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

- 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 Monitoring with Prometheus

ACP Monitoring with VictoriaMetrics

架构

监控模块架构

- 总体架构说明
- 监控系统
- 告警系统
- 通知系统

Monitoring Component Selectic

- 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

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

- 功能概览
- 主要功能
- 通知服务器
- 接收组
- 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 创建监控面板

常用函数和变量

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

Fleet Monitoring

概述

前提条件

通知

Perses 监控

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 监控数据的备份与

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

为 Monitoring 规划 Infra 节点

配置 Fleet Monitoring

将 MonitorD

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

监控组件为可观测性模块内的监控、告警、巡检和健康检查功能提供基础设施。在集群中安装 ACP Monitoring with Prometheus 或 ACP Monitoring with VictoriaMetrics。

要决定安装哪个插件，请先查看 监控组件选型指南，并选择最适合你的集群规模、存储方案和运维需求的选项。

Perses 监控仪表盘需要额外的 dashboard 组件。当你需要在 **Operations Center** > **Monitor** > **Dashboards (Perses)** 中查看、创建或迁移 dashboard 时，请安装这些组件。

开始之前

> **INFO**
>
> 某些 Monitoring 组件资源消耗较高。我们建议通过插件配置将它们部署到 infra 节点上。Prometheus 和 VictoriaMetrics 都支持插件级别的 `nodeSelector` 和 `tolerations` 设置。如果你正在评估产品且尚未准备 infra 节点，可以将这些设置留空，使组件运行在普通节点上。
>
> 有关规划 infra 节点的指导，请参见 集群节点规划。

在安装监控组件之前，请确保满足以下条件：

- 已通过 监控组件选型指南 选择了合适的监控组件。
- 在 workload cluster 中安装时，确保 `global` 集群可以访问 workload cluster 的 11780 端口。
- 如果需要为监控数据使用 storage classes 或 persistent volume 存储，请提前在 **Storage** 部分创建相应资源。

安装 **Perses Dashboard** 组件

Perses 监控 dashboard 依赖以下组件：

Component	Installation location	Purpose
Alauda Container Platform Dashboard	仅 `global` 集群	为 Perses dashboards 提供支持 RBAC 的指标查询服

Component	Installation location	Purpose
Essentials		务。
Alauda Container Platform Dashboard for Perses	`global` 集群运行唯一的 Perses server 实例。workload cluster 按需安装 operator 以提供 PersesDashboard CRD，但不会创建 Perses 实例。	提供 Perses dashboard 后端、渲染、编辑以及 PersesDashboard 资源管理能力。

Perses server 和 `observe-api` 仅在 `global` 集群中创建。当你在控制台中选择一个 workload cluster 时，控制台仍会将 Perses server 请求路由到 `global` 集群。所选的 workload cluster 决定使用哪个集群的指标和 PersesDashboard 资源。`observe-api` 会跨集群查询指标并强制执行 RBAC。

请按以下顺序安装这些组件：

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。你可以使用组件的 **Create** 操作或 OperatorHub 入口，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 for Perses**。
4. 在 `global` 集群中创建唯一的 `Perses` 实例。你可以使用组件的 **Create** 操作或 OperatorHub 入口，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. 对于每个用户需要查看或创建 Perses dashboards 的 workload cluster，安装 **Alauda Container Platform Dashboard for Perses**，以便在该集群中提供 PersesDashboard CRD。不要在 workload cluster 中创建 `Perses` 实例。
6. 转到 **Operations Center** > **Monitor** > **Dashboards (Perses)**，并验证所选集群中是否可以看到 dashboard 列表。

如果所选集群尚未安装 **Alauda Container Platform Dashboard for Perses**，Perses dashboard 页面会显示安装指南，而不是 dashboard 列表。

Perses dashboard 组件复用现有的平台指标存储，例如 VictoriaMetrics 或 Prometheus。它们不会引入独立的指标存储或 tenant。

安装 operator 仅会安装 controller 和 CRD。你必须先在 `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 Monitoring with Prometheus

从控制台安装

1. 导航到 **App Store Management** > **Cluster Plugins** 并选择目标集群。
2. 找到 **ACP Monitoring with Prometheus** 插件并点击 **Install**。
3. 配置以下参数：
   控制台会突出显示最常见的安装选项。有关可配置字段的详细说明，请参见本节中的 YAML 参考。

Parameter	Description
Scale Configuration	支持三种配置：**Small Scale**、**Medium Scale** 和 **Large Scale**： - 默认值基于平台推荐的压测值设置 - 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 storage class 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
Replica Count	设置监控组件 pod 数量 注意：Prometheus 仅支持单节点安装
Advanced Configuration	可折叠区域，用于插件级调度参数。
Node Selectors	显示在 **Advanced Configuration** 中。为 Prometheus 插件工作负载配置插件级 node selector 规则。
Node Tolerations	显示在 **Advanced Configuration** 中。为 Prometheus 插件工作负载配置插件级 toleration 规则。
Parameter Configuration	可按需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，确保该插件已经发布：

```
# 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**

创建一个 ModuleInfo 资源，在不使用任何配置参数的情况下安装该插件：

```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
      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
```

更多详细信息，请参见 监控组件容量规划。

YAML 字段参考（Prometheus）：

Field path	Description
`metadata.labels.cpaas.io/cluster-name`	安装该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）。建议最小 30 Gi。

Field path	Description
`spec.config.storage.nodes`	`storage.type=LocalVolume` 时的节点列表。最多支持 1 个节点。
`spec.config.storage.path`	`storage.type=LocalVolume` 时的基础 LocalVolume 路径。默认值：`/cpaas/monitoring`。
`spec.config.storage.storageClass`	`storage.type=StorageClass` 时的 StorageClass 名称。
`spec.config.storage.pvSelectorK`	`storage.type=PV` 时的 PV 选择器键。
`spec.config.storage.pvSelectorV`	`storage.type=PV` 时的 PV 选择器值。
`spec.config.replicas`	副本数；仅适用于 `StorageClass` / `PV` 类型。
`spec.config.components.nodeSelector`	可选。为 Prometheus 插件工作负载配置插件级 node selector 规则。
`spec.config.components.tolerations`	可选。为 Prometheus 插件工作负载配置插件级 toleration 规则。
`spec.config.components.prometheus.retention`	数据保留天数。
`spec.config.components.prometheus.scrapeInterval`	抓取间隔秒数；适用于未设置 `interval` 的 ServiceMonitors。
`spec.config.components.prometheus.scrapeTimeout`	抓取超时时间秒数；必须小于 `scrapeInterval`。
`spec.config.components.prometheus.resources`	Prometheus 的资源设置。
`spec.config.components.nodeExporter.port`	Node Exporter 端口（默认 9100）。

Field path	Description
`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 名称在创建后会发生变化，请通过标签定位该资源，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`：安装该插件的集群
- `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`：期望安装版本
- `CURRENT_VERSION`：安装前版本
- `NEW_VERSION`：可供安装的最新版本

将 **Prometheus** 工作负载放置到 **Infra** 节点

如果你希望 Prometheus 插件工作负载运行在专用 infra 节点上，请在安装或升级时配置插件级调度规则，而不是在安装后 patch 已生成的工作负载。

- 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，请确保你的 infra 节点规划和存储放置是兼容的。有关规划注意事项，请参见 How To 中的 Monitoring 指南，包括 **Planning Infra Nodes for Monitoring**。

访问已安装组件

安装完成后，可通过以下地址访问这些组件（将 `<>` 替换为实际值）：

Component	Access Address
Thanos	`<platform_access_address>/clusters/<cluster>/prometheus`
Prometheus	`<platform_access_address>/clusters/<cluster>/prometheus-0`
Alertmanager	`<platform_access_address>/clusters/<cluster>/alertmanager`

ACP Monitoring with VictoriaMetrics

前提条件

- 如果你只安装 VictoriaMetrics agent，请确保 VictoriaMetrics Center 已安装在其他集群中。

从控制台安装

1. 导航到 **App Store Management** > **Cluster Plugins** 并选择目标集群。
2. 找到 **ACP Monitoring with VictoriaMetrics** 插件并点击 **Install**。
3. 配置以下参数：
   控制台会突出显示最常见的安装选项。有关可配置字段的详细说明，请参见本节中的 YAML 参考。

Parameter	Description
Scale Configuration	支持三种配置：**Small Scale**、**Medium Scale** 和 **Large Scale**： - 默认值基于平台推荐的压测值设置 - 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
Install Agent Only	- Off：安装完整的 VictoriaMetrics 组件套件 - On：仅安装 VMAgent 采集组件，依赖 VictoriaMetrics Center
VictoriaMetrics Center	选择已安装完整 VictoriaMetrics 组件的集群
Storage Type	- **LocalVolume**：本地存储，数据存储在指定节点上 - **StorageClass**：使用 storage class 自动生成持久卷 - **PV**：使用现有持久卷
Storage Path	当 **Storage Type** 为 `LocalVolume` 时显示。指定监控数据的基础存储路径。默认值：`/cpaas/monitoring`。

Parameter	Description
Replica Count	当 **Storage Type** 为 `StorageClass` 或 `PV` 时显示。设置监控组件 pod 数量。当 **Storage Type** 为 `LocalVolume` 时，所选节点数量决定 VMStorage 副本数量。
Advanced Configuration	可折叠区域，用于插件级调度参数。
Node Selectors	显示在 **Advanced Configuration** 中。为 VictoriaMetrics 插件工作负载配置插件级 node selector 规则。
Node Tolerations	显示在 **Advanced Configuration** 中。为 VictoriaMetrics 插件工作负载配置插件级 toleration 规则。
Parameter Configuration	可按需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

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，确保该插件已经发布：

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

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

创建 **ModuleInfo**

创建一个 ModuleInfo 资源，在不使用任何配置参数的情况下安装该插件：

