可观测性

概览

概览

告警

集群通知

前提条件

安装

配置 Email 服务

概览

监控

介绍

安装

概述

架构

核心概念

操作指南

监控

告警

通知

监控面板

Perses 监控面板

实用指南

分布式追踪

关于 **Distributed Tracing**

Alauda 版 OpenTelemetry

关于 **Alauda Build of OpenTelemetry**

日志

关于 **Logging Service**

事件

介绍

使用限制

Events

操作流程

事件概览

检查

介绍

使用限制

架构

Inspection

Component Health Status

操作指南

概览

Observability 模块是 ACP 平台的核心功能，提供面向云原生应用的全面监控和可观测性能力。

该模块集成了四个关键的可观测性支柱：

- **Synthetic monitoring (probe)** 用于主动的端点测试
- **Centralized logging** 实现统一的日志管理和分析
- **Real-time monitoring** 用于指标收集和告警
- **Distributed tracing** 实现跨微服务的端到端请求追踪

通过将这些能力整合到单一平台中，帮助组织实现对应用性能的全面可视化，快速诊断问题，确保系统可靠性，并优化整个技术栈中的用户体验。

告警

集群通知

前提条件

安装

配置 Email 服务

概览

集群通知

Cluster Notification 为工作负载集群提供独立的通知能力，支持多种通道和灵活配置，以确保在分布式环境中稳定、高效地传递消息。

目录

前提条件
安装
下载插件包
使用 violet 将插件推送到平台
通过 Web 控制台安装
通过 YAML 安装
配置 Email 服务
Email Server 配置
字段说明

前提条件

- ACP version: >= v4.2
- Plugin version: >= v1.0

安装

下载插件包

登录到软件包门户，并下载最新的 灵雀云容器平台 Cluster Notification 插件包。

使用 **violet** 将插件推送到平台

使用 violet 将 灵雀云容器平台 Cluster Notification 插件推送到平台：

```
violet push aiops-notification-business.ALL.vx.x.x.tgz \
  --platform-address https://<platform-address>/ \
  --platform-token "<platform_token>"
```

对于非交互式使用，建议优先使用 `--platform-token`，而不是用户名和密码。Identity Provider (IdP) 用户必须使用平台令牌。有关所有身份验证选项，请参见 Upload Packages。

通过 **Web** 控制台安装

1. 导航到 管理员 > **Marketplace** > 集群插件
2. 搜索 **Alauda Container Platform Cluster Notification**，然后单击查看其详情
3. 单击 安装
4. 选择目标集群并确认安装

通过 **YAML** 安装

```yaml
apiVersion: cluster.alauda.io/v1alpha1
kind: ModuleInfo
metadata:
  annotations:
    cpaas.io/display-name: aiops-notification-business
    cpaas.io/module-name: '{"en": "Alauda Container Platform Cluster Noti
fication",
      "zh": "Alauda Container Platform Cluster Notification"}'
  labels:
    cpaas.io/cluster-name: <cluster-name>              # Target cluster
name
    cpaas.io/module-name: aiops-notification-business
    cpaas.io/module-type: plugin
    cpaas.io/product: Platform-Center
  name: aiops-notification-<cluster-name>            # Resource name
spec:
  version: v1.0.0                                    # Plugin version
```

配置 **Email** 服务

插件安装后，默认不包含任何通知配置。用户需要手动创建相关的 YAML 配置。该配置需要创建在与插件相同的集群中。

Email Server 配置

```yaml
apiVersion: v1
kind: Secret
metadata:
  annotations:
  labels:
    cpaas.io/notification.server.category: Email
    cpaas.io/notification.server.type: Email
  name: platform-email-server
  namespace: cpaas-system
type: NotificationServer
data:
  displayNameEn: RW1haWw=
  displayNameZh: 6YKu5Lu2
  from: dGVzdEBleGFtcGxlLmNvbQo=
  host: bWFpbC5leGFtcGxlLmNvbQo=
  insecureSkipVerify: dHJ1ZQ==
  password: MTIzNDU2Cg==
  port: NDY1
  sslEnabled: dHJ1ZQ==
  username: dGVzdEBleGFtcGxlLmNvbQo=
```

字段说明

字段	说明
displayNameEn	英文显示名称
displayNameZh	中文显示名称
from	发件人邮箱地址（例如，test@example.com）
host	Email 服务地址（例如，mail.example.com）
port	Email 服务端口
username	发件人用户名
password	发件人密码
sslEnabled	启用 SSL（true/false）

字段	说明
insecureSkipVerify	跳过证书验证（默认值：true）

> 注意：secret data 中的所有字段值都必须使用 base64 编码。用户只需更新 data 部分中的信息。

概览

使用 Alerting 为平台和工作负载可观测性配置通知投递和与告警相关的操作。

需求	继续执行
为工作负载集群配置集群通知投递。	集群通知
管理监控告警。	管理告警
管理监控通知通道。	管理通知

在安装后阶段，在团队开始依赖平台和工作负载告警之前，先配置告警通知投递。使用此 Alerting 区域查看详细的告警和通知流程。

监控

介绍

介绍

安装

安装

概述

开始之前

安装 Perses Dashboard 组件

ACP 基础监控

ACP 高级监控

架构

监控模块架构

整体架构说明

监控组件选型指南

重要说明

监控组件容量

假设与方法论

监控系统
告警系统
通知系统

组件列表
架构对比
功能对比
安装方案建议

Prometheus
VictoriaMetrics

核心概念

核心概念

监控
告警
通知
监控面板
Perses 监控面板

操作指南

指标管理

查看平台组件暴露的指标
查看 Prometheus / VictoriaMetrics 存储的
查看平台定义的所有内置指标
集成外部指标

告警管理

功能概述
主要功能
功能优势
通过 UI 创建告警策略
通过 CLI 创建资源告警
通过 CLI 创建事件告警

通知管理

功能概述
主要功能
通知服务器
接收者组
通知模板
通知规则

监控面板管理

功能概述

管理监控面板

管理图表

通过 CLI 创建监控仪表盘

常用函数和变量

Perses 监控

Fleet 监控

概述

前提条件

探针管理

功能概述

黑盒监控

黑盒告警

自定义 BlackboxExporter 监控模块

通过 CLI 创建黑盒监控项和告警

参考信息

实用指南

Prometheus 监控数据的备份与恢

功能概述

使用场景

前提条件

操作步骤

操作结果

了解更多

后续操作

VictoriaMetrics 监控数据备份与恢

功能概述

使用场景

前提条件

操作步骤

操作结果

了解更多

后续操作

从自定义命名

功能概述

使用场景

前提条件

操作步骤

操作结果

了解更多

后续操作

规划用于监控的基础设施节点

概述

配置 Fleet Monitoring

概述

将 MonitorD

概述

支持的配置方法
配置放置前
在控制台中配置放置
在 YAML 中配置放置
故障排除
了解更多
后续操作

开始之前
推送 Fleet Monitoring Operator 软件包
在 Global 集群上启用 Fleet Monitoring
将集群连接到 Fleet Monitoring
升级 Fleet Monitoring
配置采集间隔
配置自定义指标和记录规则
验证数据新鲜度
故障排除
了解更多

前提条件
迁移监控面板
转换规则
导入 Grafana JS
迁移后
相关操作

介绍

Monitoring 模块是 ACP 平台可观测性套件的核心组件，为平台管理员和运维团队提供全面的监控和告警能力。

该模块提供四项关键的监控功能：

- 指标收集，用于实时采集集群、节点、应用和容器的性能数据
- 仪表盘，用于直观展示和分析系统健康状况及性能趋势
- 告警，通过可定制的规则和阈值实现问题的主动检测
- 通知，及时将告警信息传递给运维人员

通过与 Prometheus 和 VictoriaMetrics 等开源组件的集成，Monitoring 模块帮助组织维护系统可靠性，防止停机，降低运维成本，并确保整个基础设施的最佳性能。

安装

目录

概述

开始之前

Alauda OS 上的 LocalVolume 存储路径

安装 Perses Dashboard 组件

ACP 基础监控

从控制台安装

使用 YAML 安装

将 Prometheus 工作负载放置在基础设施节点上

访问已安装的组件

ACP 高级监控

前提条件

从控制台安装

使用 YAML 安装

将 VictoriaMetrics 工作负载放置在基础设施节点上

访问已安装的组件

概述

监控组件为可观测性模块中的监控、告警、检查和健康检查功能提供基础设施。在集群中安装 ACP Basic Monitoring 或 ACP Advanced Monitoring。

要决定安装哪个插件，请先查看监控组件选择指南，然后根据集群规模、存储规划和运维要求选择最合适的选项。

Perses 监控面板需要额外的监控面板组件。在 **Operations Center** > **Monitor** > **Dashboards (Perses)** 中查看、创建或迁移监控面板时，请安装这些组件。

开始之前

> **INFO**
>
> 某些监控组件会大量消耗资源。建议通过插件配置将其放置在 infra 节点上。Prometheus 和 VictoriaMetrics 都支持插件级别的 `nodeSelector` 和 `tolerations` 设置。如果您正在评估产品且尚未配置 infra 节点，可以将这些设置留空，以便组件在通用节点上运行。
>
> 有关规划 infra 节点的指导，请参阅Infra 节点。

安装监控组件前，请确保满足以下条件：

- 参考监控组件选择指南选择合适的监控组件。
- 在业务集群中安装时，请确保 `global` 集群可以访问业务集群的 11780 端口。
- 如果需要使用存储类或持久卷存储监控数据，请提前在 **Storage** 部分创建相应资源。
- 在安装前决定由哪个集群存储监控数据。`global` 集群会安装完整的组件集，并且仅监控自身。在多集群平台上，请将聚合业务集群指标的 VictoriaMetrics Center 安装在专用业务集群中，而不是安装在 `global` 集群中；并将每个业务集群安装为仅运行 agent 的模式，指向该集群，从而使管理集群的可用性与应用监控负载相互独立。
- 如果 `global` 集群以灾难恢复方式部署，业务集群监控不得指向主集群或备用 `global` 集群。灾难恢复仅同步 `global` 集群的 etcd 数据，并排除 `cpaas-system` 等命名空间，因此监控数据不会被复制，并且在故障转移后不可用。每个 `global` 集群仅监控自身。有关同步范围，请参阅Global 集群灾难恢复。
- 如果集群节点运行 Alauda OS 且计划使用 `LocalVolume` 存储，请勿保留默认存储路径。请参阅Alauda OS 上的 LocalVolume 存储路径。

Alauda OS 上的 LocalVolume 存储路径

两个监控插件默认将 `LocalVolume` 存储路径设置为 `/cpaas/monitoring`，该路径属于运行非 Alauda OS 的节点布局。在节点运行 Alauda OS 的集群中，节点具有只读根文件系统，节点本地的平台数据保存在提供商机器配置中声明的持久磁盘上的 `/var/cpaas` 下。在这些已声明磁盘之外写入的任何内容，在平台替换节点时都不会保留。其他平台组件已遵循此约定；例如，在节点运行 Alauda OS 的集群中，etcd 备份使用 `/var/cpaas/backup`。请参阅etcd 备份与恢复。

在此类集群中使用 `LocalVolume` 存储安装任一监控插件时：

- 将存储路径设置为 `/var/cpaas/monitoring`，而不是默认路径 `/cpaas/monitoring`。在控制台中设置 **Storage Path**；在 YAML 中设置 `spec.config.storage.path`。
- 在 `spec.config.storage.nodes` 中列出的每个节点上使用持久磁盘支持该路径，并根据配置的存储容量确定磁盘大小。平台会在这些节点的此路径下创建数据目录，因此该路径必须可写。请在管理该集群的 provider 的持久磁盘字段中声明该磁盘，以确保其在节点替换后仍然保留：

Provider	持久磁盘字段
Huawei DCS	`DCSIpHostnamePool.spec.pool[].persistentDisk`
VMware vSphere	`VSphereMachineConfigPool.spec.configs[].persistentDisks[]`
Huawei Cloud Stack	`HCSMachineConfigPool.spec.configs[].persistentDisks[]`
Bare Metal	`MachineInventory.spec.storage.volumes[]`

  有关各 provider 的声明操作步骤，请参阅在 Alauda OS 上创建集群。在 Bare Metal 上，磁盘已经连接到物理主机，并且是在长期存在的 `MachineInventory` 上声明，而不是从 IaaS 平台请求。因此请遵循管理裸金属主机上的数据磁盘中的说明，并在 inventory 仍处于未分配状态时声明该卷。
- 不要在未进行规划的情况下让监控数据与平台 `/var/cpaas` 磁盘共享。监控数据会随配置的规模和保留期增长，因此应为 `/var/cpaas/monitoring` 配置专用磁盘，或调整现有 `/var/cpaas` 磁盘的大小，使其同时满足监控容量需求。

安装插件时设置此路径。对于 **ACP Basic Monitoring**，安装后无法修改存储配置。

安装 Perses Dashboard 组件

Perses 监控仪表板依赖以下组件：

组件	安装位置	用途
Alauda Container Platform Dashboard Essentials	仅限 `global` 集群	提供 Perses 仪表板使用的、支持 RBAC 的指标查询服务。
Alauda Container Platform Dashboard	`global` 集群运行唯一的 Perses server 实例。业务集群会根据需要安装 operator，以提供 PersesDashboard CRD，但不会创建 Perses 实例。	提供 Perses 仪表板后端、渲染、编辑和 PersesDashboard 资源管理功能。

Perses server 和 `observe-api` 仅在 `global` 集群中创建。在控制台中选择业务集群时，控制台仍会将 Perses server 请求路由到 `global` 集群。所选业务集群决定使用哪个集群的指标和 PersesDashboard 资源。`observe-api` 会查询跨集群指标并实施 RBAC。

请按以下顺序安装组件：

1. 在 `global` 集群中安装 **Alauda Container Platform Dashboard Essentials**。
2. 在 `global` 集群中创建一个 `ObserveAPI` 实例。你可以使用组件或 OperatorHub 条目的 **Create** 操作，也可以应用如下 CR：

   ```yaml
   apiVersion: observe.alauda.io/v1alpha1
   kind: ObserveAPI
   metadata:
     name: observe-api
     namespace: cpaas-system
   spec: {}
   ```

   验证 `observe-api` Deployment 正在运行。
3. 在 `global` 集群中安装 **Alauda Container Platform Dashboard**。
4. 在 `global` 集群中创建唯一的 `Perses` 实例。你可以使用组件或 OperatorHub 条目的 **Create** 操作，也可以应用如下 CR：

```yaml
apiVersion: perses.dev/v1alpha2
kind: Perses
metadata:
  name: cpaas-perses
  namespace: cpaas-system
  annotations:
    perses.dev/ingress.enabled: "true"
spec:
  replicas: 2
  containerPort: 8080
  storage:
    emptyDir: {}
  config:
    frontend:
      disable: true
```

验证 `cpaas-perses` Deployment 正在运行。

5. 对于用户需要查看或创建 Perses 仪表板的每个业务集群，请安装 **Alauda Container Platform Dashboard**，以便在该集群中提供 PersesDashboard CRD。不要在业务集群中创建 `Perses` 实例。
6. 前往 **Operations Center** > **Monitor** > **Dashboards (Perses)**，并确认所选集群中的仪表板列表可用。

如果所选集群尚未安装 **Alauda Container Platform Dashboard**，Perses 仪表板页面将显示安装指南，而不是仪表板列表。

Perses 仪表板组件会重复使用现有的平台指标存储，例如 VictoriaMetrics 或 Prometheus。它们不会引入单独的指标存储或租户。

安装 operator 只会安装控制器和 CRD。必须先在 `global` 集群中创建相应的 `ObserveAPI` 和 `Perses` 实例，然后才会启动实际的 `observe-api` 和 Perses server 工作负载。

ACP 基础监控

从控制台安装

1. 前往 **App Store Management** > **Cluster Plugins**，然后选择目标集群。

2. 找到 **ACP Basic Monitoring** 插件，然后点击 **Install**。

3. 配置以下参数：

   控制台会突出显示最常用的安装选项。有关可配置字段的详细信息，请参阅本节中的 YAML 参考。

参数	描述
Scale Configuration	支持三种配置：**Small Scale**、**Medium Scale** 和 **Large Scale**： - 默认值根据平台建议的负载测试值设置 - 你可以根据实际集群规模选择或自定义配额 - 默认值会随平台版本更新；对于固定配置，建议使用自定义设置
Storage Type	- **LocalVolume**：本地存储，数据存储在指定节点上 - **StorageClass**：使用存储类自动生成持久卷 - **PV**：使用现有持久卷 注意：安装后无法修改存储配置
Storage Path	当 **Storage Type** 为 `LocalVolume` 时显示。用于存储 Prometheus 数据的主机目录；必须使用绝对路径。默认值：`/cpaas/monitoring`。在 Alauda OS 上，请设置为 `/var/cpaas/monitoring`；请参阅 Alauda OS 上的 LocalVolume 存储路径。
Replica Count	设置监控组件 Pod 的数量 注意：Prometheus 仅支持单节点安装
Advanced Configuration	用于配置插件级调度参数的可折叠部分。
Node Selectors	在 **Advanced Configuration** 中显示。为 Prometheus 插件工作负载配置插件级节点选择器规则。
Node Tolerations	在 **Advanced Configuration** 中显示。为 Prometheus 插件工作负载配置插件级容忍规则。
Parameter Configuration	可根据需要调整监控组件的数据参数

4. 点击 **Install** 完成安装。

使用 YAML 安装

检查可用版本

通过检查 `global` 集群中的 ModulePlugin 和 ModuleConfig 资源，确保插件已发布：

```
# kubectl get moduleplugin | grep prometheus
prometheus                          30h
# kubectl get moduleconfig | grep prometheus
prometheus-v4.1.0                   30h
```

这表示集群中存在 ModulePlugin `prometheus`，并且已发布版本 `v4.1.0`。

创建 **ModuleInfo**

创建 ModuleInfo 资源，在不配置任何参数的情况下安装插件：

```yaml
kind: ModuleInfo
apiVersion: cluster.alauda.io/v1alpha1
metadata:
  name: global-prometheus
  labels:
    cpaas.io/cluster-name: global
    cpaas.io/module-name: prometheus
    cpaas.io/module-type: plugin
spec:
  version: v4.1.0
  config:
    storage:
      type: LocalVolume
      capacity: 40
      nodes:
        - xxx.xxx.xxx.xx
      path: /cpaas/monitoring # On Alauda OS use /var/cpaas/monitoring
      storageClass: ""
      pvSelectorK: ""
      pvSelectorV: ""
    replicas: 1
    components:
      nodeSelector:
        - key: kubernetes.io/os
          value: linux
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/infra
          operator: Exists
      prometheus:
        retention: 7
        scrapeInterval: 60
        scrapeTimeout: 45
        resources: null
      nodeExporter:
        port: 9100
        resources: null
      alertmanager:
        resources: null
      kubeStateExporter:
        resources: null
      prometheusAdapter:
        resources: null
```

```
      thanosQuery:
        resources: null
    size: Small
```

资源设置示例（Prometheus）：

```
spec:
  config:
    components:
      prometheus:
        resources:
          limits:
            cpu: 2000m
            memory: 2000Mi
          requests:
            cpu: 1000m
            memory: 1000Mi
```

有关更多详细信息，请参阅 监控组件容量规划。

YAML 字段参考（Prometheus）：

字段路径	描述
`metadata.labels.cpaas.io/cluster-name`	安装插件的目标集群名称。
`metadata.labels.cpaas.io/module-name`	必须为 `prometheus`。
`metadata.labels.cpaas.io/module-type`	必须为 `plugin`。
`metadata.name`	ModuleInfo 名称（例如 `<cluster>-prometheus`）。
`spec.version`	要安装的插件版本。
`spec.config.storage.type`	存储类型：`LocalVolume`、`StorageClass` 或 `PV`。
`spec.config.storage.capacity`	Prometheus 的存储大小（Gi）。建议至少为 30 Gi。

字段路径	描述
`spec.config.storage.nodes`	使用 `storage.type=LocalVolume` 时的节点列表。最多支持 1 个节点。
`spec.config.storage.path`	使用 `storage.type=LocalVolume` 时的 LocalVolume 基础路径。默认值：`/cpaas/monitoring`。在 Alauda OS 上，请设置为 `/var/cpaas/monitoring`；请参阅 Alauda OS 上的 LocalVolume 存储路径。
`spec.config.storage.storageClass`	使用 `storage.type=StorageClass` 时的 StorageClass 名称。
`spec.config.storage.pvSelectorK`	使用 `storage.type=PV` 时的 PV 选择器键。
`spec.config.storage.pvSelectorV`	使用 `storage.type=PV` 时的 PV 选择器值。
`spec.config.replicas`	副本数量；仅适用于 `StorageClass` / `PV` 类型。
`spec.config.components.nodeSelector`	可选。Prometheus 插件工作负载的插件级节点选择器规则。
`spec.config.components.tolerations`	可选。Prometheus 插件工作负载的插件级容忍规则。
`spec.config.components.prometheus.retention`	数据保留天数。
`spec.config.components.prometheus.scrapeInterval`	抓取间隔（秒）；适用于不带 `interval` 的 ServiceMonitor。
`spec.config.components.prometheus.scrapeTimeout`	抓取超时时间（秒）；必须小于 `scrapeInterval`。

字段路径	描述
`spec.config.components.prometheus.resources`	Prometheus 的资源设置。
`spec.config.components.nodeExporter.port`	Node Exporter 端口（默认 9100）。
`spec.config.components.nodeExporter.resources`	Node Exporter 的资源设置。
`spec.config.components.alertmanager.resources`	Alertmanager 的资源设置。
`spec.config.components.kubeStateExporter.resources`	Kube State Exporter 的资源设置。
`spec.config.components.prometheusAdapter.resources`	Prometheus Adapter 的资源设置。
`spec.config.components.thanosQuery.resources`	Thanos Query 的资源设置。
`spec.config.size`	监控规模：`Small`、`Medium` 或 `Large`。

验证安装

由于 ModuleInfo 名称会在创建后发生变化，请通过标签查找资源，以检查插件状态和版本：

```
kubectl get moduleinfo -l cpaas.io/module-name=prometheus
NAME                                             CLUSTER         MODULE
DISPLAY_NAME   STATUS    TARGET_VERSION   CURRENT_VERSION   NEW_VERSION
global-e671599464a5b1717732c5ba36079795          global          promethe
us     prometheus      Running   v4.1.0           v4.1.0            v4.1.0
```

字段说明：

- `NAME`：ModuleInfo 资源名称
- `CLUSTER`：安装插件的集群
- `MODULE`：插件名称
- `DISPLAY_NAME`：插件的显示名称

- `STATUS`：安装状态；`Running` 表示已成功安装并运行
- `TARGET_VERSION`：计划安装的版本
- `CURRENT_VERSION`：安装前的版本
- `NEW_VERSION`：可安装的最新版本

将 Prometheus 工作负载放置在基础设施节点上

如果希望 Prometheus 插件工作负载在专用基础设施节点上运行，请在安装或升级期间配置插件级调度规则，而不要在安装后修补生成的工作负载。

- 在控制台中，使用 **Advanced Configuration** 设置 **Node Selectors** 和 **Node Tolerations**。
- 在 YAML 中设置 `spec.config.components.nodeSelector` 和 `spec.config.components.tolerations`。

示例：

```yaml
config:
  components:
    nodeSelector:
      - key: kubernetes.io/os
        value: linux
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        operator: Exists
```

在应用这些调度规则之前，请确保基础设施节点规划与存储放置相兼容。有关规划注意事项，请参阅 How To 中的 Monitoring 指南，包括 **Planning Infra Nodes for Monitoring**。

访问已安装的组件

安装完成后，可以通过以下地址访问组件（将 `<>` 替换为实际值）：

组件	访问地址
Thanos	`<platform_access_address>/clusters/<cluster>/prometheus`

组件	访问地址
Prometheus	`<platform_access_address>/clusters/<cluster>/prometheus-0`
Alertmanager	`<platform_access_address>/clusters/<cluster>/alertmanager`

ACP 高级监控

前提条件

- 如果仅安装 VictoriaMetrics agent，请确保已在其他集群中安装 VictoriaMetrics Center。

从控制台安装

1. 导航至 应用商店管理 > 集群插件，然后选择目标集群。
2. 找到 **ACP** 高级监控 插件，然后点击 安装。
3. 配置以下参数：
   控制台会突出显示最常用的安装选项。有关可配置字段的详细信息，请参阅本节中的 YAML 参考。

参数	描述
规模配置	支持三种配置：小规模、中规模和大规模： - 默认值根据平台推荐的负载测试值设置 - 可以根据实际集群规模选择或自定义配额 - 默认值会随平台版本更新；如需固定配置，建议使用自定义设置
仅安装 **Agent**	- 关闭：安装完整的 VictoriaMetrics 组件套件 - 开启：仅安装 VMAgent 采集组件，该组件依赖 VictoriaMetrics Center
VictoriaMetrics Center	选择已安装完整 VictoriaMetrics 组件的集群

参数	描述
存储类型	- **LocalVolume**：使用本地存储，数据存储在指定节点上 - **StorageClass**：使用存储类自动生成持久卷 - **PV**：使用现有持久卷
存储路径	当 存储类型 为 `LocalVolume` 时显示。指定监控数据的基础存储路径。默认值：`/cpaas/monitoring`。在 Alauda OS 上设置为 `/var/cpaas/monitoring`；请参阅 Alauda OS 上的 LocalVolume 存储路径。
副本数	当 存储类型 为 `StorageClass` 或 `PV` 时显示。设置监控组件 Pod 的数量。当 存储类型 为 `LocalVolume` 时，所选节点数决定 VMStorage 副本数。
高级配置	用于插件级调度参数的可折叠部分。
节点选择器	在 高级配置 中显示。为 VictoriaMetrics 插件工作负载配置插件级节点选择器规则。
节点容忍度	在 高级配置 中显示。为 VictoriaMetrics 插件工作负载配置插件级容忍度规则。
参数配置	可以调整监控组件的数据参数 注意：数据在删除前可能会暂时超过保留期限

4. 点击 安装 完成安装。

使用 YAML 安装

检查可用版本

通过检查 `global` 集群中的 ModulePlugin 和 ModuleConfig 资源，确保插件已发布：

```
# kubectl get moduleplugin | grep victoriametrics
victoriametrics                          30h
# kubectl get moduleconfig | grep victoriametrics
victoriametrics-v4.1.0                   30h
```

这表示 ModulePlugin `victoriametrics` 存在于集群中，并且已发布版本 `v4.1.0`。

创建 **ModuleInfo**

创建 ModuleInfo 资源，在不配置任何参数的情况下安装插件：

```yaml
kind: ModuleInfo
apiVersion: cluster.alauda.io/v1alpha1
metadata:
  name: business-1-victoriametrics
  labels:
    cpaas.io/cluster-name: business-1
    cpaas.io/module-name: victoriametrics
    cpaas.io/module-type: plugin
spec:
  version: v4.1.0
  config:
    storage:
      type: LocalVolume
      capacity: 40
      nodes:
        - xxx.xxx.xxx.xx
      path: /cpaas/monitoring # On Alauda OS use /var/cpaas/monitoring
      storageClass: ""
      pvSelectorK: ""
      pvSelectorV: ""
    replicas: 1
    agentOnly: false
    agentReplicas: 1
    crossClusterDependency:
      victoriametrics: ""
    components:
      nodeSelector:
        - key: kubernetes.io/os
          value: linux
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/infra
          operator: Exists
      nodeExporter:
        port: 9100
        resources: null
      vmstorage:
        retention: 7
        resources: null
      kubeStateExporter:
        resources: null
      vmalert:
        resources: null
```

```
      prometheusAdapter:
        resources: null
      vmagent:
        scrapeInterval: 60
        scrapeTimeout: 45
        resources: null
      vminsert:
        resources: null
      alertmanager:
        resources: null
      vmselect:
        resources: null
    size: Small
```

资源设置示例（vmagent）：

```
spec:
  config:
    components:
      vmagent:
        resources:
          limits:
            cpu: 2000m
            memory: 2000Mi
          requests:
            cpu: 1000m
            memory: 1000Mi
```

有关详细信息，请参阅 监控组件容量规划。

YAML 字段参考（VictoriaMetrics）：

字段路径	描述
`metadata.labels.cpaas.io/cluster-name`	安装插件的目标集群名称。
`metadata.labels.cpaas.io/module-name`	必须为 `victoriametrics`。
`metadata.labels.cpaas.io/module-type`	必须为 `plugin`。
`metadata.name`	ModuleInfo 名称（例如 `<cluster>-`

