前开源 `mysql-operator` 版本的 BUG, 详情查看: [bugfix]update node_controller.go

修复方式有两种:

1. 可以重启 `mysql-operator`
2. 手工更新 `sys_operator` 的配置状态

```
kubectl exec mcamel-common-mysql-cluster-mysql-1 -n mcamel-system -c mysql -- mysql --defaults-file=/etc/mysql/client.conf -NB -e 'update sys_operator.status set value="1" WHERE name="configured"'
```

数据库运行正常, 使用 CR 创建数据库出现了报错

此类问题的原因有: mysql root 密码有特殊字符

```
=t Hostname : mcamel-common-mysql-cluster-mysql-1.mysql.mcamel-system , IsUpToDate : false , MasterHostname : 
I0216 03:10:58.392145      1 orchestrator_reconcile.go:548] orchestrator-reconciler "msg"="skip set read-only/writable" "key"={"Namespace":"mcamel-system","Name":"mcamel-common-mysql-cluster"} "instance"
= {"Hostname": "mcamel-common-mysql-cluster-mysql-1.mysql.mcamel-system", "IsUpToDate": "false", "MasterHostname": ""}
I0216 03:11:19.075301      1 orchestrator_reconcile.go:548] orchestrator-reconciler "msg"="skip set read-only/writable" "key"={"Namespace":"mcamel-system","Name":"mcamel-common-mysql-cluster"} "instance"
= {"Hostname": "mcamel-common-mysql-cluster-mysql-1.mysql.mcamel-system", "IsUpToDate": "false", "MasterHostname": ""}
I0216 03:11:21.209760      1 deleg.go:130] controller.mysql_database "msg"="creating MySQL database" "database"="insight" "name"="insight-database"
E0216 03:11:21.245071      1 controller.go:304] controller.mysql_database "msg"="Reconciler error" "error"="failed to create database, err: Error 1045: Access denied for user 'root'@'10.129.2.164' (using password: YES)" "name"="insight-database" "namespace"="mcamel-system"
I0216 03:11:38.372228      1 orchestrator_reconcile.go:548] orchestrator-reconciler "msg"="skip set read-only/writable" "key"={"Namespace":"mcamel-system","Name":"mcamel-common-mysql-cluster"} "instance"
= {"Hostname": "mcamel-common-mysql-cluster-mysql-0.mysql.mcamel-system", "IsUpToDate": "false", "MasterHostname": "mcamel-common-mysql-cluster-mysql-1.mysql.mcamel-system"}
I0216 03:12:13.676017      1 orchestrator_reconcile.go:548] orchestrator-reconciler "msg"="skip set read-only/writable" "key"={"Namespace":"mcamel-system","Name":"mcamel-common-mysql-cluster"} "instance"
= {"Hostname": "mcamel-common-mysql-cluster-mysql-0.mysql.mcamel-system", "IsUpToDate": "false", "MasterHostname": "mcamel-common-mysql-cluster-mysql-1.mysql.mcamel-system"}
I0216 03:12:33.383554      1 orchestrator_reconcile.go:548] orchestrator-reconciler "msg"="skip set read-only/writable" "key"={"Namespace":"mcamel-system","Name":"mcamel-common-mysql-cluster"} "instance"
= {"Hostname": "mcamel-common-mysql-cluster-mysql-0.mysql.mcamel-system", "IsUpToDate": "false", "MasterHostname": "mcamel-common-mysql-cluster-mysql-1.mysql.mcamel-system"}
I0216 03:12:33.383677      1 orchestrator_reconcile.go:548] orchestrator-reconciler "msg"="skip set read-only/writable" "key"={"Namespace":"mcamel-system","Name":"mcamel-common-mysql-cluster"} "instance"
```

获取原密码, 如果出现了 密码里面含有 - 特殊字符, 请换一个密码重装。

```
[root@master-01 ~]$ kubectl get secret -n mcamel-system mcamel-common-mysql-cluster-secret -o=jsonpath='{.data.ROOT_PASSWORD}' | base64 -d
# 如果密码是带有 '-' 时, 进入 mysql 的 shell 输入原密码出现以下错误
bash-4.4# mysql -uroot -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
```

8.3 Intro

产品优势

MySQL 具有以下产品优势。

- 行业领先
 - 支持 ActiveMQ 的快速部署与生命周期管理。ActiveMQ 是目前最流行的开源多协议数据库，支持 JavaScript, C, C++, Python, .Net 等语言编写的应用和平台，能够跨应用交换消息。通过消息控制台实现对生产集群、消费集群、消息查询等的管理操作。
 - 采用业界先进、成熟的技术路线，保证平台的稳定可靠性；同时充分考虑用户交互的便利性和界面操作的易用性，采用流行的灵活且易扩展系统架构，使用先进的技术框架路线进行设计与开发，支持基于 API 调用各组件协同工作，能够满足系统纵向集成与横向整合的信息交互。
 - 采用可靠的技术架构，支持系统稳定运行，每个组件都能提供高可用性，能够保障关键组件冗余与高可用。
 - 能够将自身运行的内在状态数据，通过采集、分析、处理后，通过合理的聚合，归纳为指标数据，方便运维人员在最短时间内了解到系统运行实时状态，同时可提供对应的 API 供其它系统读取状态指标数据。
 - 生命周期管理
 - 支持图形化或通过 YAML 创建、更新和删除 MySQL 实例。
 - 支持热部署、热更新，除通用技术服务组件升级、平台升级等重大升级改造外，系统部署和升级无需停机。
 - 图形化界面操作
 - 支持以图形化方式创建、查看、更新、删除以及弹性扩容。管理员可以根据自己的偏好，定制图形化参数。
 - 容器化消息平台
- MySQL 基于 Kubernetes 和 Docker，原生支持容器化部署，将资源利用率提高到最大。
- 兼容性和开放性
 - MySQL 兼容 Intel CPU 和国产化 CPU（如飞腾 ARM，华为鲲鹏 ARM 和海光 x86 等），兼容国产操作系统（如麒麟操作系统 v10，统信服务器操作系统 UOS 等），这些均有授权的电子证明和证书。
 - 具有开放的体系架构，提供开放标准 API 接口保证基础能力、管理服务、日志、监控等资源和服务被高效的调度、管理与使用，支持第三方管理平台通过 API 接口等方式实现相关编排与使用。

基本概念

本页的词汇表定义了 MySQL 背后的核心概念，以帮助您构建 MySQL 工作原理的思维模型，并了解文档在使用某些术语时所指代的内容。

- ACID 合规

ACID 合规性是 Atomicity、Consistency、Isolation 和 Durability 这 4 个单词的首字母缩写，是一组由原子性、一致性、隔离性和持久性组成的数据库特性，可确保高效完成数据库事务。

- ACL

访问控制列表或 ACL 是控制对系统资源访问的用户权限列表。

- Connection Status Metric (连接状态指标)

连接状态指标是衡量创建、连接和运行的线程数与数据库连接限制相关的指标。

- DBaaS (数据库即服务)

数据库即服务、托管数据库服务，简称为 DBaaS，这是一种云服务，允许用户在订阅的基础上访问云数据库系统，而无需拥有个人云数据系统。

- E2EE

英文全称为 End-to-end encryption，即端到端加密，或简称为 E2EE，是一种通信系统，它为所有人加密消息和消息服务，但接收消息的用户和发送消息的用户除外。

- Failover (故障转移)

故障转移是一种高可用性 (HA) 机制，可监控服务器的故障并在主服务器发生故障时将流量或操作重新路由到备用服务器。

- High Availability (高可用性)

高可用性 (HA) 是一种基础架构设计方法，专注于减少停机时间和消除单点故障。

- Hot Standby (热备)

热备用是侦听主节点何时发生故障以便备用节点取代其位置的行为。

- Index vs. Sequential Reads Metric (索引与顺序读取指标)

索引与顺序读取指标图显示了使用索引的读取占主服务器上所有数据库（模式）的读取总数的比例。

- LUKS Disk Encryption (LUKS 磁盘加密)

Linux 统一密钥设置磁盘加密 (LUKS) 是 Linux 存储设备的开源磁盘加密规范。

- Machine Type (机器类型)

机器类型是用于虚拟机 (VM) 实例的一组虚拟化硬件资源。

- Node Plan (节点计划)

节点计划、数据库或集群配置是节点规格的硬件计划。

- Operations Throughput Metric (运营吞吐量指标)

操作吞吐量指标是对服务器上所有数据库的获取、插入、更新和删除操作的吞吐量的度量。

- Point-In-Time-Recovery (恢复时间点)

恢复时间点（简称 PITR）确保进行自动备份，以便恢复在服务器先前状态下创建的数据。

- Port (端口)

端口是网络连接的通信端点。使用每个传输协议所用的端口号来标识一个端口。

- Read-Only Node (只读节点)

只读节点是集群主节点的副本。

- SQL Mode (SQL 模式)

SQL 模式或 `sql_mode` 是一个 MySQL 系统变量，用于配置 MySQL 服务器的操作特性。

- SSL Certificate (SSL 证书)

SSL 证书是描述网站身份的数字凭证。

- Standby Node (备用节点)

备用节点是在热备模式时设为空闲待用的节点。

- Tag (标记)

标记是与资源相关联的关键字，有助于管理资源所有权并组织对资源的查找和操作。

功能特性

本页说明有关 MySQL 的功能特性。

- 高可用架构

支持主从热备架构，灵活满足各类可用性需求，稳定可靠的性能远超业界平均水准。

- 快速部署

可在线快速部署实例，图形化界面操作简便，节省采购、部署、配置等自建数据库工作，缩短项目周期，帮助业务快速上线。

- 确保数据安全

基于实时副本技术，适配公有云、混合云和私有云，严密账户确保数据安全。

- 更低成本

可依据业务需求即时开通所需资源，无需在业务初期采购高成本硬件，有效减少初期的资产投入，避免资源闲置浪费。

- 弹性扩缩

可依据业务压力弹性扩缩数据库资源，满足不同业务阶段对于数据库性能和存储空间的需求。

- 自动运维

可设置自动备份策略、监控告警策略、自动扩容策略等。

在 DCE 5.0 中部署 MySQL 后，还将支持以下特性：

- 基于 Orchestrator 实现 MySQL 高可用和拓扑管理

- 支持单节点和主备模式

- 支持 phpmyadmin，提供管理页面

- 基于 mysqld-exporter 暴露指标

- 使用 Grafana Operator 集成 MySQL Dashboard，展示监控数据

- 使用 ServiceMonitor 对接 Prometheus 抓取指标

- 支持备份、恢复（依赖支持 S3 协议的存储）

什么是 MySQL

MySQL 是应用最广泛的开源关系数据库，是许多常见网站、应用程序和商业产品使用的主要关系数据存储，具有高吞吐、低延迟、可扩展等特性。

它是应用层协议的一个开放标准，是面向消息的中间件而设计，基于此协议的客户端与数据库来传递消息，不受产品、开发语言等条件的限制。

[创建 MySQL 实例](#)

8.4 User guide

创建 MySQL 实例

接入 MySQL 数据库后，参照以下步骤创建 MySQL 实例。

- 在实例列表中，点击右上角的 新建实例。

The screenshot shows the MySQL Database Services interface. It displays two MySQL instances:

- Instance 1:** Deployment location: kpanda-global-cluster/default, Version: 5.7.31, Deployment type: Single-node, Replicas: 1/1. Resource allocation: CPU Request 0.1 Core / Limit 1 Core, Memory Request 0.5 GB / Limit 1 GB, Disk 1 GB.
- Instance 2:** Deployment location: kpanda-global-cluster/skoala-sesa..., Version: 5.7.31, Deployment type: Single-node, Replicas: 0/1. Resource allocation: CPU Request 0.1 Core / Limit 1 Core, Memory Request 0.5 GB / Limit 1 GB, Disk 1 GB.