```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
      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
```

更多详细信息，请参见 监控组件容量规划。

YAML 字段参考（VictoriaMetrics）：

Field path	Description
`metadata.labels.cpaas.io/cluster-name`	安装该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>-`

Field path	Description
	`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 30 Gi。
`spec.config.storage.nodes`	`storage.type=LocalVolume` 时的节点列表。可以选择一个或多个节点。
`spec.config.storage.path`	`storage.type=LocalVolume` 时的基础 LocalVolume 路径。默认值：`/cpaas/monitoring`。
`spec.config.storage.storageClass`	`storage.type=StorageClass` 时的 StorageClass 名称。
`spec.config.storage.pvSelectorK`	`storage.type=PV` 时的 PV 选择器键。
`spec.config.storage.pvSelectorV`	`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。
`spec.config.agentReplicas`	启用仅 agent 模式时 vmagent 的副本数。
`spec.config.crossClusterDependency.victoriametrics`	在仅 agent 模式下提供 VictoriaMetrics Center 的目标集群。

Field path	Description
`spec.config.components.nodeSelector`	可选。为 VictoriaMetrics 插件工作负载配置插件级 node selector 规则。
`spec.config.components.tolerations`	可选。为 VictoriaMetrics 插件工作负载配置插件级 toleration 规则。
`spec.config.components.vmstorage.retention`	vmstorage 的数据保留天数。
`spec.config.components.vmagent.scrapeInterval`	抓取间隔秒数；适用于未设置 `interval` 的 ServiceMonitors。
`spec.config.components.vmagent.scrapeTimeout`	抓取超时时间秒数；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/custom metrics）。
`spec.config.components.vmagent.resources`	vmagent 的资源设置。
`spec.config.components.vminsert.resources`	vminsert 的资源设置。
`spec.config.components.vmselect.resources`	vmselect 的资源设置。

Field path	Description
`spec.config.size`	监控规模：`Small`、`Medium` 或 `Large`。

验证安装

由于 ModuleInfo 名称在创建后会发生变化，请通过标签定位该资源，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`：安装该插件的集群
- `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`：期望安装版本
- `CURRENT_VERSION`：安装前版本
- `NEW_VERSION`：可供安装的最新版本

将 VictoriaMetrics 工作负载放置到 Infra 节点

如果你希望 VictoriaMetrics 插件工作负载运行在专用 infra 节点上，请在安装或升级时配置插件级调度规则，而不是在安装后 patch 已生成的工作负载。

- 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：

```
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，请确保你的 infra 节点规划和存储放置是兼容的。有关规划注意事项，请参见 How To 中的 Monitoring 指南，包括 **Planning Infra Nodes for Monitoring**。