字段路径	描述
	`victoriametrics`）。
`spec.version`	要安装的插件版本。
`spec.config.storage.type`	存储类型：`LocalVolume`、`StorageClass` 或 `PV`。
`spec.config.storage.capacity`	VictoriaMetrics 的存储大小（Gi）。建议最小为 30 Gi。
`spec.config.storage.nodes`	使用 `storage.type=LocalVolume` 时的节点列表。可以选择一个或多个节点。
`spec.config.storage.path`	使用 `storage.type=LocalVolume` 时的基础 LocalVolume 路径。默认值：`/cpaas/monitoring`。在 Alauda OS 上设置为 `/var/cpaas/monitoring`；请参阅 Alauda OS 上的 LocalVolume 存储路径。
`spec.config.storage.storageClass`	使用 `storage.type=StorageClass` 时的 StorageClass 名称。
`spec.config.storage.pvSelectorK`	使用 `storage.type=PV` 时的 PV 选择器键。
`spec.config.storage.pvSelectorV`	使用 `storage.type=PV` 时的 PV 选择器值。
`spec.config.replicas`	副本数；适用于 `StorageClass` 和 `PV` 类型。
`spec.config.agentOnly`	是否仅安装 vmagent，而不是完整的 VictoriaMetrics 组件集。

字段路径	描述
`spec.config.agentReplicas`	仅 Agent 模式启用时 vmagent 的副本数。
`spec.config.crossClusterDependency.victoriametrics`	仅 Agent 模式启用时提供 VictoriaMetrics Center 的目标集群。
`spec.config.components.nodeSelector`	可选。VictoriaMetrics 插件工作负载的插件级节点选择器规则。
`spec.config.components.tolerations`	可选。VictoriaMetrics 插件工作负载的插件级容忍度规则。
`spec.config.components.vmstorage.retention`	vmstorage 的数据保留天数。
`spec.config.components.vmagent.scrapeInterval`	抓取间隔（秒）；适用于不含 `interval` 的 ServiceMonitor。
`spec.config.components.vmagent.scrapeTimeout`	抓取超时时间（秒）；必须小于 `scrapeInterval`。
`spec.config.components.vmstorage.resources`	vmstorage 的资源设置。
`spec.config.components.vmalert.resources`	vmalert 的资源设置。
`spec.config.components.nodeExporter.port`	Node Exporter 端口（默认 9100）。
`spec.config.components.nodeExporter.resources`	Node Exporter 的资源设置。
`spec.config.components.alertmanager.resources`	Alertmanager 的资源设置。
`spec.config.components.kubeStateExporter.resources`	Kube State Exporter 的资源设置。
`spec.config.components.prometheusAdapter.resources`	Prometheus Adapter 的资源设置（用于 HPA/自定义指标）。
`spec.config.components.vmagent.resources`	vmagent 的资源设置。

字段路径	描述
`spec.config.components.vminsert.resources`	vminsert 的资源设置。
`spec.config.components.vmselect.resources`	vmselect 的资源设置。
`spec.config.size`	监控规模：`Small`、`Medium` 或 `Large`。

验证安装

由于 ModuleInfo 名称会在创建时发生变化，请通过标签定位资源，以检查插件状态和版本：

```
kubectl get moduleinfo -l cpaas.io/module-name=victoriametrics
NAME                                             CLUSTER         MODULE
DISPLAY_NAME     STATUS    TARGET_VERSION   CURRENT_VERSION   NEW_VERSION
global-e671599464a5b1717732c5ba36079795          global          victoria
metrics   victoriametrics  Running   v4.1.0           v4.1.0            v
4.1.0
```

字段说明：

- `NAME`：ModuleInfo 资源名称
- `CLUSTER`：安装插件的集群
- `MODULE`：插件名称
- `DISPLAY_NAME`：插件显示名称
- `STATUS`：安装状态；`Running` 表示已成功安装并运行
- `TARGET_VERSION`：计划安装的版本
- `CURRENT_VERSION`：安装前的版本
- `NEW_VERSION`：可安装的最新版本

将 VictoriaMetrics 工作负载放置在基础设施节点上

如果希望 VictoriaMetrics 插件工作负载运行在专用基础设施节点上，请在安装或升级期间配置插件级调度规则，而不是在安装后修改生成的工作负载。

- 在控制台中，使用 高级配置 设置 节点选择器 和 节点容忍度。
- 在 YAML 中设置 `spec.config.components.nodeSelector` 和 `spec.config.components.tolerations`。

示例：

```yaml
config:
  components:
    nodeSelector:
      - key: kubernetes.io/os
        value: linux
    tolerations:
      - effect: NoSchedule
        key: node-role.kubernetes.io/infra
        operator: Exists
```

在应用这些调度规则之前，请确保基础设施节点规划与存储放置相兼容。有关规划注意事项，请参阅 How To 中的 Monitoring 指南，包括 **Planning Infra Nodes for Monitoring**。

访问已安装的组件

安装完成后，可以通过以下地址访问组件（将 `<>` 替换为实际值）：

组件	访问地址
VictoriaMetrics UI	`<platform_access_address>/clusters/<cluster>/vmselect-ui/vmui/?#/metrics`

> **INFO**
>
> 如果启用了 仅安装 **Agent**，集群不会在本地部署 `vmselect` 组件，因此无法在该集群中访问 VictoriaMetrics UI 地址。

架构

监控模块架构

- 整体架构说明
- 监控系统
- 告警系统
- 通知系统

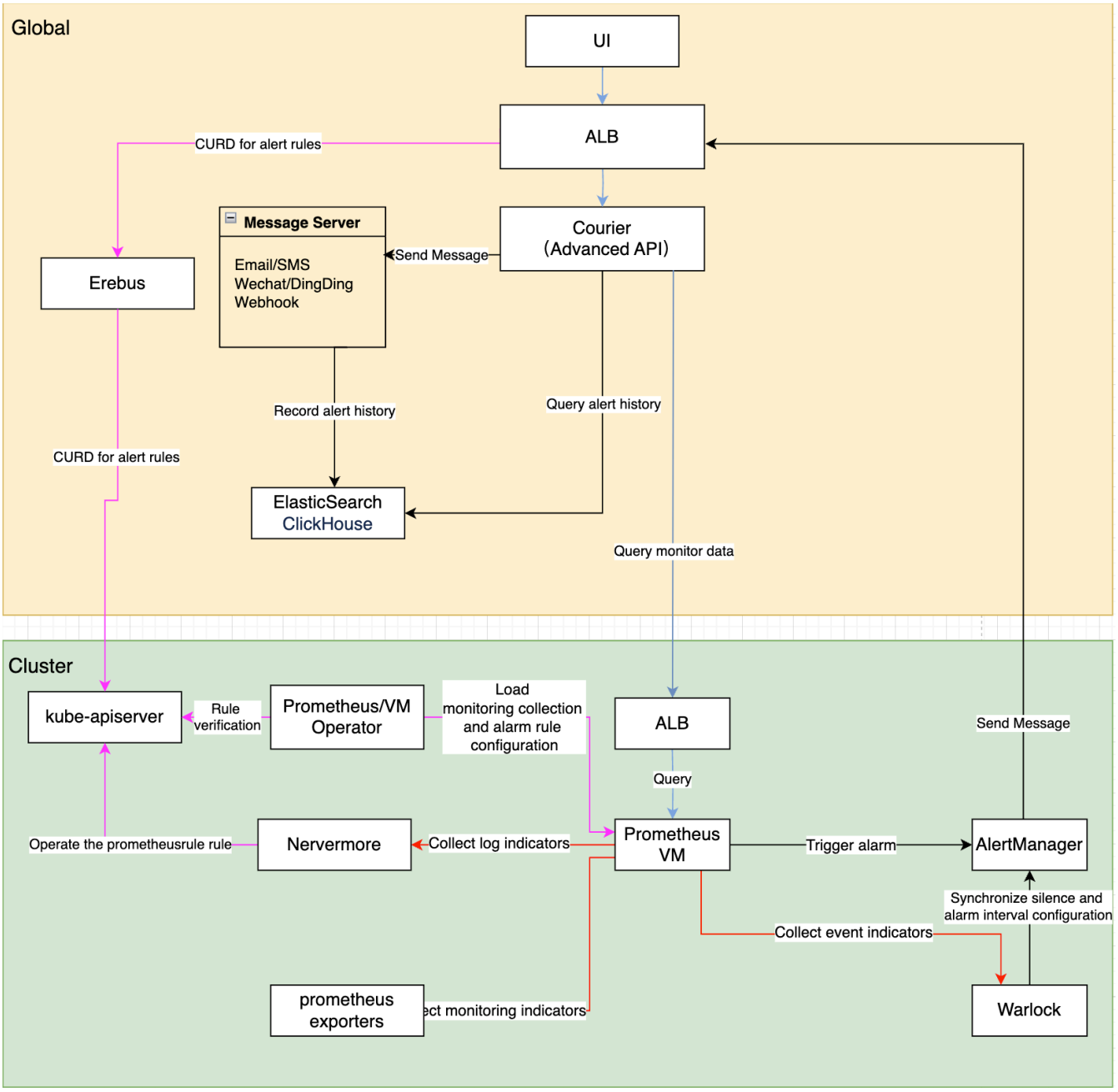
监控组件选型指南

- 重要说明
- 组件列表
- 架构对比
- 功能对比
- 安装方案建议

监控组件容量

- 假设与方法论
- Prometheus
- VictoriaMetrics

监控模块架构


Global
UI
ALB
CURD for alert rules
Message Server
Email/SMS
Wechat/DingDing
Webhook
Send Message
Courier
(Advanced API)
Erebus
Record alert history
Query alert history
CURD for alert rules
ElasticSearch
ClickHouse
Query monitor data
Cluster
kube-apiserver
Rule verification
Prometheus/VM Operator
Load monitoring collection and alarm rule configuration
ALB
Send Message
Query
Operate the prometheusrule rule
Nervermore
Collect log indicators
Prometheus VM
Trigger alarm
AlertManager
Synchronize silence and alarm interval configuration
Collect event indicators
prometheus exporters
ect monitoring indicators
Warlock

目录

整体架构说明
监控系统
数据采集与存储
数据查询与可视化
告警系统
告警规则管理
告警处理工作流
实时告警状态
通知系统
通知配置管理
通知服务器管理

整体架构说明

监控系统由以下核心功能模块组成：

1. 监控系统
   - 数据采集与存储：从多个来源采集并持久化监控指标
   - 数据查询与可视化：为监控数据提供灵活的查询和可视化能力
2. 告警系统
   - 告警规则管理：配置和管理告警策略
   - 告警触发与通知：评估告警规则并发送通知
   - 实时告警状态：提供系统当前告警状态的实时视图
3. 通知系统
   - 通知配置：管理通知模板、联系人组和策略
   - 通知服务器：管理各种通知渠道的配置

监控系统

数据采集与存储

1. Prometheus/VictoriaMetrics Operator 职责：
   - 加载并验证监控采集配置
   - 加载并验证告警规则配置
   - 将配置同步到 Prometheus/VictoriaMetrics 实例
2. 监控数据来源：
   - Nevermore：生成与日志相关的指标
   - Warlock：生成与事件相关的指标
   - Prometheus/VictoriaMetrics：通过 ServiceMonitor 发现并采集各种 exporter 的指标

数据查询与可视化

1. 监控数据查询流程：
   - 浏览器发起查询请求（路径：`/platform/monitoring.alauda.io/v1beta1`）
   - ALB 将请求转发到 Courier 组件
   - Courier API 处理查询：
     - 内置指标：通过指标接口获取 PromQL 并执行查询
     - 自定义指标：将 PromQL 直接转发到监控组件
   - 监控面板获取数据并进行展示
2. 监控面板管理流程：
   - 用户访问 `global` 集群 ALB（路径：`/kubernetes/cluster_name/apis/ait.alauda.io/v1alpha2/MonitorDashboard`）
   - ALB 将请求转发到 Erebus 组件
   - Erebus 将请求路由到目标监控集群

- Warlock 组件负责：
  - 验证监控面板配置的合法性
  - 管理 MonitorDashboard CR 资源

3. Perses 面板查询流程：
   - 用户在控制台中打开 **Operations Center** > **Monitor** > **Dashboards (Perses)**。
   - 控制台将面板请求发送到由 **Alauda Container Platform Dashboard** 部署在 `global` 集群中的 Perses 服务器。
   - Perses 服务器使用固定的 `acp-observe` 数据源，通过 `observe-api` 查询指标。
   - `observe-api` 由 **Alauda Container Platform Dashboard Essentials** 部署在 `global` 集群中，会在将查询转发到平台指标存储之前检查用户的集群、命名空间和项目权限。选定的业务集群是指标查询目标，但不会运行本地 Perses 服务器或 `observe-api`。
   - 平台指标存储（例如 VictoriaMetrics 或 Prometheus）将查询结果返回到面板。

```
User -> Console Perses dashboard -> Perses server -> observe-api
     -> Platform metric storage -> Dashboard
```

Perses 面板将 `observe-api`、Perses 服务器和 perses-operator 引入查询与渲染路径。Perses 服务器和 `observe-api` 仅运行在 `global` 集群中。业务集群会根据需要安装 Perses operator，以提供 PersesDashboard CRD 并存储其自身的面板资源。现有的 MonitorDashboard 功能仍然可用，并继续使用现有的 MonitorDashboard 管理路径。

告警系统

告警规则管理

告警规则的配置流程：

1. 用户访问 `global` 集群 ALB（路径：`/kubernetes/cluster_name/apis/monitoring.coreos.com/v1/prometheusrules`）
2. 请求经过 ALB -> Erebus -> 目标集群 kube-apiserver

3. 各组件的职责：

- Prometheus/VictoriaMetrics Operator：
  - 验证告警规则的合法性
  - 管理 PrometheusRule CR
- Nevermore：监听并处理日志告警指标
- Warlock：监听并处理事件告警指标

告警处理工作流

1. 告警评估：

- PrometheusRule/VMRule 定义告警规则
- Prometheus/VictoriaMetrics 定期评估规则

2. 告警通知：

- 告警触发后发送到 Alertmanager
- Alertmanager -> ALB -> Courier API
- Courier API 负责分发通知

3. 告警存储：

- 告警历史记录存储在 ElasticSearch/ClickHouse 中

实时告警状态

1. 状态采集：

- `global` 集群中的 Courier 生成指标：
  - cpaas_active_alerts：当前活动告警
  - cpaas_active_silences：当前静默配置
- Global Prometheus 每 15 秒采集一次

2. 状态显示：

- 前端通过 Courier API 查询并显示实时状态

通知系统

通知配置管理

通知模板、通知联系人组和通知策略的管理流程如下：

1. 用户通过浏览器访问 `global` 集群的标准 API
   - 访问路径：`/apis/ait.alauda.io/v1beta1/namespaces/cpaas-system`
2. 管理相关资源：
   - 通知模板：apiVersion: "ait.alauda.io/v1beta1", kind: "NotificationTemplate"
   - 通知联系人组：apiVersion: "ait.alauda.io/v1beta1", kind: "NotificationGroup"
   - 通知策略：apiVersion: "ait.alauda.io/v1beta1", kind: "Notification"
3. Courier 负责：
   - 验证通知模板的合法性
   - 验证通知联系人组的合法性
   - 验证通知策略的合法性

通知服务器管理

1. 用户通过浏览器访问 `global` 集群的 ALB
   - 访问路径：`/kubernetes/global/api/v1/namespaces/cpaas-system/secrets`
2. 管理并提交通知服务器配置
   - 资源名称：platform-email-server
3. Courier 负责：
   - 验证通知服务器配置的合法性

监控组件选型指南

在安装集群监控时，平台提供了两种监控组件供您选择：VictoriaMetrics 和 Prometheus。本文将详细介绍这两种组件的特点及适用场景，帮助您做出最合适的选择。

目录

- 重要说明
- 组件列表
  - Prometheus 相关组件
  - VictoriaMetrics 相关组件
- 架构对比
  - Prometheus 架构
  - VictoriaMetrics 架构
- 功能对比
- 安装方案建议
  - 监控安装架构概览
    - Prometheus 安装方式
    - VictoriaMetrics 安装方式
  - 选型建议
    - 适合使用 VictoriaMetrics 的场景
    - 适合使用 Prometheus 的场景

重要说明

- 安装集群监控组件时，只能选择 VictoriaMetrics 或 Prometheus 其中之一。
- 从 3.18 版本开始，VictoriaMetrics 已升级为 Beta 状态，满足生产环境使用条件。
- VictoriaMetrics 适用于有高可用性要求和多集群监控的场景。
- Prometheus 适用于单集群监控场景，尤其适合规模较小的情况。

组件列表

Prometheus 相关组件

组件名称	功能描述
Prometheus Server	核心服务器，负责采集、存储和查询监控数据
Exporters	监控数据采集组件，通过 HTTP 接口暴露监控指标
AlertManager	告警管理中心，处理告警规则和通知
PushGateway	支持监控数据的推送模式，用于特殊网络环境中的数据传输

VictoriaMetrics 相关组件

组件名称	功能描述
VMStorage	监控数据存储引擎
VMInsert	数据写入组件，负责数据分发和存储
VMSelect	查询服务组件，提供数据查询能力
VMAlert	告警规则评估和处理组件
VMAgent	监控指标采集组件

架构对比

Prometheus 架构

Prometheus 是一个成熟的开源监控系统，也是 CNCF 继 Kubernetes 之后的第二个毕业项目。它具有以下特点：

- 强大的数据采集能力。
- 灵活的查询语言 PromQL。
- 完善的生态系统。
- 支持千节点规模的集群监控。

VictoriaMetrics 架构

VictoriaMetrics 是新一代高性能时序数据库和监控解决方案，具有以下优势：

- 更高的数据压缩比。
- 更低的资源消耗。
- 原生支持集群高可用。
- 更简单的运维管理。

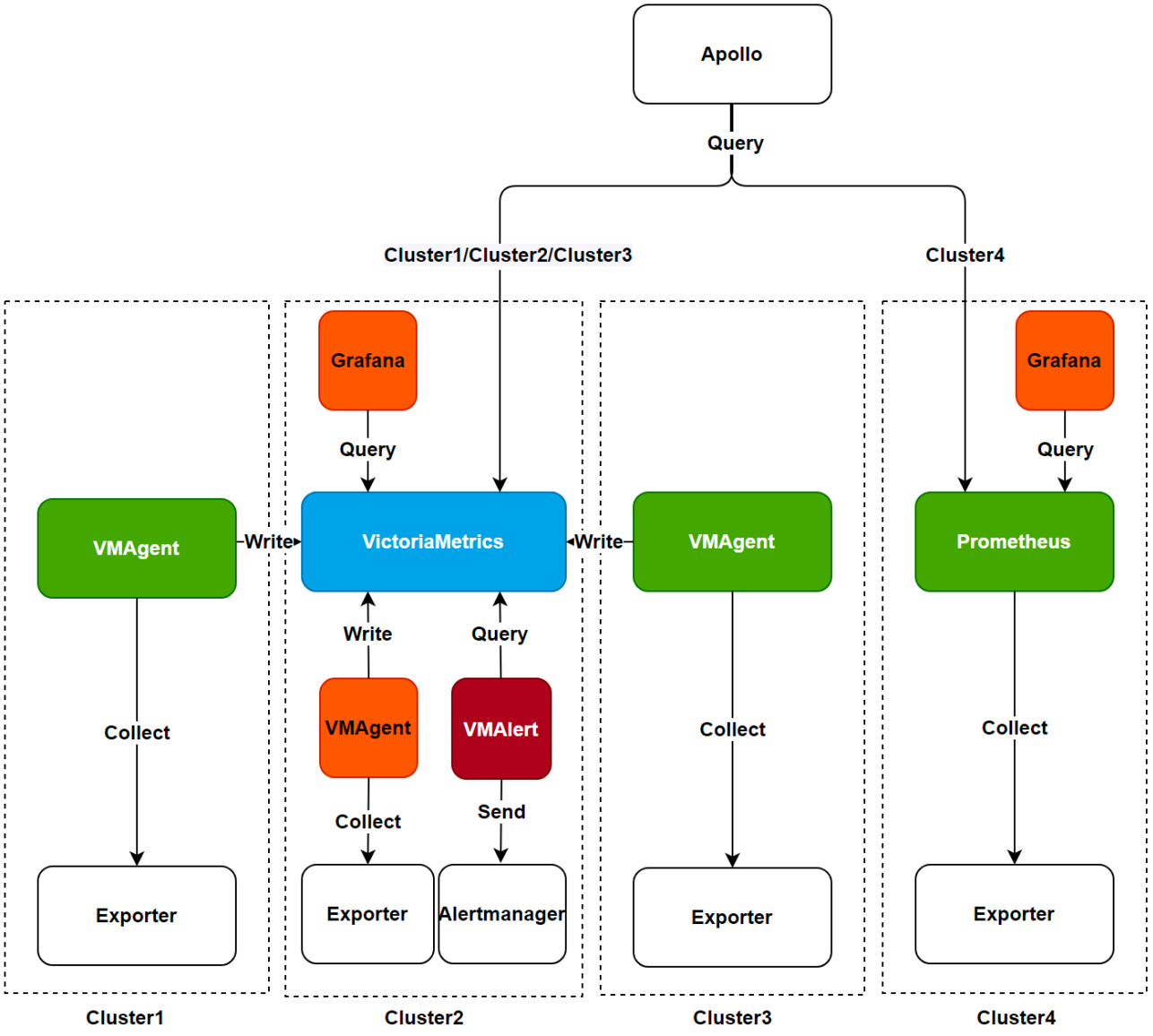
功能对比

功能	**Prometheus**	**VictoriaMetrics**	说明
高可用安装	❌	✅	VictoriaMetrics 支持真正的集群高可用，数据一致性更好
单节点安装	✅	✅	两者均支持单节点安装模式
长期数据存储	需要远程存储	原生支持	VictoriaMetrics 更适合长期数据存储
资源效率	较高	更优	VictoriaMetrics 具有更好的资源利用率

功能	Prometheus	VictoriaMetrics	说明
Community 支持	非常成熟	快速发展	Prometheus 拥有更大的 Community 生态系统

安装方案建议

监控安装架构概览



上图展示了平台支持的监控组件的安装架构和数据流。平台提供以下两种安装方式供选择：

注意：更换监控组件时，请确保现有组件已完全卸载，且监控数据不支持跨组件迁移。

Prometheus 安装方式

此方式对应上图中 **cluster4** 的架构：

- 使用 Prometheus 组件采集和处理监控数据。
- 通过监控图表查询和展示数据。
- 适用于单集群场景。

VictoriaMetrics 安装方式

VictoriaMetrics 支持以下两种安装模式：

1. 单集群安装模式

   - 对应上图中 **cluster2** 的架构。
   - 所有 VictoriaMetrics 组件安装在同一集群中。
   - 使用 VMAgent 采集数据并写入 VictoriaMetrics。
   - VMAlert 负责告警规则评估。
   - 通过监控图表查询和展示数据。 提示：当数据规模低于每秒 100 万时，建议使用此模式。

2. 多集群安装模式

   - 对应上图中 **cluster1/cluster2/cluster3** 的架构。
   - 在业务集群中安装 VMAgent 作为数据采集智能体。
   - VMAgent 将数据写入中心监控集群中的 VictoriaMetrics。
   - 中心监控集群是专用的业务集群，而不是 `global` 集群。 `global` 集群安装自己的完整组件集，仅监控自身，从而使管理集群可用性与应用监控负载保持独立。
   - 当 `global` 集群部署了容灾时，业务集群中的 VMAgent 不得指向主 `global` 集群或备 `global` 集群，因为监控数据不会在它们之间复制。
   - 支持跨多个集群的统一监控管理。 提示：在安装 VMAgent 之前，请确保监控集群中已安装 VictoriaMetrics 服务。

选型建议

适合使用 **VictoriaMetrics** 的场景

- 高性能和可扩展性需求：适用于处理高吞吐量数据和长期存储的监控场景。
- 成本效益考量：需要优化存储和计算资源成本。
- 高可用性要求：需要对监控组件提供高可用性保障。
- 多集群管理：需要统一管理多个集群的监控数据。

适合使用 **Prometheus** 的场景

- 小规模单集群：监控规模较小，无高可用性要求。
- 现有 **Prometheus** 用户：已拥有完整的 Prometheus 监控系统。
- 简单稳定性要求：追求简单可靠的监控方案。
- 深度生态集成：与 Prometheus 生态系统紧密集成，迁移成本较高。

监控组件容量规划

监控组件负责存储从平台中一个或多个集群收集的指标数据。因此，您需要提前评估监控规模，并根据本文档中的指南规划监控组件所需的资源。

目录

假设与方法论

Prometheus

小规模 — 10 个 worker 节点，500 个双容器 Pod

中等规模 — 50 个 worker 节点，2000 个双容器 Pod

大规模 — 500 个 worker 节点，10000 个双容器 Pod

VictoriaMetrics

小规模 — 10 个 worker 节点，500 个双容器 Pod

中等规模 — 50 个 worker 节点，2000 个双容器 Pod

大规模 — 500 个 worker 节点，10000 个双容器 Pod

假设与方法论

- 本文档中的数据来自受控实验室性能报告，旨在作为生产规划的容量基线。
- 磁盘示例的保留时间为7天；其他保留目标请按比例调整。
- 存储基线符合上述警告（SSD，约6000 IOPS，约250MB/s 读写，独立挂载）。

- 测试工作负载涵盖了典型的监控页面，如“acp ns overview page”和“platform region detail page”。

Prometheus

以下是按规模划分的 Prometheus 及相关组件（Thanos Query、Thanos Sidecar 等）的容量建议。

小规模 — 10 个 worker 节点，500 个双容器 Pod

- 指标摄取速率：约2800 样本/秒

Component	Container	Replicas	CPU Limit	Memory Limit	D ap
courier-api	courier	2	2C	4Gi	-
kube-prometheus-thanos-query	thanos-query	1	1C	1Gi	-
prometheus-kube-prometheus-0	prometheus	1	2C	8Gi	2(

中等规模 — 50 个 worker 节点，2000 个双容器 Pod

- 指标摄取速率：约7294 样本/秒

Component	Container	Replicas	CPU Limit	Memory Limit	D ap
courier-api	courier	2	4C	4Gi	-
kube-prometheus-thanos-query	thanos-query	1	2.5C	8Gi	-

Component	Container	Replicas	CPU Limit	Memory Limit	D ap
prometheus-kube-prometheus-0	prometheus	1	4C	8Gi	4(

大规模 — 500 个 worker 节点，10000 个双容器 Pod

- 指标摄取速率：约 41575 样本/秒

Component	Container	Replicas	CPU Limit	Memory Limit	D ap
courier-api	courier	2	6C	4Gi	-
kube-prometheus-thanos-query	thanos-query	1	2C	6Gi	-
prometheus-kube-prometheus-0	prometheus	1	8C	20Gi	1(

VictoriaMetrics

以下是按规模划分的 VictoriaMetrics 组件容量建议。

小规模 — 10 个 worker 节点，500 个双容器 Pod

- 指标摄取速率：约 3274 样本/秒

Component	Container	Replicas	CPU Limit	Memory Limit	D ap
courier-api	courier	1	2C	4Gi	-

Component	Container	Replicas	CPU Limit	Memory Limit	D ap
vmselect-cluster	proxy	1	1C	200Mi	-
vmselect	vmselect	1	500m	1Gi	-
vmstorage-cluster	vmstorage	1	500m	2Gi	3(

中等规模 — 50 个 worker 节点，2000 个双容器 Pod

- 指标摄取速率：约6940 样本/秒

Component	Container	Replicas	CPU Limit	Memory Limit	D ap
courier-api	courier	2	4C	4Gi	-
vmselect-cluster	proxy	1	1C	200Mi	-
vmselect	vmselect	1	2C	2Gi	-
vmstorage-cluster	vmstorage	1	2C	2Gi	1(

大规模 — 500 个 worker 节点，10000 个双容器 Pod

- 指标摄取速率：约34300 样本/秒

Component	Container	Replicas	CPU Limit	Memory Limit	D ap
courier-api	courier	2	6C	4Gi	-

Component	Container	Replicas	CPU Limit	Memory Limit	D a
vmselect-cluster	proxy	1	2C	200Mi	-
vmselect	vmselect	1	5C	3Gi	-
vmstorage-cluster	vmstorage	1	2C	6Gi	3(