- 在创建 MySQL 实例页面中，配置基本信息后，点击下一步。

The screenshot shows the 'Create MySQL Instance' wizard, Step 1: Basic Information. The steps are numbered 1 to 4 at the top.

基本信息

MySQL 实例名称: mysql001
2~63 个字符，只能包含小写字母、数字及分隔符("-")，且必须以小写字母开头、字母或数字结尾。

描述: (empty)
描述信息不超过 256 个字符。

部署位置

集群: mcamel-test
命名空间: mcamel-system

Buttons: 取消, 下一步

- 选择部署类型、CPU、内存和存储等 规格配置 后，点击下一步。

基本信息

规格配置

服务设置

配置确认

版本 * 5.7.31

部署类型 * 单机 主备

使用主备模式，能更加保证数据的准确性和一致性。

副本数 * 1

CPU 配额 * 请求量 0.1 Core | 限制量 1 Core

内存配额 * 请求量 0.5 GB | 限制量 1 GB

存储类 * local-path

存储容量 * 0.5 GB

取消 上一步 下一步

4. 设置用户名和密码等服务设置，默认采用 ClusterIP 作为访问方式。

基本信息

规格配置

服务设置

配置确认

服务设置

访问类型 * 集群内访问(ClusterIP) ②

端口配置	协议	端口名称	服务端口	容器端口
	TCP	mysql	3306	3306
<small>+ 添加</small>				
注释	key	value		
<small>+ 添加</small>				

访问设置

用户名 * root

取消 上一步 下一步

5. 确认基本信息、规格配置、服务设置的信息准确无误后，点击 **确认**。

The screenshot shows the fourth step of a four-step MySQL instance creation wizard. The top navigation bar includes the DaoCloud logo, user 'admin', and a bell icon. The main area displays configuration details:

- 访问设置 (Access Settings):**
 - 用户名 (Username): root
 - 密码 (Password): *****
 - 管理工具 (Management Tool): 启用 phpMyAdmin (Enable phpMyAdmin)
 - phpMyAdmin CPU 配额 (phpMyAdmin CPU Quota): 请求量 0.1 Core / 限制量 1 Core
 - phpMyAdmin 内存配额 (phpMyAdmin Memory Quota): 请求量 0.5 G / 限制量 1 G
 - phpMyAdmin 访问类型 (phpMyAdmin Access Type): 集群内访问(ClusterIP) (Cluster IP access)
- phpMyAdmin 端口配置 (phpMyAdmin Port Configuration):**

协议 (Protocol)	端口名称 (Port Name)	服务端口 (Service Port)	容器端口 (Container Port)
TCP	phpmyadmin	80	80

At the bottom are '取消' (Cancel), '上一步' (Previous Step), and a blue '确认' (Confirm) button.

6. 返回实例列表，屏幕将提示 创建实例成功。

The screenshot shows the MySQL Database Services list. A success message '创建实例成功' (Instance created successfully) is displayed at the top right. The list contains two entries:

- 运行中 (Running):**
 - 部署位置 (Deployment Location): kpanda-global-cluster/default
 - 版本 (Version): 5.7.31
 - 部署类型 (Deployment Type): 单机 (Single Machine)
 - 副本数 (Replica Count): 1 副本
 - 资源配额 (Resource Quota):
 - CPU: 请求量 0.1 Core / 限制量 1 Core
 - 内存 (Memory): 请求量 0.5 GB / 限制量 1 GB
 - 磁盘 (Disk): 1 GB
 - 状态 (Status): 正常/全部副本数 (Normal / All replicas available) - 1 / 1
- aa (Pending):**
 - 部署位置 (Deployment Location): kpanda-global-cluster/skoala-sesa...
 - 版本 (Version): 5.7.31
 - 部署类型 (Deployment Type): 单机 (Single Machine)
 - 副本数 (Replica Count): 1 副本
 - 资源配额 (Resource Quota):
 - CPU: 请求量 0.1 Core / 限制量 1 Core
 - 内存 (Memory): 请求量 0.5 GB / 限制量 1 GB
 - 磁盘 (Disk): 1 GB
 - 状态 (Status): 正常/全部副本数 (Normal / All replicas available) - 0 / 1

删除 MySQL 实例

如果想要删除一个 MySQL 实例，可以执行如下操作：

- 在 MySQL 实例列表中，点击右侧的 **...** 按钮，在弹出菜单中选择 **删除实例**。

The screenshot shows the MySQL Database Service interface. At the top, there's a search bar and a 'New Instance' button. Below it, a table lists two MySQL instances:

	部署位置	资源配额	正常/全部副本数
aa	kpanda-global-cluster/default	CPU 请求量 0.1 Core / 限制量 1 Core 内存 请求量 0.5 GB / 限制量 1 GB 磁盘 1 GB	0 / 1
bb	kpanda-global-cluster/skoala-sesa...	CPU 请求量 0.1 Core / 限制量 1 Core 内存 请求量 0.5 GB / 限制量 1 GB 磁盘 1 GB	0 / 1

A context menu is open over the 'aa' instance, with the following options:

- 更新实例
- 删除实例** (highlighted with a red border)
- ... (three dots)

- 在弹窗中输入该实例的名称，确认无误后，点击 **删除** 按钮。

The screenshot shows a confirmation dialog box with the following content:

确认删除实例 aa 吗？

确认删除实例 aa 吗？删除后对应的数据将会全部丢失，请谨慎操作。

请输入aa

aa

删除 取消

!!! warning

删除实例后，该实例相关的所有信息也会被全部删除，请谨慎操作。

查看 MySQL 日志

操作步骤

通过访问每个 MySQL 的实例详情，页面可以支持查看 MySQL 的日志。

- 在 MySQL 实例列表中，选择想要查看的日志，点击 实例名称 进入到实例详情页面。

The screenshot shows the MySQL Database Service interface. At the top, there's a search bar and several navigation buttons. Below, a list of MySQL instances is displayed in cards:

- mcamel-common-mysql-cluster**: Status: 未就绪 (Not Ready). Details: 部署位置: kpanda-global-cluster/mcamel-sys..., 版本: 5.7, 部署类型: 主备, 副本数: 2副本. Resource usage: CPU 请求量 0.2 Core, 内存 请求量 0.293 GB, 磁盘 50 GB. Status: 正常/全部副本数 1 / 2.
- test-mysql**: Status: 运行中 (Running). Details: 部署位置: mcamel-test/mcamel-system, 版本: 8.0.31, 部署类型: 单机, 副本数: 1副本. Resource usage: CPU 请求量 0 Core, 内存 请求量 0 GB, 磁盘 1 GB. Status: 正常/全部副本数 1 / 1.
- test-mysql-8029**: Status: 运行中 (Running). Details: 部署位置: mcamel-test/mcamel-system, 版本: 8.0.29, 部署类型: 单机, 副本数: 1副本. Resource usage: CPU 请求量 0.1 Core, 内存 请求量 0.5 GB, 磁盘 1 GB. Status: 正常/全部副本数 1 / 1.

- 在实例的左侧菜单栏，会发现有一个日志查看的菜单栏选项。

The screenshot shows the MySQL instance detail page for 'test-mysql'. The left sidebar has a '日志查看' (Log View) option highlighted with a red circle containing a number 1. The main content area includes:

- 基本信息**: 显示了实例名称 (test-mysql, 运行中), 部署位置 (mcamel-test/mcamel-system), 创建时间 (2023-02-21 14:03), 版本 (8.0.31).
- 资源配额**: 显示了 CPU 使用率 (0%), 内存使用率 (0%), 磁盘存储量 (1 GB).
- 监控告警 (最近 10 条)**: 显示了最近 10 条告警记录的列表，包括规则名称、告警级别、资源类型、资源名称、描述、发生时间、持续时间等。

- 点击 日志查看 即可进入到日志查看页面（Insight 日志查看）。

日志查看说明

在日志查看页面，我们可以很方便的进行日志查看，常用操作说明如下：

- 支持 自定义日志时间范围，在日志页面右上角，可以方便地切换查看日志的时间范围（可查看的日志范围以 可观测系统设置内保存的日志时长为准）
- 支持 关键字检索日志，左侧检索区域支持查看更多的日志信息
- 支持 日志量分布查看，中上区域柱状图，可以查看在时间范围内的日志数量分布
- 支持 查看日志的上下文，点击右侧 上下文 图标即可
- 支持 导出日志

The screenshot shows the DaoCloud observability platform's log search feature. The left sidebar has a tree structure with '可观测性' (Observability) as the root, followed by '概览', '仪表盘', '资源监控', '场景监控', '数据查询', '指标查询', '日志查询' (selected), '链路查询', '告警中心', '采集管理', and '系统管理'. The main area is titled '日志查询' (Log Search). It includes a search bar with placeholder '请输入日志内容' (Enter log content) (2), a dropdown for '集群' (Cluster) set to 'kpanda-global-cluster', a histogram showing log volume from 00:34 to 00:48 (3), and a table of logs (4). Each log entry has a timestamp, logger, and message. A context menu (5) is open over a log entry from 'elastic-operator-0 manager' on 2023-02-27 at 00:48, with options like '查看上下文' (View Context).