访问已安装组件

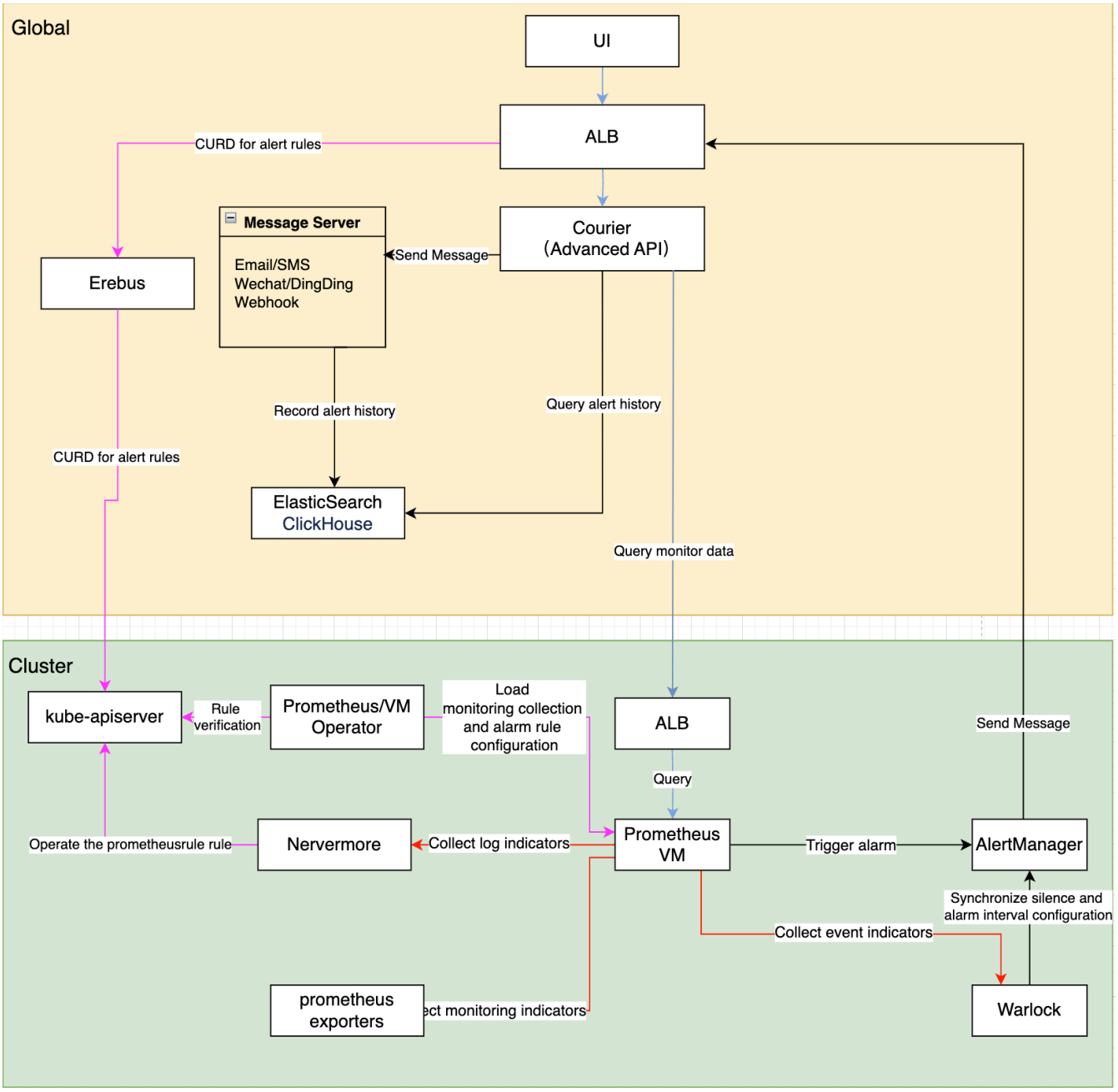
安装完成后，可通过以下地址访问该组件（将 `<>` 替换为实际值）：

Component	Access Address
VictoriaMetrics UI	`<platform_access_address>/clusters/<cluster>/vmselect-ui/vmui/?#/metrics`

> **INFO**
>
> 如果启用了 **Install Agent Only**，则该集群不会本地部署 `vmselect` 组件，因此该集群中无法使用 VictoriaMetrics UI 地址。

架构

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

总体架构说明

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：评估告警规则并分发通知
- 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：管理各类通知渠道的配置

监控系统

数据采集与存储

1. Prometheus/VictoriaMetrics Operator 职责：
   - 加载并校验监控采集配置
   - 加载并校验告警规则配置
   - 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 发现并采集各类 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（Path: `/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 indicators 接口获取 PromQL 并进行查询
     - 自定义指标：直接将 PromQL 转发到监控组件
   - 监控 dashboard 获取数据并展示
2. 监控 dashboard 管理流程：
   - 用户访问 `global` 集群的 ALB（Path: `/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：

- 校验监控 dashboard 配置的合法性
- 管理 MonitorDashboard CR 资源

3. Perses dashboard 查询流程：

- 用户在控制台中打开 **Operations Center** > **Monitor** > **Dashboards (Perses)**。
- 控制台通过 **Alauda Container Platform Dashboard for Perses** 向部署在 `global` 集群中的 Perses server 发送 dashboard 请求。
- Perses server 使用固定的 `acp-observe` 数据源，通过 `observe-api` 查询指标。
- 部署在 `global` 集群中的 **Alauda Container Platform Dashboard Essentials** 提供的 `observe-api` 会先检查用户的 cluster、namespace 和 project 权限，然后再将查询转发到平台指标存储。所选的 workload cluster 是指标查询目标，但它不会运行本地的 Perses server 或 `observe-api`。
- 平台指标存储（例如 VictoriaMetrics 或 Prometheus）将查询结果返回给 dashboard。

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

Perses dashboards 在查询和渲染路径中引入了 `observe-api`、Perses server 和 perses-operator。Perses server 和 `observe-api` 仅运行在 `global` 集群中。workload cluster 可按需安装 Perses operator，以提供 PersesDashboard CRD 并保存其自身的 dashboard 资源。现有的 MonitorDashboard 能力仍然可用，并继续使用现有的 MonitorDashboard 管理路径。

告警系统

告警规则管理

告警规则配置流程如下：

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

- Prometheus/VictoriaMetrics Operator：
  - 校验告警规则的合法性
  - 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：
   - 触发后，告警会发送到 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 生成以下指标：
     - 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. 状态展示：
   - 前端通过 Courier API 查询并展示实时状态

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：
   - Notification Template: apiVersion: "ait.alauda.io/v1beta1", kind: "NotificationTemplate"
   - Notification Contact Group: apiVersion: "ait.alauda.io/v1beta1", kind: "NotificationGroup"
   - Notification Policy: apiVersion: "ait.alauda.io/v1beta1", kind: "Notification"
3. Courier 负责：
   - 校验通知模板的合法性
   - 校验通知联系人组的合法性
   - 校验通知策略的合法性

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：
   - 校验通知服务器配置的合法性

Monitoring Component Selection Guide

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

- 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 适用于高可用需求及多集群监控场景。
- Prometheus 适用于单集群监控场景，尤其是规模较小的情况。

组件列表

Prometheus 相关组件

组件名称	功能描述
Prometheus Server	负责采集、存储和查询监控数据的核心服务器
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，负责告警规则和通知处理
PushGateway	支持监控数据的推送模式，适用于特殊网络环境下的数据传输

VictoriaMetrics 相关组件

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

架构对比

Prometheus 架构

Prometheus 是成熟的开源监控系统，是 CNCF 继 Kubernetes 之后的第二个毕业项目，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及监控解决方案，具备以下优势：