核心概念

目录

- 监控
  - 指标
  - PromQL
  - 内置指标
  - Exporter
  - ServiceMonitor
- 告警
  - 告警规则
  - 告警策略
- 通知
  - 通知策略
  - 通知模板
- 监控面板
  - 监控面板
  - 图表
  - 数据源
  - 变量
- Perses 监控面板
  - PersesDashboard
  - 图表

图表组

变量

数据源

文件夹和标签

监控

指标

指标用于定量描述系统的运行状态，每个指标由四个基本元素组成：

- 指标名称：用于标识被监控对象，例如 `cpu_usage`
- 指标值：具体的测量值，例如 `85.5`
- 时间戳：记录测量时间
- 标签：用于多维数据分类，例如 `{pod="nginx-1", namespace="default"}`

PromQL

PromQL 是 Prometheus 的查询语言，用于查询和聚合监控系统中的指标数据。

内置指标

平台基于长期运维经验预置了一系列常用监控指标。您可以在配置告警规则或创建监控面板时直接使用这些指标，无需额外配置。

Exporter

Exporter 是用于采集监控数据的组件，主要职责包括：

- 从目标系统采集原始监控数据
- 将数据转换为标准的时间序列指标格式

- 通过 HTTP 接口提供可查询的指标数据

ServiceMonitor

ServiceMonitor 用于以声明式方式管理监控配置，主要定义：

- 监控目标的选择条件
- 指标采集接口的配置
- 采集任务的执行参数（间隔、超时等）

告警

告警规则

告警规则定义触发告警的具体条件：

- 告警表达式：使用 PromQL 语句描述触发告警的条件
- 告警阈值：明确的触发边界值
- 持续时间：条件必须持续满足的时长
- 告警级别：区分告警严重程度（例如 P0/P1/P2）

告警策略

告警策略将多个告警规则组织在一起，便于统一配置：

- 告警目标：规则的目标范围
- 通知方式：发送告警的渠道
- 发送间隔：重复发送告警通知的时间间隔

通知

通知策略

通知策略管理发送告警消息的规则：

- 接收人：告警通知的目标用户
- 通知渠道：支持的消息发送方式
- 通知模板：消息内容格式的定义

通知模板

通知模板用于自定义告警消息的展示格式：

- 标题模板：告警消息标题的格式
- 内容模板：告警详情的组织方式
- 变量替换：支持动态数据填充

监控面板

监控面板

监控面板是多个相关图表的集合，用于提供系统状态的整体视图。它支持灵活的布局安排，并且可以按行或按列组织图表。

图表

图表是监控数据的可视化表示，支持多种展示类型。

数据源

监控数据源的配置。目前仅支持将当前集群的监控组件作为数据源，暂不支持自定义数据源。

变量

变量用于充当值的占位符，可用于指标查询。通过监控面板顶部的变量选择器，您可以动态调整查询条件，使图表内容实时更新。

Perses 监控面板

PersesDashboard

PersesDashboard 是 Alauda Container Platform 4.4 中引入的监控面板资源。它基于开源 Perses 渲染引擎，由 **Operations Center** > **Monitor** > **Dashboards (Perses)** 使用。

PersesDashboard 与现有的 MonitorDashboard 资源并存。MonitorDashboard 使用现有的 Grafana 风格监控面板模型，而 PersesDashboard 使用 Perses 模型。当您希望在 Perses 监控面板入口中渲染现有 MonitorDashboard 资源时，可以将其迁移为 PersesDashboard 资源。

图表

图表是在 Perses 监控面板中的单个可视化单元，例如时序图、柱状图、仪表盘或表格。

图表组

图表组用于在监控面板中组织多个图表。图表组可用于构建监控面板结构，并对相关图表进行折叠或展开。

变量

变量是监控面板参数，用于动态过滤图表查询。Perses 监控面板支持列表变量和文本变量。

数据源

Perses 监控面板的指标数据源固定为平台指标数据源，对用户透明。查看或编辑 Perses 监控面板时，您无需选择数据源。

文件夹和标签

文件夹和标签用于组织和搜索 Perses 监控面板。

操作指南

指标管理

查看平台组件暴露的指标

查看 Prometheus / VictoriaMetrics 存储的

查看平台定义的所有内置指标

集成外部指标

告警管理

功能概述

主要功能

功能优势

通过 UI 创建告警策略

通过 CLI 创建资源告警

通过 CLI 创建事件告警

通知管理

功能概述

主要功能

通知服务器

接收者组

通知模板

通知规则

通知

监控面板管理

功能概述

管理监控面板

管理图表

通过 CLI 创建监控仪表盘

常用函数和变量

Perses 监控

Fleet 监控

概述

前提条件

探针管理

功能概述

黑盒监控

黑盒告警

自定义 BlackboxExporter 监控模块

通过 CLI 创建黑盒监控项和告警

参考信息

指标管理

平台的监控系统基于 Prometheus / VictoriaMetrics 收集的指标。本文档将指导您如何管理这些指标。

目录

查看平台组件暴露的指标

查看 Prometheus / VictoriaMetrics 存储的所有指标

前提条件

操作步骤

查看平台定义的所有内置指标

前提条件

操作步骤

集成外部指标

前提条件

操作步骤

查看平台组件暴露的指标

平台内集群组件的监控方式是通过 `ServiceMonitor` 抽取暴露的指标。平台中的指标通过 `/metrics` 端点公开，您可以使用以下示例命令查看平台中某个组件暴露的指标：

```
curl -s http://<Component IP>:<Component metrics port>/metrics | grep 'TY
PE\|HELP'
```

示例输出：

```
# HELP controller_runtime_active_workers Number of currently used workers
per controller
# TYPE controller_runtime_active_workers gauge
# HELP controller_runtime_max_concurrent_reconciles Maximum number of con
current reconciles per controller
# TYPE controller_runtime_max_concurrent_reconciles gauge
# HELP controller_runtime_reconcile_errors_total Total number of reconcil
iation errors per controller
# TYPE controller_runtime_reconcile_errors_total counter
# HELP controller_runtime_reconcile_time_seconds Length of time per recon
ciliation per controller
```

查看 **Prometheus / VictoriaMetrics** 存储的所有指标

您可以查看集群中可用的指标列表，帮助您基于这些指标编写所需的 PromQL。

前提条件

1. 您已获取用户 Token
2. 您已获取平台地址

操作步骤

> 使用 `curl` 命令运行以下命令获取指标列表：
>
> ```
> curl -k -X 'GET' -H 'Authorization: Bearer <Your token>' 'https://<Yo
> ur platform access address>/v2/metrics/<Your cluster name>/prometheu
> s/label/__name__/values'
> ```
>
> 示例输出：

```
{
  "status": "success",
  "data": [
    "ALERTS",
  "ALERTS_FOR_STATE",
  "advanced_search_cached_resources_count",
  "alb_error",
  "alertmanager_alerts",
  "alertmanager_alerts_invalid_total",
  "alertmanager_alerts_received_total",
  "alertmanager_cluster_enabled"]
}
```


查看平台定义的所有内置指标

为了简化用户使用，平台内置了大量常用指标。您在配置告警或监控面板时可以直接使用这些指标，无需自行定义。以下将介绍如何查看这些指标。

前提条件

1. 您已获取用户 Token
2. 您已获取平台地址

操作步骤

使用 `curl` 命令运行以下命令获取指标列表：

```
curl -k -X 'GET' -H 'Authorization: Bearer <Your token>' 'https://<Your platform access address>/v2/metrics/<Your cluster name>/indicators'
```

示例输出：

```
[
  {
  "alertEnabled": true, (1)
  "annotations": {
   "cn": "计算组件中容器的 CPU 利用率",
   "descriptionEN": "Cpu utilization for pods in workload",
   "descriptionZH": "计算组件中容器的 CPU 利用率",
   "displayNameEN": "CPU utilization of the pods",
   "displayNameZH": "计算组件中容器的 CPU 利用率",
   "en": "Cpu utilization for pods in workload",
   "features": "SupportDashboard", (2)
   "summaryEN": "CPU usage rate {{.externalLabels.comparison}}{{.exte
rnalLabels.threshold}} of Pod ({{.labels.pod}})",
   "summaryZH": "Pod ({{.labels.pod}}) 的 CPU 使用率 {{.externalLabel
s.comparison}}{{.externalLabels.threshold}}"
  },
  "displayName": "计算组件中容器的 CPU 利用率",
  "kind": "workload",
  "multipleEnabled": true,  (3)
  "name": "workload.pod.cpu.utilization",
  "query": "avg by (kind,name,namespace,pod) (avg by (kind,name,names
pace,pod,container)(cpaas_advanced_container_cpu_usage_seconds_total_
irate5m{kind=~\"{{.kind}}\",name=~\"{{.name}}\",namespace=~\"{{.names
pace}}\",container!=\"\",container!=\"POD\"}) / avg by (kind,name,nam
espace,pod,container)(cpaas_advanced_kube_pod_container_resource_limi
ts{kind=~\"{{.kind}}\",name=~\"{{.name}}\",namespace=~\"{{.namespac
e}}\",resource=\"cpu\"}))", (4)
  "summary": "Pod ({{.labels.pod}}) 的 CPU 使用率 {{.externalLabels.co
mparison}}{{.externalLabels.threshold}}",
  "type": "metric",
  "unit": "%",
  "legend": "{{.namespace}}/{{.pod}}",
  "variables": [ (5)
   "namespace",
   "name",
   "kind"
  ]
 }
]
```

(1) 此指标是否支持用于配置告警

2. 此指标是否支持用于监控面板
3. 此指标是否支持用于配置多资源告警
4. 指标定义的 PromQL 语句
5. 指标 PromQL 语句中可用的变量

集成外部指标

除了平台内置指标外，您还可以通过 `ServiceMonitor` 或 `PodMonitor` 集成您的应用或第三方应用暴露的指标。本节以以 Pod 形式安装在同一集群中的 Elasticsearch Exporter 为例进行说明。

前提条件

您已安装应用并通过指定接口暴露指标。本文档假设您的应用安装在 `cpaas-system` 命名空间，并暴露了 `http://<elasticsearch-exporter-ip>:9200/_prometheus/metrics` 端点。

操作步骤

1. 创建 Service/Endpoint 以供 Exporter 暴露指标

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    chart: elasticsearch
    service_name: cpaas-elasticsearch
  name: cpaas-elasticsearch
  namespace: cpaas-system
spec:
  clusterIP: 10.105.125.99
  ports:
  - name: cpaas-elasticsearch
    port: 9200
    protocol: TCP
    targetPort: 9200
  selector:
    service_name: cpaas-elasticsearch
  sessionAffinity: None
  type: ClusterIP
```

2. 创建 `ServiceMonitor` 对象描述应用暴露的指标：

```yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    app: cpaas-monitor
    chart: cpaas-monitor
    heritage: Helm
    prometheus: kube-prometheus (1)
    release: cpaas-monitor
  name: cpaas-elasticsearch-Exporter
  namespace: cpaas-system (2)
spec:
  jobLabel: service_name (3)
  namespaceSelector: (4)
    any: true
  selector: (5)
    matchExpressions:
      - key: service_name
        operator: Exists
  endpoints:
    - port: cpaas-elasticsearch (6)
      path: /_prometheus/metrics (7)
      interval: 60s (8)
      honorLabels: true
      basicAuth: (9)
        password:
          key: ES_PASSWORD
          name: acp-config-secret
        username:
          key: ES_USER
          name: acp-config-secret
```

(1) ServiceMonitor 应同步到哪个 Prometheus；operator 会根据 Prometheus CR 的 serviceMonitorSelector 配置监听对应的 ServiceMonitor 资源。如果 ServiceMonitor 的标签不匹配 Prometheus CR 的 serviceMonitorSelector 配置，则该 ServiceMonitor 不会被 operator 监控。

(2) operator 会根据 Prometheus CR 的 serviceMonitorNamespaceSelector 配置监听哪些命名空间的 ServiceMonitor；如果 ServiceMonitor 不在 Prometheus CR 的 serviceMonitorNamespaceSelector 中，则该 ServiceMonitor 不会被 operator 监控。

③ Prometheus 收集的指标会添加一个 job 标签，值为对应 jobLabel 的 service 标签值。

④ ServiceMonitor 根据 namespaceSelector 配置匹配对应的 Service。

⑤ ServiceMonitor 根据 selector 配置匹配 Service。

⑥ ServiceMonitor 根据 port 配置匹配 Service 的端口。

⑦ 访问 Exporter 的路径，默认为 /metrics。

⑧ Prometheus 抓取 Exporter 指标的间隔。

⑨ 如果访问 Exporter 路径需要认证，则需添加认证信息；也支持 bearer token、tls 认证等方式。

3. 检查 ServiceMonitor 是否被 Prometheus 监控
   访问监控组件的 UI，检查是否存在 job `cpaas-elasticsearch-exporter`。

   - Prometheus UI 地址：`https://<Your platform access address>/clusters/<Cluster name>/prometheus-0/targets`
   - VictoriaMetrics UI 地址：`https://<Your platform access address>/clusters/<Cluster name>/vmselect-ui/vmui/?#/metrics`

告警管理

目录

功能概述
主要功能
功能优势
通过 UI 创建告警策略
前提条件
操作流程
选择告警类型
配置告警规则
其他配置
其他说明
通过 CLI 创建资源告警
前提条件
操作流程
通过 CLI 创建事件告警
前提条件
操作流程
通过告警模板创建告警策略
前提条件
操作流程
创建告警模板

使用告警模板创建告警策略
设置告警静默
通过 UI 设置
通过 CLI 设置
配置告警规则建议

功能概述

平台的告警管理功能旨在帮助用户全面监控并及时发现系统异常。通过利用预置的系统告警和灵活的自定义告警能力，结合标准化的告警模板和分层管理机制，为运维人员提供完整的告警解决方案。

无论是平台管理员还是业务人员，都可以在各自权限范围内便捷地配置和管理告警策略，实现对平台资源的有效监控。

主要功能

- 内置系统告警策略：基于常见故障诊断思路，针对 `global` 集群和工作负载集群预设丰富的告警规则。
- 自定义告警规则：支持基于多种数据源创建告警规则，包括预置监控指标、自定义监控指标、黑盒监控项、平台日志数据和平台事件数据。
- 告警模板管理：支持创建和管理标准化告警模板，快速应用于类似资源。
- 告警通知集成：支持通过多种渠道将告警信息推送给运维人员。
- 告警视图隔离：区分平台管理告警和业务告警，确保不同角色人员关注各自的告警信息。
- 实时告警查看：提供实时告警，集中展示当前处于告警状态的资源数量及详细告警信息。
- 告警历史查看：支持查看一段时间内的历史告警记录，便于运维人员和管理员分析近期监控告警情况。

功能优势

- 全面的监控覆盖：支持对集群、节点、计算组件等多种资源类型的监控，内置丰富的系统告警策略，无需额外配置即可使用。
- 高效的告警管理：通过告警模板实现标准化配置，提高运维效率；告警视图分离，便于不同角色人员快速定位相关告警。
- 及时的问题发现：告警通知自动触发，确保问题及时发现，支持多渠道告警推送，实现主动预防问题。
- 完善的权限管理：严格的告警策略访问控制，确保告警信息安全可控。

通过 UI 创建告警策略

前提条件

- 已配置通知策略（如需配置自动告警通知）。
- 目标集群已安装监控组件（创建基于监控指标的告警策略时必需）。
- 目标集群已安装日志存储组件和日志采集组件（创建基于日志和事件的告警策略时必需）。

操作流程

1. 进入 运维中心 > 告警 > 告警策略。
2. 点击 创建告警策略。
3. 配置基础信息。

选择告警类型

资源告警

- 按资源类型分类的告警类型（如某命名空间下的 deployment 状态）。
- 资源选择说明：
  - 未选择参数时默认为“任意”，支持自动关联新添加的资源。
  - 选择“全选”时，仅作用于当前资源。
  - 选择多个命名空间时，资源名称支持正则表达式（如 `cert.*`）。

事件告警

- 按具体事件分类的告警类型（如 Pod 状态异常）。
- 默认选择指定资源下的所有资源，支持自动关联新添加的资源。

配置告警规则

点击 添加告警规则，根据告警类型配置以下参数：

资源告警参数

参数	说明
表达式	Prometheus 格式的监控指标算法，如 `rate(node_network_receive_bytes{instance="$server",device!~"lo"}[5m])`
指标单位	自定义监控指标单位，可手动输入或从平台预置单位中选择
图例参数	控制图表中曲线对应的名称，格式为 `{{.LabelName}}` ，如 `{{.hostname}}`
时间范围	日志/事件查询的时间窗口
日志内容	日志内容查询字段（如 Error），多个查询字段之间用 OR 连接
事件原因	事件原因查询字段（Reason，如 BackOff、Pulling、Failed 等），多个查询字段之间用 OR 连接
触发条件	由比较运算符、告警阈值和持续时间（可选）组成。根据实时值/日志条数/事件条数与告警阈值的比较，以及实时值在告警阈值范围内的持续时间，判断是否触发告警。
告警级别	分为 Critical、Serious、Warning 和 Info 四个级别。可根据告警规则对业务的影响，为对应资源设置合理的告警级别。

事件告警参数

参数	说明
时间范围	事件查询的时间窗口

参数	说明
事件监控项	支持监控事件级别或事件原因，多个字段之间用 OR 连接
触发条件	基于事件数量进行比较判断
告警级别	与资源告警级别定义相同

其他配置

1. 选择一个或多个已创建的通知策略。
2. 配置告警发送间隔。
   - 全局：使用平台默认配置。
   - 自定义：可根据告警级别设置不同的发送间隔。
   - 选择“不重复”时，仅在告警触发和恢复时发送通知。

其他说明

1. 在告警规则的“更多”选项中，可设置 labels 和 annotations。
2. 具体配置请参考 Prometheus Alerting Rules Documentation↗。
3. 注意：labels 中不要使用 `$value` 变量，可能导致告警异常。

通过 CLI 创建资源告警

前提条件

- 已配置通知策略（如需配置自动告警通知）。
- 目标集群已安装监控组件（创建基于监控指标的告警策略时必需）。
- 目标集群已安装日志存储组件和日志采集组件（创建基于日志和事件的告警策略时必需）。

操作流程

1. 新建 YAML 配置文件，命名为 `example-alerting-rule.yaml`。
2. 在 YAML 文件中添加 PrometheusRule 资源并提交。以下示例创建了名为 policy 的新告警策略：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  annotations:
    alert.cpaas.io/cluster: global # 告警所在集群名称
    alert.cpaas.io/kind: Cluster # 资源类型
    alert.cpaas.io/name: global # 资源对象，支持单个、多个（用 | 分隔）或任意（.*）
    alert.cpaas.io/namespace: cpaas-system # 告警对象所在命名空间，支持单个、多个（用 | 分隔）或任意（.*）
    alert.cpaas.io/notifications: '["test"]'
    alert.cpaas.io/repeat-config: '{"Critical":"never","High":"5m","Medium":"5m","Low":"5m"}'
    alert.cpaas.io/rules.description: '{}'
    alert.cpaas.io/rules.disabled: '[]'
    alert.cpaas.io/subkind: ''
    cpaas.io/description: ''
    cpaas.io/display-name: policy # 告警策略展示名称
  labels:
    alert.cpaas.io/owner: System
    alert.cpaas.io/project: cpaas-system
    cpaas.io/source: Platform
    prometheus: kube-prometheus
    rule.cpaas.io/cluster: global
    rule.cpaas.io/name: policy
    rule.cpaas.io/namespace: cpaas-system
  name: policy
  namespace: cpaas-system
spec:
  groups:
    - name: general # 告警规则组名称
      rules:
        - alert: cluster.pod.status.phase.not.running-tx1ob-e998f0b94854ee1eade5ae79279e005a
          annotations:
            alert_current_value: '{{ $value }}' # 当前值通知，保持默认
          expr: (count(min by(pod)(kube_pod_container_status_ready{}) !=1) or on() vector(0))>2
          for: 30s # 持续时间
          labels:
            alert_cluster: global # 告警所在集群名称
            alert_for: 30s # 持续时间
            alert_indicator: cluster.pod.status.phase.not.running # 告警
```

```
规则指标名称（自定义告警指标名称即为 custom）
            alert_indicator_aggregate_range: '30' # 告警规则聚合时间，单位
秒
            alert_indicator_blackbox_name: '' # 黑盒监控项名称
            alert_indicator_comparison: '>' # 告警规则比较方式
            alert_indicator_query: '' # 告警规则日志查询（仅日志告警）
            alert_indicator_threshold: '2' # 告警规则阈值
            alert_indicator_unit: '' # 告警规则指标单位
            alert_involved_object_kind: Cluster # 告警规则所属对象类型：Clu
ster|Node|Deployment|Daemonset|Statefulset|Middleware|Microservice|Stor
age|VirtualMachine
            alert_involved_object_name: global # 告警规则所属对象名称
            alert_involved_object_namespace: '' # 告警规则所属对象命名空间
            alert_name: cluster.pod.status.phase.not.running-tx1ob # 告
警规则名称
            alert_namespace: cpaas-system # 告警规则所在命名空间
            alert_project: cpaas-system # 告警规则所属对象项目名称
            alert_resource: policy # 告警规则所在告警策略名称
            alert_source: Platform # 告警规则所在告警策略数据类型：Platform-
平台数据 Business-业务数据
            severity: High # 告警规则严重级别：Critical-严重, High-严重, Med
ium-警告, Low-信息
```

通过 CLI 创建事件告警

前提条件

- 已配置通知策略（如需配置自动告警通知）。
- 目标集群已安装监控组件（创建基于监控指标的告警策略时必需）。
- 目标集群已安装日志存储组件和日志采集组件（创建基于日志和事件的告警策略时必需）。

操作流程

1. 新建 YAML 配置文件，命名为 `example-alerting-rule.yaml`。
2. 在 YAML 文件中添加 PrometheusRule 资源并提交。以下示例创建了名为 policy2 的新告警策略：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  annotations:
    alert.cpaas.io/cluster: global
    alert.cpaas.io/events.scope:
      '[{"names":["argocd-gitops-redis-ha-haproxy"],"kind":"Deployment","operator":"=","namespaces":["*"]}]'
      # names: 事件告警的资源名称；若名称为空，operator 无效。
      # kind: 触发事件告警的资源类型。
      # namespace: 触发事件告警的资源所属命名空间。空数组表示非命名空间资源；当 ns 为 ['*'] 时表示所有命名空间。
      # operator: 选择器 =, !=, =~, !~
    alert.cpaas.io/kind: Event # 告警类型，Event（事件告警）
    alert.cpaas.io/name: '' # 资源告警使用，事件告警为空
    alert.cpaas.io/namespace: cpaas-system
    alert.cpaas.io/notifications: '["acp-qwtest"]'
    alert.cpaas.io/repeat-config: '{"Critical":"never","High":"5m","Medium":"5m","Low":"5m"}'
    alert.cpaas.io/rules.description: '{}'
    alert.cpaas.io/rules.disabled: '[]'
    cpaas.io/description: ''
    cpaas.io/display-name: policy2
  labels:
    alert.cpaas.io/owner: System
    alert.cpaas.io/project: cpaas-system
    cpaas.io/source: Platform
    prometheus: kube-prometheus
    rule.cpaas.io/cluster: global
    rule.cpaas.io/name: policy2
    rule.cpaas.io/namespace: cpaas-system
  name: policy2
  namespace: cpaas-system
spec:
  groups:
    - name: general
      rules:
        - alert: cluster.event.count-6sial-34c9a378e3b6dda8401c2d728994ce2f
          # 6sial-34c9a378e3b6dda8401c2d728994ce2f 可自定义以保证唯一性
          annotations:
            alert_current_value: '{{ $value }}' # 当前值通知，保持默认
          expr: round(((avg
```

```
            by(kind,namespace,name,reason)(increase(cpaas_event_count{n
amespace=~".*",id="policy2-cluster.event.count-6sial"}[300s])))
            + (avg
            by(kind,namespace,name,reason)(abs(increase(cpaas_event_cou
nt{namespace=~".*",id="policy2-cluster.event.count-6sial"}[300s]))))
            / 2)>2
          # policy2 中的 id 需为告警策略名称；6sial 必须与前置告警规则名称匹配
          for: 15s # 持续时间
          labels:
            alert_cluster: global # 告警所在集群名称
            alert_for: 15s # 持续时间
            alert_indicator: cluster.event.count # 告警规则指标名称（自定义
告警指标名称即为 custom）
            alert_indicator_aggregate_range: '300' # 告警规则聚合时间，单位
秒
            alert_indicator_blackbox_name: ''
            alert_indicator_comparison: '>' # 告警规则比较方式
            alert_indicator_event_reason: ScalingReplicaSet # 事件原因
            alert_indicator_threshold: '2' # 告警规则阈值
            alert_indicator_unit: pieces # 告警规则指标单位，事件告警保持不变
            alert_involved_object_kind: Event
            alert_involved_object_options: Single
            alert_name: cluster.event.count-6sial # 告警规则名称
            alert_namespace: cpaas-system # 告警规则所在命名空间
            alert_project: cpaas-system # 告警规则所属对象项目名称
            alert_repeat_interval: 5m
            alert_resource: policy2 # 告警规则所在告警策略名称
            alert_source: Platform # 告警规则所在告警策略数据类型：Platform-
平台数据 Business-业务数据
            severity: High # 告警规则严重级别：Critical-严重，High-严重，Med
ium-警告，Low-信息
```

通过告警模板创建告警策略

告警模板是针对类似资源的告警规则和通知策略的组合。通过告警模板，可以轻松快速地为平台上的集群、节点或计算组件创建告警策略。

前提条件

- 已配置通知策略（如需配置自动告警通知）。
- 目标集群已安装监控组件（创建基于监控指标的告警策略时必需）。

操作流程

创建告警模板

1. 在左侧导航栏点击 运维中心 > 告警 > 告警模板。
2. 点击 创建告警模板。
3. 配置告警模板的基础信息。
4. 在 告警规则 区域，点击 添加告警规则，根据以下参数说明添加告警规则：

参数	说明
表达式	Prometheus 格式的监控指标算法，如 `rate(node_network_receive_bytes{instance="$server",device!~"lo"}[5m])`
指标单位	自定义监控指标单位，可手动输入或从平台预置单位中选择
图例参数	控制图表中曲线对应的名称，格式为 `{{.LabelName}}` ，如 `{{.hostname}}`
时间范围	日志/事件查询的时间窗口
日志内容	日志内容查询字段（如 Error），多个查询字段之间用 OR 连接
事件原因	事件原因查询字段（Reason，如 BackOff、Pulling、Failed 等），多个查询字段之间用 OR 连接
触发条件	由比较运算符、告警阈值和持续时间（可选）组成。
告警级别	分为 Critical、Serious、Warning 和 Info 四个级别。可根据告警规则对业务的影响，为对应资源设置合理的告警级别。

5. 点击 创建。

使用告警模板创建告警策略

1. 在左侧导航栏点击 运维中心 > 告警 > 告警策略。
   提示：可通过顶部导航栏切换目标集群。
2. 点击 创建告警策略 按钮旁的展开按钮 > 模板创建告警策略。

3. 配置部分参数，参考以下说明：

参数	说明
模板名称	选择要使用的告警模板名称。模板按集群、节点和计算组件分类。选择模板后，可查看告警模板中设置的告警规则、通知策略等信息。
资源类型	选择模板是针对 集群、节点 还是 计算组件 的告警策略模板；对应的资源名称将显示。

4. 点击 创建。

设置告警静默

支持对集群、节点和计算组件的告警进行静默设置。通过对特定告警策略设置静默，可控制该告警策略下所有规则在静默时间内触发时不发送通知消息。支持永久静默和自定义时间静默。

例如：平台升级或维护时，许多资源可能出现异常状态，导致大量告警触发，运维人员在升级或维护完成前频繁收到告警通知。设置告警策略静默可避免此类情况发生。

注意：静默状态持续到静默结束时间后，静默设置将自动清除。

通过 UI 设置

1. 在左侧导航栏点击 运维中心 > 告警 > 告警策略。
2. 点击需静默的告警策略右侧操作按钮 > 设置静默。
3. 切换 告警静默 开关至开启状态。
   提示：该开关控制静默设置是否生效。取消静默只需关闭开关。
4. 根据以下说明配置相关参数：
   提示：若未选择静默范围或资源名称，默认为 任意，表示后续的 删除/添加 资源操作将对应删除静默/添加静默告警策略；若选择“全选”，则仅作用于当前选定的资源范围，后续的 删除/添加 资源操作不再处理。

参数	说明
静默范围	静默设置生效的资源范围。

参数	说明
资源名称	静默设置针对的资源对象名称。
静默时间	告警静默的时间范围。静默时间开始时，告警进入静默状态；若静默结束时间后告警策略仍处于告警状态或再次触发告警，告警通知将恢复。永久：静默设置持续到告警策略被删除。自定义：自定义静默开始和结束时间，时间间隔不得少于 5 分钟。

5. 点击 设置。
   提示：从设置静默到静默开始前，告警策略的静默状态为 静默等待。此期间，策略内规则触发告警时正常发送通知；静默开始至结束期间，告警策略状态为 静默中，规则触发告警时不发送通知。

通过 **CLI** 设置

1. 指定要设置静默的告警策略资源名称，执行命令：