时间	日志
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.268Z","log.logger":"elasticsearch-controller","message":"Ending reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters","took":0.466213453}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.242Z","log.logger":"zen2","message":"Ensuring no voting exclusions are set","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"} 5
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.196Z","log.logger":"transport","message":"Skipping pod because it has no IP yet","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","pod_name":"mcamel-common-es-cluster-masters-es-masters-2"}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:54.802Z","log.logger":"elasticsearch-controller","message":"Starting reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"}

更新 MySQL 实例

如果想要更新或修改 MySQL 的资源配置，可以按照本页说明操作。

- 在实例列表中，点击右侧的 **...** 按钮，在弹出菜单中选择 **更新实例**。

MySQL 数据库服务 Default ↗

aa ● 运行中

部署位置 kpanda-global-cluster/default 资源配额 CPU 请求量 0.1 Core / 限制量 1 Core
版本 5.7.31 内存 请求量 0.5 GB / 限制量 1 GB
部署类型 单机 磁盘 1 GB
副本数 1 副本 正常/全部副本数

... 更新实例 删除实例

- 基本信息：只能修改描述。然后点击 **下一步**。

更新实例 aa

1 2 3

基本信息 规格配置 服务设置

基本信息

MySQL 实例名称 aa

描述 这是一个 test 实例
描述信息不超过 256 个字符。

部署位置

集群 kpanda-global-cluster

命名空间 default

取消 下一步

- 修改规格配置后点击 **下一步**。

更新实例 aa

1. 基本信息

2. 规格配置

3. 服务设置

版本 * 5.7.31

部署类型 * 单机 主备

使用主备模式，能更加保证数据的准确性和一致性。

副本数 * 1

CPU 配额 * 请求量 0.1 Core 限制量 1 Core

内存配额 * 请求量 0.5 GB 限制量 1 GB

存储类 local-path

存储容量 * 1 GB

[取消](#) [上一步](#) [下一步](#)

4. 修改服务设置后点击 确认。

更新实例 aa

1. 基本信息

2. 规格配置

3. 服务设置

访问类型 * 集群内访问(ClusterIP) [?](#)

端口配置	协议	端口名称	服务端口	容器端口
	TCP	mysql	3306	3306
+ 添加				
注释	key	value		
+ 添加				

访问设置

管理工具 [?](#) 关闭 phpMyAdmin

[取消](#) [上一步](#) [确认](#)

9. Rabbitmq

9.1 RabbitMQ Release Notes

本页列出 RabbitMQ 消息队列的 Release Notes，便于您了解各版本的演进路径和特性变化。

v0.8.0

发布日期：2023-02-23

API

- 新增 `mcamel-rabbitmq helm-docs` 模板文件。
- 新增 `mcamel-rabbitmq` 应用商店中的 Operator 只能安装在 `mcamel-system`。
- 新增 `mcamel-rabbitmq` 支持 cloud shell。
- 新增 `mcamel-rabbitmq` 支持导航栏单独注册。
- 新增 `mcamel-rabbitmq` 支持查看日志。
- 修复 `mcamel-rabbitmq` 实例名太长导致自定义资源无法创建的问题。
- 修复 `mcamel-rabbitmq` 工作空间 Editor 用户无法查看实例密码。
- 升级 `mcamel-rabbitmq` 升级离线镜像检测脚本。

文档

- 新增 日志查看操作说明，支持自定义查询、导出等功能。

v0.7.0

发布日期：2022-12-25

API

- 新增 `mcamel-rabbitmq` NodePort 端口冲突提前检测。
- 新增 `mcamel-rabbitmq` 节点亲和性配置。

UI

- 优化 `mcamel-rabbitmq-ui` 中间件样式走查优化。

v0.6.4

发布日期: 2022-11-28

- 新增 获取用户列表接口
- 改进 密码校验调整为 MCamel 中等密码强度
- 新增 支持多架构的镜像, 配置方式为 `depend.arm64-img.rabbitClusterImageFormat: xxxx`
- 新增 支持 sc 扩容拦截, 当 sc 不支持扩容的时候, 直接拦截掉
- 新增 返回列表或者详情时的公共字段
- 新增 返回 alerts
- 新增 校验 Service 注释
- 修复 页面控制台可能访问到错误的端口

v0.6.1

发布日期: 2022-10-27

API

- 新增 增加覆盖率
- 新增 前端的 UI 注册功能
- 新增 性能增强
- 新增 列表页增加分页功能
- 新增 增加修改配置的功能
- 新增 增加返回可修改配置项的功能
- 新增 更改创建实例的限制为集群级别, 原来为 namespace 级别
- 新增 增加监控地址的拼接功能
- 新增 增加可以修改版本号的功能
- 新增 修改底层 update 逻辑为 patch 逻辑
- 新增 RabbitMQ e2e 测试覆盖率 17.24% 左右
- 新增 增加 RabbitMQ 性能压测报告
- 新增 增加 RabbitMQ bug 抽查
- 新增 对接 ghippo 增加 workspace 接口
- 新增 对接 insight 通过 crd 注入 dashboard
- 新增 将时间戳 api 字段统一调整为 int64
- 新增 单测覆盖率提升到 53%
- 优化 更新 release note 脚本, 执行 release-process 规范

文档

- 新增 新增功能说明
- 新增 创建 RabbitMQ
- 新增 RabbitMQ 数据迁移
- 新增 实例监控
- 新增 首次进入 RabbitMQ
- 新增 适用场景

9.2 Intro

产品优势

RabbitMQ 具有以下产品优势。

- 行业领先
 - 支持 ActiveMQ 的快速部署与生命周期管理。ActiveMQ 是目前最流行的开源多协议消息中间件，支持 JavaScript, C, C++, Python, .Net 等语言编写的应用和平台，能够跨应用交换消息。通过消息控制台实现对生产集群、消费集群、消息查询等的管理操作。
 - 采用业界先进、成熟的技术路线，保证平台的稳定可靠性；同时充分考虑用户交互的便利性和界面操作的易用性，采用流行的灵活且易扩展系统架构，使用先进的技术框架路线进行设计与开发，支持基于 API 调用各组件协同工作，能够满足系统纵向集成与横向整合的信息交互。
 - 采用可靠的技术架构，支持系统稳定运行，每个组件都能提供高可用性，能够保障关键组件冗余与高可用。
 - 能够将自身运行的内在状态数据，通过采集、分析、处理后，通过合理的聚合，归纳为指标数据，方便运维人员在最短时间内了解到系统运行实时状态，同时可提供对应的 API 供其它系统读取状态指标数据。
 - 生命周期管理
 - 支持图形化或通过 YAML 创建、更新和删除 RabbitMQ 实例。
 - 支持热部署、热更新，除通用技术服务组件升级、平台升级等重大升级改造外，系统部署和升级无需停机。
 - 图形化界面操作
 - 支持以图形化方式创建、查看、更新、删除以及弹性扩容。管理员可以根据自己的偏好，定制图形化参数。
 - 容器化消息平台
- RabbitMQ 基于 Kubernetes 和 Docker，原生支持容器化部署，将资源利用率提高到最大。
- 兼容性和开放性
 - RabbitMQ 兼容 Intel CPU 和国产化 CPU（如飞腾 ARM，华为鲲鹏 ARM 和海光 x86 等），兼容国产操作系统（如麒麟操作系统 v10，统信服务器操作系统 UOS 等），这些均有授权的电子证明和证书。
 - 具有开放的体系架构，提供开放标准 API 接口保证基础能力、管理服务、日志、监控等资源和服务被高效的调度、管理与使用，支持第三方管理平台通过 API 接口等方式实现相关编排与使用。

基本概念

本节列出有关 RabbitMQ 涉及的专有名词及术语，方便您更好地理解相关概念并使用 RabbitMQ 消息队列。

- **消息 (Message)**

消息一般分为两部分：消息体和标签。标签也称为消息头，主要用来描述这条消息。消息体是消息的内容，是一个 json 体或者数据等。

消息体是不透明的，而消息头则由一系列的可选属性组成，这些属性包括 routing-key（路由键）、priority（相对于其他消息的优先权）、delivery-mode（指出该消息可能需要持久性存储）等。

生产者发布消息，消费者使用消息，生产者和消费者彼此并无直接关系。

- **消息标识 (Message ID)**

消息标识是消息的可选属性，类型为一个短字符串 (short string)。

- **队列 (Queue)**

队列用于存储消息，生产者将消息送到队列，消费者从队列中获取消息。多个消费者可以同时订阅同一个队列，队列里的消息分配给不同的消费者。每个消息都会被投入到一个或多个队列里。

- **消息存活时间**

消息在队列中的有效期。某条消息在队列中的留存时间超过配置的消息存活时间时，则该消息过期。消息存活时间的值必须为非负整型数，单位为毫秒。例如，某条消息的存活时间的值是 1000，则代表该消息最多会在队列中存活 1 秒。

- **延时消息**

生产者将消息发布到消息队列 RabbitMQ 版服务端，但并不期望这条消息立马投递，而是延迟一定时间后才投递到消费者进行消费，该消息即延时消息。

- **生产者 (Publisher)**

消息的生产者，也是一个向交换器发布消息的客户端应用程序。即向队列发布消息的一方。发布消息的最终目的在于将消息内容传递给其他系统/模块，使对方按照约定处理该消息。

- **消费者 (Consumer)**

消息的消费者，表示一个从消息队列中取得消息的客户端应用程序。消费者订阅 RabbitMQ 的队列。当消费者使用一条消息时，只是使用消息的消息体。在消息路由的过程中，会丢弃标签，存入到队列中的只有消息体。

- **代理 (Broker)**

消息中间件的服务节点，即消息队列服务器实体。

功能特性

RabbitMQ 的通用功能特性包括：

- 可靠性 (Reliability)

RabbitMQ 使用一些机制来保证可靠性，如持久化、传输确认、发布确认。

- 消息集群 (Clustering)

多个 RabbitMQ 服务器可以组成一个集群，形成一个逻辑 Broker。

- 高可用队列 (Highly Available Queues)

队列可以在集群中的主机上进行镜像，使得在部分节点出问题的情况下队列仍然可用。

- 多种协议 (Multi-protocol)

RabbitMQ 支持多种消息队列协议，比如 STOMP、MQTT 等。

- 多语言客户端 (Many Clients)

RabbitMQ 几乎支持所有常用语言，比如 Java、.NET、Ruby 等。

- 管理界面 (Management UI)

RabbitMQ 提供了一个易用的图形用户界面，使得用户可以监控和管理消息 Broker 的方方面面。

- 跟踪机制 (Tracing)

如果消息异常，RabbitMQ 提供了消息跟踪机制，用户可以轻松找出发生了什么。

- 插件机制 (Plugin System)

RabbitMQ 提供了许多插件，支持从多方面进行扩展，也可以编写自己的插件。

在 DCE 5.0 中部署 RabbitMQ 后，还将支持以下特性：

- 支持单节点和多节点 RabbitMQ 集群部署
- 支持 RabbitMQ Management 插件，提供管理页面
- 支持 RabbitMQ Prometheus 插件，暴露监控指标
- 使用 ServiceMonitor 对接 Prometheus 抓取指标
- 支持 RabbitMQ 集群的扩容和滚动升级