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

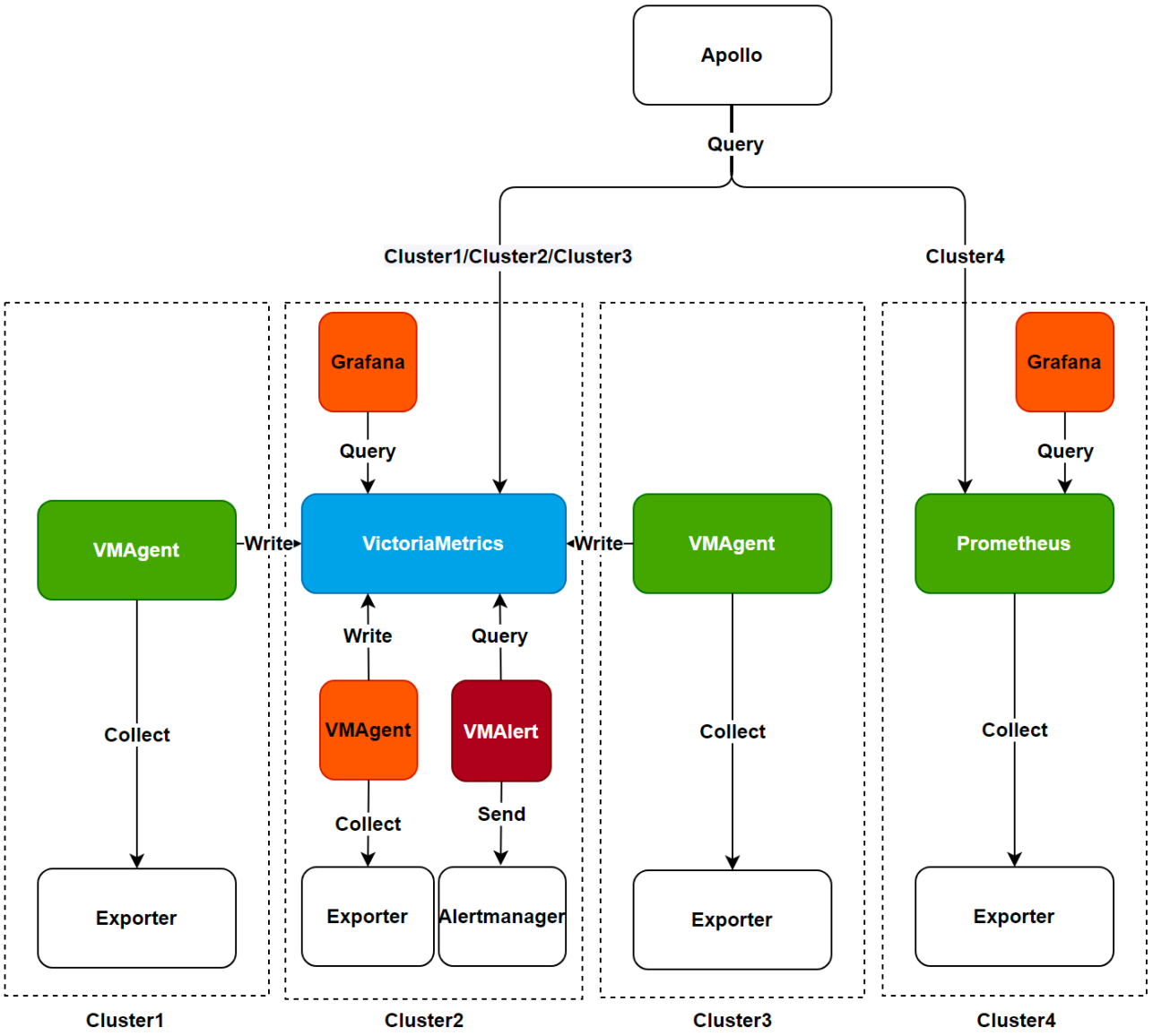
功能对比

功能	**Prometheus**	**VictoriaMetrics**	说明
高可用安装	❌	✅	VictoriaMetrics 支持真正的集群高可用，且数据一致性更好
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 资源利用率更高

功能	Prometheus	VictoriaMetrics	说明
社区支持	非常成熟	快速发展	Prometheus 拥有更大的社区生态

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安装架构及数据流向。平台提供以下两种安装方式供选择：

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

Prometheus 安装方式

该方式对应上图中的 **cluster4** 架构：

- 使用 Prometheus 组件采集和处理监控数据。
- 通过监控面板查询和展示数据。
- 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
   - 对应上图中的 **cluster2** 架构。
   - 所有 VictoriaMetrics 组件安装在同一集群内。
   - 使用 VMAgent 采集数据并写入 VictoriaMetrics。
   - VMAlert 负责告警规则评估。
   - 通过监控面板查询和展示数据。 提示：建议数据规模低于每秒 100 万时使用此模式。
2. 多集群安装模式
   - 对应上图中的 **cluster1/cluster2/cluster3** 架构。
   - 在业务集群中安装 VMAgent 作为数据采集智能体。
   - VMAgent 将数据写入中央监控集群中的 VictoriaMetrics。
   - 支持多集群统一监控管理。 提示：请确保在安装 VMAgent 前，VictoriaMetrics 服务已部署在监控集群中。

选择建议

适合使用 **VictoriaMetrics** 的场景

- 高性能及可扩展性需求：适用于处理高吞吐数据和长期存储的监控场景。
- 成本效益考虑：需要优化存储和计算资源成本。
- 高可用需求：对监控组件的高可用性有保障要求。
- 多集群管理：需要跨多个集群统一管理监控数据。

适合使用 **Prometheus** 的场景

- 单集群小规模：监控规模较小，无高可用需求。
- 已有 **Prometheus** 用户：已有完整的 Prometheus 监控体系。
- 简单稳定需求：追求简单可靠的监控方案。
- 深度生态集成：与 Prometheus 生态紧密集成，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

- 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

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

功能概览

主要功能

通知服务器

接收组

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 创建监控面板

常用函数和变量

Perses 监控

Fleet Monitoring

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

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

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

功能概览

通过通知功能，您可以将平台的监控和告警能力集成起来，及时向通知接收人发送预警信息，提醒相关人员采取必要措施以解决问题或避免故障。

主要功能

- 通知服务器：通知服务器提供向平台上的接收组发送通知消息的服务，例如邮件服务器。
- 接收组：接收组是具有相似逻辑特征的一组通知接收人，通过对接收通知消息的实体进行分类，可以降低维护成本。
- 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。用于规范通知规则的告警通知消息内容和格式。例如，自定义邮件通知的主题和内容。
- 通知规则：通知规则是一组用于定义如何向特定联系人发送通知消息的规则。在告警、巡检和登录认证等需要通知外部服务的场景中，必须使用通知规则。

通知服务器

通知服务器提供向平台上的接收人发送通知消息的服务。平台当前支持以下通知服务器：

- 企业通讯工具服务器：支持集成企业微信、钉钉和飞书内置应用，用于向个人发送通知。
- 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，或向您指定的服务器发送 WebHook。

> **WARNING**
>
> 只能添加一个企业通讯工具服务器。

企业通讯工具服务器

企业微信

1. 按照以下示例配置通知服务器参数。填写完成后，切换到 集群管理 > 资源管理 中的 `global` 集群，并创建资源对象。

```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，或者在用户的 个人信息 中更新用户的 企业微信 **ID**，以确保用户能够正常接收消息。

钉钉

1. 按照以下示例配置通知服务器参数。填写完成后，切换到 集群管理 > 资源管理 中的 `global` 集群，并创建资源对象。

```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，或者在用户的 个人信息 中更新用户的 钉钉 **ID**，以确保用户能够正常接收消息。