```
kubectl edit PrometheusRule <TheNameOfThealertPolicyYouWantToSet>
```

2. 按示例修改资源，添加静默注解并提交。

```
---
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  annotations:
    alert.cpaas.io/cluster: global
    alert.cpaas.io/kind: Node
    alert.cpaas.io/name: 0.0.0.0
    alert.cpaas.io/namespace: cpaas-system
    alert.cpaas.io/notifications: '[]'
    alert.cpaas.io/rules.description: '{}'
    alert.cpaas.io/rules.disabled: '[]'
    alert.cpaas.io/rules.version: '23'
    alert.cpaas.io/silence.config:
      '{"startsAt":"2025-02-08T08:01:37Z","endsAt":"2025-02-22T08:01:37
Z","creator":"leizhu@alauda.io","resources":{"nodes":[{"name":"192.168.
36.11","ip":"192.168.36.11"},{"name":"192.168.36.12","ip":"192.168.36.1
2"},{"name":"192.168.36.13","ip":"192.168.36.13"}]}}'
      # 节点级告警策略的静默配置，包括开始时间、结束时间、创建者等；若静默范围包含特
定节点，请按示例追加 resources.node 信息。若需静默所有资源，无需 resources 字
段。
    # alert.cpaas.io/silence.config: '{"startsAt":"2025-02-08T08:04:50
Z","endsAt":"2199-12-31T00:00:00Z","creator":"leizhu@alauda.io","name":
["alb-operator-ctl","apollo"],"namespace":["cpaas-system"]}'
    # 工作负载级告警策略的静默配置，包括开始时间、结束时间、创建者等；若静默范围包含
特定工作负载，请按示例追加 name 和 namespace 信息。若需静默所有资源，无需 name 和
namespace 字段。
    # 设置 endsAt 字段为 2199-12-31T00:00:00Z 表示永久静默。
    alert.cpaas.io/subkind: ''
    cpaas.io/creator: leizhu@alauda.io
    cpaas.io/description: ''
    cpaas.io/display-name: policy3
    cpaas.io/updated-at: 2025-02-08T08:01:42Z
  labels:
  ## 排除无关信息
```

配置告警规则建议

更多的告警规则并不总是更好。冗余或复杂的告警规则可能导致告警风暴，增加维护负担。建议在配置告警规则前阅读以下指导，确保自定义规则既能达到预期目的，又保持高效。

- 尽量少创建新规则：仅创建满足具体需求的规则。通过尽量少的规则数量，可以构建更易管理和集中的监控告警体系。
- 关注症状而非原因：创建通知用户症状的规则，而非症状的根本原因。这样，当相关症状出现时，用户能收到告警，并可进一步调查触发告警的根本原因。此策略可显著减少需创建的规则总数。
- 变更前规划和评估需求：先明确哪些症状重要，以及希望用户在症状出现时采取何种行动。然后评估现有规则，判断是否可通过修改实现目标，而无需为每个症状创建新规则。通过修改现有规则和谨慎创建新规则，有助于简化告警体系。
- 提供清晰的告警信息：创建告警信息时，包含症状描述、可能原因和建议操作。信息应清晰简洁，提供排查流程或相关信息链接，帮助用户快速评估情况并做出响应。
- 合理设置严重级别：为规则分配严重级别，指示用户在症状触发告警时应如何响应。例如，将严重级别设置为 Critical，表示需相关人员立即处理。通过设定严重级别，帮助用户判断告警响应优先级，确保及时处理紧急问题。

通知管理

目录

功能概述
主要功能
通知服务器
企业通信工具服务器
邮件服务器
Webhook 类型服务器
为 Webhook 通知配置自定义请求标头
接收者组
将接收者组与标头 Secret 关联
通知模板
创建模板
引用变量
电子邮件中的特殊格式标记语言
通知规则
前提条件
操作步骤
为项目设置通知规则
前提条件
操作步骤

功能概述

通过通知，您可以集成平台的监控和告警功能，及时向通知接收者发送预警信息，提醒相关人员采取必要措施来解决问题或避免故障。

主要功能

- 通知服务器：通知服务器为平台上的接收者组提供发送通知消息的服务，例如邮件服务器。
- 接收者组：接收者组是一组具有相似逻辑特征的通知接收者。通过对接收通知消息的实体进行分类，可以减轻维护负担。
- 通知模板：通知模板是由自定义内容、内容变量和内容格式参数组成的标准化结构，用于统一通知规则中告警通知消息的内容和格式。例如，自定义邮件通知的主题和内容。
- 通知规则：通知规则是定义如何向特定联系人发送通知消息的一组规则。对于告警、巡检和登录认证等需要通知外部服务的场景，使用通知规则至关重要。

通知服务器

通知服务器为平台上的接收者提供发送通知消息的服务。目前，平台支持以下通知服务器：

- 企业通信工具服务器：支持集成企业微信、钉钉和飞书内置应用，向个人发送通知。
- 邮件服务器：通过邮件服务器使用电子邮件发送通知。
- **Webhook** 类型服务器：支持集成企业微信群机器人、钉钉群机器人、飞书群机器人，或向指定服务器发送 WebHook。

> **WARNING**
>
> 只能添加一个企业通信工具服务器。

企业通信工具服务器

企业微信

1. 按照以下示例配置通知服务器参数。填写参数后，切换到 `global` 集群，在集群管理 > 资源管理中创建资源对象。

```yaml
# WeChat Work corpId, corpSecret, agentId acquisition methods can be referenced in the official documentation: https://developer.work.weixin.qq.com/document/path/90665
apiVersion: v1
kind: Secret
type: NotificationServer
metadata:
  labels:
    cpaas.io/notification.server.type: CorpWeChat
    cpaas.io/notification.server.category: Corp
  name: platform-corp-wechat-server
  namespace: cpaas-system
data:
  displayNameZh: 企业微信 # Server's Chinese display name, encoded in base64 by default
  displayNameEn: WeChat # Server's English display name, encoded in base64 by default
  corpId: # Corporate ID, encoded in base64 by default
  corpSecret: # Application secret, encoded in base64 by default
  agentId: # Corporate application ID, encoded in base64 by default
```

2. 创建完成后，需要在平台的用户角色管理 > 用户管理中，或在用户的个人信息中更新用户的企业微信 **ID**，以确保用户可以正常接收消息。

钉钉

1. 按照以下示例配置通知服务器参数。填写参数后，切换到 `global` 集群，在集群管理 > 资源管理中创建资源对象。

```yaml
# DingTalk appKey, appSecret, agentId acquisition method: https://open-dev.dingtalk.com/fe/app#/corp/app
apiVersion: v1
kind: Secret
type: NotificationServer
metadata:
  labels:
    cpaas.io/notification.server.type: CorpDingTalk
    cpaas.io/notification.server.category: Corp
  name: platform-corp-dingtalk-server
  namespace: cpaas-system
data:
  displayNameZh: 钉钉 # Server's Chinese display name, encoded in base64 by default
  displayNameEn: DingTalk # Server's English display name, encoded in base64 by default
  appKey: # Application key, encoded in base64 by default
  appSecret: # Application secret, encoded in base64 by default
  agentId: # Application agent_id, encoded in base64 by default
```

2. 创建完成后，需要在平台的用户角色管理 > 用户管理中，或在用户的个人信息中更新用户的钉钉 **ID**，以确保用户可以正常接收消息。

飞书

1. 按照以下示例配置通知服务器参数。填写参数后，切换到 `global` 集群，在集群管理 > 资源管理中创建资源对象。

```
# Feishu appId, appSecret acquisition methods: https://open.feishu.cn/app/
apiVersion: v1
kind: Secret
type: NotificationServer
metadata:
  labels:
    cpaas.io/notification.server.type: CorpFeishu
    cpaas.io/notification.server.category: Corp
  name: platform-corp-feishu-server
  namespace: cpaas-system
data:
  displayNameZh: 飞书 # Server's Chinese display name, encoded in base64 by default
  displayNameEn: Feishu # Server's English display name, encoded in base64 by default
  appId: # Application ID, encoded in base64 by default
  appSecret: # Application secret, encoded in base64 by default
```

2. 创建完成后，需要在平台的用户角色管理 > 用户管理中，或在用户的个人信息中更新用户的飞书 **ID**，以确保用户可以正常接收消息。

邮件服务器

1. 在左侧导航栏中，单击平台设置 > 通知服务器。
2. 单击立即配置。
3. 参考以下说明配置相关参数。

参数	描述
服务地址	支持 SMTP 协议的通知服务器地址，例如 `smtp.yeah.net`。
端口	通知服务器的端口号。选中使用 **SSL** 后，必须输入 SSL 端口号。
服务器配置	使用 **SSL**：Secure Socket Layer（SSL）是一种标准安全技术。SSL 开关用于控制是否在服务器与客户端之间建立加密链接。 跳过不安全验证：insecureSkipVerify 开关用于控制是否验证客户端证书和服

参数	描述
	务器主机名。如果启用，则不会验证证书，也不会验证证书中的主机名与服务器主机名是否一致。
发件人邮箱	通知服务器中的发件人邮箱账户，用于发送通知邮件。
启用身份验证	如果需要身份验证，请配置邮件服务器的用户名和授权码。

4. 单击确定。

Webhook 类型服务器

支持集成企业微信群机器人、钉钉群机器人、飞书群机器人，或向指定的 Webhook 服务器发送 HTTP 请求。

企业微信群机器人

1. 在左侧导航栏中，单击集群管理 > 集群。
2. 单击 `global` 集群旁的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的控制平面节点上执行以下命令：

```
kubectl patch secret -n cpaas-system platform-wechat-server -p '{"dat
a":{"enable":"dHJ1ZQo="}}'
```

提示：`dHJ1ZQo=` 是 true 的 base64 编码值；要禁用，请将 `dHJ1ZQo=` 替换为 `ZmFsc2UK`，后者是 false 的 base64 编码值。

钉钉群机器人

1. 在左侧导航栏中，单击集群管理 > 集群。
2. 单击 `global` 集群旁的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的控制平面节点上执行以下命令：

```
kubectl patch secret -n cpaas-system platform-dingtalk-server -p '{"dat
a":{"enable":"dHJ1ZQo="}}'
```

提示：`dHJ1ZQo=` 是 true 的 base64 编码值；要禁用，请将 `dHJ1ZQo=` 替换为 `ZmFsc2UK`，后者是 false 的 base64 编码值。

飞书群机器人

1. 在左侧导航栏中，单击集群管理 > 集群。
2. 单击 `global` 集群旁的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的控制平面节点上执行以下命令：

```
kubectl patch secret -n cpaas-system platform-feishu-server -p '{"dat
a":{"enable":"dHJ1ZQo="}}'
```

提示：`dHJ1ZQo=` 是 true 的 base64 编码值；要禁用，请将 `dHJ1ZQo=` 替换为 `ZmFsc2UK`，后者是 false 的 base64 编码值。

Webhook 服务器

1. 在左侧导航栏中，单击集群管理 > 集群。
2. 单击 `global` 集群旁的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的控制平面节点上执行以下命令：

```
kubectl patch secret -n cpaas-system platform-webhook-server -p '{"dat
a":{"enable":"dHJ1ZQo="}}'
```

提示：`dHJ1ZQo=` 是 true 的 base64 编码值；要禁用，请将 `dHJ1ZQo=` 替换为 `ZmFsc2UK`，后者是 false 的 base64 编码值。

为 **Webhook** 通知配置自定义请求标头

如果目标 Webhook 端点需要自定义 HTTP 请求标头，请在 `cpaas-system` 命名空间中创建 Secret，稍后将其与接收者组关联。

1. 在左侧导航栏中，单击集群管理 > 集群。
2. 单击 `global` 集群旁的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的控制平面节点上创建一个类似于以下内容的 YAML 文件。

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: webhook-header-secret
  namespace: cpaas-system
type: NotificationSender
data:
  X-Secret: <base64-encoded-header-value>
```

4. 将 YAML 文件保存为 `webhook-header.yaml`，并应用资源。

```bash
kubectl apply -f webhook-header.yaml
```

INFO

1. `data` 下的每个键都用作 HTTP 标头名称。
2. `data` 下的每个值都必须进行 base64 编码，然后才能应用到集群。
3. 如果端点需要多个标头，请在 `data` 下添加更多键值对。

接收者组

接收者组是一组具有相似逻辑特征的通知接收者。例如，您可以将运维团队设置为接收者组，以便在配置通知规则时轻松选择和管理。

INFO

1. 平台支持多种通知服务器，并会根据通知服务器配置显示相应通知类型的配置选项。
2. 如果需要使用 Webhook 类型服务器作为通知接收者，则必须在接收者组中配置相关 URL。
3. 如果 Webhook 端点需要自定义请求标头，则必须将接收者组与 `cpaas-system` 命名空间中类型为 `NotificationSender` 的 Secret 关联。

1. 在左侧导航栏中，单击运维中心 > 通知。

2. 切换到接收者组页签。
3. 单击创建接收者组，并按照以下说明配置相关参数。

参数	描述
Email	为整个接收者组添加一个电子邮件地址。平台会向此电子邮件地址以及该组中所有联系人的电子邮件地址发送通知。
Webhook URL/WeChat 群机器人**/DingTalk** 群机器人**/Feishu** 群机器人	根据已配置的通知服务器填写相应通知方式的 URL。配置完成后，该组中的联系人将通过此方式收到通知。
联系人配置	单击添加联系人，将现有平台用户添加到接收者组。请确保所选联系人的联系信息（电话、电子邮件、接口回调）准确无误，以避免遗漏消息通知。

4. 单击添加。

将接收者组与标头 **Secret** 关联

如果已配置的 Webhook 端点需要自定义请求标头，请将接收者组与在配置 Webhook 通知的自定义请求标头中创建的 Secret 关联。

1. 在左侧导航栏中，单击集群管理 > 集群。
2. 单击 `global` 集群旁的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的控制平面节点上编辑相应的接收者组（`NotificationGroup`）资源。

```
kubectl edit notificationgroups.ait.alauda.io -n cpaas-system <notification-group-name>
```

4. 将 `cpaas.io/notification.webhook.config` 注解添加到 `metadata.annotations`。其值必须是为自定义请求标头创建的 Secret 名称。

```yaml
apiVersion: ait.alauda.io/v1beta1
kind: NotificationGroup
metadata:
  name: <notification-group-name>
  namespace: cpaas-system
  annotations:
    cpaas.io/notification.webhook.config: webhook-header-secret
```

INFO

Webhook URL 在接收者组中配置，而自定义请求标头则通过所引用的 Secret 进行配置。

通知模板

通知模板是一种标准化结构，由自定义内容、内容变量和内容格式参数组成。它用于统一通知规则中告警通知消息的内容和格式。

平台管理员或运维人员可以设置通知模板，根据不同的告警通知方式自定义通知消息的内容和格式，帮助用户快速获取关键告警信息并提高运维效率。

平台管理员可以将平台通知模板设为公开。项目通知规则可以选择公开模板，因此项目管理员无需在每个项目中重复创建模板，即可复用由平台维护的消息格式。

INFO

平台支持多种通知服务器，系统会根据通知服务器配置显示相应的通知类型模板。如果未配置通知服务器，则默认不显示相应的通知模板。

创建模板

1. 在左侧导航栏中，单击运维中心 > 通知。
2. 切换到模板页签。
3. 单击创建模板。
4. 在基本信息区域中配置以下参数。

参数	说明
消息类型	根据通知用途选择消息类型。 告警消息：发送由告警规则触发的告警消息，配合平台的告警功能使用； 组件异常消息：发送由某些组件异常触发的通知信息。
公开	将模板设为项目通知规则可用。此选项仅适用于平台通知模板。模板公开后，项目管理员创建或更新通知规则时，可以从平台公开模板组中选择该模板。公开模板在模板列表中标记为只读的公开标签，并在详情页显示公开状态。再次将公开模板设为私有时，请在提交更改前于对话框中确认影响。

5. 在模板配置区域中，参考不同的模板类型配置变量和内容格式参数。

INFO

1. 模板内容只能由变量、变量显示名称以及平台支持的特殊格式标记语言组成。只要符合语法规则，就可以自由组合变量和其他元素。
2. 模板中只能使用平台支持的变量。可以修改变量显示名称和内容格式，但不能修改变量本身。请参阅引用变量和电子邮件中的特殊格式标记语言。
3. 平台根据实际运维场景为各种通知类型提供默认通知模板内容，可满足大多数通知消息设置需求。如果没有特殊要求，可以直接使用默认模板内容。
4. 如果再次将平台公开模板设为私有，该模板仍会在平台公开模板组中显示，但处于禁用状态。创建新的通知规则时无法选择该模板，但已使用该模板的现有通知规则仍可继续发送通知。
5. 如果任何项目通知规则引用了平台公开模板，则该模板受删除保护。删除模板前，请先移除所有引用。

6. 单击创建。

引用变量

变量是通知消息（NotificationMessage）中标签或注解的键，格式为 `{{.labelKey}}`。为方便用户快速获取关键信息，可以为变量指定自定义显示名称，例如：`Alert Level: {{ .externalLabels.severity }}`。

通知规则根据通知模板向用户发送通知消息时，模板中的变量会引用通知消息（实际监控数据）中对应的标签值。最终，监控数据将以标准化的内容格式发送给用户。

平台默认提供以下基本变量：

显示名称	变量	说明
告警状态	`{{ .externalLabels.status }}`	例如：告警中。
告警级别	`{{ .externalLabels.severity }}`	例如：严重。
告警集群	`{{ .labels.alert_cluster }}`	例如：发生告警的集群 1。
告警对象	`{{ .externalLabels.object }}`	发生告警的资源类型和名称，例如节点 192.168.16.53。
规则名称	`{{ .labels.alert_resource }}`	告警规则的名称，例如 cpaas-node-rules。
告警描述	`{{ .externalLabels.summary }}`	告警规则的描述。
触发值	`{{ .externalLabels.currentValue }}`	触发告警的监控值。
告警时间	`{{ dateFormatWithZone .startsAt "2006-01-02 15:04:05" "Asia/Chongqing" }}`	告警开始时间。
恢复时间	`{{ dateFormatWithZone .endsAt "2006-01-02 15:04:05" "Asia/Chongqing" }}`	告警结束时间。
指标名称	`{{ .labels.alert_indicator }}`	监控指标的名称。

电子邮件中的特殊格式标记语言

电子邮件通知中常用的 HTML 格式标签及其说明如下表所示：

内容元素	标签	说明
文本	-	支持输入中英文文本内容。
字体	`<font color="#FF0000">Set Font Color</font>` `<b>Bold Font</b>`	设置字体格式。
标题	`<h1>Level 1 Title</h1>`，最高支持 h6（六级标题）。	设置标题级别。
段落	`<p>Paragraph</p>`	插入普通段落文本。
引用	`<q>Quote</q>`	插入简短引用内容。
超链接	`<a href="//www.example.com">Hyperlink</a>`	插入超链接。

通知规则

通知规则是一组用于定义如何向指定联系人发送通知消息的规则。对于需要向外部服务发送通知的场景，例如告警、巡检和登录认证，使用通知规则至关重要。

> **INFO**
>
> 平台支持多种通知服务器，并会根据通知服务器配置显示与通知类型对应的通知方式。如果未配置通知服务器，默认不会显示相应的通知方式。

前提条件

要使用企业通信工具服务器通知联系人，用户必须先在个人信息中修改联系信息，填写 `WeChat Work ID`。

操作步骤

1. 在左侧导航栏中，单击运维中心 > 通知。

2. 单击创建规则，并按照以下说明配置相关参数。

参数	描述
接收组	接收组是通知接收者的逻辑集合，平台将使用指定的通知方式向其发送通知。
通知接收者	选择添加一个或多个通知接收者，平台将根据接收者在个人信息中的联系方式发送通知。
通知方式	支持企业微信、钉钉、飞书、企业微信群机器人、钉钉群机器人、飞书群机器人和 **WebHook URL** 等多种方式，并支持多选。 注意：配置通知服务器后才会显示部分参数。
通知模板	选择用于展示通知信息的通知模板。在项目视图中，模板按来源分组。您可以选择项目模板或平台公共模板。

3. 单击创建。

为项目设置通知规则

默认情况下，平台的通知规则、通知模板和接收组按租户隔离。作为项目管理员，您无法查看或使用其他项目或平台管理员配置的通知规则、私有通知模板或接收组。平台公共通知模板除外：创建或更新通知规则时，您可以选择这些模板，但无法在项目视图中编辑或删除它们。

前提条件

1. 您已联系平台管理员完成通知服务器配置。
2. 如果需要通过企业通信工具发送通知，还需确保待通知联系人的通信工具 ID 已在个人信息中正确配置。

操作步骤

1. 在项目管理视图中，单击*项目名称*。
2. 在左侧导航栏中，单击通知。

3. 切换到接收组页签，参考接收组创建接收组。

> **TIP**
>
> 如果无需通过接收组管理通知联系人，或无需向 Webhook 类型通知服务器发送通知，则可以跳过此步骤。

4. 切换到模板页签，参考通知模板创建通知模板。如果已有适用的平台公共模板，可以跳过此步骤，并在创建通知规则时选择该模板。
5. 切换到规则页签，参考通知规则创建通知规则。

监控面板管理

INFO

从 Alauda Container Platform 4.4 开始，Perses 监控面板 作为新的监控面板功能可用。Perses 监控面板与现有的 MonitorDashboard 功能可共存。您可以继续使用本页面管理 MonitorDashboard，也可以将选定的 MonitorDashboard 资源迁移到 PersesDashboard 资源。

目录

功能概述
- 主要功能
- 优势
- 使用场景
- 前提条件
- 监控面板与监控组件之间的关系

管理监控面板
- 创建监控面板
- 导入监控面板
- 添加变量
- 添加图表
- 添加分组
- 切换监控面板
- 其他操作

管理图表
图表描述
图表配置描述
常规参数
图表特殊参数
通过 CLI 创建监控仪表盘
常用函数和变量
常用函数
常用变量
变量用例一
变量用例二
使用内置指标时的注意事项

功能概述

平台提供了强大的监控面板管理功能，旨在替代传统的 Grafana 工具，为用户提供更加全面、灵活的监控体验。此功能聚合平台内的各种监控数据，呈现统一的监控视图，从而显著提升配置效率。

主要功能

- 支持为业务视图和平台视图配置自定义监控面板。
- 支持在业务视图中查看平台视图中配置的公开共享监控面板，并根据业务所属的命名空间隔离数据。
- 支持管理监控面板中的图表，允许用户添加、删除和修改图表，对图表进行缩放，以及通过拖放移动图表。
- 支持在监控面板中设置自定义变量，用于筛选查询数据。
- 支持在监控面板中配置用于管理图表的分组。分组可以根据自定义变量重复显示。
- 支持的图表类型包括：趋势图、阶梯线图、柱状图、水平柱状图、条形仪表图、仪表盘图、表格、统计图、XY 图、饼图、文本。

- 支持一键导入 Grafana 监控面板。

优势

- 支持用户自定义监控场景，不受预定义模板限制，真正实现个性化监控体验。
- 提供丰富的可视化选项，包括折线图、柱状图、饼图，以及灵活的布局和样式选项。
- 与平台的角色权限无缝集成，允许业务视图定义自己的监控面板，同时确保数据隔离。
- 与容器平台的各种功能深度集成，可即时访问容器、网络、存储等监控数据，为用户提供全面的性能观测和故障诊断能力。
- 完全兼容 Grafana 监控面板 JSON，便于从 Grafana 迁移并继续使用。

使用场景

- **IT** 运维管理：作为 IT 运维团队的一员，您可以使用监控面板统一展示和管理容器平台的各种性能指标，例如 CPU、内存、网络流量等。通过自定义监控报告和告警规则，您可以及时发现并定位系统问题，提升运维效率。
- 应用性能分析：对于应用开发人员和测试人员，监控面板提供多种丰富的可视化选项，以直观展示应用运行状态和资源消耗。您可以针对不同的应用场景自定义专用监控视图，深入分析应用性能瓶颈，为优化提供依据。
- 多集群管理：对于管理多个容器集群的用户，监控面板可以聚合来自不同集群的监控数据，使您能够一目了然地掌握系统的整体运行状态。
- 故障诊断：发生系统问题时，监控面板可以提供全面的性能数据和分析工具，帮助您快速定位问题根因。您可以根据告警信息快速查看相关监控指标的波动，深入分析故障。

前提条件

目前，监控面板仅支持查看由平台中安装的监控组件采集的监控数据。因此，在配置监控面板之前，您应做好以下准备：

- 确保要配置监控面板的集群已安装监控组件，具体为 **ACP Basic Monitoring** 或 **ACP Advanced Monitoring** 插件。
- 确保您希望在监控面板中显示的数据已由监控组件采集。

监控面板与监控组件之间的关系

- 监控面板资源存储在 Kubernetes 集群中。您可以使用顶部的 **Cluster** 页签在不同集群之间切换视图。
- 监控面板依赖集群中的监控组件查询数据源。因此，在使用监控面板之前，请确保当前集群已成功安装监控组件，且监控组件运行正常。
- 监控面板默认会从对应集群请求监控数据。如果您在集群中以代理模式安装了 **VictoriaMetrics** 插件，我们会为您请求存储集群，以查询该集群对应的数据，无需进行特殊配置。

管理监控面板

监控面板由一个或多个图表组成，这些图表按一个或多个行组织和排列，以便清晰地查看相关信息。图表可以从数据源查询原始数据，并将其转换为平台支持的一系列可视化效果。

创建监控面板

1. 单击 创建监控面板，参考以下说明配置相关参数。

参数	描述
文件夹	监控面板所在的文件夹；您可以输入文件夹名称或选择现有文件夹。
标签	监控面板的标签；切换监控面板时，您可以通过顶部的标签筛选，快速查找现有监控面板。
设为主监控面板	启用后，创建成功时将当前监控面板设为主监控面板；下次进入监控面板功能时，默认显示主监控面板的数据。
变量	创建监控面板时添加变量，以便在添加的图表中将其用作指标参数，也可以将其用作监控面板首页上的筛选条件。

2. 添加完成后，单击 创建，完成监控面板的创建。接下来，您需要添加变量、添加图表和添加分组，以完成整体布局设计。

导入监控面板

平台支持直接导入 Grafana JSON，并将其转换为监控面板进行展示。

- 目前仅支持导入 V8+ 版本的 Grafana JSON；低版本将无法导入。
- 如果导入的监控面板中包含平台不支持的图表，这些图表可能会显示为不支持的图表类型，但您可以修改图表设置以使其正常显示。
- 导入监控面板后，您可以像往常一样执行各种管理操作，导入的图表与在平台中创建的图表没有区别。

添加变量

1. 在变量表单区域，单击 添加。

查询

`Query` 类型的变量可以根据时间序列的特征维度筛选数据。您可以指定查询表达式，以动态计算并生成查询结果。

参数	描述
查询设置	定义查询设置时，除了使用 PromQL 查询时间序列外，平台还提供了一些常用变量和函数。请参阅常用函数和变量。
正则表达式	使用正则表达式，可以从变量查询返回的内容中筛选出所需的值，使变量中的每个选项名称更符合预期。您可以在变量值预览中预览筛选结果是否符合预期。
选择设置	- 多选：在监控面板首页的顶部筛选器中选择此项后，可以同时选择多个选项。您需要在图表的查询表达式中引用此变量，才能查看与变量值对应的数据。 - 全部：勾选后，筛选选项中将启用包含全部的选项，以便选择所有变量数据。

常量

常量变量是值固定且在整个监控面板中保持不变的静态变量，通常用于存储环境标识符、固定阈值或配置参数，以供多个图表引用，而不会显示为筛选选项。

参数	描述
常量值	常量变量的值。

自定义

自定义变量允许用户定义预设的静态选项列表，这些选项将作为监控面板上的下拉筛选器显示。此类变量通常用于手动选择特定服务、团队或类别，无需执行动态数据查询。

参数	描述
自定义设置	输入以逗号分隔的选项值，每个选项使用 display_name : value 格式（例如 Production : prod, Staging : stage, Development : dev）；如果显示名称与值相同，也可以直接列出值。