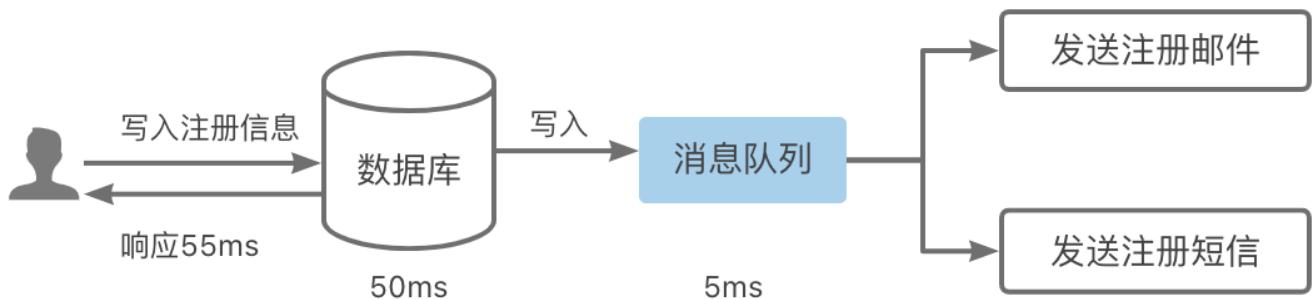
适用场景

RabbitMQ 适用的场景广泛，本节列出了几个典型场景。

异步处理

场景说明：用户注册后，需要发送注册邮件和注册短信。

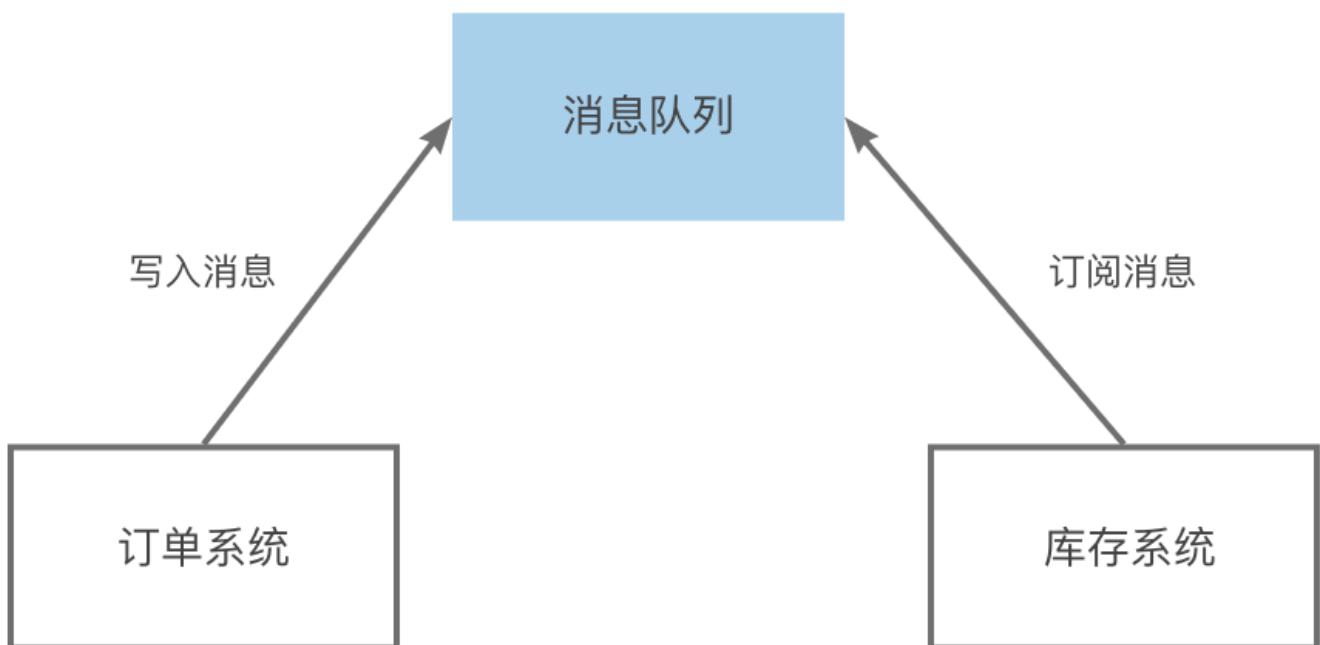
引入消息队列后，用户的响应时间就等于写入数据库的时间 + 写入消息队列的时间（这个可以忽略不计）。引入消息队列后处理后，响应时间是串行的 3 倍，是并行的 2 倍。



应用解耦

场景说明：在促销活动时用户下单数量激增，用户下单后，订单系统需要通知库存系统。

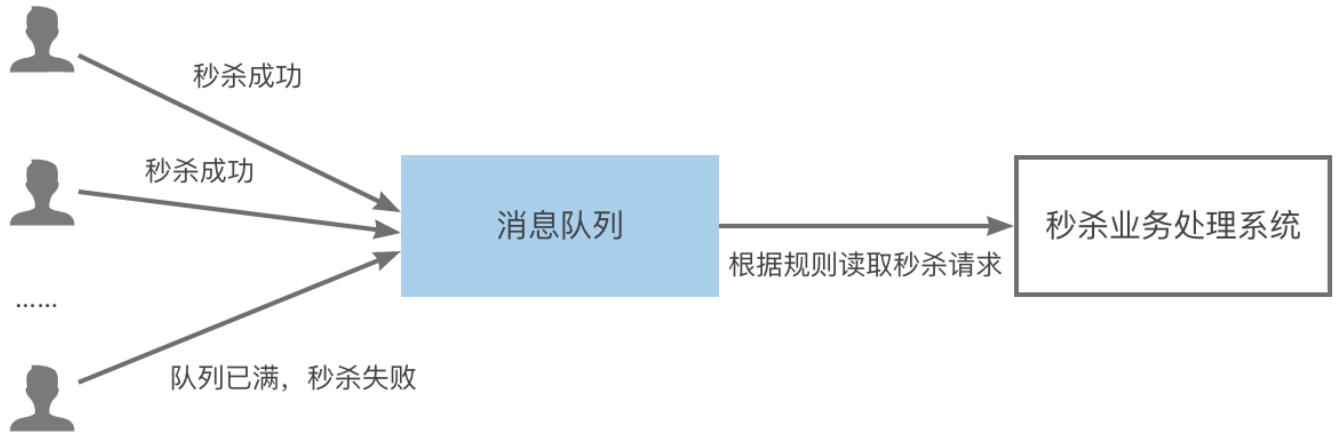
传统的做法就是订单系统调用库存系统的接口，这会导致库存系统出现故障时，订单会失败。如果使用消息队列（如下图），用户下单后，订单系统完成持久化处理，将消息写入消息队列，返回用户订单下单成功。订阅下单的消息，获取下单消息，进行库操作。就算库存系统出现故障，消息队列也能保证消息的可靠投递，不会导致消息丢失。



流量削峰

场景说明：某平台策划的秒杀活动，一般会因为流量过大，导致应用不可用。

为了解决这个问题，一般在应用前端加入消息队列。通过消息队列可以控制活动人数，超过此一定阀值的订单直接丢弃，同时可以缓解短时间的高流量压垮应用。服务器收到用户的请求之后，首先写入消息队列，假如消息队列长度超过最大值，则直接抛弃用户请求或跳转到错误页面。秒杀业务根据消息队列中的请求信息，再做后续处理。



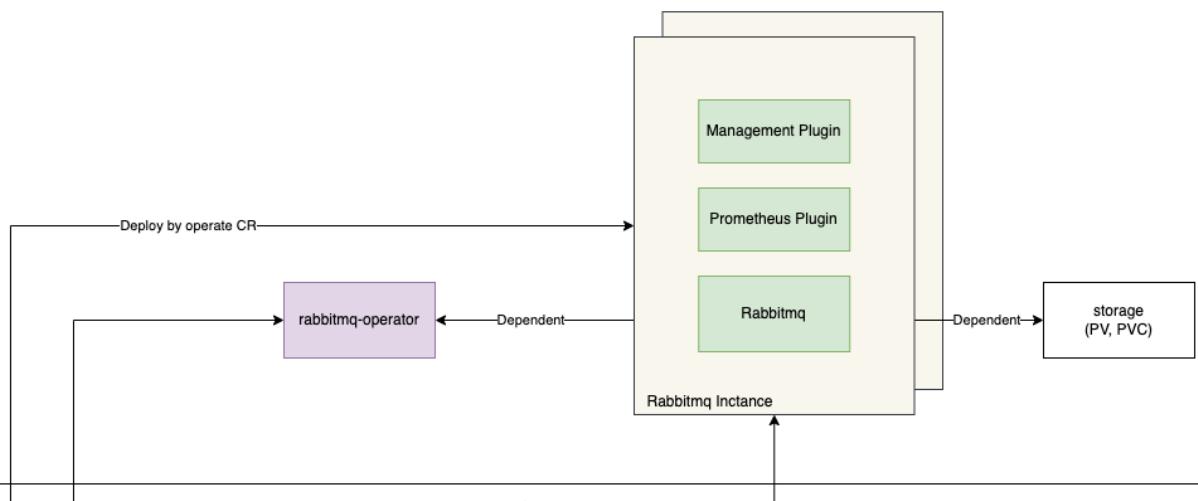
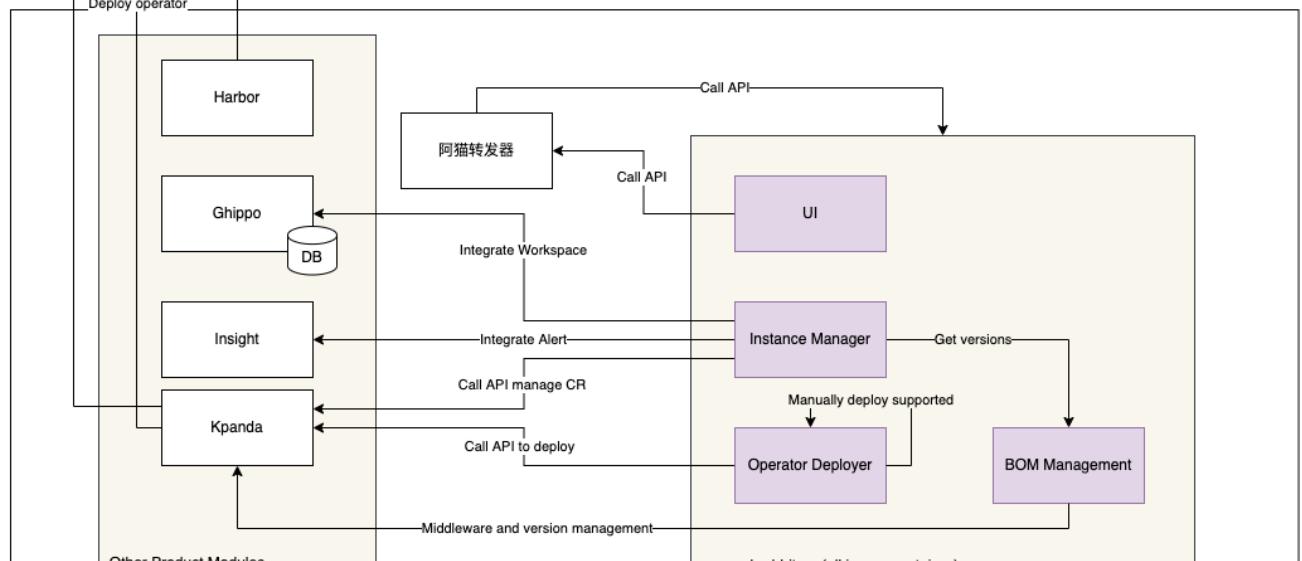
什么是 RabbitMQ

RabbitMQ 是实现了高级消息队列协议（AMQP）的开源消息代理软件（亦称面向消息的中间件），具有高吞吐、低延迟、可扩容等特性。

RabbitMQ 是应用层协议的一个开放标准，为面向消息的中间件设计，基于此协议的客户端与消息中间件来传递消息，不受产品、开发语言等条件的限制。这是一个企业级的真正具备低延迟、高并发、高可用、高可靠特性，可支撑万亿级数据洪峰的分布式消息中间件服务。