飞书

1. 按照以下示例配置通知服务器参数。填写完成后，切换到 集群管理 > 资源管理 中的 `global` 集群，并创建资源对象。

```
# 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，或者在用户的 个人信息 中更新用户的 飞书 **ID**，以确保用户能够正常接收消息。

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。当勾选 使用 **SSL** 时，必须填写 SSL 端口号。
服务器配置	使用 **SSL**：SSL（Secure Socket Layer）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连接。 跳过不安全验证：`insecureSkipVerify` 开关用于控制是否验证客户端证书

参数	描述
	和服务器主机名。启用后，将不会验证证书以及证书中的主机名与服务器主机名的一致性。
发件人邮箱	通知服务器中的发件人邮箱账号，用于发送通知邮件。
启用认证	如果需要认证，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，或向您指定的 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` 集群旁边的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的主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；如需禁用，请将 `dHJ1ZQo=` 替换为 `ZmFsc2UK`，即 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` 集群旁边的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的主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；如需禁用，请将 `dHJ1ZQo=` 替换为 `ZmFsc2UK`，即 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` 集群旁边的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的主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；如需禁用，请将 `dHJ1ZQo=` 替换为 `ZmFsc2UK`，即 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` 集群旁边的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的主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；如需禁用，请将 `dHJ1ZQo=` 替换为 `ZmFsc2UK`，即 false 的 base64 编码值。

为 **Webhook** 通知配置自定义请求头

如果目标 Webhook 端点需要自定义 HTTP 请求头，请先在 `cpaas-system` 命名空间中创建一个 Secret，后续再将其与接收组关联。

1. 在左侧导航栏中，单击 集群管理 > 集群。
2. 单击 `global` 集群旁边的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的主节点上创建如下所示的 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 请求头名称。
2. `data` 下的每个值在应用到集群之前都必须进行 base64 编码。
3. 如果端点需要多个请求头，请在 `data` 下添加更多键值对。

接收组

接收组是一组具有相似逻辑特征的通知接收人。例如，您可以将运维团队设置为接收组，以便在配置通知规则时进行快速选择和管理。

INFO

1. 平台支持多种通知服务器，并会根据通知服务器配置展示对应通知类型的相关配置项。
2. 如果需要使用 Webhook 类型服务器作为通知接收人，必须在接收组中配置相关 URL。
3. 如果 Webhook 端点需要自定义请求头，则必须将接收组关联到 `cpaas-system` 命名空间中类型为 `NotificationSender` 的 Secret。

1. 在左侧导航栏中，单击 运维中心 > 通知。

2. 切换到 接收组 选项卡。
3. 单击 创建接收组，并按照以下说明配置相关参数。

参数	描述
邮箱	为整个接收组添加一个邮箱。平台将向该邮箱以及组内所有联系人的邮箱发送通知。
Webhook URL/企业微信群机器人**/**钉钉群机器人**/**飞书群机器人	请根据已配置的通知服务器填写对应的通知方式 URL。配置完成后，该组中的联系人将通过此方式接收通知。
联系人配置	单击 添加联系人 将平台中已有的用户添加到接收组中。请确保所选联系人的联系信息（电话、邮箱、接口回调）的准确性，以避免漏收消息通知。

4. 单击 添加。

将接收组与 **Header Secret** 关联

如果已配置的 Webhook 端点需要自定义请求头，请将接收组与 为 Webhook 通知配置自定义请求头 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` 集群旁边的操作按钮 > **CLI** 工具。
3. 在 `global` 集群的主节点上编辑对应的接收组（`NotificationGroup`）资源。

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

4. 向 `metadata.annotations` 添加 `cpaas.io/notification.webhook.config` 注解。其值必须为用于自定义请求头的 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 在接收组中配置，自定义请求头通过引用的 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。用于规范通知规则的告警通知消息内容和格式。

平台管理员或运维人员可以根据不同的告警通知方式设置通知模板，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升运维效率。

平台管理员可以将平台通知模板设为公开。公开模板可在项目通知规则中被选用，因此项目管理员无需在每个项目中重复创建模板，即可复用平台维护的消息格式。

INFO

平台支持多种通知服务器，并会根据通知服务器配置展示对应通知类型的模板。如果未配置通知服务器，则默认不会展示对应的通知模板。

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 选项卡。
3. 单击 创建模板。
4. 在 基本信息 区域中，配置以下参数。

参数	描述
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，与平台告警功能配合使用； 组件异常消息：发送由某些组件异常触发的通知信息。
公开	使模板可用于项目通知规则。该选项适用于平台通知模板。模板设为公开后，项目管理员在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会带有只读的 公开 标签，并在详情页显示公开状态。当您将公开模板重新设为私有时，在提交更改前，请在对话框中确认影响。

5. 在 模板配置 区域中，引用不同的模板类型以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，变量和其他元素可以自由组合。
2. 模板中只能使用平台支持的变量。您可以修改变量显示名称和内容格式，但不能修改变量本身。请参见 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。如无特殊要求，您可以直接使用默认模板内容。
4. 如果平台公开模板重新设为私有，它仍会在平台公开模板组中可见，但会处于禁用状态。该模板不能用于新建通知规则，但已在使用该模板的现有通知规则仍可继续发送通知。
5. 如果平台公开模板被任何项目通知规则引用，则会受到删除保护。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}}`。为了方便用户快速获取关键信息，可以为变量设置自定义显示名称，例如：`告警级别: {{ .externalLabels.severity }}`。

当通知规则基于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中对应的标签值（实际监控数据）。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础变量：

显示名称	变量	描述
告警状态	`{{ .externalLabels.status }}`	例如：Alerting。
告警级别	`{{ .externalLabels.severity }}`	例如：Critical。
告警集群	`{{ .labels.alert_cluster }}`	例如：发生告警的 Cluster 1。
告警对象	`{{ .externalLabels.object }}`	发生告警的资源类型和名称，例如 node 192.168.16.53。
规则名称	`{{ .labels.alert_resource }}`	告警规则名称，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 }}`	监控指标名称。

电子邮件中的特殊格式标记语言

在电子邮件通知中，下表列出了常见的 HTML 格式标签及其说明：

内容元素	标签	描述
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>`，最多支持 h6（header 6）。	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特定联系人发送通知消息的规则。在告警、巡检和登录认证等需要通知外部服务的场景中，必须使用通知规则。

> **INFO**
>
> 平台支持多种通知服务器，并会根据通知服务器配置展示对应通知类型的通知方式。如果未配置通知服务器，则默认不会展示对应的通知方式。

前提条件

要使用 企业通讯工具服务器 通知联系人，用户必须先在 个人信息 中填写其 `WeChat Work ID`，以修改自己的联系信息。

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	接收组是一组逻辑上归类的通知接收人，平台将使用指定的通知方式向其发送通知。
通知接收人	可选择添加一个或多个通知接收人，平台将根据接收人在 个人信息 中配置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、**WebHook URL** 等多种方式，并支持多选。 注意：配置通知服务器后将显示部分参数。
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开模板。

3. 单击 创建。

为项目设置通知规则

平台的通知规则、通知模板和接收组默认按租户隔离。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。平台公开通知模板除外：您可以在创建或更新通知规则时选择它们，但不能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讯工具进行通知，还需要确保待通知联系人已在 个人信息 中正确配置其通讯工具 ID。

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 选项卡，参考 接收组 创建接收组。

> **TIP**
>
> 如果您不需要通过接收组管理通知联系人，或者不需要通知 Webhook 类型通知服务器，可以跳过此步骤。

4. 切换到 模板 选项卡，参考 通知模板 创建通知模板。如果已有合适的平台公开模板，则可以跳过此步骤，并在创建通知规则时选择该模板。
5. 切换到 规则 选项卡，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，提供了 Perses Monitoring Dashboards 作为新的监控面板能力。Perses 监控面板与现有的 MonitorDashboard 能力可以共存。您可以继续使用本页面进行 MonitorDashboard 管理，或者将选定的 MonitorDashboard 资源迁移为 PersesDashboard 资源。

目录

功能概述

该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，为用户提供更全面、更灵活的监控体验。此功能会聚合平台内的多种监控数据，以统一的监控视图呈现，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 namespace 实现数据隔离。
- 支持管理监控面板中的图表，允许用户添加、删除、修改图表，对图表进行放大/缩小，并通过拖拽移动图表。
- 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分组以管理图表。分组可基于自定义变量重复显示。
- 支持的图表类型包括：趋势图、阶梯折线图、柱状图、横向柱状图、条形仪表图、仪表图、表格、Stat 图表、XY 图表、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配置。
- 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。
- 与容器平台的多种功能深度集成，可即时获取容器、网络、存储等监控数据，为用户提供全面的性能观察和故障诊断能力。
- 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 运维团队的一部分，您可以使用监控面板统一展示和管理容器平台的各类性能指标，例如 CPU、内存、网络流量等。通过自定义监控报表和告警规则，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，监控面板提供了多种丰富的可视化选项，可直观展示应用运行状态和资源消耗。您可以根据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，并为优化提供依据。