文本框

文本框变量允许用户直接输入文本，通常用于指定无需动态数据查询的特定值或参数。

参数	描述
文本框值	文本框变量的默认值。

2. 单击 确定，添加一个或多个变量。

添加图表

向当前创建的监控面板添加多个图表，以显示不同资源的数据。

提示：单击图表右下角可自定义图表大小；单击图表任意位置可调整图表顺序。

1. 单击 添加图表，参考以下说明配置相关参数。

- 图表预览：此区域将动态显示与已添加指标对应的数据信息。
- 添加指标：在此区域配置图表标题和监控指标。
- 添加方式：支持使用内置指标或原生自定义指标。两种方式的指标将合并，并同时生效。
  - 内置指标：选择平台内置的常用指标和图例参数，以显示当前图表中的数据信息。
    - 注意：添加到图表中的所有指标必须使用统一单位；无法在一个图表中添加多个单位的指标。

  - 原生：自定义指标单位、指标表达式和图例参数。指标表达式遵循 PromQL 语法；详情请参阅 PromQL 官方文档↗。

- 图例参数：控制图表中曲线对应的名称。您可以使用文本或模板：

  - 规则：输入值必须采用 `{{.xxxx}}` 格式；例如，`{{.hostname}}` 会替换为表达式返回的 hostname 标签对应的值。
  - 提示：如果输入的图例参数格式不正确，图表中曲线对应的名称将以原始格式显示。

- 即时切换：启用 即时 开关后，将通过 Prometheus 的 Query 接口查询即时值并进行排序，例如统计图表和仪表盘图表。关闭后，将使用 `query_range` 方法进行计算，查询特定时间段内的一系列数据。
- 图表设置：支持选择不同的图表类型来可视化指标数据。请参阅管理图表。

2. 单击 保存，完成图表添加。
3. 您可以在监控面板中添加一个或多个图表。
4. 添加图表后，您可以执行以下操作，确保图表的显示效果和大小符合预期。

   - 单击图表右下角，自定义图表大小。
   - 单击图表任意位置，调整图表顺序。
   - 单击 编辑 按钮，修改图表设置。
   - 单击 删除 按钮，删除图表。
   - 单击 复制 按钮，复制图表。

5. 调整完成后，单击监控面板页面上的 保存 按钮，保存修改。

添加分组

分组是监控面板中的逻辑分隔符，可用于将图表归类在一起。

1. 单击 添加图表 下拉菜单 > 添加分组，并参考以下说明配置相关参数。

- 分组：分组的名称。
- 重复：支持禁用重复，或选择用于当前图表的变量。

  - 禁用重复：不选择变量，使用默认创建的分组。

- 参数变量：选择当前图表中创建的变量，监控面板将针对变量的每个对应值生成一行相同的子分组。子分组不支持修改、删除或移动图表。

2. 添加分组后，您可以执行以下操作，管理监控面板中的图表显示。

- 可以折叠或展开分组，以隐藏监控面板中的部分内容。折叠分组中的图表不会发送查询。
- 将图表移入分组，使该图表由该分组管理。分组将管理其与下一个分组之间的所有图表。
- 折叠分组后，您也可以将该分组管理的所有图表一起移动。
- 折叠和展开分组也属于对监控面板的调整。如果希望下次重新打开此监控面板时保留该状态，请单击 保存 按钮。

切换监控面板

将创建的自定义监控面板设为主监控面板。下次进入监控面板功能时，默认显示主监控面板的数据。

1. 在左侧导航栏中，单击 运营中心 > 监控 > 监控面板。
2. 默认进入主监控面板。单击 切换监控面板。
3. 您可以通过标签筛选或按名称搜索来查找监控面板，并通过 主监控面板 开关切换主监控面板。

其他操作

您可以单击监控面板页面右侧的操作按钮，根据需要对监控面板执行操作。

操作	描述
YMAL	打开存储在 Kubernetes 集群中的监控面板 CR 资源实际代码。您可以通过编辑 YAML 中的参数修改监控面板的所有内容。
导出表达式	以 CSV 格式导出当前监控面板中使用的指标及其对应的查询表达式。
复制	复制当前监控面板；您可以根据需要编辑图表，并将其保存为新的监控面板。
设置	修改当前监控面板的基本信息，例如更改标签和添加更多变量。
删除	删除当前监控面板。

管理图表

平台提供多种可视化方式，以支持不同的使用场景。本章将主要介绍这些图表类型、配置选项和使用方法。

图表描述

编号	图表名称	描述	建议使用场景
1	趋势图	通过一个或多个线段显示数据随时间的变化趋势。	显示随时间变化的趋势，例如 CPU 利用率、内存使用量等的变化。
2	阶梯线图	在折线图的基础上，通过水平和垂直线段连接数据点，形成阶梯状结构。	适合显示离散事件的时间戳，例如告警数量。
3	柱状图	使用垂直矩形柱表示数据量，柱的高度表示数值。	柱状图便于直观比较数值差异，有助于发现模式和异常，适用于关注数值变化的场景，例如 pod 数量、节点数量等。
4	横向柱状图	与柱状图类似，但使用水平矩形柱表示数据。	当数据维度较多时，横向柱状图可以更好地利用空间布局并提高可读性。
5	仪表盘图	使用半环或环形表示指标的当前值及其占总值的比例。	直观反映关键监控指标的当前状态，例如系统 CPU 利用率和内存使用量。建议使用带颜色变化的告警阈值来表示异常情况。
6	仪表条图	使用垂直矩形柱显示指标的当前值及其占比。	直观反映关键指标的当前状态，例如目标完成进度和系统负载。当同一指标存在多个类别时，更推荐使用仪表条图，例如可用磁盘空间或磁盘利用率。
7	饼图	使用扇区显示部分与整体之间的比例关	适合展示不同维度下整体数据的组成，例如一段时间内 4XX、

编号	图表名称	描述	建议使用场景
		系。	3XX 和 2XX 响应码的占比。
8	表格	以行列形式组织数据，便于查看和比较具体数值。	适合显示结构化的多维数据，例如节点详细信息、pod 详细信息等。
9	统计图	显示单个关键指标的当前值，通常需要配合文本说明。	适合显示重要监控指标的实时值，例如 pod 数量、节点数量、当前告警数量等。
10	散点图	使用笛卡尔坐标绘制一系列数据点，反映两个变量之间的相关性。	适合分析两个指标之间的关系，通过数据点的分布发现线性相关、聚类等模式，帮助挖掘指标之间的关系。
11	文本卡片	以卡片形式显示关键文本信息，通常包含标题和简短描述。	适合展示文本信息，例如图表描述和故障排查说明。

图表配置描述

常规参数

参数	描述
基本信息	根据选定的指标数据选择合适的图表类型，并添加标题和描述；您可以添加一个或多个链接，通过单击标题旁边对应的链接名称快速访问。
标准设置	原生指标数据使用的单位。此外，仪表盘图和仪表条图还支持配置 总值 字段，该字段将在图表中以 当前值/总值 的百分比形式显示。
工具提示	工具提示用于控制将鼠标悬停在图表上时是否显示实时数据，并支持按选定条件排序。
阈值参数	配置图表的阈值开关；启用后，图表中将以选定的颜色显示阈值，从而可以设置阈值大小。

参数	描述
值	设置值的计算方式，例如最新值或最小值。此配置选项仅适用于统计图和仪表盘图。
值映射	重新定义指定值、范围、正则表达式或特殊值，例如将 100 定义为满负载。此配置选项仅适用于统计图、表格和仪表盘图。

图表特殊参数

图表类型	参数	描述
趋势图	图表样式	您可以选择折线图或面积图作为显示样式；折线图更侧重于反映指标的趋势变化，而面积图更关注总量和部分占比的变化。请根据实际需求进行选择。
仪表盘图	仪表盘图设置	显示方向：当您需要在一个图表中查看多个指标时，可以设置这些指标以横向或纵向排列。 单位重新定义：您可以为每个指标设置独立的单位；如果未设置，平台将显示 标准设置 中的单位。
统计图	统计图设置	显示方向：当您需要在一个图表中查看多个指标时，可以设置这些指标以横向或纵向排列。 图表模式：您可以在统计图中添加图表，以显示指标随时间的变化趋势。
饼图	饼图设置	最大切片数：您可以设置此参数以减少饼图中的切片数量，降低占比较小但数量较多的类别造成的干扰。超出数量的切片将合并并显示为 其他。 标签显示字段：您可以设置饼图标签中显示的字段。
饼图	图表样式	您可以选择饼图或环形图作为显示样式。
表格	表格设置	隐藏列：您可以通过此参数减少表格中的列数，重点查看主要列信息。 列对齐方式：您可以通过此参数修改列中数据的对齐方式。

图表类型	参数	描述
		显示名称和单位：您可以通过此参数修改使用的列名称和单位。
文本卡片	图表样式	样式：您可以选择在富文本编辑框或 HTML 中编辑要在文本卡片中显示的内容。

通过 **CLI** 创建监控仪表盘

1. 创建一个名为 `example-dashboard.yaml` 的新 YAML 配置文件。
2. 将 MonitorDashboard 资源添加到 YAML 文件中并提交。以下示例创建了一个名为 demo-v2-dashboard1 的监控仪表盘：

```yaml
kind: MonitorDashboard
apiVersion: ait.alauda.io/v1alpha2
metadata:
  annotations:
    cpaas.io/dashboard.version: '3'
    cpaas.io/description: '{"zh":"描述信息","en":""}' # Description fiel
d
    cpaas.io/operator: admin
  labels:
    cpaas.io/dashboard.folder: demo-v2-folder1 # Folder
    cpaas.io/dashboard.is.home.dashboard: 'False' # Is it the main dash
board?
  name: demo-v2-dashboard1 # Name
  namespace: cpaas-system # Namespace (all management view creations wi
ll occur in this ns)
spec:
  body: # All information fields
    titleZh: 更新显示名称 # Built-in field for Chinese display name (this
field is created under the Chinese language)
    title: english_display_name # Built-in field for English display na
me (this field is created under the English language) Built-in dashboar
ds can set bilingual translations.
    templating: # Custom variables
      list:
        - hide: 0 # 0 means not hidden; 1 means only the label is hidde
n; 2 means both label and value are hidden
          label: 集群 # Built-in variable display name (label is set to
the appropriate name based on the language, e.g., cluster in English)
          name: cluster # Built-in variable name (unique)
          options: # Define dropdown options; if a query retrieves dat
a, it will use requested data; otherwise, it will use options. A defaul
t value can be set (generally only used for setting default values)
            - selected: false # Whether to default select
              text: global
              value: global
          type: custom # Custom variable type; currently, only built-in
(custom) and query are supported (Importing Grafana will support consta
nt custom interval (after import, it will be changed to a custom variab
le and will not support auto))

        - allValue: '' # Select all, passing options with the format xx
x|xxx|xxx; can set allValue for conversion (Grafana retrieves all data
for the current variable as xxx|xxx|xxx, adjustments will ensure consis
```

```
tency)
          current: null # Current value of the variable; if not set, de
faults to the first in the list
          definition: query_result(kube_namespace_labels) # Query expre
ssion for data retrieval
          hide: 0 # 0 means not hidden; 1 means only the label is hidde
n; 2 means both label and value are hidden
          includeAll: true # Whether to select all
          label: ns # Built-in variable display name
          multi: true # Whether multiple selections are allowed
          name: ns # Variable name (unique)
          options: []
          query: ''
          regex: /.*namespace=\"(.*?)\".*/ # Regex expression for extra
cting variable values
          sort: 2 # Sorting: 1 - ascending alphabetical order; 2 - desc
ending alphabetical order (only these two support temporarily); 3 - asc
ending numerical order; 4 - descending numerical order
          type: query # Custom variable type
    time: # Dashboard time
      from: now-30m # Start time
      to: now # End time
    repeat: '' # Row repeat configuration; chooses custom variable
    collapsed: 'false' # Row collapsed or expanded configuration
    description: '123' # Description (tooltip after title)
    targets: # Data sources
      - indicator: cluster.node.ready # Metric
        expr: sum (cpaas_pod_number{cluster=\"\"}>0) # PromQL expressio
n
        instant: false # Query mode true retrieves data at a specific t
ime
        legendFormat: '' # Legend
        range: true # Default querying range when retrieving data
        refId: 指标1 # Unique identifier for display name of data source
    gridPos: # Information on the dashboard's positional layout
      h: 8 # Height
      w: 12 # Width (width corresponds to 24 grid units)
      x: 0 # Horizontal position
      y: 0 # Vertical position
    panels: # Panel data
      title: 图表标题tab # Panel name
      type: table # Panel type; currently supports timeseries, barchar
t, stat, gauge, table, bargauge, row, text, pie (step chart, scatter pl
ot, bar chart, configurable through drawStyle attribute)
```

```
      id: a2239830-492f-4d27-98f3-cb7ecb77c56f # Unique identifier
      links: # Links
        - targetBlank: true # Open in a new tab
          title: '1' # Name
          url: '1' # URL address
      transformations: # Data transformations
        - id: 'organize' # Type organize; used for sorting, rearranging order, showing fields, whether to display
          options:
            excludeByName: # Hidden fields
              cluster_cpu_utilization: true
            indexByName: # Sort
              cluster_cpu_utilization: 0,
              Time: 1
            renameByName: # Rename
              Time: ''
              cluster_cpu_utilization: '222'
        - id: 'merge' # Merging data
          options:
      fieldConfig: # For defining panel properties and appearance
        defaults: # Default configuration
          custom: # Custom graphic attributes
            align: 'left' # Table alignment: left, center, right
            cellOptions: # Table threshold configuration
              type: color-text # Only supports text for threshold color settings
            spanNulls: false # true connects null values; false does not connect; number == 0 connects null values according to 0
            drawStyle: line # Panel types: line, bars for bar charts, points for point charts
            fillOpacity: 20 # Exists when drawStyle is area (currently does not support configuration, area defaults to 20)
            thresholdsStyle: # Configures how to display thresholds (currently only supports line)
              mode: line # Threshold display format (area not supported currently)
            lineInterpolation: 'stepBefore' # Step chart configuration; defaults to only supporting stepBefore (stepAfter will be supported later)
          decimals: 3 # Decimal points
          min: 0 # Minimum value (currently not supported for page configuration, only supports imports that have been adapted)
          max: 1 # Maximum value (page configuration only applies to stat gauge barGauge pie)
```

```
          unit: '%' # Unit
          mappings: # Value mapping configuration (currently only suppo
rts value and range types; special types supported on data)
            - options: # Value mapping rules
                '1': # Corresponding value
                  index: 0
                  text: 'Running' # Displayed as Running when value is
1
              type: value # Value mapping type
            - options: # Range mapping rules
                from: 2 # Range start value
                to: 3 # Range end value
                result: # Mapping result
                  index: 1
                  text: 'Error' # Values from 2 to 3 will display as Er
ror
              type: range # Mapping type for range
            - type: special # Mapping type for special scenarios
              options:
                match: null # nan null null+nan empty true false
                result:
                  text: xxx
                  index: 2
          thresholds: # Threshold configuration
            mode: absolute # Threshold configuration mode, absolute val
ue mode (currently only supports absolute and percentage mode; percenta
ge mode is not supported yet)
            steps: # Threshold steps
              - color: '#a7772f' # Threshold color
                value: '2' # Threshold value
              - color: '#007AF5' # Default value with no value is the B
ase
        overrides: # Override configuration
          - matcher:
              id: byName # Match based on field name
              options: node # Corresponding name
            properties: # Override configuration; id currently only sup
ports displayName unit
              - id: displayName # Display name override
                value: '1' # Overridden display name
              - id: unit # Unit override
                value: GB/s # Unit value
              - id: noValue # No value display
                value: No value display
```

```
      options:
        orientation: horizontal # Control the layout direction of panel
s; applies to gauge and barGauge (stat will be supported later)
        legend: # Legend configuration
          calcs: # Calculating methods (only displays when the legend p
osition is on the right)
            - latest # Currently only supports most recent value
          placement: right # Legend position (right or bottom; defaults
to bottom)
          placementRightTop: '' # Configuration for the upper right
          showLegend: true # Whether to display the legend
        tooltip: # Tooltips
          mode: multi # Mode dual selection (only multi-mode supported)
All data displayed when the mouse hovers over
          sort: asc # Sorting: asc or desc
        reduceOptions: # Value calculating method (used for aggregating
data)
          calcs: # Calculating methods (latest, minimum, maximum, avera
ge, sum)
            - latest
          limit: 3 # Pie limits the number of slices
        textMode: 'value' # Stat configuration; defines style for displ
aying metric value; options are auto, value, value_and_name, name, none
(currently not supported in the page configuration, but supported in im
ports)
        colorMode: 'value' # Stat configuration; defines color mode for
displaying metric values; options are none, value, background (defaults
to value; not supported in configuration but adapted in import)
        displayLabels: ['name', 'value', 'percent'] # Fields displayed
in pie chart labels
        pieType: 'pie' # Pie chart type; options are pie and donut
        mode: 'html' # Text chart type mode; options are html and richT
ext
        content: '<div>xxx</div>' # Content for text chart type
        footer:
          enablePagination: true # Table pagination enabled
```

常用函数和变量

常用函数

定义查询设置时，除了使用 PromQL 设置查询外，平台还提供以下常用函数，供您自定义查询设置时参考。

函数	用途
label_names()	返回 Prometheus 中的所有标签，例如 label_names()。
label_values(label)	返回 Prometheus 中所有受监控指标的指定标签名称对应的所有可选值，例如 label_values(job)。
label_values(metric, label)	返回 Prometheus 中指定指标的标签名称对应的所有可选值，例如 label_values(up, job)。
metrics(metric)	返回指标字段中符合指定正则表达式模式的所有指标名称，例如 metrics(cpaas_active)。
query_result(query)	返回指定 Prometheus 查询的查询结果，例如 query_result(up)。

常用变量

定义查询设置时，您可以将常用函数组合为变量，以便快速定义自定义变量。以下是一些常见的变量定义，供您参考：

变量名称	查询函数
cluster	`label_values(cpaas_cluster_info,cluster)`
node	`label_values(node_load1, instance)`
namespace	`query_result(kube_namespace_labels)`
deployment	`label_values(kube_deployment_spec_replicas{namespace="$namespace"}, deployment)`
daemonset	`label_values(kube_daemonset_status_number_ready{namespace="$namespace" daemonset)`

变量名称	查询函数
statefulset	`label_values(kube_statefulset_replicas{namespace="$namespace"}, statefulset)`
pod	`label_values(kube_pod_info{namespace=~"$namespace"}, pod)`
vmcluster	`label_values(up, vmcluster)`
daemonset	`label_values(kube_daemonset_status_number_ready{namespace="$namespace" daemonset)`

变量用例一

使用 **query_result(query)** 函数查询值：`node_load5`，并提取 IP。

1. 在 查询设置 中填写 `query_result(node_load5)`。
2. 在 变量值预览 区域，预览示例为 `node_load5{container="node-exporter",endpoint="metrics",host_ip="192.168.178.182",instance="192.168.178.182:9100"}`。
3. 在 正则表达式 中填写 `/.*instance="(.*?):.*/` 以筛选值。
4. 在 变量值预览 区域，预览示例为 `192.168.176.163`。

变量用例二

1. 添加第一个变量：namespace，使用 **query_result(query)** 函数查询值：`kube_namespace_labels`，并提取 namespace。
   - 查询设置：`query_result(kube_namespace_labels)`。
   - 变量值预览：`kube_namespace_labels{container="exporter-kube-state", endpoint="kube-state-metrics", instance="12.3.188.121:8080", job="kube-state", label_cpaas_io_project="cpaas-system", namespace="cert-manager", pod="kube-prometheus-exporter-kube-state-55bb6bc67f-lpgtx", project="cpaas-system", service="kube-prometheus-exporter-kube-state"}`。
   - 正则表达式：`/.+namespace=\"(.*?)\".*/`。

- 在 变量值预览 区域，预览示例中包含多个 namespace，例如 `argocd`、`cpaas-system` 等。

2. 添加第二个变量：deployment，并引用之前创建的变量：

   - 查询设置：`kube_deployment_spec_replicas{namespace=~"$namespace"}`。
   - 正则表达式：`/.+deployment="(.*?)",.*/`。

3. 在当前仪表盘中添加一个图表，并引用之前添加的变量，例如：

   - 指标名称：计算组件下的 **pod** 内存使用量。
   - 键值对：`kind`：`Deployment`、`name`：`$deployment`、`namespace`：`$namespace`。

4. 添加图表并保存后，您可以在仪表盘首页查看相应的图表信息。

使用内置指标时的注意事项

> **WARNING**
>
> 以下指标使用自定义变量 `namespace`、`name` 和 `kind`，不支持多选或选择全部。
>
> - `namespace` 仅支持选择特定 namespace；
> - `name` 仅支持三种计算组件：`deployment`、`daemonset`、`statefulset`；
> - `kind` 仅支持指定以下类型之一：`Deployment`、`DaemonSet`、`StatefulSet`。

- `workload.cpu.utilization`
- `workload.memory.utilization`
- `workload.network.receive.bytes.rate`
- `workload.network.transmit.bytes.rate`
- `workload.gpu.utilization`
- `workload.gpu.memory.utilization`
- `workload.vgpu.utilization`
- `workload.vgpu.memory.utilization`

Perses 监控仪表板

目录

- 功能概述
- 先决条件
- 访问 Perses 仪表板
- 查看仪表板
  - 仪表板列表
  - 仪表板视图
- 创建和编辑仪表板
  - 创建仪表板
  - 使用仪表板编辑器
  - 添加图表
  - 添加变量
  - 编辑仪表板 JSON
- 图表类型
- 导入仪表板
- 管理仪表板
- 故障排除
  - 页面显示安装指南而不是仪表板
  - 仪表板没有数据
  - 某些内置仪表板显示 `xxx-pd` 等资源名称
- 相关操作

功能概述

Perses 监控仪表板在 Alauda Container Platform 4.4 中推出，是基于开源 Perses 渲染引擎的新型仪表板功能。您可以查看内置仪表板、创建自定义仪表板、导入仪表板 JSON，以及管理集群的指标图表。

Perses 监控仪表板与现有的 监控仪表板 功能并存。现有的 MonitorDashboard 条目和资源仍然可用。您可以在需要通过 Perses 仪表板条目进行渲染时，将现有的 MonitorDashboard 资源迁移到 PersesDashboard 资源。有关详细信息，请参阅将 MonitorDashboard 迁移到 Perses。

第一阶段侧重于指标仪表板：

- 支持常见的指标图表类型。
- 平台指标数据源固定，不会在 UI 中公开。
- 此阶段不支持日志、跟踪和性能分析可视化。

先决条件

使用 Perses 监控仪表板之前，请安装以下组件：

组件	安装位置	用途
Alauda Container Platform Dashboard	`global` 集群运行唯一的 Perses server 实例。工作负载集群会根据需要安装 operator，以提供 PersesDashboard CRD，但不会创建 Perses 实例。	提供 Perses 仪表板后端、渲染和编辑功能。
Alauda Container Platform Dashboard Essentials	仅限 `global` 集群	提供 Perses 仪表板使用的、支持 RBAC 的指标查询服务。

有关安装说明，请参阅安装。如果所选集群尚未安装 **Alauda Container Platform Dashboard**，Perses 仪表板页面将显示安装指南，而不是仪表板列表。

访问 **Perses** 仪表板

1. 登录容器平台管理视图。
2. 在左侧导航栏中，前往 运营中心 > 监控 > 仪表板 **(Perses)**。
3. 使用页面顶部的 集群选择器，选择要查看其仪表板的集群。

查看仪表板

仪表板列表

仪表板列表会显示您可以访问所选集群中的 Perses 仪表板。

项目	说明
列	列表显示 仪表板、文件夹和创建时间。
搜索和筛选	您可以按名称、文件夹或标签搜索或筛选仪表板。
内置仪表板	内置仪表板会标记为 内置。内置仪表板的 编辑和删除功能已禁用，但可以使用复制功能。
自定义仪表板	自定义仪表板支持 编辑、复制和删除。

仪表板视图

单击仪表板名称以打开仪表板视图。

在仪表板视图中，您可以：

- 查看按图表组组织的图表。
- 使用仪表板变量筛选图表数据。
- 选择时间范围和刷新间隔。
- 在仪表板支持下载时下载仪表板数据。
- 单击 `{}` ，以只读模式打开 仪表板 **JSON**。

- 在图表中打开 查询查看器，以只读模式查看 PromQL 查询、图例和最小步长。

由于平台指标数据源固定，查询查看器不会显示数据源信息。

对于内置仪表板，仪表板视图中的 编辑和设置功能已禁用。

创建和编辑仪表板

创建仪表板

1. 在 Perses 仪表板列表页面上，单击 创建。
2. 配置仪表板信息。

字段	必填	描述
名称	是	仪表板显示名称。
描述	否	仪表板描述。
文件夹	是	用于组织仪表板的文件夹。
标签	否	用于组织和搜索仪表板的标签。

3. 确认创建。平台将打开仪表板编辑器。

资源名称由后端生成。您只需在 UI 中提供仪表板显示名称。

使用仪表板编辑器

仪表板编辑器提供以下主要操作：

操作	描述
变量	添加或编辑仪表板变量。
添加图表	向仪表板添加图表。

操作	描述
添加图表组	添加图表组以组织图表。
保存	保存仪表板。
取消	退出编辑，不保存最新更改。

添加图表

1. 在仪表板编辑器中，单击 添加图表。
2. 在 添加图表 抽屉中，配置图表名称、组、描述和图表类型。
3. 在 查询 选项卡中配置查询。

字段	描述
查询语言	使用原生 PromQL。指标名称和标签支持自动补全。
查询类型	范围查询使用 **ACP Courier Query**，即时查询使用 **ACP Courier Instant Query**。Bar、Gauge 和 Stat 图表通常使用即时查询来显示当前值。
添加查询	向一个图表添加多个查询。

4. 根据需要配置其他选项卡，例如 常规设置、查询设置、链接和 **JSON**。
5. 检查预览并保存图表。

查询区域会自动使用平台指标数据源，不显示数据源选择器。

添加变量

在编辑器中使用 变量 添加仪表板变量。

变量类型	描述
列表	根据指标查询结果动态生成可选值。

变量类型	描述
文本	使用手动输入的文本。文本变量支持 常量设置。

编辑仪表板 **JSON**

在编辑模式下，单击 `{}` 以打开 编辑仪表板 **JSON**。您可以直接编辑仪表板定义。

图表类型

在此阶段，Perses 监控仪表板支持以下指标图表类型：

图表类型	使用场景
时序图	查看时序趋势。
柱状图	比较或排名。
仪表盘图	查看当前值或利用率。
统计图	突出显示单个值或关键指标。
饼图	查看比例和组成。
表格	查看结构化的多维数据。
散点图	查看分布和相关性。
直方图	查看直方图分布。
热力图	查看热度分布。
状态历史	查看随时间变化的状态。
Markdown	显示描述性文本或章节内容。

此阶段不支持日志、trace 和性能分析图表。

导入仪表板

1. 在仪表板列表页面，打开 创建 下拉菜单，然后选择 导入仪表板。
2. 配置导入信息。

字段	必填	描述
名称	是	仪表板显示名称。
描述	否	仪表板描述。
文件夹	是	用于整理仪表板的文件夹。
仪表板 **JSON**	是	Perses 或 Grafana 仪表板 JSON。您可以上传文件或粘贴 JSON 内容。

3. 确认导入。

可以直接导入 Perses JSON。导入时，Grafana JSON 会转换为 Perses 模型。此转换会丢失部分信息，仅保留当前版本支持的图表类型和函数。

管理仪表板

操作	描述
复制	创建现有仪表板的可编辑副本。内置仪表板也可以复制为自定义仪表板。
设置	更新仪表板名称、描述、文件夹和标签。
切换	打开对话框，按文件夹和搜索关键字切换到其他仪表板。
删除	删除自定义仪表板。无法删除内置仪表板。

故障排除

页面显示安装指南而不是仪表板

所选集群尚未安装 **Alauda Container Platform Dashboard**。请在该集群中安装此组件，然后重新打开 仪表板（**Perses**）。

仪表板没有数据

检查以下项目：

- **Alauda Container Platform Dashboard Essentials** 已安装并在 `global` 集群中运行。
- 所选集群中的监控数据采集和存储正常运行。
- 当前用户有权限查询目标集群、命名空间或项目的指标。