其系统架构及数据流如下图。

Worker Clusters**Deploy operator****Global Cluster**

创建 RabbitMQ 实例 { .md-button .md-button--primary }

9.3 User guide

创建 RabbitMQ

在 RabbitMQ 消息队列中，执行以下操作：

1. 在右上角点击 新建实例 。

实例名	状态	部署位置	版本	副本数	资源配额	磁盘	正常/全部副本数
mcamelt1	运行中	mcamel-test/mcamel-system	3.8.30	三副本	CPU 请求量 0.2 Core / 限制量 0.2 C... 内存 请求量 0.5 GB / 限制量 0.5 GB 磁盘 3 GB	3 / 3	正常/全部副本数
test000106	运行中	mcamel-test/mcamel-system	3.8.30	三副本	CPU 请求量 0.1 Core / 限制量 0.2 C... 内存 请求量 0.1 GB / 限制量 0.5 GB 磁盘 30 GB	0 / 3	正常/全部副本数

2. 在 创建 RabbitMQ 实例 页面中，设置基本信息后，点击下一步。

创建 RabbitMQ 实例

基本信息

RabbitMQ 实例名称 * instance01
2 ~ 63 个字符，只能包含小写字母、数字及分隔符("-")，且必须以小写字母开头、字母或数字结尾。

描述
描述信息不超过 256 个字符。

部署位置

集群 * kpanda-global-cluster

命名空间 * mcamel-system

下一步 取消

3. 配置规格后，点击下一步。

- 版本：RabbitMQ 的版本号，当前仅支持 RabbitMQ 3.7.20。
- 副本数：支持 1、3、5、7 副本数。
- 资源配额：根据实际情况选择规则。
- 存储卷：选择 RabbitMQ 实例的存储卷和储存空间总量。



4. 服务设置后，点击下一步。

- 访问方式：可以选择集群内访问还是 Nodeport 访问。
- 服务设置：设置连接 RabbitMQ 实例的用户名、密码。

DaoCloud

admin

← 创建 RabbitMQ 实例

基本信息

规格配置

服务设置

配置确认

服务设置

访问类型 • 集群内访问(ClusterIP) 节点端口(NodePort)

端口配置	协议	端口名称	服务端口	容器端口
TCP	amqp	5672	5672	x
TCP	management	15672	15672	x
TCP	prometheus	15692	15692	x

+ 添加

注释

key value

+ 添加

访问设置

用户名 • testname
4~64个字符，以英文字母开头，由英文小写字母、数字、下划线、中划线组成。

密码 • ········
6~18个字符，必须同时包含英文大小写、数字、特殊符号(!@#\$%^&_=+/-=+)

确认密码 • ········
该用户名、密码用于 RabbitMQ 网页控制台默认管理员用户密码，请谨慎操作。

取消 上一步 下一步

5. 确认实例信息无误，点击确认完成创建。

DaoCloud

admin

← 创建 RabbitMQ 实例

基本信息 规格配置 服务设置 配置确认 (4)

基本信息

RabbitMQ 实例名称: instance01
部署位置: kpanda-global-cluster/mcamel-system

规格配置

版本: 3.8.30
副本数: 单副本
CPU 配额: 请求量 0.2 Core 限制量 0.2 Core
内存配额: 请求量 0.5 G 限制量 0.5 G
存储类: storagetest9
存储容量: 0.5

服务设置

服务设置

访问类型: 集群内访问(ClusterIP)

端口配置	协议	端口名称	服务端口	容器端口
	TCP	amqp	5672	5672
	TCP	management	15672	15672

取消 上一步 确认

6. 在实例列表页查看实例是否创建成功。刚创建的实例状态为未就绪，等几分钟后该状态变为运行中。

DaoCloud

RabbitMQ 消息队列 Default

创建实例成功

admin

搜索 新建实例

instance01 未就绪

部署位置 kpanda-global-cluster/mcamel-syst... 资源配额 CPU 请求量 0.2 Core / 限制量 0.2 C...
版本 3.8.30 内存 请求量 0.5 GB / 限制量 0.5 GB
副本数 单副本 磁盘 0.5 GB 0 / 1 正常/全部副本数

mq123 运行中

部署位置 kpanda-global-cluster/mcamel-syst... 资源配额 CPU 请求量 0.1 Core / 限制量 1 Core
版本 3.8.30 内存 请求量 0.5 GB / 限制量 1 GB
副本数 单副本 磁盘 2 GB 1 / 1 正常/全部副本数

共 2 项 1 / 1 10 项

删除 RabbitMQ

如果想要删除一个消息队列，可以执行如下操作：

- 在消息队列中，点击右侧的 **...** 按钮，在弹出菜单中选择 **删除实例**。

队列名称	状态	部署位置	版本	副本数	资源配额	正常/全部副本数
mcamelt1	运行中	mcamel-test/mcamel-system	3.8.30	三副本	CPU 请求量 0.2 Core / 限制量 0.2 Core 内存 请求量 0.5 GB / 限制量 0.5 GB 磁盘 3 GB	更新实例 删除实例 正常/全部副本数
test000106	运行中	mcamel-test/mcamel-system	3.8.30	三副本	CPU 请求量 0.1 Core / 限制量 0.2 Core 内存 请求量 0.1 GB / 限制量 0.5 GB 磁盘 30 GB	0 / 3 正常/全部副本数

- 在弹窗中输入该消息队列的名称，确认无误后，点击 **删除** 按钮。



!!! warning

删除实例后，该实例相关的所有消息也会被全部删除，请谨慎操作。

实例监控

RabbitMQ 内置了 Prometheus 和 Grafana 监控模块。

- 在消息队列页面中，点击某个名称。

RabbitMQ 消息队列

Default

搜索 新建实例

mcamel1	运行中	...		
部署位置	mcamel-test/mcamel-system	资源配额	CPU 请求量 0.2 Core / 限制量 0.2 Core	3 / 3
版本	3.8.30	内存	请求量 0.5 GB / 限制量 0.5 GB	正常/全部副本数
副本数	三副本	磁盘	3 GB	

test000106	运行中	...		
部署位置	mcamel-test/mcamel-system	资源配额	CPU 请求量 0.1 Core / 限制量 0.2 Core	0 / 3
版本	3.8.30	内存	请求量 0.1 GB / 限制量 0.5 GB	正常/全部副本数
副本数	三副本	磁盘	30 GB	

- 在左侧导航栏，点击 实例监控，可以接入监控模块。

DaoCloud

mcamel1 Default

RabbitMQ 实例: mcamelt1 / 实例监控

mcamel-system / RabbitMQ-Overview

Last 15 minutes

Ready messages	Incoming messages / s	Publishers	Connections	Queues
N/A	N/A	N/A	N/A	N/A
Unacknowledged messages	Outgoing messages / s	Consumers	Channels	Nodes
N/A	N/A	N/A	N/A	N/A

▼ NODES

Messages ready to be delivered to consumers

Total connections

Total channels

Total queues

各项监控指标如下。

Panel 名	指标名称	说明
connections	连接数	该指标用于统计 RabbitMQ 实例中的总连接数。
channels	通道数	该指标用于统计 RabbitMQ 实例中的总通道数。
queues	队列数	该指标用于统计 RabbitMQ 实例中的总队列数。
consumers	消费者数	该指标用于统计 RabbitMQ 实例中的总消费者数。
publish	生产速率	统计 RabbitMQ 实例中实时消息生产速率。
socket_used	Socket 连接数	该指标用于统计当前节点 RabbitMQ 所使用的 Socket 连接数。
CPU Usage	CPU 使用量	该指标用于统计节点 CPU 使用量。
Memory Usage	内存使用量	该指标用于统计节点内存使用量。

查看 RabbitMQ 日志

操作步骤

通过访问每个 RabbitMQ 的实例详情，页面可以支持查看 RabbitMQ 的日志。

- 在 RabbitMQ 实例列表中，选择想要查看的日志，点击 实例名称 进入到实例详情页面。

The screenshot shows the DaoCloud platform's RabbitMQ instance management interface. At the top, there's a navigation bar with the DaoCloud logo and user information. Below it is a search bar and a 'New Instance' button. The main area is titled 'RabbitMQ 消息队列' and shows two instances: 'test-mq-li' and 'test-priv'. Each instance card provides details like deployment location, version, replica count, and access address. The 'test-mq-li' instance is currently selected, indicated by a red border around its card.

- 在实例的左侧菜单栏，会发现有一个日志查看的菜单栏选项。

This screenshot shows the detailed view of the 'test-mq-li' RabbitMQ instance. The left sidebar has a 'Logs' section with a red box highlighting the 'Logs' link. The main content area includes sections for 'Basic Information', 'Resource Allocation', and 'Monitoring Settings'. The 'Basic Information' section shows the instance name, status, deployment location, creation time, and access address. The 'Resource Allocation' section shows CPU, memory, and disk resources. The 'Monitoring Settings' section includes NodePort, access address, and password fields.

- 点击「日志查看」即可进入到日志查看页面（Insight 日志查看）。

日志查看说明

在日志查看页面，我们可以很方便的进行日志查看，常用操作说明如下：

- 支持 自定义日志时间范围，在日志页面右上角，可以方便地切换查看日志的时间范围（可查看的日志范围以 可观测系统设置内保存的日志时长为准）
- 支持 关键字检索日志，左侧检索区域支持查看更多的日志信息
- 支持 日志量分布查看，中上区域柱状图，可以查看在时间范围内的日志数量分布
- 支持 查看日志的上下文，点击右侧 上下文 图标即可
- 支持 导出日志

The screenshot shows the DaoCloud observability platform's log search feature. The left sidebar has a tree structure with '可观测性' (Observability) as the root, followed by '概览', '仪表盘', '资源监控', '场景监控', '数据查询', '指标查询', '日志查询' (selected), '链路查询', '告警中心', '采集管理', and '系统管理'. The main area is titled '日志查询' (Log Search). It includes a search bar with placeholder '请输入日志内容' (Enter log content) (2), a '筛选' (Filter) section with a '搜索' input field, a '集群' (Cluster) dropdown set to 'kpanda-global-cluster', and a histogram showing log volume from 00:34 to 00:48. Below the histogram is a table of logs (日志) with columns '时间' (Time) and '日志' (Log). The first log entry is highlighted (5) with a context menu containing '查看上下文' (View Context), '下载' (Download), and other options. The top right corner shows a user 'admin' and a time range selector '最近 15 分钟'.