- 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，让您一眼掌握系统的整体运行状态。
- 故障诊断：当系统出现问题时，监控面板会为您提供全面的性能数据和分析工具，帮助快速定位问题根因。您可以根据告警信息迅速查看相关监控指标的波动，进行深入的故障分析。

前提条件

目前，监控面板仅支持查看平台内已安装监控组件所采集的监控数据。因此，在配置监控面板之前，您需要做好如下准备：

- 确保要配置监控面板的集群已经安装了监控组件，即 **ACP Monitor with Prometheus** 或 **ACP Monitor with VictoriaMetrics** 插件。
- 确保您希望在监控面板中展示的数据已被监控组件采集。

监控面板与监控组件的关系

- 监控面板资源存储在 Kubernetes 集群中。您可以通过顶部的 **Cluster** 选项卡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运行正常。
- 监控面板默认会请求对应集群的监控数据。如果您在集群中以代理模式安装了 **VictoriaMetrics** 插件，我们会替您向存储集群发起请求，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

监控面板是由一个或多个图表组成的集合，按一行或多行进行组织和排列，以清晰展示相关信息。这些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	说明
Folder	监控面板所在的文件夹；您可以输入或选择现有文件夹。
Label	监控面板的标签；在切换时可通过顶部标签筛选快速查找现有监控面板。
Set as Main Dashboard	启用后，创建成功时将当前监控面板设为主监控面板；再次进入监控面板功能时，将默认展示主监控面板数据。
Variables	创建监控面板时可添加变量，在已添加的图表中作为指标参数引用，也可作为监控面板首页的筛选条件。

2. 添加完成后，单击 创建 完成监控面板的创建。接下来，您需要继续 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，并将其转换为用于展示的监控面板。

- 目前仅支持 V8+ 版本的 Grafana JSON；低版本将无法导入。
- 如果导入的监控面板中有任何图表不在平台支持范围内，它们可能会显示为 不支持的图表类型，但您可以修改图表设置以实现正常显示。
- 导入监控面板后，您可以像平常一样进行所有管理操作，其行为与平台中创建的图表没有区别。

添加变量

1. 在变量表单区域中，单击 添加。

查询

`Query` 类型的变量允许您基于时间序列的特征维度筛选数据。可以指定查询表达式，动态计算并生成查询结果。

参数	说明
Query Settings	在定义查询设置时，除了可以使用 PromQL 查询时间序列外，平台还提供了一些常用变量和函数。请参考 Common Functions and Variables。
Regular Expression	通过正则表达式，您可以从变量查询返回的内容中筛选出所需值，使变量中的各个选项名称更符合预期。您可以在 **Variable Value Preview** 中预览筛选后的值是否符合预期。
Selection Settings	- **Multiple Selection**：在监控面板首页顶部筛选器中选中后，允许同时选择多个选项。您需要在图表的查询表达式中引用该变量，以查看与变量值对应的数据。 - **All**：如果勾选，筛选项中将启用包含 **All** 的选项，以选择全部变量数据。

常量

常量变量是具有固定值的静态变量，在整个监控面板中保持不变，通常用于存储环境标识、固定阈值或需要在多个图表间引用但不作为筛选项展示的配置参数。

参数	说明
Constant Value	常量变量的值。

自定义

自定义变量允许用户定义一组预设的静态选项，这些选项会作为下拉筛选器显示在监控面板上，通常用于手动选择特定服务、团队或类别，而无需动态数据查询。

参数	说明
Custom Settings	输入用逗号分隔的选项值，格式为每个选项的 `display_name : value`（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，通常用于指定不需要动态数据查询的特定值或参数。

参数	说明
Textbox Value	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

在当前创建的监控面板中添加多个图表，以展示不同资源的数据。

提示：您可以单击图表右下角来自定义图表大小；单击图表中的任意位置可重新排列图表顺序。

1. 单击 添加图表，参考以下说明配置相关参数。

- **Panel Preview**：该区域会动态展示与已添加指标对应的数据内容。
- **Add Metric**：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使用原生自定义指标，两种方式会取并集并同时生效。

  - 内置指标：选择平台内置的常用指标和图例参数，以展示当前图表下的数据内容。
    - 注意：添加到图表中的所有指标必须使用统一单位；不支持在一个图表中添加多种单位的指标。
  - 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；详情请参考 PromQL Official Documentation ↗。
- 图例参数：控制图表中曲线对应的名称。可使用文本或模板：
  - 规则：输入值必须为 `{{.xxxx}}` 格式；例如，`{{.hostname}}` 会将其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。
- **Instant** 开关：当 **Instant** 开关开启时，会通过 Prometheus 的 Query 接口查询即时值并进行排序，例如统计图表和仪表图；如果关闭，则使用 `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。请参考 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，您可以使用以下操作，确保图表的展示和大小符合预期。
   - 单击图表右下角可自定义其大小。
   - 单击图表中的任意位置可重新排列图表顺序。
   - 单击 编辑 按钮可修改图表设置。
   - 单击 删除 按钮可删除图表。
   - 单击 复制 按钮可复制图表。
5. 调整完成后，单击监控面板页面上的 保存 按钮以保存修改。

添加分组

分组是监控面板中的逻辑分隔，可将图表归组在一起。

1. 单击 添加图表 下拉菜单 > 添加分组，参考以下说明配置相关参数。

- 分组：分组名称。

- 重复：支持禁用重复或选择当前图表的变量。
  - 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。子分组不支持对图表进行修改、删除或移动。

2. 添加分组后，您可以对分组执行以下操作，以管理监控面板中的图表展示。
   - 分组可以折叠或展开，以隐藏监控面板中的部分内容。折叠分组中的图表不会发起查询。
   - 将图表移动到分组中后，该图表将由该分组管理。该分组会管理位于其与下一个分组之间的所有图表。
   - 当分组处于折叠状态时，您还可以将该分组管理的所有图表一并移动。
   - 分组的折叠和展开也属于对监控面板的调整。如果您希望下次重新打开该监控面板时保持此状态，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。再次进入监控面板功能时，将默认展示主监控面板数据。

1. 在左侧导航栏中，单击 **Operations Center** > **Monitoring** > **Monitoring Dashboards**。
2. 默认进入主监控面板。单击 **Switch Dashboard**。
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 **Main Dashboard** 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相应操作。

操作	说明
YMAL	打开存储在 Kubernetes 集群中的监控面板实际 CR 资源代码。您可以通过编辑 YAML 中的参数来修改监控面板的所有内容。
Export Expression	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。

操作	说明
Copy	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。
Settings	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。
Delete	删除当前监控面板。

管理图表

平台提供了多种可视化方式，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。

图表说明

序号	图表名称	说明	建议使用场景
1	趋势图	通过一条或多条线段显示数据随时间的变化趋势。	展示时间上的趋势变化，例如 CPU 利用率、内存使用量等变化。
2	阶梯折线图	在线图基础上，通过水平和垂直线段连接数据点，形成阶梯状结构。	适合展示离散事件的时间点，例如告警数量。
3	柱状图	使用垂直矩形柱表示数据的大小，柱子的高度代表值。	柱状图直观地比较数值差异，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升可读性。
5	仪表图	使用半圆或环形图形表示指标当前值及其占总量的比例。	直观反映关键监控指标的当前状态，例如系统 CPU 利用率、内

序号	图表名称	说明	建议使用场景
			存使用量。建议搭配颜色变化的告警阈值来表示异常情况。
6	条形仪表图	使用垂直矩形条展示指标当前值及其占比。	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，更推荐使用条形仪表图，例如可用磁盘空间或利用率。
7	饼图	使用扇形展示各部分与整体的比例关系。	适合展示不同维度下整体数据的组成，例如一段时间内 4XX、3XX 和 2XX 响应码的占比。
8	表格	以行列格式组织数据，便于查看和比较具体数值。	适合展示结构化的多维数据，例如节点详细信息、Pod 详细信息等。
9	统计图表	展示单个关键指标的当前值，通常需要文字说明。	适合展示重要监控指标的实时值，例如 Pod 数量、节点数量、当前告警数等。
10	散点图	使用笛卡尔坐标系绘制一系列数据点，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，通过数据点分布发现线性相关、聚类等模式，有助于挖掘指标之间的关系。
11	文本卡片	以卡片形式展示关键信息文本，通常包含标题和简短说明。	适合展示文本信息，例如图表说明和故障排查说明。

图表配置说明

通用参数

参数	说明
基本信息	根据所选指标数据选择合适的图表类型，并添加标题和说明；您还可以添加一个或多个链接，通过选择标题旁边对应的链接名称可快速访问。

参数	说明
标准设置	原生指标数据使用的单位。此外，仪表图和条形仪表图还支持配置 **Total Value** 字段，在图表中将以 **Current Value/Total Value** 的百分比形式显示。
工具提示	工具提示是鼠标悬停在图表上时实时数据显示的开关，并支持选择排序。
阈值参数	配置图表的阈值开关；启用后，阈值将在图表中以选定颜色显示，从而支持阈值大小设置。
Value	设置数值的计算方式，例如最近值或最小值。此配置项仅适用于统计图表和仪表图。
Value Mapping	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。此配置项仅适用于统计图表、表格和仪表图。

图表特殊参数

图表类型	参数	说明
趋势图	**Graph Style**	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趋势变化，而面积图则更强调总量和部分占比变化。请根据实际需求选择。
仪表图	**Gauge Chart Settings**	**Show Direction**：当您需要在一个图表中查看多个指标时，可以设置这些指标水平排列或垂直排列。 **Unit Redefinition**：您可以为每个指标设置独立单位；如果未设置，平台将显示 标准设置 中的单位。
统计图表	**Stat Chart Settings**	**Show Direction**：当您需要在一个图表中查看多个指标时，可以设置这些指标水平排列或垂直排列。 **Graph Mode**：您可以在统计图表中添加图形，以展示指标随时间变化的趋势。
饼图	**Pie Chart Settings**	**Maximum Number of Slices**：您可以通过此参数减少饼图中的扇区数量，以降低占比较低但数量较多的类别带来的干扰。超出的扇区将合并并显示为 **Others**。

图表类型	参数	说明
		Label Display Fields：您可以设置饼图标签中显示的字段。
饼图	**Graph Style**	您可以选择 pie 或 donut 作为显示样式。
表格	**Table Settings**	**Hide Columns**：您可以通过此参数减少表格中的列数，聚焦于部分主要列信息。 **Column Alignment**：您可以通过此参数修改列内数据的对齐方式。 **Display Name and Unit**：您可以通过此参数修改所使用的列名和单位。
文本卡片	**Graph Style**	**Style**：您可以选择使用富文本编辑框或 HTML 来编辑需要在文本卡片中显示的内容。

通过 **CLI** 创建监控面板

1. 创建一个名为 `example-dashboard.yaml` 的 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 的监控面板：