某些内置仪表板显示 `xxx-pd` 等资源名称

如果源仪表板不包含 `cpaas.io/display-name` 注释，某些迁移的内置仪表板可能会显示 Kubernetes 资源名称，而不是显示名称。这不会影响仪表板渲染或指标查询。

相关操作

- 将 MonitorDashboard 迁移到 Perses
- 安装
- 监控仪表板

Fleet 监控

目录

概述
前提条件
访问 Fleet 监控
内置监控面板
Fleet 监控概览
Fleet 监控项目配额
筛选数据
添加自定义 Fleet 监控监控面板
限制

概述

Fleet 监控为平台管理员提供全局多集群监控视图。它帮助您了解哪些集群已连接、监控数据是否为最新，以及整个集群群组是否存在资源容量或配额风险。

Fleet 监控不取代单集群监控。当您需要进行集群群组级治理、容量分析和长期趋势分析时，请使用 Fleet 监控。当您需要针对特定集群、命名空间、工作负载或指标进行详细故障排查时，请使用单集群监控。

前提条件

使用 Fleet 监控前，请确保满足以下要求：

- **Alauda Container Platform Fleet Monitoring Central Service** 已安装在 Global 集群上。
- Global 集群上存在 `FleetMonitoringHub` 资源。
- **Alauda Container Platform Fleet Monitoring Cluster Service** 已安装在您要纳入 Fleet 监控的每个集群上。
- 您要纳入的每个集群上都存在 `FleetMonitoringAgent` 资源。
- 已连接的集群由 Global 集群管理，并已启用 Monitoring 功能。
- 您具有查看 Fleet 监控页面和监控数据的权限。

有关启用步骤，请参阅 配置 Fleet 监控。

访问 **Fleet** 监控

要打开 Fleet 监控，请转到 **Platform** > **Observe** > **Fleet Monitoring**。

该页面显示默认的 Fleet 监控监控面板。使用 **Switch** 在内置 Fleet 监控监控面板和自定义 Fleet 监控监控面板之间切换。

在第一个版本中，Fleet 监控页面不提供 **Create**、**Import** 或图表编辑操作。若要添加自定义 Fleet 监控监控面板，请使用现有的 Monitoring Dashboard 资源工作流。有关更多信息，请参阅 添加自定义 Fleet 监控监控面板。

内置监控面板

Fleet 监控包含用于集群群组级监控的预设监控面板。

Fleet 监控概览

Fleet 监控概览监控面板是默认的预设监控面板。它帮助您回答以下问题：

- 哪些集群已连接到 Fleet 监控？

- 集群群组中有多少节点、Pod、项目、CPU 核心和内存资源？
- 集群群组当前的 CPU 和内存使用率及请求率是多少？
- 哪些节点的 CPU 或内存使用率较高？
- 集群群组级 CPU 和内存利用率趋势如何？

使用此监控面板执行日常集群群组健康检查和容量审查，并快速识别需要关注的集群。

监控面板包含以下信息：

区域	描述
集群群组清单	集群数量、节点数量、Pod 数量、项目数量、CPU 总量、内存总量、过期采集数量和活动告警数量。
集群群组利用率	CPU 使用率、CPU 请求率、内存使用率和内存请求率。
节点排名	已连接集群中按 CPU 使用率和内存使用率排名靠前的节点。
利用率趋势	CPU 利用率趋势和内存利用率趋势。
集群详细信息	集群级资源和利用率详细信息。

Fleet 监控项目配额

Fleet 监控项目配额监控面板是一个预设监控面板，可帮助您了解已连接集群中的项目配额分配和使用情况。

使用此监控面板查看项目配额分布，并识别存在配额分配或使用风险的项目。

监控面板包含以下信息：

区域	描述
定义和阈值	配额、已分配、已使用和使用率的定义，以及正常、高和接近上限使用率的阈值含义。

区域	描述
配额使用概览	项目数量、配额对象数量、高使用率对象数量、CPU 和内存使用率、配额总量、已分配量和已使用量。
配额分布	未分配、已分配但未使用和已占用资源中的 CPU 与内存配额分布。
配额使用排名	按 CPU 配额使用率和内存配额使用率排名靠前的项目。
项目配额详细信息	项目级配额分配、使用情况和风险详细信息。

此监控面板读取 Fleet 级汇总指标，这些指标每小时在 Global 集群上评估一次。新创建的项目或新连接的集群最多可能需要约一小时才会显示在此监控面板上。

筛选数据

Fleet 监控监控面板提供变量，帮助您将视图缩小到特定的集群集合或项目集合。

变量	描述
Cluster Label Key	选择用于筛选的集群标签键。
Cluster Label Value	为所选集群标签键选择一个值。
Cluster	选择一个或多个集群。可通过集群标签变量缩小此列表。
Project	选择一个或多个项目。此变量在项目配额监控面板上可用。
Quota Resource Type	在 `limits` 和 `requests` 之间切换。此变量在项目配额监控面板上可用。
时间范围	控制监控面板查询的时间范围。

Cluster 变量可以列出所有已知集群。**Connected Clusters** 指标仅统计实际写入 Fleet 监控数据的集群。因此，集群列表和已连接集群数量可能不同。

添加自定义 **Fleet** 监控监控面板

您可以使用现有的 Monitoring Dashboard 资源工作流，将自定义多集群监控面板添加到 Fleet 监控中。

请使用以下任一方法：

- 在现有的 Dashboard 页面中，在 Global 集群上创建监控面板，并通过页面操作添加 `fleet-monitoring` 标签。
- 将 `MonitorDashboard` YAML 资源提交到 Global 集群上安装 **Alauda Container Platform Fleet Monitoring Central Service** 的命名空间。确保资源具有 `cpaas.io/dashboard.tag.fleet-monitoring: "true"` 标签。此标签会将 `fleet-monitoring` 标签添加到监控面板。

示例：

```yaml
apiVersion: ait.alauda.io/v1alpha2
kind: MonitorDashboard
metadata:
  name: my-fleet-dashboard
  namespace: <fleet-monitoring-namespace>
  labels:
    cpaas.io/dashboard.folder: fleet
    cpaas.io/dashboard.tag.fleet-monitoring: "true"
    cpaas.io/dashboard.tag.multi-cluster: "true"
    cpaas.io/published: "true"
spec:
  body: {}
```

将 `<fleet-monitoring-namespace>` 替换为 Global 集群上安装 **Alauda Container Platform Fleet Monitoring Central Service** 的命名空间。

系统不会验证自定义监控面板是否使用 Fleet 监控指标。监控面板作者必须确保监控面板使用在 Fleet 监控上下文中有效的数据源、指标和变量。

对于多集群监控面板，请使用 `cluster` 标签标识源集群。如果原始指标已经具有 `cluster` 标签，Fleet 监控会将原始值保留为 `exported_cluster`。

在当前平台查询路径中，直接查询 Fleet 指标时，Fleet 监控监控面板查询应显式包含 `vmcluster=~".*"`。如果没有此选择器，监控查询代理可能会将查询范围缩小到 Global 监控后端，并且不会返回已连接集群的 Fleet 指标数据。

示例：

```
avg_over_time(fleet:node:node_load15:avg{vmcluster=~".*",cluster="g1-c1"}
[1h])
```

```
last_over_time(fleet:node:node_load15:avg:avg_over_time_1h{vmcluster=~".
*",cluster="g1-c1"}[2h])
```

限制

Fleet 监控第一个版本存在以下限制：

- Fleet 监控不提供专用的多集群告警页面。
- Fleet 监控数据可由现有告警机制使用，但告警规则仍通过现有告警工作流进行管理。
- Fleet 监控不提供供普通用户自助配置所采集指标或记录规则的页面。
- Fleet 监控不会回填历史数据。只有在启用 Fleet 监控后才会采集数据。
- Fleet 监控不适用于秒级故障排查。请使用单集群监控进行详细故障排查。

探针管理

目录

功能概述
黑盒监控
前提条件
操作流程
黑盒告警
前提条件
操作流程
自定义 BlackboxExporter 监控模块
操作流程
通过 CLI 创建黑盒监控项和告警
前提条件
操作流程
参考信息

功能概述

平台的探针功能基于 Blackbox Exporter 实现，允许用户通过 ICMP、TCP 或 HTTP 对网络进行探测，以快速定位平台上发生的故障。

与依赖平台已有的各种监控指标的白盒监控系统不同，黑盒监控关注的是结果。当白盒监控无法覆盖影响服务可用性的所有因素时，黑盒监控能够快速发现故障并基于故障触发告警。例如，当某个 API 接口异常时，黑盒监控可以及时将此类问题暴露给用户。

> **WARNING**
>
> 探针功能不支持在内核版本 3.10 及以下的节点上使用 ICMP 探测 IPv6 地址。若需使用此场景，请将节点内核版本升级至 3.11 及以上。

黑盒监控

创建黑盒监控项时，可以选择 ICMP、TCP 或 HTTP 探测方式，定期探测指定的目标地址。

前提条件

监控组件必须已安装在集群中，且监控组件运行正常。

操作流程

1. 在左侧导航栏点击 运维中心 > 监控 > 黑盒监控。
   提示：黑盒监控为集群级功能，点击顶部导航栏可切换集群。
2. 点击 创建黑盒监控项。
3. 按照以下说明配置相关参数。

参数	说明
探测方式	**ICMP**：通过 ping 输入的 目标地址（域名或 IP）探测服务器可用性。 **TCP**：通过监听 目标地址 中指定的 `<域名:端口>` 或 `<IP:端口>` 探测主机业务端口。 **HTTP**：探测输入的 目标地址 URL，检查网站连通性。 提示：HTTP 探测方式默认仅支持 GET 请求，若需 POST 请求，请参考 自定义 BlackboxExporter 监控模块。

参数	说明
探测间隔	探测的时间间隔。
目标地址	探测目标地址，最长支持 128 字符。 不同探测方式的输入格式如下： **ICMP**：域名或 IP 地址，例如 `10.165.94.31`。 **TCP**：`<域名:端口>` 或 `<IP:端口>`，例如 `172.19.155.133:8765`。 **HTTP**：以 http 或 https 开头的 URL，例如 `http://alauda.cn/`。

4. 点击 创建。
   创建成功后，可在列表页实时查看最新探测结果，并基于黑盒监控项创建告警策略。当检测到故障时，系统会自动触发告警，通知相关人员进行处理。

> **WARNING**
>
> 黑盒监控项创建成功后，系统需要约 5 分钟同步配置。同步期间不会进行探测，且无法查看探测结果。

黑盒告警

前提条件

- 监控组件必须已安装在集群中，且监控组件运行正常。
- 黑盒监控项已成功创建，且系统已完成配置同步，黑盒监控页面可见探测结果。

操作流程

1. 在左侧导航栏点击 运维中心 > 告警 > 告警策略。
   提示：告警策略为集群级功能，点击顶部导航栏可切换集群。请确保切换至刚配置黑盒监控项的集群。
2. 点击 创建告警策略。
3. 按照以下说明配置相关参数；更多参数信息请参考 创建告警策略。

- 告警类型：请选择 资源告警。
- 资源类型：请选择 集群。
- 点击 添加告警规则。
  - 告警类型：请选择 黑盒告警。
  - 黑盒监控项：请选择所需的黑盒监控项。
  - 指标名称：请选择希望监控并触发告警的指标。平台当前支持的指标为 **Connectivity** 和 **HTTP Status Code**。
    - **Connectivity**：所有黑盒监控项均可选择该指标，触发条件为 **“!= 1”** 表示黑盒监控项的目标地址不可达。
    - **HTTP Status Code**：仅当所选黑盒监控项的探测方式为 **HTTP** 时可选。可输入三位正整数作为触发条件值，例如设置为 **“> 299”** 表示响应码为 3XX、4XX 或 5XX 时触发告警。
  - 通知策略：请选择预先配置的通知策略。
  - 点击 添加。

4. 点击 创建。提交告警策略后，可在告警策略列表中查看该策略。

自定义 **BlackboxExporter** 监控模块

您还可以通过向 BlackboxExporter 配置文件中添加自定义监控模块，增强黑盒监控功能。例如，添加 **http_post_2xx** 模块后，当黑盒监控的探测方式设置为 `HTTP` 时，即可探测 POST 请求方法的状态。

黑盒监控的配置文件位于集群中 Prometheus 组件安装的命名空间内，默认名称为 `cpaas-monitor-prometheus-blackbox-exporter`，可根据实际名称进行修改。

> **TIP**
>
> 该配置文件为与命名空间相关的 ConfigMap 资源，可通过平台的管理功能 集群管理 **>** 资源管理 快速查看和更新。

操作流程

1. 通过向配置文件中 **key** `modules` 添加自定义监控模块，更新黑盒监控配置。

   以添加 **http_post_2xx** 模块为例：

```
blackbox.yaml: |
  modules:
    http_post_2xx:                    # HTTP POST 探测模块
      prober: http
      timeout: 5s
      http:
        method: POST                  # 探测请求方法
        headers:
          Content-Type: application/json
        body: '{}'                    # 探测时携带的请求体内容
```

   黑盒监控配置文件的完整 YAML 示例，请参考 参考信息。

2. 通过以下任一方式激活配置。

   - 删除 Blackbox Exporter 组件 **cpaas-monitor-prometheus-blackbox-exporter** 的 Pod，重启组件。
   - 执行以下命令调用 reload API，刷新配置文件：

```
curl -X POST -v <Pod IP>:9115/-/reload
```

通过 CLI 创建黑盒监控项和告警

前提条件

- 已配置通知策略（若需告警自动通知）。
- 目标集群已安装监控组件。

操作流程

1. 新建 YAML 配置文件，命名为 `example-probe.yaml`。

2. 在 YAML 文件中添加 PrometheusRule 资源并提交。以下示例创建名为 `prometheus-liveness` 的新告警策略：

```yaml
apiVersion: monitoring.coreos.com/v1
kind: Probe
metadata:
  annotations:
    cpaas.io/creator: jhshi@alauda.io # 探针项创建者
    cpaas.io/updated-at: '2021-05-25T08:08:45Z' # 探针项最后更新时间
    cpaas.io/display-name: 'Prometheus prober' # 探针项描述
  creationTimestamp: '2021-05-10T02:04:33Z' # 探针项创建时间
  labels:
    prometheus: kube-prometheus # 用于 prometheus 名称的标签值
  name: prometheus-liveness # 探针项名称
  namespace: cpaas-system # prometheus 命名空间
spec:
  jobName: prometheus-liveness # 探针项名称
  prober:
    url: cpaas-monitor-prometheus-blackbox-exporter:9115 # Blackbox 指标
URL, 从特性中获取
  module: http_2xx # 探针项模块名称
  targets:
    staticConfig:
      static:
        - http://www.prometheus.io # 探针目标地址
      labels:
        module: http_2xx # 探针项模块名称
        prober: http # 探测方式
  interval: 30s # 探测间隔
  scrapeTimeout: 10s
```

3. 新建 YAML 配置文件，命名为 `example-alerting-rule.yaml`。

4. 在 YAML 文件中添加 PrometheusRule 资源并提交。以下示例创建名为 `policy` 的新告警策略：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  annotations:
    alert.cpaas.io/cluster: global # 告警所属集群名称
    alert.cpaas.io/kind: Cluster # 资源类型
    alert.cpaas.io/name: global # 黑盒监控项所在集群名称
    alert.cpaas.io/namespace: cpaas-system # prometheus 命名空间，保持默认
    alert.cpaas.io/notifications: '["test"]'
    alert.cpaas.io/repeat-config: '{"Critical":"never","High":"5m","Med
ium":"5m","Low":"5m"}'
    alert.cpaas.io/rules.description: '{}'
    alert.cpaas.io/rules.disabled: '[]'
    alert.cpaas.io/subkind: ''
    cpaas.io/description: ''
    cpaas.io/display-name: policy # 告警策略显示名称
  labels:
    alert.cpaas.io/owner: System
    alert.cpaas.io/project: cpaas-system
    cpaas.io/source: Platform
    prometheus: kube-prometheus
    rule.cpaas.io/cluster: global
    rule.cpaas.io/name: policy
    rule.cpaas.io/namespace: cpaas-system
  name: policy
  namespace: cpaas-system
spec:
  groups:
    - name: general # 告警规则名称
      rules:
        - alert: cluster.blackbox.probe.success-y97ah-9833444d918cab96c
43e9ab6efc172cf
          annotations:
            alert_current_value: '{{ $value }}' # 通知时的当前值，保持默认
          expr:
            max by (job, instance) (probe_success{job=~"test",
            instance=~"https://demo.at-servicecenter.com/"})!=1
            # 连通性告警场景，务必修改黑盒监控项名称和目标地址
          for: 30s # 持续时间
          labels:
            alert_cluster: global # 告警所属集群名称
            alert_for: 30s # 持续时间
            alert_indicator: cluster.blackbox.probe.success # 保持不变
```

```
            alert_indicator_aggregate_range: '0' # 保持不变
            alert_indicator_blackbox_instance: https://demo.at-servicec
enter.com/ # 黑盒监控目标地址
            alert_indicator_blackbox_name: test # 黑盒监控项名称
            alert_indicator_comparison: '!=' # 连通性告警保持配置不变
            alert_indicator_query: '' # 日志告警使用，无需配置
            alert_indicator_threshold: '1' # 告警规则阈值，1 表示连通性，保
持不变
            alert_indicator_unit: '' # 告警规则指标单位
            alert_involved_object_kind: Cluster # 黑盒告警保持不变
            alert_involved_object_name: global # 黑盒监控项所在集群
            alert_involved_object_namespace: '' # 告警规则所属对象命名空间
            alert_name: cluster.blackbox.probe.success-y97ah # 告警规则名
称
            alert_namespace: cpaas-system # 告警规则所在命名空间
            alert_project: cpaas-system # 告警规则所属对象项目名称
            alert_resource: policy # 告警规则所在告警策略名称
            alert_source: Platform # 告警规则数据类型：Platform-平台数据，Bu
siness-业务数据
            severity: High # 告警规则级别：Critical-灾难，High-严重，Medium
-警告，Low-提示
        - alert: cluster.blackbox.http.status.code-235el-99b0095b6b6669
415043e14ae84f43bc
          annotations:
            alert_current_value: '{{ $value }}'
            alert_notifications: '["message"]'
          expr:
            max by(job, instance) (probe_http_status_code{job=~"test",
            instance=~"https://demo.at-servicecenter.com/"})>200
            # HTTP 状态码告警场景，务必修改黑盒监控项名称和目标地址
          for: 30s
          labels:
            alert_cluster: global
            alert_for: 30s
            alert_indicator: cluster.blackbox.http.status.code
            alert_indicator_aggregate_range: '0'
            alert_indicator_blackbox_instance: https://demo.at-servicec
enter.com/
            alert_indicator_blackbox_name: test
            alert_indicator_comparison: '>'
            alert_indicator_query: ''
            alert_indicator_threshold: '299' # 告警规则阈值，HTTP 状态码告
警场景应为三位数，例如大于 299（3XX、4XX、5XX）表示错误
            alert_indicator_unit: ''
```

```
alert_involved_object_kind: Cluster
alert_involved_object_name: global
alert_involved_object_namespace: ''
alert_involved_object_options: Single
alert_name: cluster.blackbox.http.status.code-235el
alert_namespace: cpaas-system
alert_project: cpaas-system
alert_resource: policy33
alert_source: Platform
severity: High
```

参考信息

黑盒监控 YAML 配置文件的完整示例如下：

```
apiVersion: v1
data:
  blackbox.yaml: |
    modules:
      http_2xx_example:               # HTTP 探测示例
        prober: http
        timeout: 5s                   # 探测超时时间
        http:
          valid_http_versions: ["HTTP/1.1", "HTTP/2.0"]
# 返回信息中的版本，通常默认
          valid_status_codes: []  # 默认为 2xx                          # 有效
响应码范围，返回码在此范围内视为探测成功
          method: GET                 # 请求方法
          headers:                    # 请求头
            Host: vhost.example.com
            Accept-Language: en-US
            Origin: example.com
          no_follow_redirects: false  # 是否允许重定向
          fail_if_ssl: false
          fail_if_not_ssl: false
          fail_if_body_matches_regexp:
            - "Could not connect to database"
          fail_if_body_not_matches_regexp:
            - "Download the latest version here"
          fail_if_header_matches: # 验证未设置 Cookie
            - header: Set-Cookie
              allow_missing: true
              regexp: '.*'
          fail_if_header_not_matches:
            - header: Access-Control-Allow-Origin
              regexp: '(\*|example\.com)'
          tls_config:                  # https 请求的 TLS 配置
            insecure_skip_verify: false
          preferred_ip_protocol: "ip4" # 默认为 "ip6"                  # 优
先使用的 IP 协议版本
          ip_protocol_fallback: false  # 不回退到 "ip6"
      http_post_2xx:                   # 带请求体的 HTTP 探测示例
        prober: http
        timeout: 5s
        http:
          method: POST                 # 探测请求方法
          headers:
            Content-Type: application/json
```

实用指南

Prometheus 监控数据的备份与恢

功能概述
使用场景
前提条件
操作步骤
操作结果
了解更多
后续操作

VictoriaMetrics 监控数据备份与恢

功能概述
使用场景
前提条件
操作步骤
操作结果
了解更多
后续操作

从自定义命名

功能概述
使用场景
前提条件
操作步骤
操作结果
了解更多
后续操作

规划用于监控的基础设施节点

概述
支持的配置方法
配置放置前
在控制台中配置放置
在 YAML 中配置放置
故障排除
了解更多
后续操作

配置 Fleet Monitoring

概述
开始之前
推送 Fleet Monitoring Operator 软件包
在 Global 集群上启用 Fleet Monitoring
将集群连接到 Fleet Monitoring
升级 Fleet Monitoring
配置采集间隔
配置自定义指标和记录规则
验证数据新鲜度
故障排除

将 MonitorD

概述
前提条件
迁移监控面板
转换规则
导入 Grafana J
迁移后
相关操作

了解更多

Prometheus 监控数据的备份与恢复

目录

功能概述
使用场景
前提条件
操作步骤
备份数据
方法 1：备份存储目录（推荐）
方法 2：快照备份
恢复数据
操作结果
了解更多
TSDB 数据格式说明
数据备份注意事项
后续操作

功能概述

Prometheus 监控数据以 TSDB（时间序列数据库）格式存储，支持备份和恢复功能。监控数据存储在 Prometheus 容器内的指定路径中（由配置 `storage.tsdb.path` 指定，默认值为 `/prometheus`）。

```
template:
  spec:
    containers:
      - args:
          - '--storage.tsdb.path=/prometheus' # Directory for storing mon
itoring data in the Prometheus container
```

使用场景

- 系统迁移期间保留历史监控数据
- 防止意外事件导致数据丢失
- 将监控数据迁移到新的 Prometheus 实例

前提条件

- 已安装 ACP Basic Monitoring 插件（计算组件的名称为 `prometheus-kube-prometheus-0`，类型为 `StatefulSet`）
- 拥有集群管理员权限
- 确保目标存储位置有足够的存储空间

操作步骤

备份数据

开始备份前，请注意：Prometheus 存储监控数据时，会先将采集到的数据放入缓存，然后定期写入存储目录。以下备份方法以存储目录作为数据源，因此不包含备份时缓存中的数据。

方法 1：备份存储目录（推荐）

1. 使用 kubectl cp 命令进行备份：

```
kubectl cp -n cpaas-system prometheus-kube-prometheus-0-0:/prometheus -c prometheus <target storage path>
```

2. 从存储后端备份（根据安装时选择的存储类型）：

- **LocalVolume**：从节点上的 `<storage path>/prometheus` 目录复制，其中 `<storage path>` 是插件安装时配置的路径：默认值为 `/cpaas/monitoring`，在 Alauda OS 上为 `/var/cpaas/monitoring`
- **PV**：从 PV 挂载目录复制（建议将 PV 的 **persistentVolumeReclaimPolicy** 设置为 `Retain`）
- **StorageClass**：从 PV 挂载目录复制

方法 2：快照备份

1. 启用 Admin API：

```
kubectl edit -n cpaas-system prometheus kube-prometheus-0
```

添加配置：

```
spec:
  enableAdminAPI: true
```

注意：更新并保存配置后，Prometheus Pod（Pod 名称：prometheus-kube-prometheus-0-0）将会重启。请等待所有 Pod 状态变为 Running 后，再执行后续操作。

2. 创建快照：

```
curl -XPOST <Prometheus Pod IP>:9090/api/v1/admin/tsdb/snapshot
```

恢复数据

1. 将备份数据复制到 Prometheus 容器：

```
kubectl cp ./prometheus-backup cpaas-system/prometheus-kube-prometheus-0-0:/prometheus/
```

2. 将数据移至指定目录：

```
kubectl exec -it -n cpaas-system prometheus-kube-prometheus-0-0 -c prometheus sh
mv /prometheus/prometheus-backup/* /prometheus/
```

快捷方式：如果插件安装时使用的存储类型为 **LocalVolume**，只需将备份数据直接复制到安装插件的节点上的 `<storage path>/prometheus/prometheus-db/` 目录即可。`<storage path>` 是安装时配置的路径：默认值为 `/cpaas/monitoring`，在 Alauda OS 上为 `/var/cpaas/monitoring`。

操作结果

- 备份完成后，可以在目标存储路径看到完整的 TSDB 格式监控数据
- 恢复完成后，Prometheus 将自动加载历史监控数据

了解更多

TSDB 数据格式说明

TSDB 格式数据结构示例：

```
├── 01FXP317QBANGAX1XQAXCJK9DB
│   ├── chunks
│   │   └── 000001
│   ├── index
│   ├── meta.json
│   └── tombstones
├── chunks_head
│   ├── 000022
│   └── 000023
├── queries.active
└── wal
    ├── 00000020
    ├── 00000021
    ├── 00000022
    ├── 00000023
    └── checkpoint.00000019
        └── 00000000
```

数据备份注意事项

- 备份数据不包括备份时的缓存数据
- 建议定期执行数据备份
- 使用 PV 存储时，建议将 **persistentVolumeReclaimPolicy** 设置为 `Retain`

后续操作

- 验证监控数据是否已正确恢复
- 定期安排数据备份计划
- 如果使用快照备份方式，可以选择禁用 Admin API

VictoriaMetrics 监控数据备份与恢复

目录

功能概述
使用场景
前提条件
操作步骤
1. 确认存储路径
2. 执行数据备份
3. 执行数据恢复
操作结果
了解更多
后续操作

功能概述

VictoriaMetrics 监控数据备份与恢复功能支持定期备份监控数据，并在需要时恢复数据，确保监控数据的安全性和可用性。

使用场景

- 定期备份监控数据，防止数据丢失
- 系统迁移期间迁移数据
- 灾难恢复
- 重建测试环境数据

前提条件

- 集群中已安装 ACP Advanced Monitoring 插件
- 确保有足够的备份存储空间
- 具有访问 VictoriaMetrics 存储路径的权限

操作步骤

1. 确认存储路径

VictoriaMetrics 的监控数据存储在容器的指定路径中，该路径由 `-storageDataPath` 参数指定，默认为 `/vm-data`。

配置示例：

```
spec:
  template:
    spec:
      containers:
        - args:
            - '-storageDataPath=/vm-data'
```

注意：ACP Advanced Monitoring 插件中的计算组件名称为 `vmstorage-cluster`，类型为 `StatefulSet`。

2. 执行数据备份

使用 vmbackup 工具备份数据；详细操作请参阅 vmbackup 官方文档↗。

3. 执行数据恢复

使用 vmrestore 工具恢复备份数据；详细操作请参阅 vmrestore 官方文档↗。

操作结果

完成备份后，您将获得完整的监控数据备份文件。执行恢复操作后，监控数据将恢复到备份时的状态。

了解更多

- VictoriaMetrics 官方文档↗
- 数据备份最佳实践↗
- 数据恢复故障排除↗

后续操作

- 验证备份数据的完整性
- 设置定期备份计划
- 定期测试恢复流程
- 监控备份任务的执行状态

从自定义命名的网络接口收集网络数据

目录

功能概述
使用场景
前提条件
操作步骤
操作结果
了解更多
后续操作

功能概述

通过修改监控组件的配置，平台支持从自定义命名的网络接口收集网络数据，使您能够在监控页面查看这些接口的网络流量信息。

使用场景

当您的节点使用自定义命名的网络接口（不遵循 `eth.|en.|wl.*|ww.*` 命名约定）且需要在平台中监控这些接口的网络流量数据时，适用于此功能。

前提条件

- 已创建一个业务集群
- 您拥有平台管理员权限
- 已知自定义网络接口的命名规则

操作步骤

1. 点击平台顶部导航栏中的 CLI 工具图标。
2. 点击 **global**。
3. 在 `global` 集群中，查找与您的业务集群对应的 moduleinfo 资源名称：

   ```
   kubectl get moduleinfo | grep -E 'prometheus|victoriametrics'
   ```

   示例输出：

   ```
   global-6448ef7f7e5e3924c1629fad826372e7     global     prometheus
   prometheus                          Running   v3.15.0-zz231204040711-9d
   1fc12474c2   v3.15.0-zz231204040711-9d1fc12474c2   v3.15.0-zz2312040407
   11-9d1fc12474c2
   ovn-0954f21f0359720e8c115804376b3e7e        ovn        prometheus
   prometheus                          Running   v3.15.0-zz231204040711-9d
   1fc12474c2   v3.15.0-zz231204040711-9d1fc12474c2   v3.15.0-zz2312040407
   11-9d1fc12474c2
   ```

4. 编辑业务集群的 moduleinfo 资源：

   ```
   kubectl edit moduleinfo <moduleinfo resource name of the workload clust
   er>
   ```

5. 添加或修改 valuesOverride 字段：