时间	日志
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.268Z","log.logger":"elasticsearch-controller","message":"Ending reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters","took":0.466213453}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.242Z","log.logger":"zen2","message":"Ensuring no voting exclusions are set","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"} 5
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.196Z","log.logger":"transport","message":"Skipping pod because it has no IP yet","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","pod_name":"mcamel-common-es-cluster-masters-es-masters-2"}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:54.802Z","log.logger":"elasticsearch-controller","message":"Starting reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"}

RabbitMQ 数据迁移

RabbitMQ 的数据包括元数据（RabbitMQ 用户、vhost、队列、交换和绑定）和消息数据，其中消息数据存储在单独的消息存储库中。

由于业务需要，要求把 `rabbitmq-cluster-a` 集群上的数据迁移到 `rabbitmq-cluster-b` 集群上。

数据迁移步骤

!!! info

从 v3.7.0 开始，RabbitMQ 将所有消息数据存储在 `msg_stores/vhosts` 目录中，并存储在每个 vhost 的子目录中。每个 vhost 目录都以哈希命名，并包含一个带有 vhost 名称的 `.vhost` 文件，因此可以单独备份特定 vhost 的消息集。[了解\[更多信息\] \(https://www.rabbitmq.com/backup.html\)](https://www.rabbitmq.com/backup.html)。

RabbitMQ 数据迁移，可以采用如下两种方案：

- 方案一：不迁移数据，先切换生产端，再切换消费端。
- 方案二：先迁移数据，然后同时切换生产端和消费端。

方案一

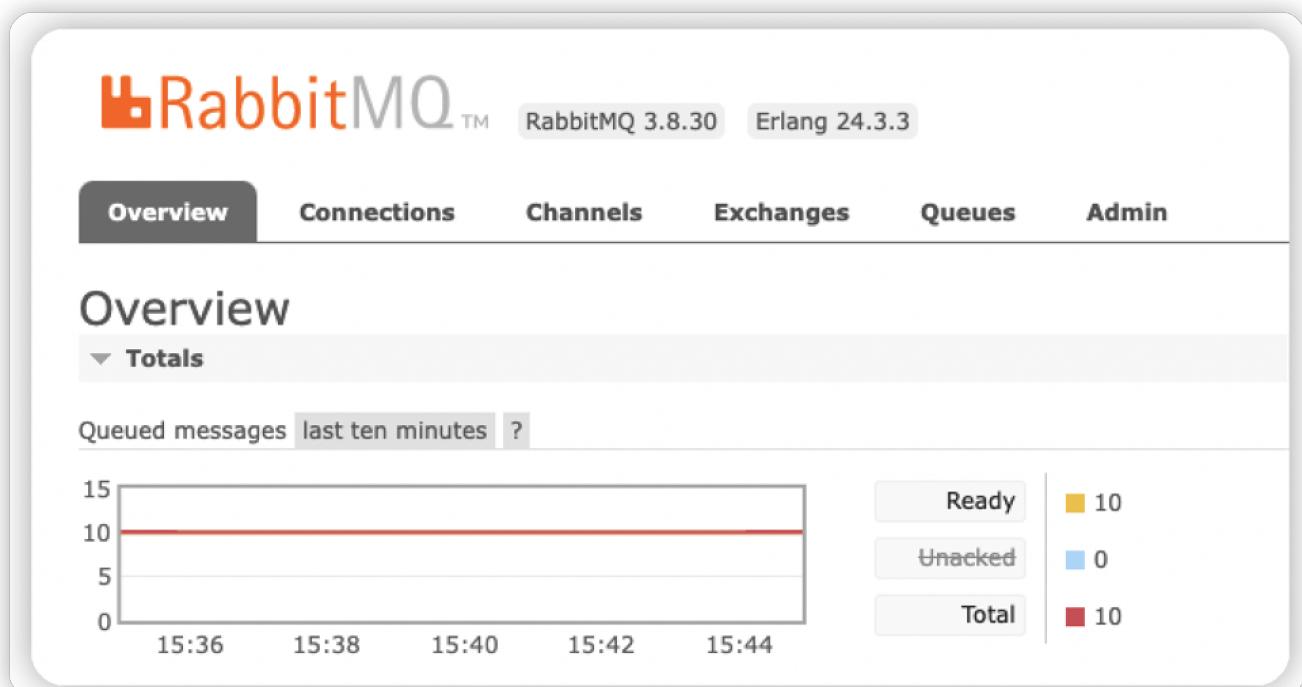
不迁移数据，先切换生产端，再切换消费端。

操作流程

1. 将消息生产端切换到集群 `rabbitmq-cluster-b`，不再生产消息到 `rabbitmq-cluster-a` 集群中。
2. 消费端同时消费 `rabbitmq-cluster-a` 和 `rabbitmq-cluster-b` 集群中的消息，当 `rabbitmq-cluster-a` 集群中消息全部消费完后，将消息消费端切换到 `rabbitmq-cluster-b` 集群中，完成数据迁移。

验证方法

- 在 RabbitMQ Management Web UI 页面查看。



- 调用 API 查看

```
shell
curl -s -u username:password -XGET http://ip:port/api/overview
```

参数说明：

- username：使用 rabbitmq-cluster-a 集群的 RabbitMQ Management WebUI 的帐号
- password：使用 rabbitmq-cluster-a 集群的 RabbitMQ Management WebUI 的密码
- ip：使用 rabbitmq-cluster-a 集群的 RabbitMQ Management WebUI 的 IP 地址
- port：使用 rabbitmq-cluster-a 集群的 RabbitMQ Management WebUI 的端口号
- 在 Overview 视图中，消费消息数（Ready）以及未确定的消息数（Unacked）都为 0，说明消费完成。



方案二

先迁移数据，然后同时切换生产端和消费端。借助 shovel 插件完成数据迁移。

shovel 迁移数据的原理是消费 rabbitmq-cluster-a 集群中的消息，将消息生产到 rabbitmq-cluster-b 集群中，迁移后 rabbitmq-cluster-a 集群中的消息被清空，建议离线迁移，业务会出现中断。

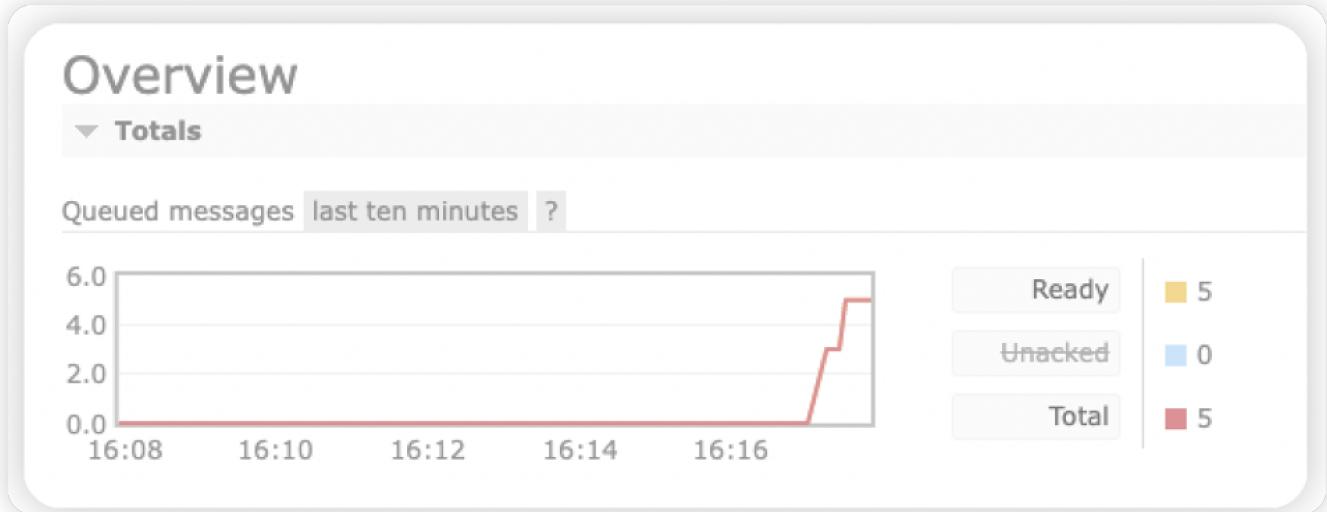
rabbitmq-cluster-a 和 rabbitmq-cluster-b 均需要开启 shovel 插件。

The screenshot shows the RabbitMQ Admin interface with the 'Admin' tab selected (indicated by a red box labeled 1). The main panel displays the 'Dynamic Shovels' configuration. The 'Source' section (highlighted with red circles 3 and 4) includes fields for 'Name', 'Source' (set to 'AMQP 0.9.1'), 'URI' (containing '源 rabbitmq 服务地址 amqp://'), 'Queue' (containing '需要迁移队列名称'), 'Prefetch count', 'Auto-delete' (set to 'Never'), and 'Destination' (set to 'AMQP 0.9.1'). The 'Destination' section (highlighted with red circle 5) includes fields for 'URI' (containing '目标 rabbitmq 服务地址 amqp://'), 'Queue' (containing '迁移后队列名称'), 'Add forwarding headers' (set to 'No'), 'Reconnect delay' (set to '5'), and 'Acknowledgement mode' (set to 'On confirm'). A red box labeled '2' covers the 'Shovel Management' tab.

参数说明：

- Name: 配置 shovel 的名称。
- Source: 指定协议类型、连接的源集群地址，源端的类型。
- Prefech count: 表示 shovel 内部缓存（从源端集群到目的集群之间的缓存部分）的消息条数。
- Auto-delete: 默认为 Never，表示不删除本集群消息，如果设置为 After initial length transferred，则在消息转移完成后删除。
- Destination: 指定协议类型，连接目标集群地址，目标端的类型。
- Add forwarding headers: 设置为 true，则会在转发的消息内添加 x-shovelled 的 header 属性。
- Reconnect delay: 指定在 Shovel link 失效的情况下，重新建立连接前需要等待的时间，单位为秒。如果设置为 0，则不会进行重连动作，即 Shovel 会在首次连接失效时停止工作。默认为 5 秒。
- Acknowledgement mode: 参考 Federation 的配置。
- no ack 表示无须任何消息确认行为；
- on publish 表示 Shovel 会把每一条消息发送到目的端之后再向源端发送消息确认；
- on confirm 表示 Shovel 会使用 publisher confirm 机制，在收到目的端的消息确认之后再向源端发送消息确认。

迁移前消息数量



配置 shovel 信息

Overview Connections Channels Exchanges Queues Admin

Dynamic Shovels

▼ Shovels

Name	Source	Destination	Reconnect Delay	Ack mode	Auto-delete
rabbitmq-cluster-a-test1	amqp091 amqp://admin:[redacted]@10.6.51.81:30363 test-1 queue	amqp091 amqp://admin:[redacted]@10.6.51.84:31786 new-test-1 queue	0	on-confirm	never

shovel 运行状态

Shovel Status

Name	Node	State	Source	Destination	Last changed	Operations
rabbitmq-cluster-a-test1	rabbit@rabbitmq-cluster-a-server-1.rabbitmq-cluster-a-nodes.mcamel-system	running	amqp091 amqp://10.6.51.81:30363 test-1 queue	amqp091 amqp://10.6.51.84:31786 new-test-1 queue	2022-07-18 8:21:06	<button>Restart</button>
dynamic						

当 shovel 状态为“running”时，表示迁移开始。等数据迁移完成后，将生产端、消费端切换至 rabbitmq-cluster-b 集群中，完成迁移过程。