```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

在定义查询设置时，除了可以使用 PromQL 设置查询外，平台还提供以下常用函数供您在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 中所有监控指标里该标签名称的所有可选值，例如 `label\_values(job)`。
label_values(metric, label)	返回 Prometheus 中指定指标里该标签名称的所有可选值，例如 `label\_values(up{}, job)`。
metrics(metric)	返回指标字段中符合所定义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 查询的结果，例如 `query\_result(up{})`。

常用变量

在定义查询设置时，您可以将常用函数组合成变量，以便快速定义自定义变量。以下是一些可供参考的常用变量定义：

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)`

变量使用场景一

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

1. 在 **Query Settings** 中填写 `query_result(node_load5)`。
2. 在 **Variable Value Preview** 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 **Regular Expression** 中填写 `/.*instance="(.*?):.*/` 以筛选该值。
4. 在 **Variable Value Preview** 区域，预览示例为 `192.168.176.163`。

变量使用场景二

1. 添加第一个变量 namespace，使用 **query_result(query)** 函数查询 `kube_namespace_labels` 的值，并提取 namespace。
   - **Query Settings**：`query_result(kube_namespace_labels)`。
   - **Variable Value Preview**：`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"}`。
   - **Regular Expression**：`/.+namespace=\"(.*?)\".*/`。

- 在 **Variable Value Preview** 区域，预览示例包含多个 namespace，例如 `argocd`、`cpaas-system` 等。

2. 添加第二个变量 deployment，并引用前面创建的变量：

- **Query Settings**：`kube_deployment_spec_replicas{namespace=~"$namespace"}`。
- **Regular Expression**：`/.+deployment="(.*?)",.*/`。

3. 向当前监控面板添加一个图表，并引用前面添加的变量，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，您可以在监控面板首页查看对应的图表信息。

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 监控面板

目录

功能概述

Perses 监控面板在 Alauda Container Platform 4.4 中作为一项基于开源 Perses 渲染引擎的新监控面板能力引入。你可以查看内置监控面板、创建自定义监控面板、导入监控面板 JSON，以及管理集群的指标图表。

Perses 监控面板与现有的 Monitoring Dashboards 功能共存。现有的 MonitorDashboard 条目和资源仍然可用。当你希望在 Perses 监控面板条目中渲染现有监控面板时，可以将现有的 MonitorDashboard 资源迁移为 PersesDashboard 资源。更多信息请参见 将 MonitorDashboard 迁移到 Perses。

第一阶段聚焦于指标监控面板：

- 支持常见的指标图表类型。
- 平台指标数据源是固定的，不会在 UI 中暴露。
- 此阶段不支持日志、追踪和性能剖析可视化。

前提条件

在使用 Perses 监控面板之前，请先安装以下组件：

组件	安装位置	用途
Alauda Container Platform Dashboard for Perses	`global` 集群运行唯一的 Perses server 实例。工作负载集群按需安装 operator 以提供 PersesDashboard CRD，但不会创建 Perses 实例。	提供 Perses 监控面板后端、渲染和编辑能力。
Alauda Container Platform Dashboard Essentials	仅 `global` 集群	提供 Perses 监控面板使用的、支持 RBAC 的指标查询服务。

安装说明请参见 Installation。如果所选集群尚未安装 **Alauda Container Platform Dashboard for Perses**，Perses 监控面板页面将显示安装指南，而不是监控面板列表。

访问 **Perses** 监控面板

1. 登录到容器平台管理视图。
2. 在左侧导航栏中，进入 **Operations Center** > **Monitor** > **Dashboards (Perses)**。
3. 使用页面顶部的 **Cluster** 选择器，选择要查看其监控面板的集群。

查看监控面板

监控面板列表

监控面板列表会显示你在所选集群中可以访问的 Perses 监控面板。

项目	描述
列	列表显示 **Dashboard**、**Folder** 和 **Created**。
搜索和筛选	你可以按名称、文件夹或标签搜索或筛选监控面板。
内置监控面板	内置监控面板标记为 **Built-In**。内置监控面板的 **Edit** 和 **Delete** 被禁用，而 **Duplicate** 可用。
自定义监控面板	自定义监控面板支持 **Edit**、**Duplicate** 和 **Delete**。

监控面板视图

点击监控面板名称以打开监控面板视图。

在监控面板视图中，你可以：

- 按图表组查看图表。
- 使用监控面板变量筛选图表数据。
- 选择时间范围和刷新间隔。
- 当监控面板支持时，下载监控面板数据。

- 点击 `{}` 以只读模式打开 **Dashboard JSON**。
- 从图表中打开 **Query Viewer**，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是固定的，Query Viewer 不会显示数据源信息。

对于内置监控面板，监控面板视图中的 **Edit** 和 **Settings** 会被禁用。

创建和编辑监控面板

创建监控面板

1. 在 Perses 监控面板列表页面上，点击 **Create**。
2. 配置监控面板信息。

字段	必填	描述
Name	是	监控面板显示名称。
Description	否	监控面板描述。
Folder	是	用于组织监控面板的文件夹。
Tags	否	用于组织和搜索监控面板的标签。

3. 确认创建。平台将打开监控面板编辑器。

资源名称由后端生成。你只需在 UI 中提供监控面板显示名称。

使用监控面板编辑器

监控面板编辑器提供以下主要操作：

操作	描述
Variables	添加或编辑监控面板变量。

操作	描述
Add Panel	向监控面板添加图表。
Add Panel Group	添加图表组以组织图表。
Save	保存监控面板。
Cancel	不保存最新更改并退出编辑。

添加图表

1. 在监控面板编辑器中，点击 **Add Panel**。
2. 在 **Add Panel** 抽屉中，配置图表名称、组、描述和图表类型。
3. 在 **Query** 选项卡中配置查询。

字段	描述
Query language	使用原生 PromQL。指标名称和标签支持自动补全。
Query Type	对于范围查询，使用 **ACP Courier Query**；对于即时查询，使用 **ACP Courier Instant Query**。即时查询通常由 Bar、Gauge 和 Stat 图表用于当前值视图。
Add Query	为一个图表添加多个查询。