```
spec:
  valuesOverride:# If this field does not exist, you need to add the va
luesOverride field under spec along with the parameters below
    ait/chart-cpaas-monitor:
      global:
        indicator:
          networkDevice: eth.*|em.*|en.*|wl.*|ww.*|[A-Z].*i|custom_inte
rface # Replace custom_interface with the custom regular expression to
ensure correct matching of the network interface name
```

6. 等待 10 分钟后，检查节点监控页面上的网络相关图表，确保更改已生效。

操作结果

配置生效后，您可以在平台监控页面查看自定义命名网络接口的以下数据：

- 网络流量数据
- 网络吞吐量
- 网络包统计信息

了解更多

- 有关网络监控的更多信息，请参阅 Monitoring Architecture Documentation

后续操作

- 监控自定义网络接口的性能指标
- 基于监控数据设置告警规则

规划用于监控的基础设施节点

目录

概述
支持的配置方法
配置放置前
在控制台中配置放置
Prometheus
VictoriaMetrics
在 YAML 中配置放置
Prometheus
VictoriaMetrics
故障排除
Monitoring 工作负载仍调度在通用节点上
Monitoring 工作负载无法调度到选定的基础设施节点上
了解更多
后续操作

概述

为 Monitoring 插件规划并配置基础设施节点。使用插件配置将 Monitoring 工作负载放置在基础设施节点上，而不是在安装后修补生成的工作负载。

支持的配置方法

- 对于 **ACP Basic Monitoring**，可以在控制台中通过 **Advanced Configuration** 配置放置，也可以在 Installation 中通过 YAML 配置 `spec.config.components.nodeSelector` 和 `spec.config.components.tolerations`。
- 对于 **ACP Advanced Monitoring**，可以在控制台中通过 **Advanced Configuration** 配置放置，也可以在 Installation 中通过 YAML 配置 `spec.config.components.nodeSelector` 和 `spec.config.components.tolerations`。

不要将修补生成的 Deployments、StatefulSets 或其他由插件管理的工作负载作为放置 Monitoring 工作负载的标准方式。

配置放置前

在配置放置之前，请确保满足以下条件：

- 根据 Infra Nodes 规划基础设施节点。
- 确认存储使用的是 `LocalVolume`，还是使用 `spec.nodeAffinity` 的其他持久卷。
- 确保所选基础设施节点同时满足调度规则和存储放置约束。

在控制台中配置放置

Prometheus

从控制台安装或升级 **ACP Basic Monitoring** 时，展开 **Advanced Configuration** 并配置以下字段：

控制台字段	描述
Node Selectors	为 Prometheus 工作负载设置插件级别的节点选择器规则。
Node Tolerations	为 Prometheus 工作负载设置插件级别的容忍规则。

VictoriaMetrics

从控制台安装或升级 **ACP Advanced Monitoring** 时，展开 **Advanced Configuration** 并配置以下字段：

控制台字段	描述
Node Selectors	为 VictoriaMetrics 工作负载设置插件级别的节点选择器规则。
Node Tolerations	为 VictoriaMetrics 工作负载设置插件级别的容忍规则。

在 **YAML** 中配置放置

Prometheus

如果希望 Prometheus 插件工作负载在专用基础设施节点上运行，请在安装或升级期间配置插件级别的调度规则。

示例：

```
spec:
  config:
    components:
      nodeSelector:
        - key: kubernetes.io/os
          value: linux
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/infra
          operator: Exists
```

VictoriaMetrics

如果希望 VictoriaMetrics 插件工作负载在专用基础设施节点上运行，请在安装或升级期间配置插件级别的调度规则。

示例：

```
spec:
  config:
    components:
      nodeSelector:
        - key: kubernetes.io/os
          value: linux
      tolerations:
        - effect: NoSchedule
          key: node-role.kubernetes.io/infra
          operator: Exists
```

当 `storage.type` 为 `LocalVolume` 时，可以选择一个或多个节点。请确保每个选定的存储节点也符合配置的节点选择器规则。

故障排除

Monitoring 工作负载仍调度在通用节点上

检查以下事项：

- 目标节点具有预期的标签。
- 配置的容忍度与基础设施节点上的污点相匹配。
- 插件已使用最新的调度配置进行升级或重新应用。

Monitoring 工作负载无法调度到选定的基础设施节点上

此问题通常表明选定的节点不满足一项或多项调度或存储约束。

常见原因：

- 基础设施节点没有 `nodeSelector` 引用的标签。
- 基础设施节点具有未被配置的容忍度覆盖的污点。
- 选定的 `LocalVolume` 节点或 PV `nodeAffinity` 规则指向基础设施节点组之外的节点。

了解更多

- 安装
- 监控组件容量规划

后续操作

- 验证 Monitoring 工作负载是否运行在预期的基础设施节点上。
- 检查选定的基础设施节点是否仍满足容量规划目标。

配置 Fleet Monitoring

目录

概述

开始之前

推送 Fleet Monitoring Operator 软件包

在 Global 集群上启用 Fleet Monitoring

将集群连接到 Fleet Monitoring

升级 Fleet Monitoring

- 兼容性矩阵
- 升级路径指南
- 升级操作步骤

配置采集间隔

配置自定义指标和记录规则

- 确定是否仅需要 Agent 侧规则，还是同时需要 Agent 侧和 Hub 侧规则
- 配置已连接集群
- 验证已连接集群的渲染结果
- 添加自定义 Hub 汇总规则
- 验证 Hub 侧汇总
- 查询自定义 Fleet 指标
- 常见错误

验证数据新鲜度

故障排除

未显示 Fleet Monitoring 数据

Connected Clusters 中缺少集群

自定义监控面板未显示在 Fleet Monitoring 中

监控面板中缺少新集群或项目

数据新鲜度异常

了解更多

概述

Fleet Monitoring 使用两项服务：

- **Alauda Container Platform Fleet Monitoring Central Service** 在 global 集群上运行，并启用 Fleet Monitoring 的 Global 侧功能。
- **Alauda Container Platform Fleet Monitoring Cluster Service** 在要纳入 Fleet Monitoring 的每个集群上运行。

启用 Fleet Monitoring 后，已连接的集群会将 fleet 级指标写入 Global VictoriaMetrics 存储。内置 Fleet Monitoring 监控面板使用这些指标显示 fleet 健康状况、资源使用情况、数据新鲜度和项目配额使用情况。

开始之前

配置 Fleet Monitoring 前，请确保满足以下要求：

- 你具有平台管理员权限，或具有安装 Operator 和创建 Fleet Monitoring 资源所需的权限。
- Global 集群具有可用的 Fleet Monitoring 数据 remote write 端点。支持带 write 端点的 VictoriaMetrics。如果 Monitoring 功能公开的端点接受 remote write 流量，也可以使用基于 Prometheus 或 Thanos 的 Monitoring。
- 要连接的每个集群都由 Global 集群管理。
- 要连接的每个集群都已启用 Monitoring 功能。
- Fleet Monitoring Operator 软件包已推送到所需集群，或已在 OperatorHub 中可用。Fleet Monitoring 以 Agnostic Operator 的形式交付，默认不包含在平台安装中。

- New Web Console 插件部署功能可用。Fleet Monitoring Operator 通过 New Web Console OperatorHub 工作流部署。有关更多信息，请参阅安装 New Web Console。

推送 **Fleet Monitoring Operator** 软件包

在从 OperatorHub 安装 Fleet Monitoring 前，使用 `violet` 推送 Fleet Monitoring Operator 软件包。

将 **Alauda Container Platform Fleet Monitoring Central Service** 推送到 Global 集群：

```
violet push <path/to/fleet-monitoring-central-service-operator-package> \
  --platform-address "https://<your-platform-domain>" \
  --platform-token "<platform_token>" \
  --clusters "global"
```

将 **Alauda Container Platform Fleet Monitoring Cluster Service** 推送到要安装 Cluster Service 的每个集群。如果还要将 Global 集群纳入 Fleet Monitoring 数据，请在集群列表中包含 `global`：

```
violet push <path/to/fleet-monitoring-cluster-service-operator-package> \
  --platform-address "https://<your-platform-domain>" \
  --platform-token "<platform_token>" \
  --clusters "global,<workload-cluster-1>,<workload-cluster-2>"
```

推送软件包后，相应的 Operator 会显示在所选集群的 **Marketplace** > **OperatorHub** 中。有关 `violet push` 的更多信息，请参阅上架软件包。

在 **Global** 集群上启用 **Fleet Monitoring**

1. 转到 **Administrator**。
2. 在左侧导航栏中，单击 **Marketplace** > **OperatorHub**。
3. 在页面顶部，选择 `global` 集群。
4. 搜索 **Alauda Container Platform Fleet Monitoring Central Service**。

5. 如果 Operator 状态不是 **Installed**，单击 **Install**。除非环境需要不同的 channel、namespace 或升级策略，否则保留默认安装配置。

参数	推荐配置
Channel	使用 Operator 软件包提供的默认 channel。
Installation Mode	`Cluster`。Operator 管理该集群的 Fleet Monitoring 资源。
Installation Place	使用 Operator 软件包提供的推荐 namespace。
Upgrade Strategy	`Manual`，除非平台升级策略要求自动升级。

6. 在 OperatorHub 中确认 **Alauda Container Platform Fleet Monitoring Central Service** 的状态为 **Installed**。
   如果选择 `Manual` 升级策略，且 OperatorHub 显示待处理的安装计划，请批准该安装计划以完成安装。

7. 确认 Global 集群具有可用的 VictoriaMetrics write 端点。
   Fleet Monitoring 使用 Global VictoriaMetrics write 端点接收来自已连接集群的数据。如果 Global 集群只有 Prometheus 或 Thanos Query 可用，则已连接集群无法将 Fleet Monitoring 数据写入 Global 集群。

8. 在 Global 集群上创建 `FleetMonitoringHub` 资源。

```yaml
apiVersion: monitoring.alauda.io/v1alpha1
kind: FleetMonitoringHub
metadata:
  name: fleet-monitoring-hub
spec: {}
```

`FleetMonitoringHub` 是集群范围的资源。请勿设置 `metadata.namespace`。

字段	必需	描述
`metadata.name`	是	必须为 `fleet-monitoring-hub`。

字段	必需	描述
`metadata.namespace`	否	请勿设置此字段。`FleetMonitoringHub` 是集群范围的资源。
`spec`	是	使用空对象 `{}`。创建资源会启用 Fleet Monitoring 的 Global 侧功能。

9. 验证 `FleetMonitoringHub` 状态。

```
kubectl get fleetmonitoringhub fleet-monitoring-hub -o yaml
```

检查以下条件是否已就绪：

条件	描述
`ConfigurationReady`	Global 存储和数据库信息可用。
`DashboardsReady`	已应用内置 Fleet Monitoring 监控面板。
`GlobalRulesReady`	已应用 Global 记录规则。

也可以检查阶段：

```
kubectl get fleetmonitoringhub fleet-monitoring-hub -o jsonpath='{.status.phase}{"\n"}'
```

预期阶段为 `Ready`。

将集群连接到 **Fleet Monitoring**

在要纳入 Fleet Monitoring 的每个集群上重复执行以下步骤。

1. 转到 **Administrator**。
2. 在左侧导航栏中，单击 **Marketplace** > **OperatorHub**。

3. 在页面顶部，选择目标集群。
4. 搜索 **Alauda Container Platform Fleet Monitoring Cluster Service**。
5. 如果 Operator 状态不是 **Installed**，单击 **Install**。除非环境需要不同的 channel、namespace 或升级策略，否则保留默认安装配置。

参数	推荐配置
Channel	使用 Operator 软件包提供的默认 channel。
Installation Mode	`Cluster`。Operator 管理所选集群的 Fleet Monitoring 资源。
Installation Place	使用 Operator 软件包提供的推荐 namespace。
Upgrade Strategy	`Manual`，除非平台升级策略要求自动升级。

6. 在 OperatorHub 中确认 **Alauda Container Platform Fleet Monitoring Cluster Service** 的状态为 **Installed**。
   如果选择 `Manual` 升级策略，且 OperatorHub 显示待处理的安装计划，请批准该安装计划以完成安装。
7. 在目标集群上创建 `FleetMonitoringAgent` 资源。

```yaml
apiVersion: monitoring.alauda.io/v1alpha1
kind: FleetMonitoringAgent
metadata:
  name: fleet-monitoring-agent
spec:
  interval: 5m
```

`FleetMonitoringAgent` 是集群范围的资源。请勿设置 `metadata.namespace`。

字段	必需	描述
`metadata.name`	是	必须为 `fleet-monitoring-agent`。
`metadata.namespace`	否	请勿设置此字段。`FleetMonitoringAgent` 是集群范围的资源。

字段	必需	描述
`spec.interval`	否	采集间隔。支持的值为 `5m`、`10m`、`15m` 和 `30m`。默认值为 `5m`。

若要将 Global 集群本身纳入 Fleet Monitoring 数据，还需在 Global 集群上安装 **Alauda Container Platform Fleet Monitoring Cluster Service**，并在那里创建 `FleetMonitoringAgent` 资源。

8. 验证 `FleetMonitoringAgent` 状态。

```
kubectl get fleetmonitoringagent fleet-monitoring-agent -o yaml
```

检查以下条件：

条件	描述
`ConfigurationReady`	集群名称、本地 Monitoring 访问信息和数据库信息可用。
`ResourcesApplied`	已应用 Fleet Monitoring 资源，例如规则和 workload 资源。
`Ready`	集群已准备好写入 Fleet Monitoring 数据，或因受支持的原因而被有意跳过。

也可以检查阶段和检测到的集群名称：

```
kubectl get fleetmonitoringagent fleet-monitoring-agent -o jsonpath='Ph
ase: {.status.phase}{"\n"}Cluster: {.status.cluster}{"\n"}'
```

预期阶段为 `Ready`。常见的 `Ready` 原因包括：

- `WorkloadReady`：集群部署了 VMAgent，已准备好写入 Fleet Monitoring 数据。
- `SkippedForGlobal`：该集群是 Global 集群，因此 Cluster Service 跳过向同一 Global 存储部署 VMAgent。
- `SkippedBackendReuse`：集群复用了 Global VictoriaMetrics 后端，因此 Cluster Service 跳过部署 Fleet Monitoring VMAgent，以避免写入环路。

如果目标集群向 Global 存储写入数据，请确认数据库信息可用：

```
kubectl -n <fleet-monitoring-namespace> get secret fleet-monitoring-database
```

将 `<fleet-monitoring-namespace>` 替换为安装 **Alauda Container Platform Fleet Monitoring Cluster Service** 的 namespace。

9. 打开 **Platform** > **Observe** > **Fleet Monitoring**，确认集群显示在监控面板数据中。

升级 Fleet Monitoring

Fleet Monitoring 通过安装时使用的同一 OperatorHub 工作流进行升级。升级不会删除 Fleet Monitoring 数据。已连接集群写入 Global 存储的指标仍然可用。

兼容性矩阵

下表列出了支持的版本：

Fleet Monitoring 版本	支持的 **Alauda Container Platform** 版本
1.0.1	4.3.x、4.4.x
1.0.0	4.3.x

升级路径指南

- 补丁级升级：同一 minor 版本内的 patch 版本之间兼容，可以直接升级，例如 `1.0.0` → `1.0.1`。
- 升级顺序：先升级 Global 集群上的 **Alauda Container Platform Fleet Monitoring Central Service**，再升级每个已连接集群上的 **Alauda Container Platform Fleet Monitoring Cluster Service**。
- 两项服务在升级窗口期间可以运行不同版本。Global 集群升级期间，已连接集群会继续写入 Fleet Monitoring 数据。

升级操作步骤

1. 使用 `violet push` 推送新的 Operator 软件包，方式与初始安装相同。将 Central Service 软件包推送到 Global 集群，并将 Cluster Service 软件包推送到每个已连接集群。
2. 转到 **Administrator** > **Marketplace** > **OperatorHub**，然后选择目标集群。
3. 打开 Fleet Monitoring Operator 并开始升级。
   - 使用 `Automatic` 升级策略时，OperatorHub 会在新版本可用后升级 Operator。
   - 使用 `Manual` 升级策略时，请批准待处理的安装计划以开始升级。
4. 在运行 Fleet Monitoring Operator 的每个集群上重复前两步。
5. 确认已安装新版本。

   ```
   kubectl -n <fleet-monitoring-namespace> get csv
   ```

   Fleet Monitoring `ClusterServiceVersion` 必须以 `Succeeded` 阶段报告新版本。
6. 按照前面章节中的说明，再次验证 `FleetMonitoringHub` 和 `FleetMonitoringAgent` 状态。

请注意以下事项：

- 升级时不要删除并重新创建 `FleetMonitoringHub` 或 `FleetMonitoringAgent`。这两个资源都会保留，并由新 Operator 版本进行协调。
- 如果新版本提供了新的 VMAgent 镜像，部署 VMAgent 的集群上的 Fleet Monitoring VMAgent pod 会重启。采集会在重启期间暂停，然后自动恢复。

配置采集间隔

采集间隔在每个已连接集群的 `FleetMonitoringAgent` 资源上配置。

支持的值：

- `5m`
- `10m`
- `15m`

- `30m`

示例：

```
apiVersion: monitoring.alauda.io/v1alpha1
kind: FleetMonitoringAgent
metadata:
  name: fleet-monitoring-agent
spec:
  interval: 10m
```

更新 `spec.interval` 后，Fleet Monitoring Cluster Service 会协调本地采集配置。

在部署 Fleet Monitoring VMAgent 的集群上，VMAgent 采集间隔遵循 `spec.interval`，而 Fleet Monitoring 记录规则继续按照系统为联邦配置的间隔进行评估。在 Global 集群以及复用 Global VictoriaMetrics 后端、无需部署 Fleet Monitoring VMAgent 的集群上，本地 Fleet Monitoring 规则遵循 `spec.interval`。

配置自定义指标和记录规则

Fleet Monitoring 包含内置指标和记录规则。集群管理员可以在每个已连接集群上追加自定义指标和记录规则。

要将用户定义的指标报告到 Fleet Monitoring，请使用以下工作流：

1. 在已连接的业务集群上定义本地 Fleet 记录规则，将源指标转换为 Fleet 指标名称。
2. 将该记录指标名称添加到 Fleet allowlist，使 Fleet Monitoring VMAgent 将其联邦并 remote-write 到 Global 集群。
3. 如果需要 1 小时聚合等 Fleet 级汇总，请在 Global 集群上另行添加自定义 Hub 侧 `PrometheusRule`。
4. 使用 Fleet Monitoring 查询或监控面板验证记录指标和汇总指标。

确定是否仅需要 **Agent** 侧规则，还是同时需要 **Agent** 侧和 **Hub** 侧规则

选择以下模式之一：

- 需要 Global 集群上的原始 Fleet 指标并可以直接查询时，仅使用已连接集群的 ConfigMap。
- 如果还需要用于监控面板或长期视图的 1 小时聚合等 Fleet 级汇总，则同时使用已连接集群的 ConfigMap 和 Global 集群自定义 `PrometheusRule`。

目标示例：

- 已连接集群上的源指标：`node_load15`
- 已连接集群上记录的 Fleet 指标：`fleet:node:node_load15:avg`
- Global 集群上的可选 1 小时汇总：`fleet:node:node_load15:avg:avg_over_time_1h`

配置已连接集群

在已连接集群上安装 **Alauda Container Platform Fleet Monitoring Cluster Service** 的 namespace 中，创建或更新 `fleet-monitoring-custom-metrics` ConfigMap。

此 ConfigMap 有两个作用：

- `metrics.yaml` 将指标名称添加到 Fleet Monitoring VMAgent federate allowlist。
- `recording-rules-prometheus.yaml` 或 `recording-rules-victoriametrics.yaml` 定义生成 Fleet 指标的本地记录规则。

示例：

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: fleet-monitoring-custom-metrics
  namespace: <fleet-monitoring-namespace>
data:
  metrics.yaml: |
    default:
      - fleet:node:node_load15:avg
  recording-rules-prometheus.yaml: |
    groups:
      - name: fmval-custom
        rules:
          - record: fleet:node:node_load15:avg
            expr: avg by (node) (node_load15)
  recording-rules-victoriametrics.yaml: |
    groups:
      - name: fmval-custom
        rules:
          - record: fleet:node:node_load15:avg
            expr: avg by (node) (node_load15)
```

将 `<fleet-monitoring-namespace>` 替换为安装 **Alauda Container Platform Fleet Monitoring Cluster Service** 的 namespace。

`metrics.yaml` 会将指标名称追加到内置 allowlist。

对于记录规则，Cluster Service 仅加载与本地 Monitoring 堆栈匹配的键：

- 基于 Prometheus 的集群使用 `recording-rules-prometheus.yaml`
- 基于 VictoriaMetrics 的集群使用 `recording-rules-victoriametrics.yaml`

自定义配置可以追加指标和规则，但不会删除或覆盖内置默认值。

如果 ConfigMap 格式无效或包含无效规则，Fleet Monitoring 会保留内置默认值，并在 `FleetMonitoringAgent` 状态中报告错误。

在部署 Fleet Monitoring VMAgent 的已连接业务集群中，Agent reconciler 会将渲染后的 Fleet Monitoring 记录规则组间隔规范化为 `1m`。不要依赖 ConfigMap 中的自定义间隔值来控制最终渲染的本地 Fleet Monitoring 规则间隔。

对于自定义 Fleet 指标，请使用 Fleet 命名约定记录指标，例如 `fleet:node:node_load15:avg`。这样可以确保指标与 Fleet Monitoring 监控面板、汇总和查询模式兼容。

Fleet Monitoring 查询和监控面板要求记录的时间序列带有 `cluster` 标签。当规则尚未定义该标签时，Agent reconciler 会自动向渲染后的 Fleet Monitoring 记录规则添加 `cluster=<local-cluster-name>`。

验证已连接集群的渲染结果

更新 ConfigMap 后，Fleet Monitoring Agent 会自动协调本地规则和 VMAgent 配置，无需重启。

检查渲染后的本地规则：

```
kubectl -n <fleet-monitoring-namespace> get prometheusrule fleet-monitoring-agent-recording-rules -o yaml
```

确认：

- 自定义组显示在 `spec.groups` 中
- 自定义记录指标显示在规则列表中
- 渲染后的规则带有 `labels.cluster=<connected-cluster-name>`

检查渲染后的 VMAgent federate allowlist：

```
kubectl -n <fleet-monitoring-namespace> get configmap fleet-monitoring-vmagent -o yaml
```

确认自定义记录指标显示在 `data.prometheus.yml` 的 `params.match[]` 下。

添加自定义 **Hub** 汇总规则

如果希望自定义 Fleet 指标具有 1 小时聚合等 Fleet 级汇总，请在 Global 集群上、安装 **Alauda Container Platform Fleet Monitoring Central Service** 的 namespace 中创建单独的 `PrometheusRule`。

示例：

```
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  name: fmval-custom-hub-rollup
  namespace: <fleet-monitoring-namespace>
  labels:
    prometheus: kube-prometheus
    rule.cpaas.io/is-record: "true"
    alert.cpaas.io/owner: System
    alert.cpaas.io/project: ""
    monitoring.alauda.io/hub: fleet-monitoring-hub
spec:
  groups:
    - name: fmval-custom-rollup-1h
      interval: 1h
      rules:
        - record: fleet:node:node_load15:avg:avg_over_time_1h
          expr: avg_over_time(fleet:node:node_load15:avg[1h])
```

将 `<fleet-monitoring-namespace>` 替换为安装 **Alauda Container Platform Fleet Monitoring Central Service** 的 namespace。

此自定义 `PrometheusRule` 是追加配置，不需要复制内置 Hub 规则，也不应使用 Fleet Monitoring operator 所有权元数据创建。

验证 **Hub** 侧汇总

检查自定义 Hub 侧规则：

```
kubectl -n <fleet-monitoring-namespace> get prometheusrule fmval-custom-hub-rollup -o yaml
```

确认：

- 规则存在于 Global 集群上
- 规则使用预期的 Fleet 指标作为输入

- 汇总输出指标名称与计划使用的监控面板或查询表达式一致

查询自定义 Fleet 指标

通过平台 Monitoring API 或 Fleet Monitoring 监控面板查询 Fleet 指标时，请在选择器中明确包含 `vmcluster=~".*"`。在当前平台查询路径中，省略此选择器可能导致查询代理将查询范围缩小到 Global Monitoring 后端，并且不返回已连接集群的 Fleet 数据。

查询示例：

- 原始自定义 Fleet 指标：

```
avg_over_time(fleet:node:node_load15:avg{vmcluster=~".*",cluster="g1-c
1"}[1h])
```

- 1 小时汇总指标：

```
last_over_time(fleet:node:node_load15:avg:avg_over_time_1h{vmcluster=
~".*",cluster="g1-c1"}[2h])
```

常见错误

请注意以下问题：

- 在 `cpaas-system` 中创建 `fleet-monitoring-custom-metrics`，但 Fleet Monitoring 安装在其他 namespace（例如 `fleet-monitoring`）中
- 将源指标名称而不是记录的 Fleet 指标名称添加到 `metrics.yaml`
- 定义本地记录规则，却未将记录的 Fleet 指标名称添加到 `metrics.yaml`
- 期望已连接集群 ConfigMap 中的自定义间隔在渲染后仍然有效
- 查询 Fleet 指标时未在选择器中包含 `vmcluster=~".*"`
- 尚未创建相应的自定义 Hub 侧规则，却期望 Global Fleet 汇总指标已经存在

验证数据新鲜度

连接集群后，打开 **Platform** > **Observe** > **Fleet Monitoring**，在概览监控面板上检查以下信息：

- **Connected Clusters**
- **Stale Clusters**
- **Last Write Ago**
- **Data Freshness Exceptions**

如果集群显示在 **Cluster** 变量中，但未计入已连接集群，则该集群可能已被平台识别，但尚未写入 Fleet Monitoring 数据。检查该集群是否已从 OperatorHub 安装 **Alauda Container Platform Fleet Monitoring Cluster Service**，并且具有已就绪的 `FleetMonitoringAgent`。

故障排除

未显示 **Fleet Monitoring** 数据

检查以下项目：

- Global 集群已安装 **Alauda Container Platform Fleet Monitoring Central Service**。
- `FleetMonitoringHub` 资源存在且条件已就绪。
- Global 集群具有可用的 VictoriaMetrics 存储和 write 端点。仅 Prometheus 的 Monitoring 无法接收 Fleet Monitoring remote write 数据。
- 已应用内置监控面板和 Global 规则。

Connected Clusters 中缺少集群

在目标集群上检查以下项目：

- 已安装 **Alauda Container Platform Fleet Monitoring Cluster Service**。
- `FleetMonitoringAgent` 资源存在。
- 集群由 Global 集群管理。
- 集群已启用 Monitoring 功能。
- `FleetMonitoringAgent` 状态未报告缺少数据库信息或 Monitoring 功能信息无效。

- `fleet-monitoring-database` Secret 包含指向 Global VictoriaMetrics write 端点的 `remoteWriteURL`。
- `fleet-monitoring-vmagent` 日志未报告 remote write 错误。如果日志显示 `405 Method Not Allowed`，则 `remoteWriteURL` 可能指向 Prometheus 或 Thanos Query 端点，而不是 VictoriaMetrics write 端点。

自定义监控面板未显示在 **Fleet Monitoring** 中

检查该监控面板是否在 Global 集群上创建，并带有以下标签：

```
cpaas.io/dashboard.tag.fleet-monitoring: "true"
```