迁移后消息数量

- rabbitmq-cluster-a 集群消息情况

Overview Connections Channels Exchanges Queues Admin

Cluster rabbitmq-cluster-a User admin Log out

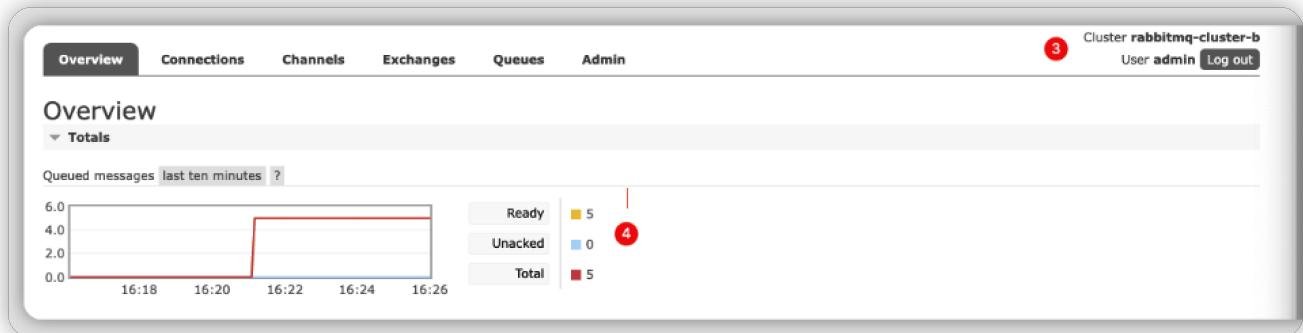
Overview

▼ Totals

Queued messages last ten minutes ? ②

Ready	0
Unacked	0
Total	0

- rabbitmq-cluster-b 集群消息情况



更新 RabbitMQ

如果想要更新或修改 RabbitMQ 的资源配置，可以按照本页说明操作。

- 在消息队列中，点击右侧的 **...** 按钮，在弹出菜单中选择 **更新实例**。

RabbitMQ 消息队列

mcamelt1 ● 运行中

部署位置 **mcamel-test/mcamel-system** 资源配额 CPU 请求量 0.2 Core / 限制量 0.2 Core
版本 3.8.30 内存 请求量 0.5 GB / 限制量 0.5 GB
副本数 三副本 磁盘 3 GB 正常/全部副本数

更新实例 (highlighted with a red box)

test000106 ● 运行中

部署位置 **mcamel-test/mcamel-system** 资源配额 CPU 请求量 0.1 Core / 限制量 0.2 Core
版本 3.8.30 内存 请求量 0.1 GB / 限制量 0.5 GB
副本数 三副本 磁盘 30 GB 正常/全部副本数

0 / 3

- 修改基本信息后，点击 **下一步**。

更新实例 instance01

1 基本信息 2 规格配置 3 服务设置

基本信息

RabbitMQ 实例名称 **instance01**

描述

描述信息不超过 256 个字符。

部署位置

集群 **kpanda-global-cluster**

命名空间 **mcamel-system**

取消 下一步

- 修改规格配置后，点击 **下一步**。

更新实例 instance01

基本信息 规格配置 服务设置

版本 • 3.8.30

副本数 • 单副本 三副本 五副本 七副本
请谨慎选择“单节点”模式，推荐使用多节点模式保证实例高可用。

CPU 配额 • 请求量 0.2 Core 限制量 0.2 Core

内存配额 • 请求量 0.5 GB 限制量 0.5 GB

存储类 storagetest9

存储容量 • 0.6 GB

取消 上一步 下一步

4. 修改服务设置后，点击确认。

更新实例 instance01

基本信息 规格配置 服务设置

访问类型 • 集群内访问(ClusterIP) 节点端口(NodePort)

端口配置	协议	端口名称	服务端口	容器端口	节点端口
	TCP	amqp	5672	5672	自动补全
	TCP	management	15672	15672	自动补全
	TCP	prometheus	15692	15692	自动补全
+ 添加					
注释	key	value			
+ 添加					

取消 上一步 确认

5. 返回消息队列，屏幕右上角将显示消息：更新实例成功。

The screenshot shows the DaoCloud interface for managing RabbitMQ instances. At the top, there's a navigation bar with the DaoCloud logo and a user account section for 'admin'. A green success message '更新实例成功' (Update instance successful) is displayed in the top right corner. The main area is titled 'RabbitMQ 消息队列' (RabbitMQ Message Queue) and shows two instances:

- instance01** (运行中): Deployment location: kpanda-global-cluster/mcamel-syst..., Version: 3.8.30, Replica count: 单副本. Resource allocation: CPU Request 0.2 Core / Limit 0.2 C..., Memory Request 0.5 GB / Limit 0.5 GB, Disk 0.6 GB. Status: 正常/全部副本数 1 / 1.
- mq123** (运行中): Deployment location: kpanda-global-cluster/mcamel-syst..., Version: 3.8.30, Replica count: 单副本. Resource allocation: CPU Request 0.1 Core / Limit 1 Core, Memory Request 0.5 GB / Limit 1 GB, Disk 2 GB. Status: 正常/全部副本数 1 / 1.

At the bottom, it says '共 2 项' (2 items total), page '1 / 1', and a dropdown for '10 项' (10 items).

查看消息队列

本节说明如何查看 RabbitMQ 消息队列。

- 在消息队列页面中，点击某个名称。

队列名称	状态	操作
mcamel1	运行中	...
test000106	运行中	...

mcamel1 队列详细信息：

参数	值	备注
部署位置	mcamel-test/mcamel-system	
版本	3.8.30	
副本数	三副本	
资源配额	CPU 请求量 0.2 Core / 限制量 0.2 Core 内存 请求量 0.5 GB / 限制量 0.5 GB 磁盘 3 GB	3 / 3 正常/全部副本数

test000106 队列详细信息：

参数	值	备注
部署位置	mcamel-test/mcamel-system	
版本	3.8.30	
副本数	三副本	
资源配额	CPU 请求量 0.1 Core / 限制量 0.2 Core 内存 请求量 0.1 GB / 限制量 0.5 GB 磁盘 30 GB	0 / 3 正常/全部副本数

- 进入消息队列概览，查看访问设置、资源配置和 Pod 列表等信息。

基本信息

参数	值	备注
实例名称	mq123	运行中
版本	3.8.30	单副本
部署位置	kpanda-global-cluster/mcamel-sys...	创建时间 2022-07-19 17:59
副本数	-	-

访问设置

参数	值
访问方式	NodePort
集群 IPv4	10.98.63.145
页面控制台用户名	admin
页面控制台地址	http://10.6.182.101:31001

资源配置

资源	限制量	请求量	备注
CPU	1 Core	0.1 C...	
内存	1 GB	0.5 GB	
磁盘	2 GB	-	

容器组列表

容器组	运行状态	容器组 IP	重启次数	CPU 使用量/限制量	内存使用量/限制量	创建时间
mq123-server-0	运行中	192.168.130.97	0	0.224 Core / 1 Core	241.455 GB / 100...	2022-07-19 17:59

10. Redis

10.1 Redis 缓存服务 Release Notes

本页列出 Redis 缓存服务的 Release Notes，便于您了解各版本的演进路径和特性变化。

v0.5.0

发布日期：2023-02-23

API

- 新增 `mcamel-redis helm-docs` 模板文件。
- 新增 `mcamel-redis` 应用商店中的 Operator 只能安装在 `mcamel-system`。
- 新增 `mcamel-redis` 支持 cloud shell。
- 新增 `mcamel-redis` 支持导航栏单独注册。
- 新增 `mcamel-redis` 支持查看日志。
- 新增 `mcamel-redis` 更新单例/集群模式 Operator 的版本。
- 新增 `mcamel-redis` 展示 common Redis 集群。
- 新增 `mcamel-redis` Operator 对接 chart-syncer。
- 修复 `mcamel-redis` 实例名太长导致自定义资源无法创建的问题。
- 修复 `mcamel-redis` 工作空间 Editor 用户无法查看实例密码。
- 修复 `mcamel-redis` 不能解析出正确的 Redis 版本号。
- 修复 `mcamel-redis` 无法修改 Port 的问题。
- 升级 `mcamel-redis` 升级离线镜像检测脚本。

文档

- 新增 日志查看操作说明，支持自定义查询、导出等功能。

v0.4.0

发布日期：2022-12-25

API

- 新增 `mcamel-redis` NodePort 端口冲突提前检测。
- 新增 `mcamel-redis` 节点亲和性配置。
- 修复 `mcamel-redis` 单例和集群设置 nodeport 无效的问题。
- 修复 `mcamel-redis` 集群模式下，从节点不可以设置为 0 的问题。

v0.2.6

发布日期：2022-11-28

API

- 修复 更新 Redis 时校验部分字段错误

- 改进 密码校验调整为 MCamel 低等密码强度
- 改进 提升哨兵模式的版本依赖, v1.1.1=>v1.2.2, 重要变更为支持 k8s 1.25+
- 新增 支持在 ARM 环境安装主备模式的 Redis 集群
- 新增 sc 扩容提示
- 新增 返回列表或者详情时的公共字段
- 新增 增加返回告警列表
- 新增 校验 Service 注释
- 修复 服务地址展示错误

v0.2.2

发布日期: 2022-10-26

API

- 新增 获取用户列表接口
- 新增 支持 arm 架构
- 新增 redis 实例全生命周期的管理
- 新增 redis 实例的监控部署
- 新增 支持 redis 哨兵, 单例和集群一键部署
- 新增 支持 ws 权限隔离
- 新增 支持在线动态扩容
- 升级 release notes 脚本

10.2 Intro

适用场景

很多大型电商网站、视频直播和游戏平台等，存在 大规模数据访问，对 数据查询效率要求高，且 数据结构简单，不涉及太多关联查询。这种场景使用 Redis，在速度上对传统磁盘数据库有很大优势，能够有效减少数据库磁盘 IO，提高数据查询效率，减轻管理维护工作量，降低数据库存储成本。Redis 对传统磁盘数据库是一个重要的补充，成为了互联网应用，尤其是支持高并发访问的互联网应用必不可少的基础服务之一。

以下举几个典型样例：

1. 电商网站 - 秒杀抢购

电商网站的商品类目、推荐系统以及秒杀抢购活动，适宜使用 Redis 缓存数据库。

例如秒杀抢购活动，并发高，对于传统关系型数据库来说访问压力大，需要较高的硬件配置（如磁盘 IO）支撑。Redis 数据库单节点 QPS 支撑能达到 10 万，轻松应对秒杀并发。实现秒杀和数据加锁的命令简单，使用 SET、GET、DEL、RPUSH 等命令即可。

2. 视频直播 - 消息弹幕

直播间的在线用户列表，礼物排行榜，弹幕消息等信息，都适合使用 Redis 中的 SortedSet 结构进行存储。

例如弹幕消息，可使用 ZREVRANGEBYSCORE 排序返回，在 Redis 5.0 中，新增了 zpopmax、zpopmin 命令，更加方便消息处理。

3. 游戏应用 - 游戏排行榜

在线游戏一般涉及排行榜实时展现，比如列出当前得分或战力最高的 10 个用户。使用 Redis 的有序集合存储用户排行榜非常合适，有序集合使用非常简单，提供多达 20 个操作集合的命令。

4. 社交 App - 返回最新评论/回复

在 web 类应用中，常有“最新评论”之类的查询，如果使用关系型数据库，往往涉及到按评论时间逆排序，随着评论越来越多，排序效率越来越低，且并发频繁。

使用 Redis 的 List（链表）来存储最新 1000 条评论，当请求的评论数在这个范围，就不需要访问磁盘数据库，直接从缓存中返回，减少数据库压力的同时，提升 App 的响应速度。