不带此标签的监控面板没有 `fleet-monitoring` 标签，因此不会列在 Fleet Monitoring 的 **Switch** 列表中。

同时检查监控面板资源的 namespace。Fleet Monitoring 页面会列出安装 **Alauda Container Platform Fleet Monitoring Central Service** 的 namespace 中的监控面板，以及 `cpaas-system` 和平台监控面板配置发布的 namespace 中的监控面板。在其他 namespace 中创建的监控面板不会列出。

监控面板中缺少新集群或项目

某些图表（例如项目配额图表和长期趋势图表）读取 Global 集群每小时评估一次的 Fleet 级汇总指标。新连接的集群或新创建的项目可能最多需要约一小时才会显示在这些图表中，即使原始 Fleet 指标已经可用。

如果一个汇总间隔后图表仍为空，请按照Connected Clusters 中缺少集群中的检查步骤操作。

数据新鲜度异常

检查以下项目：

- 已连接集群正在运行。
- Fleet Monitoring Cluster Service pod 健康。
- 已连接集群上的本地 Monitoring 组件健康。

- 已连接集群可以向 Global VictoriaMetrics 存储写入数据。
- `FleetMonitoringAgent` 状态未报告配置或资源应用错误。

了解更多

- Fleet Monitoring
- Monitoring 监控面板管理

将 MonitorDashboard 迁移到 Perses

目录

概述
前提条件
迁移监控面板
转换规则
导入 Grafana JSON
迁移后
相关操作

概述

从 Alauda Container Platform 4.4 开始，除了现有的 MonitorDashboard 功能外，还可以使用 Perses 监控监控面板。您可以将现有的 MonitorDashboard 资源迁移为 PersesDashboard 资源，并在 **Operations Center** > **Monitor** > **Dashboards (Perses)** 中查看。

迁移会将 Grafana 风格的 MonitorDashboard 模型转换为 PersesDashboard 模型。源 MonitorDashboard 会保留，并且仍可通过现有的监控监控面板入口访问。

迁移是选择性的。您可以选择要迁移的 MonitorDashboard 资源，并通过添加迁移标签 `kubectl` 触发转换。

前提条件

迁移监控面板前，请确保满足以下要求：

- 目标集群已安装 **Alauda Container Platform Dashboard**。
- `global` 集群已安装 **Alauda Container Platform Dashboard Essentials**。
- 您拥有源 MonitorDashboard 资源的读取权限。
- 您拥有目标 PersesDashboard 资源的创建或写入权限。

有关组件安装，请参阅安装。

迁移监控面板

通过向 MonitorDashboard 资源添加 `cpaas.io/migrate-to-perses=true` 标签来触发迁移。

1. 列出 MonitorDashboard 资源，并确定要迁移的监控面板。

```
kubectl get monitordashboards.ait.alauda.io -A
```

2. 向 MonitorDashboard 资源添加迁移标签。

```
kubectl label monitordashboards.ait.alauda.io <name> -n <namespace> cpa
as.io/migrate-to-perses=true
```

3. 验证是否已生成 PersesDashboard 资源。
   生成的 PersesDashboard 名称为 `<source-monitor-dashboard-name>-pd`。

```
kubectl get persesdashboards.perses.dev <name>-pd -n <namespace>
```

4. 在控制台中，转到 **Operations Center** > **Monitor** > **Dashboards (Perses)**，确认列表中显示了已迁移的监控面板，并且可以打开。

INFO

- 要迁移多个监控面板，请对每个监控面板运行标签命令。您可以批量迁移监控面板。

- 迁移不会删除或修改源 MonitorDashboard。原始监控面板及其入口仍可访问。
- 运行该命令的用户必须拥有读取 MonitorDashboard 资源和写入 PersesDashboard 资源的权限。

转换规则

迁移工具会将 MonitorDashboard 内容转换为 PersesDashboard 模型。

转换项	描述
图表类型	Grafana 风格的图表类型会映射为 Perses 图表类型，例如时间序列图表、统计图表、仪表盘图表、条形图表、饼图表和表格。不支持的图表类型会保留为占位 Markdown 面板。
查询	查询会被重写为使用平台指标查询网关 `observe-api`。监控面板不会公开数据源选择。
显示名称和元数据	在可用时，会保留显示名称、描述、标签和文件夹信息。
即时查询	即时值查询会转换为 **ACP Courier Instant Query**。

导入 **Grafana JSON**

如果您拥有 Grafana 监控面板 JSON，而不是 MonitorDashboard 资源，可以从 Perses Monitoring Dashboards 导入。在 **Perses Monitoring Dashboards** 中打开 **Create** 下拉菜单，选择 **Import dashboard**，然后提供 JSON 内容。

Grafana JSON 导入使用有损的 Grafana 到 Perses 转换。只有当前平台版本支持的图表类型和函数会被保留。

迁移后

- 生成的 Perses 监控面板是可编辑的自定义监控面板。

- 您仍可通过现有的监控面板入口访问源 MonitorDashboard。
- 如果源监控面板不包含 `cpaas.io/display-name` 注解，迁移后的监控面板列表中可能会显示 Kubernetes 资源名称。这不会影响渲染或指标查询。

相关操作

- Perses Monitoring Dashboards
- 安装
- 监控面板

分布式追踪

关于 **Distributed Tracing**

关于 Distributed Tracing

Alauda Distributed Tracing 记录单个请求在分布式应用中跨服务的运行路径。它帮助团队端到端地跟踪请求路径，理解跨服务边界的延迟，并在云原生微服务环境中排查故障。

在 microservices 架构中，一个用户操作通常会触发多个 service、database、message queue 和基础设施组件上的工作。Distributed Tracing 会将这些工作单元关联为单个 trace，使运维人员可以分析完整的执行路径，而不是孤立地检查每个 service。

trace 表示请求在系统中的完整路径。每个 trace 包含一个或多个 span，每个 span 记录一个逻辑工作单元，包括 operation name、start time、duration 和相关元数据。父子 span 关系展示上游和下游操作之间的连接方式。

Alauda Distributed Tracing 提供以下能力：

- 端到端请求可见性：用于查看分布式事务的完整执行路径
- 延迟分析：用于识别缓慢的 service、操作或下游依赖
- 根因分析：用于定位跨 service 边界的故障
- 可扩展的 **trace** 存储和查询：基于 Jaeger，并支持多个后端

Alauda Distributed Tracing 可与 Alauda Build of OpenTelemetry 配合使用，通过 OTLP 等 open telemetry 标准接收、处理并转发 trace 数据。它还可以与 Alauda Service Mesh 集成，从而在排查问题时将 service-to-service 流量与 trace 数据一并分析。

> **Note**
>
> 因为 Alauda Distributed Tracing 的发版周期与灵雀云容器平台不同，所以 Alauda Distributed Tracing 的文档现在作为独立的文档站点托管在 Alauda Distributed Tracing ↗。

Alauda 版 OpenTelemetry

关于 **Alauda Build of OpenTelemetry**

关于 Alauda Build of OpenTelemetry

Alauda Build of OpenTelemetry 基于开源 OpenTelemetry 项目，提供了一种标准化、厂商中立的方式，用于从云原生原生应用中收集遥测数据。它帮助团队构建一致的可观测性流水线，用于跟踪、指标和日志，而无需将应用埋点绑定到单一后端。

OpenTelemetry Collector 是接收、处理和导出遥测数据的核心运行时。它可以作为智能体在靠近工作负载的位置运行，也可以作为集中式网关运行，因此既适用于应用级埋点，也适用于平台级遥测流水线。

Alauda Build of OpenTelemetry 提供以下功能：

- 多信号采集：用于跟踪、指标和日志
- 协议和格式转换：用于连接不同的遥测来源和后端
- 遥测处理：在导出之前对数据进行增强、过滤、批处理和转换
- 灵活的导出目标：将遥测数据转发到可观测性后端，例如分布式跟踪系统
- 自动埋点支持：减少收集基础遥测所需的手动代码修改量

通过在工作负载和可观测性后端之间引入 OpenTelemetry，团队可以将应用埋点与后端实现细节解耦。这使得在集群之间标准化遥测采集、在不同后端之间迁移，以及在数据存储或查询之前控制遥测流量都变得更加容易。

Note

因为 Alauda Build of OpenTelemetry v2 的发版周期与灵雀云容器平台不同，所以 Alauda Build of OpenTelemetry v2 的文档现在作为独立的文档站点托管在 Alauda Build of OpenTelemetry v2。

日志

关于 **Logging Service**

关于 Logging Service

Logging 模块是 ACP 平台可观测性套件的核心组件，提供全面的日志管理能力，实现高效且可靠的日志处理。

该模块提供四项关键的日志功能：

- 日志采集，实现对应用程序、容器和基础设施组件日志的自动收集
- 日志存储，基于 ElasticSearch 和 ClickHouse 后端，支持可扩展且持久的日志保存
- 日志查询，支持对大量日志数据的快速且灵活的搜索

通过与 Filebeat、ElasticSearch 和 ClickHouse 等强大的开源组件集成，该模块使组织能够高效处理海量日志，加快故障排查，确保合规性要求，并实时获得宝贵的运营洞察。

> **Note**
>
> 因为 Logging Service 的发版周期与灵雀云容器平台不同，所以 Logging Service 的文档现在作为独立的文档站点托管在 Logging Service ↗。

事件

介绍

使用限制

Events

操作流程

事件概览

介绍

该平台集成了 Kubernetes 事件，记录 Kubernetes 资源的重要状态变化及各种运行状态变化。它还提供存储、查询和可视化的能力。当集群、节点或 Pod 等资源出现异常时，用户可以通过分析事件来确定具体原因。

基于从事件中识别出的根本原因，用户可以为工作负载创建告警策略。当关键事件数量达到告警阈值时，可以自动触发告警，通知相关人员及时干预，从而降低平台的运维风险。

目录

使用限制

使用限制

该功能依赖于日志服务。请确保平台内已预先安装 ACP Log Essentails、ACP Log Collector 和 ACP Log Storage 插件。

Note

因为 Logging Service 的发版周期与灵雀云容器平台不同，所以 Logging Service 的文档现在作为独立的文档站点托管在 Logging Service ↗。

Events

目录

操作流程

事件概览

操作流程

1. 点击左侧导航栏中的 **Operations Center** > **Events**。
   提示：通过顶部导航栏的下拉选择框切换集群以查看对应的事件。

事件概览

事件页面默认展示最近 30 分钟内发生的重要事件概览（可选择或自定义时间范围），以及资源事件记录。

- 重要事件概览：该卡片展示选定时间范围内重要事件的原因及发生该事件的资源数量。
  - 注意：当同一资源多次发生该类型事件时，资源数量不会累加。
  - 例如：节点重启事件的资源数量为 20，表示在选定时间范围内，有 20 个资源发生了该事件，同一资源可能多次发生。

- 资源事件记录：在重要事件概览区域下方，展示选定时间范围内符合查询条件的所有事件记录。您可以点击重要事件卡片筛选对应类型的事件，或展开视图输入查询条件进行搜索。查询条件如下：
  - 资源类型：发生事件的 Kubernetes 资源类型，如 Pod。
  - 命名空间：发生事件的 Kubernetes 资源所在的命名空间。
  - 事件原因：事件发生的原因。
  - 事件级别：事件的重要程度，如 Warning。
  - 资源名称：发生事件的 Kubernetes 资源名称，可选择或输入多个名称。

TIP

- 点击事件记录中资源名称旁的视图图标，可在弹出的 **Event Details** 对话框中查看事件详细信息。
- 事件原因左侧图标的颜色表示事件级别。绿色图标表示该事件级别为 `Normal`，该事件可忽略；橙色图标表示该事件级别为 `Warning`，说明资源存在异常，应关注该事件以防止事故发生。

检查

介绍

介绍

使用限制

架构

架构

Inspection

Component Health Status

操作指南

巡检

执行巡检

巡检配置

Component Health Status

Procedures to Operate

巡检报告说明

介绍

Inspection 模块是 ACP 平台可观测性套件的核心组件，提供自动化巡检和评估功能，用于全面监控资源并管理风险。

该模块提供四项核心巡检能力：

- 资源巡检：自动评估集群、节点、Pod、证书和其他平台资源，以识别风险和使用模式
- 实时监控：实时跟踪巡检任务进度，并即时了解资源的运行状态
- 可视化报告：直观展示巡检结果，包括资源风险、使用信息和运行洞察
- 报告生成：生成可下载的 PDF 或 Excel 格式巡检报告，其中包含全面的分析和建议

通过将这些能力与基于角色的访问控制和自动化评估算法相结合，该模块可帮助组织降低人工巡检成本，主动识别资源异常，降低业务风险，并通过系统化的健康评估维持平台的最佳性能。

目录

使用限制

使用限制

- 平台上的部分巡检项依赖于集群已安装监控组件。请确保每个集群均已提前安装 ACP Basic Monitoring 插件或 ACP Advanced Monitoring 插件。

- 平台巡检支持通过电子邮件发送巡检结果。请确保已提前完成邮件通知服务器配置。

借助容器平台的巡检功能，用户可以更高效地管理和维护容器环境，提升系统稳定性和安全性。

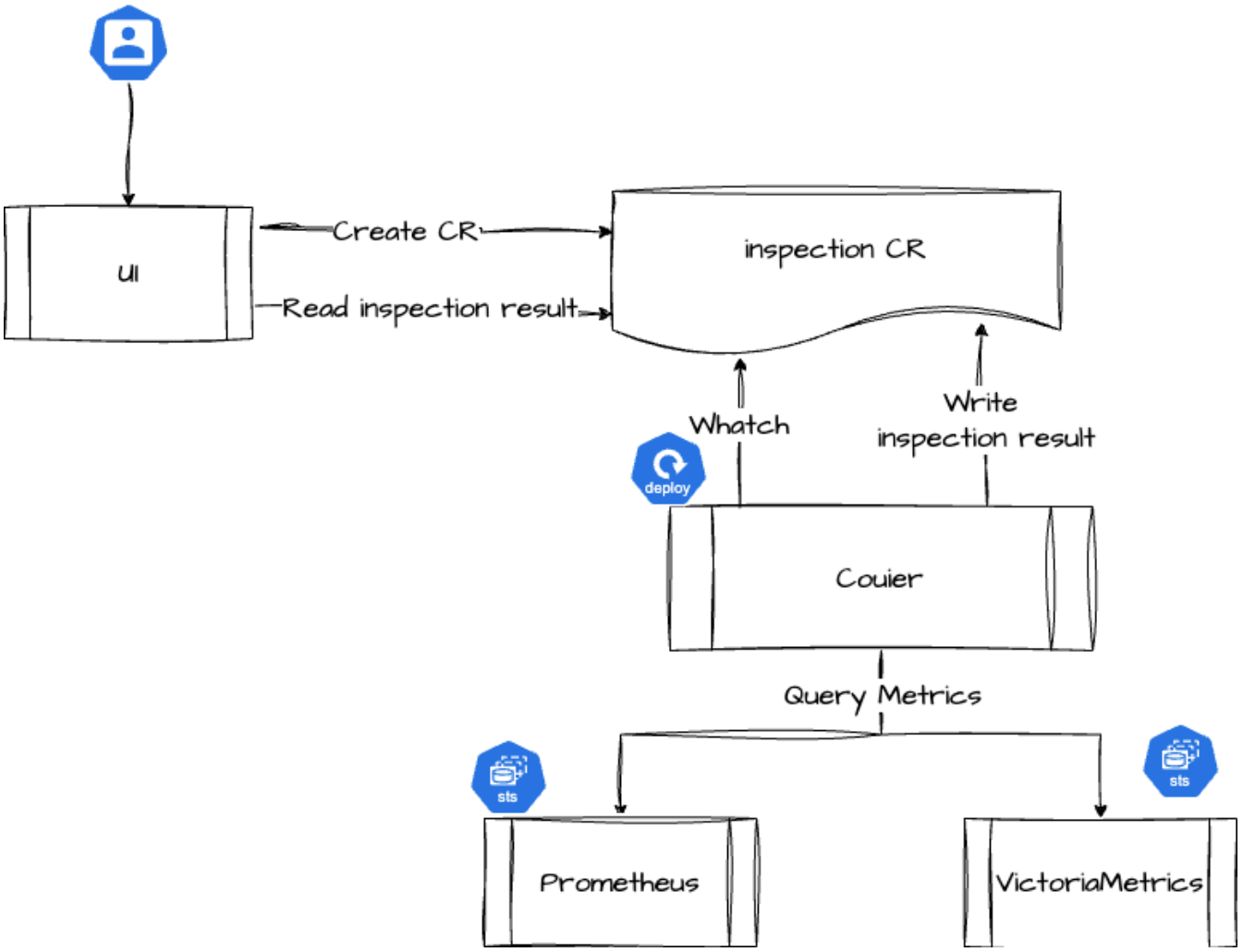
架构

目录

Inspection

Component Health Status

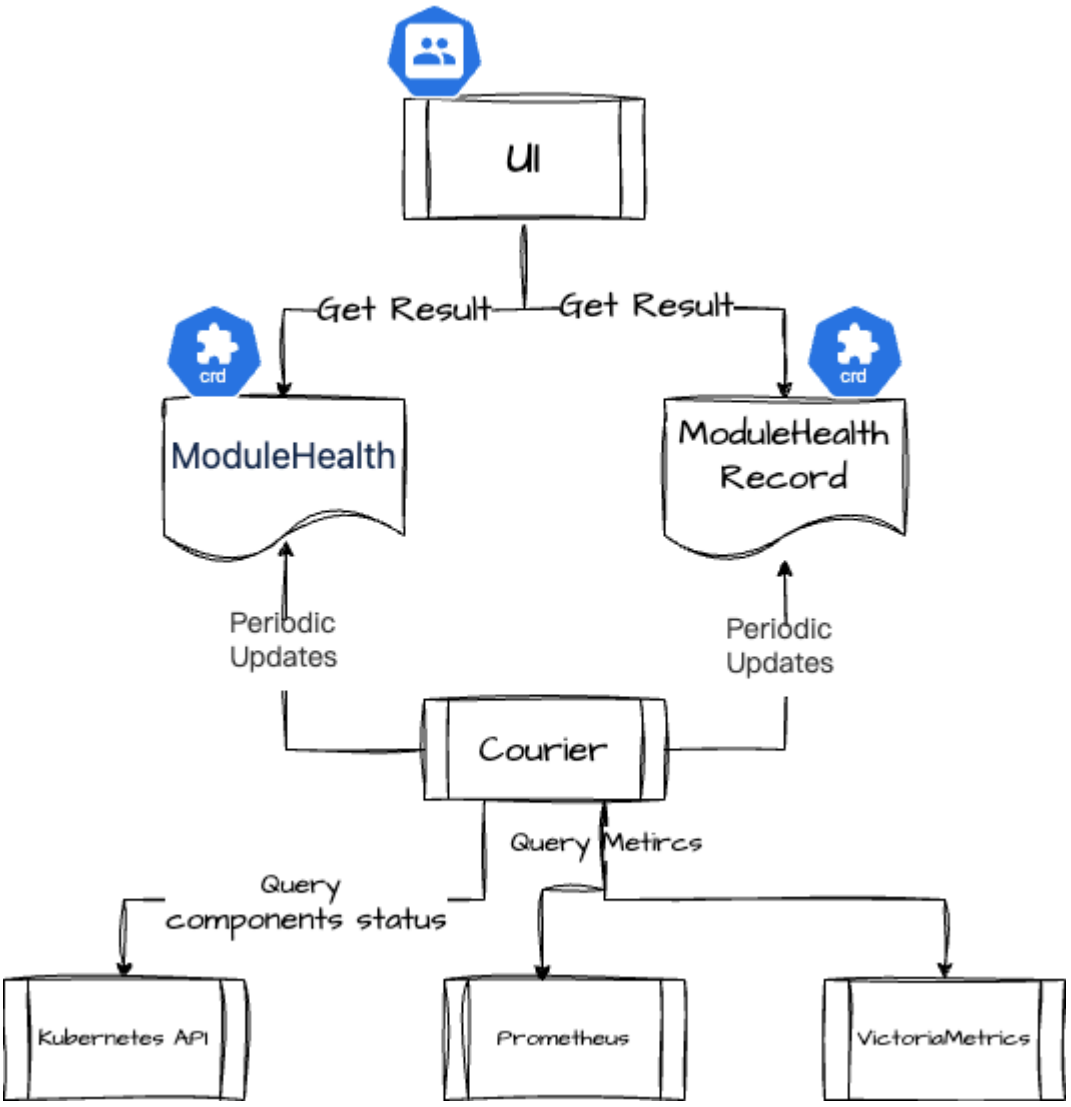
Inspection



Inspection 模块由平台组件 Courier 和监控组件共同提供，涉及以下业务流程：

- 创建 inspection 任务：平台向 `global` 集群提交 inspection 类型的 CR。
- 执行 inspection 任务：Courier 组件监控 inspection 类型 CR 的生成，并向各集群的监控组件查询与 inspection 相关的各类指标数据。
- 写入 inspection 结果：Courier 组件完成对各 inspection 项的评估后，将 inspection 结果写回对应的 inspection CR。
- 查看 inspection 结果：用户可通过平台查看 inspection 任务的状态和结果，数据来源于对应的 inspection CR。

Component Health Status



Component Health Status 由平台组件 Courier 和监控组件共同提供，涉及以下业务流程：

- 预定义组件监控列表：平台在 `global` 集群预定义了两类 CRD，用于定义需监控的组件列表及监控方式：
  - ModuleHealth：定义需监控的组件及监控方式。
  - ModuleHealthRecord：定义各集群对应组件的监控结果。
- 定期监控组件状态：Courier 会 watch ModuleHealth，检查指定功能，然后将 inspection 结果写入 ModuleHealth 和 ModuleHealthRecord 的 CR 资源。
- 组件状态判定：Courier 会请求 Kubernetes 和监控组件的数据，以判定组件的实际状态及存在的问题。
  - Kubernetes：检查组件是否安装及组件副本数是否正常。
  - Prometheus / VictoriaMetrics：基于各组件提供的指标，查询并判定组件是否能正常提供服务。

- 查看组件健康状态：用户可通过平台查看各组件的健康状态，数据来源于对应的 ModuleHealth 和 ModuleHealthRecord CR 资源。

操作指南

巡检

执行巡检

巡检配置

巡检报告说明

Component Health Status

Procedures to Operate

巡检

目录

执行巡检
巡检配置
巡检报告说明
最近一次巡检
资源风险巡检
资源利用率巡检

执行巡检

1. 点击左侧导航栏中的运维中心 > 巡检 > 基础巡检。
   提示：巡检页面显示最近一次巡检的巡检数据信息。在巡检过程中，您可以实时查看已完成巡检的资源数据。
2. 在基础巡检页面，支持以下操作：

   - 执行巡检：点击页面右上角的巡检按钮，对平台执行巡检。
   - 下载巡检报告：点击页面右上角的下载报告按钮，在弹出对话框中选择报告格式（PDF 和 Excel），然后点击下载，系统会将相应格式的报告下载到本地计算机。

     - PDF 格式的巡检报告不包含资源风险详情页数据；
     - Excel 格式的巡检报告包含巡检中的所有数据；

- 支持同时下载两种格式的报告。

巡检配置

巡检配置	描述
定时巡检	自动化任务执行时间规则，支持输入 Crontab 表达式。 提示：点击输入框可展开平台预设的触发规则模板，选择适当的模板，通过简单修改快速设置触发规则。
巡检记录保留	要保留的巡检记录数量。
邮件通知	选择邮件通知联系人。 注意：通知联系人必须已配置电子邮件。
巡检报告名称	平台内置巡检通知模板用于通知联系人的名称。
巡检配置项	根据平台针对证书、集群主机和 Pod 的默认巡检项，修改告警阈值或禁用巡检项。

巡检报告说明

最近一次巡检

在最近一次巡检信息区域，可以查看最近一次巡检的相关信息：

- 巡检时间：最近一次巡检的开始和结束时间。
- 巡检资源总数：最近一次巡检中巡检的资源总数（集群、节点、Pod、证书）。
- 风险：存在风险的资源数量，包括分类为故障和警告的资源。

资源风险巡检

在资源风险巡检页面，可以查看 `global` 集群、Kubernetes 灵雀云 提供的业务集群、第三方集群，以及这些集群下所有节点、Pod 和证书的风险信息概览。

点击相应资源类型（集群、节点、**Pod**、证书）卡片上的风险详情按钮，进入该资源类型的风险详情页面。在详情页面，可以查看该资源最近一次巡检的信息，以及存在故障和警告的资源列表。

- 点击资源名称，跳转到资源详情页面。
- 点击列表中名称字段右侧的展开按钮，展开故障和警告的判断条件及原因。

有关各资源的风险状态判断标准（故障、警告），请参阅下表。

注意：每种资源类型的故障和警告可能根据多个条件进行判断；当资源的巡检数据符合任一判断条件时，即视为一条风险数据。

资源类型	巡检范围	故障判断条件	警告判断条件
集群	- `global` 集群 - Kubernetes 灵雀云 提供的业务集群 - 第三方集群	- 集群状态为异常； - apiserver 连接异常	- 集群扩容（节点数/Pod 数/指标数）后，监控组件资源配置未更新。 - 日志数据量和日志采集频率增加后，日志组件资源配置未更新。 - 集群 CPU 使用率超过 60%； - 集群内存使用率超过 60%； - 集群 ETCD 组件中的任一 Pod 处于非 Running 状态； - 集群中的任一主机处于非 Ready 状态； - 集群中任意两个节点的时间差超过 40 秒； - 集群 CPU 请求率（实际请求值 / 总量）超过 60%； - 集群内存请求率（实际请求值 / 总量）超过 80%； - 集群未安装监控组件； - 集群监控组件异常； - 集群 **kube-controller-manager** 组件中的任一 Pod 处于非 Running 状态；

资源类型	巡检范围	故障判断条件	警告判断条件
			- 集群 **kube-scheduler** 组件中的任一 Pod 处于非 Running 状态； - 集群 **kube-apiserver** 组件中的任一 Pod 处于非 Running 状态。
节点	- 所有控制平面节点 - 所有工作节点	- 节点状态为异常； - 节点上的 **node-exporter** 组件 Pod 处于非 Running 状态； - 节点上的 **kubelet** 组件 Pod 处于非 Running 状态。	- 节点 inode 可用数小于 1000 - 节点 CPU 使用率超过 60%； - 节点内存使用率超过 60%； - 节点目录磁盘空间使用率超过 60%； - 节点系统负载超过 200% 且持续时间超过 15 分钟； - 过去一天至少发生过一次 NodeDeadlock（节点死锁）事件； - 过去一天至少发生过一次 NodeOOM（内存不足）事件； - 过去一天至少发生过一次 NodeTaskHung（任务挂起）事件。
Pod	所有 Pod	- Pod 状态为错误； - Pod 处于启动状态超过 5 分钟。	- Pod CPU 使用率超过 80%； - Pod 内存使用率超过 80%； - 过去 5 分钟内 Pod 的重启次数大于或等于 1。
证书	- Certmanager 证书 - Kubernetes 证书	证书状态为已过期。	证书有效期少于 29 天。

资源利用率巡检

点击资源利用率巡检页签，进入资源利用率巡检页面。

在资源利用率巡检页面，可以查看 `global` 集群、第三方集群和 Kubernetes 灵雀云 提供的业务集群的 CPU、内存和磁盘总量、使用量及使用率，以及平台上的集群、节点、Pod 和项目等资源数量。

- 资源使用统计：可以查看 `global` 集群、第三方集群和 Kubernetes 灵雀云 提供的业务集群的 CPU、内存和磁盘总量及总使用率。
- 平台资源数量：可以查看平台上运行的资源数量。

Component Health Status

平台健康状态页面展示了已安装在平台上的功能的健康状态统计数据。当您的账户拥有与平台相关的管理或审计权限时，还可以查看特定功能的详细健康数据，包括：未安装该功能的集群列表、已安装该功能的集群健康状态，以及与该功能相关的集群内组件的检测数据。这有助于您快速识别问题，提高平台的运维效率。

目录

Procedures to Operate

Procedures to Operate

1. 导航至已安装产品的视图页面或平台中心（平台管理、项目管理、运维中心）。
2. 点击导航栏右上角的问号按钮 > **Platform Health Status**。
3. 查看功能卡片；功能卡片显示该功能的健康状态信息。如果功能组件存在异常，会在卡片上以 `fault` 形式体现。
4. 点击功能卡片上的健康/故障值，页面右侧将展开详细健康状态页面，您可以查看故障组件的详细问题信息。