什么是 Redis 缓存服务

Redis 缓存服务是 DaoCloud 提供的一款内存数据库缓存服务，兼容了 Redis 和 Memcached 两种内存数据库引擎，为您提供开箱即用、安全可靠、弹性扩容、便捷管理的在线分布式缓存能力，满足用户高并发及数据快速访问的业务诉求。

- **开箱即用**

提供单机、高可用集群、Cluster 集群、读写分离类型的缓存实例，拥有从 128M 到 1024G 的丰富内存规格。您可以通过 UI 控制台直接创建，无需单独准备服务器资源。

所有 Redis 版本采用容器化部署，秒级完成创建。

- **安全可靠**

借助 DaoCloud 全局管理、审计日志等安全管理服务，全方位保护实例数据的存储与访问。

灵活的容灾策略，主备/集群实例从单集群内部署，到支持多集群多云部署。

- **弹性伸缩**

提供对实例内存规格的在线扩容与缩容服务，帮助您实现基于实际业务量的成本控制，达到按需使用的目标。

- **便捷管理**

可视化 Web 管理界面，在线完成实例重启、参数修改等操作。还提供 RESTful API，方便您进一步实现实例自动化管理。

[创建 Redis 实例](#)

10.3 User guide

创建 Redis 实例

接入 Redis 缓存服务后，参照以下步骤创建 Redis 实例。

- 在 Redis 缓存服务的实例列表中，点击 新建部署 按钮。

Redis 缓存服务 Default

搜索 新建实例

test-redis-1		运行中		...
部署位置	mcamel-test/mcamel-system	资源配额	CPU 请求量 0.1 Core / 限制量 1 Core	3 / 3
版本	6.2.5	内存	请求量 0.5 GB / 限制量 1 GB	正常/全部副本数
部署类型	主备模式	磁盘	1 GB	
副本数	3 副本			

test-cin		运行中		...
部署位置	mcamel-test/mcamel-system	资源配额	CPU 请求量 0.1 Core / 限制量 1 Core	1 / 1
版本	6.2.5	内存	请求量 0.5 GB / 限制量 1 GB	正常/全部副本数
部署类型	单机模式	磁盘	1 GB	
副本数	1 副本			

- 在创建 Redis 实例页面中，配置基本信息后，点击下一步。

DaoCloud admin

← 创建 Redis 实例

1 基本信息 2 规格配置 3 服务设置 4 配置确认

基本信息

Redis 实例名称 * redis01
2~63 个字符，只能包含小写字母、数字及分隔符("-")，且必须以小写字母开头、字母或数字结尾。

描述

描述信息不超过 256 个字符。

部署位置

集群 * mcamel-test

命名空间 * mcamel-system

安装环境检测 ✓ 集群状态检测正常，可继续安装

取消 下一步

- 选择部署类型、CPU、内存和存储等 规格配置 后，点击 下一步 。

基本信息

规格配置

服务设置

配置确认

版本 * 6.2.5

部署类型 * 单机模式 主备模式 集群模式

副本数 * 1

CPU 配额 * 请求量 0.1 Core 限制量 1 Core

内存配额 * 请求量 0.5 GB 限制量 1 GB

存储类 * hwameistor-storage-lvm-hdd-ha

存储容量 * 1 GB

取消 上一步 下一步

4. 设置用户名和密码等服务设置，默认采用 ClusterIP 作为访问方式。

基本信息

规格配置

服务设置

配置确认

服务设置

访问类型 * 集群内访问(ClusterIP) 节点端口(NodePort)

端口配置

服务端口 6379
容器内部服务端口，无需变更

注释 + 添加

访问设置

密码 * *****
6~18 个字符，必须同时包含英文大小写、数字、特殊符号(!@#\$%^&*-/?=-_+)

取消 上一步 下一步

5. 确认基本信息、规格配置、服务设置的信息准确无误后，点击 确认。

基本信息

Redis 实例名称: redis01
部署位置: mcamel-test/mcamel-system

规格配置

版本: 6.2.5
部署类型: 单机模式
副本数: 1 副本
CPU 配额: 请求量 0.1 Core 限制量 1 Core
内存配额: 请求量 0.5 GB 限制量 1 GB
存储类: hwameistor-storage-lvm-hdd-ha

服务设置

配置确认

取消 上一步 确认

6. 返回实例列表，屏幕将提示 创建实例成功。新创建的实例状态为 未就绪，等待片刻后将变为 运行中。

Redis 缓存服务

redis01 ● 未就绪

部署位置: mcamel-test/mcamel-system
资源配额: CPU 请求量 0.1 Core / 限制量 1 Core
内存 请求量 0.5 GB / 限制量 1 GB
磁盘 1 GB
正常/全部副本数: 0 / 1

版本: 6.2.5
部署类型: 单机模式
副本数: 1 副本

test-redis-1 ● 运行中

部署位置: mcamel-test/mcamel-system
资源配额: CPU 请求量 0.1 Core / 限制量 1 Core
内存 请求量 0.5 GB / 限制量 1 GB
磁盘 1 GB
正常/全部副本数: 3 / 3

版本: 6.2.5
部署类型: 主备模式
副本数: 3 副本

创建实例成功

删除 Redis 实例

如果想要删除一个 Redis 实例，可以执行如下操作：

- 在 Redis 实例列表中，点击右侧的 **...** 按钮，在弹出菜单中选择 **删除实例**。

The screenshot shows the Redis Cache Service interface. At the top, there's a search bar and a '新建实例' (Create New Instance) button. Below that, the 'Default' cluster is selected. Two instances are listed:

- redis01**: Status: 未就绪 (Not Ready). Configuration: 部署位置: mcamel-test/mcamel-system, 版本: 6.2.5, 部署类型: 单机模式 (Single Node Mode), 副本数: 1 副本 (1 Replica). Resource Allocation: CPU 请求量 0.1 Core / 限制量 1 Core, 内存 请求量 0.5 GB / 限制量 1 GB, 磁盘 1 GB. A modal window titled '删除实例' (Delete Instance) is open over this instance, with the '删除实例' button highlighted by a red box.
- test-redis-1**: Status: 运行中 (Running). Configuration: 部署位置: mcamel-test/mcamel-system, 版本: 6.2.5, 部署类型: 主备模式 (Master-Slave Mode), 副本数: 3 副本 (3 Replicas). Resource Allocation: CPU 请求量 0.1 Core / 限制量 1 Core, 内存 请求量 0.5 GB / 限制量 1 GB, 磁盘 1 GB. Status: 正常/全部副本数 (Normal/All Replicas).

- 在弹窗中输入该实例的名称，确认无误后，点击 **删除** 按钮。

The screenshot shows a confirmation dialog box titled '确认删除实例 redis01 吗?' (Are you sure you want to delete instance redis01?). The dialog contains the following text: '确认删除实例 redis01 吗？删除后对应的数据将会全部丢失，请谨慎操作。' (Are you sure you want to delete instance redis01? Deleting it will result in the loss of all data, please operate with caution.) and a note '请输入redis01' (Please enter redis01). The input field contains 'redis01'. At the bottom, there are two buttons: '删除' (Delete) highlighted with a red box and '取消' (Cancel).

!!! warning

删除实例后，该实例相关的所有信息也会被全部删除，请谨慎操作。

查看 Redis 日志

操作步骤

通过访问每个 Redis 的实例详情，页面可以支持查看 Redis 的日志。

- 在 Redis 实例列表中，选择想要查看的日志，点击 实例名称 进入到实例详情页面。

- 在实例的左侧菜单栏，会发现有一个日志查看的菜单栏选项。

- 点击 日志查看 即可进入到日志查看页面（Insight 日志查看）。

日志查看说明

在日志查看页面，我们可以很方便的进行日志查看，常用操作说明如下：

- 支持 自定义日志时间范围，在日志页面右上角，可以方便地切换查看日志的时间范围（可查看的日志范围以 可观测系统设置内保存的日志时长为准）
- 支持 关键字检索日志，左侧检索区域支持查看更多的日志信息
- 支持 日志量分布查看，中上区域柱状图，可以查看在时间范围内的日志数量分布
- 支持 查看日志的上下文，点击右侧 上下文 图标即可
- 支持 导出日志

The screenshot shows the DaoCloud observability platform's log search feature. On the left, there's a sidebar with various monitoring and management sections. The main area is titled "日志查询" (Log Search) and includes a search bar (2), a cluster selection dropdown (3), a histogram showing log volume over time (3), and a detailed log table (4). A context menu (5) is open over a specific log entry, showing options like "查看上下文" (View Context).

时间	日志
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.268Z","log.logger":"elasticsearch-controller","message":"Ending reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters","took":0.466213453}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.242Z","log.logger":"zen2","message":"Ensuring no voting exclusions are set","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"} 5
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:55.196Z","log.logger":"transport","message":"Skipping pod because it has no IP yet","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","namespace":"mcamel-system","pod_name":"mcamel-common-es-cluster-masters-es-masters-2"}
2023-02-27 00:48	elastic-operator-0 manager {"log.level":"info","@timestamp":"2023-02-26T16:48:54.802Z","log.logger":"elasticsearch-controller","message":"Starting reconciliation run","service.version":"2.3.0+a5aadbd","service.type":"eck","ecs.version":"1.4.0","iteration":2802,"namespace":"mcamel-system","es_name":"mcamel-common-es-cluster-masters"}

更新 Redis 实例

如果想要更新或修改 Redis 的资源配置，可以按照本页说明操作。

- 在实例列表中，点击右侧的 **...** 按钮，在弹出菜单中选择 **更新实例**。

redis01 ● 运行中

部署位置: mcamel-test/mcamel-system
版本: 6.2.5
部署类型: 单机模式
副本数: 1 副本

资源配额: CPU 请求量 0.1 Core / 限制量 1 Core
内存 请求量 0.5 GB / 限制量 1 GB
磁盘 1 GB

正常/全部副本数

更新实例

删除实例

test-redis-1 ● 运行中

部署位置: mcamel-test/mcamel-system
版本: 6.2.5
部署类型: 主备模式
副本数: 3 副本

资源配额: CPU 请求量 0.1 Core / 限制量 1 Core
内存 请求量 0.5 GB / 限制量 1 GB
磁盘 1 GB

3 / 3 正常/全部副本数

- 基本信息: 只能修改描述。然后点击下一步。

更新实例 redis01

1 2 3

基本信息 规格配置 服务设置

基本信息

Redis 实例名称: redis01

描述:

描述信息不超过 256 个字符。

部署位置

集群: mcamel-test
命名空间: mcamel-system

下一步

- 修改规格配置后点击 **下一步**。

更新实例 redis01

基本信息

版本 * 6.2.5

部署类型 * 单机模式

副本数 * 1

CPU 配额 * 请求量 0.1 Core 限制量 1 Core

内存配额 * 请求量 0.5 GB 限制量 1 GB

存储类 hwameistor-storage-lvm-hdd-ha

存储容量 * 1

取消 上一步 下一步

4. 修改服务设置后点击确认。

更新实例 redis01

基本信息

规格配置

服务设置

访问类型 * 集群内访问(ClusterIP) 节点端口(NodePort)

端口配置

服务端口 6379
容器内部服务端口，无需变更

注释 + 添加

取消 上一步 确认

部署类型 主备模式 磁盘 1 GB

副本数 3 副本