4. 根据需要配置其他选项卡，例如 **General Settings**、**Query Settings**、**Links** 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，不会显示数据源选择器。

添加变量

在编辑器中使用 **Variables** 来添加监控面板变量。

变量类型	描述
List	根据指标查询结果动态生成可选择的值。
Text	使用手动输入的文本。文本变量支持 **Constant** 设置。

编辑监控面板 **JSON**

在编辑模式下，点击 `{}` 打开 **Edit Dashboard JSON**。你可以直接编辑监控面板定义。

图表类型

Perses 监控面板在此阶段支持以下指标图表类型：

图表类型	使用场景
Time Series Chart	时间序列趋势。
Bar Chart	柱状对比或排名。
Gauge Chart	当前值或利用率。
Stat Chart	高亮单个值或关键指标。
Pie Chart	占比和组成。
Table	结构化的多维数据。
Scatter Chart	分布和相关性。
Histogram	直方图分布。
HeatMap	热度分布。
Status History	随时间变化的状态。
Markdown	说明文本或章节内容。

此阶段不支持日志、追踪和性能剖析图表。

导入监控面板

1. 在监控面板列表页面上，打开 **Create** 下拉菜单并选择 **Import dashboard**。
2. 配置导入信息。

字段	必填	描述
Name	是	监控面板显示名称。
Description	否	监控面板描述。
Folder	是	用于组织监控面板的文件夹。
Dashboard JSON	是	Perses 或 Grafana 监控面板 JSON。你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。Grafana JSON 会在导入过程中转换为 Perses 模型。该转换是有损的，只保留当前版本支持的图表类型和功能。

管理监控面板

操作	描述
Duplicate	创建现有监控面板的可编辑副本。内置监控面板也可以复制为自定义监控面板。
Settings	更新监控面板名称、描述、文件夹和标签。
Switch	打开对话框，可按文件夹和搜索关键字切换到另一个监控面板。
Delete	删除自定义监控面板。内置监控面板无法删除。

故障排查

页面显示安装指南而不是监控面板

所选集群尚未安装 **Alauda Container Platform Dashboard for Perses**。请在该集群中安装该组件，然后再次打开 **Dashboards (Perses)**。

监控面板没有数据

请检查以下项目：

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

某些内置监控面板显示诸如 `xxx-pd` 之类的资源名称

如果源监控面板不包含 `cpaas.io/display-name` 注解，某些迁移后的内置监控面板可能会显示 Kubernetes 资源名称而不是显示名称。这不会影响监控面板渲染或指标查询。

相关操作

- 将 MonitorDashboard 迁移到 Perses
- Installation
- Monitoring Dashboards

Fleet Monitoring

目录

概述

Fleet Monitoring 为平台管理员提供全局多集群监控视图。它可以帮助你了解哪些集群已连接、监控数据是否最新，以及 fleet 是否存在资源容量或配额风险。

Fleet Monitoring 不能替代单集群监控。对于 fleet 级治理、容量分析和长期趋势，请使用 Fleet Monitoring；当你需要针对特定集群、namespace、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

在使用 Fleet Monitoring 之前，请确保满足以下要求：

- **Alauda Container Platform Fleet Monitoring Central Service** 已安装在 global 集群上。
- `FleetMonitoringHub` 资源存在于 global 集群上。
- **Alauda Container Platform Fleet Monitoring Cluster Service** 已安装在你希望纳入 Fleet Monitoring 的每个集群上。
- `FleetMonitoringAgent` 资源存在于你希望纳入的每个集群上。
- 已连接的集群由 global 集群管理，并且已启用 Monitoring 功能。
- 你有权限查看 Fleet Monitoring 页面和监控数据。

启用步骤请参见 配置 Fleet Monitoring。

访问 **Fleet Monitoring**

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

该页面将显示默认的 Fleet Monitoring 监控面板。使用 **Switch** 可在内置和自定义 Fleet Monitoring 监控面板之间切换。

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

内置监控面板

Fleet Monitoring 包含用于 fleet 级监控的预设监控面板。

Fleet Monitoring 概览

Fleet Monitoring 概览 监控面板是默认的预设监控面板。它可以帮助你回答以下问题：

- 哪些集群已连接到 Fleet Monitoring ?

- fleet 中有多少节点、pod、project、CPU 核心和内存资源？
- 当前 fleet 范围内的 CPU 和内存使用量及请求量处于什么水平？
- 哪些节点的 CPU 或内存使用率更高？
- fleet 级 CPU 和内存利用率趋势如何？

可将此监控面板用于日常 fleet 健康检查、容量审查，以及快速识别需要关注的集群。

该监控面板包含以下信息：

区域	说明
Fleet inventory	集群数量、节点数量、pod 数量、project 数量、CPU 总量、内存总量、过期采集数量以及活跃告警数量。
Fleet utilization	CPU 使用率、CPU 请求率、内存使用率以及内存请求率。
Node ranking	已连接集群中按 CPU 使用量和内存使用量排序的前几个节点。
Utilization trends	CPU 利用率趋势和内存利用率趋势。
Cluster details	集群级资源和利用率详情。

Fleet Monitoring Project Quota

Fleet Monitoring Project Quota 监控面板是一个预设监控面板，可帮助你了解已连接集群中的 project 配额分配和使用情况。

可使用此监控面板审查 project 配额分布，并识别存在配额分配或使用风险的 project。

该监控面板包含以下信息：

区域	说明
Definitions and thresholds	配额、已分配、已使用和使用率的定义，以及正常、高和接近上限使用的阈值含义。

区域	说明
Quota usage overview	project 数量、配额对象数量、高使用量对象数量、CPU 和内存使用率、配额总量、已分配量和已使用量。
Quota distribution	未分配、已分配未使用和已占用资源之间的 CPU 和内存配额分布。
Quota usage ranking	按 CPU 配额使用量和内存配额使用量排序的前几个 project。
Project quota details	project 级配额分配、使用情况和风险详情。

筛选数据

Fleet Monitoring 监控面板提供变量，可帮助你将视图缩小到特定的集群集合或 project 集合。

变量	说明
Cluster Label Key	选择用于筛选的集群标签键。
Cluster Label Value	选择所选集群标签键的值。
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 project。此变量可用于 project quota 监控面板。
Quota Resource Type	在 `limits` 和 `requests` 之间切换。此变量可用于 project quota 监控面板。
Time range	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** 指标只统计实际正在写入 Fleet Monitoring 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 Monitoring** 监控面板

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

可使用以下任一方法：

- 在现有的 Dashboard 页面中，在 global 集群上创建一个监控面板，并通过页面操作添加 `fleet-monitoring` 标签。
- 将 `MonitorDashboard` YAML 资源提交到 **Alauda Container Platform Fleet Monitoring Central Service** 安装所在的命名空间中，且该命名空间位于 global 集群上。确保该资源包含 `cpaas.io/dashboard.tag.fleet-monitoring: "true"` 标签。此标签会为监控面板添加 `fleet-monitoring` 标签。

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: {}
```

请将 `<fleet-monitoring-namespace>` 替换为 **Alauda Container Platform Fleet Monitoring Central Service** 所安装的命名空间。

系统不会验证自定义监控面板是否使用 Fleet Monitoring 指标。监控面板的作者必须确保该监控面板使用适用于 Fleet Monitoring 场景的数据源、指标和变量。

对于多集群监控面板，请使用 `cluster` 标签来标识源集群。如果原始指标已经包含 `cluster` 标签，Fleet Monitoring 会将原始值保留为 `exported_cluster`。

在当前平台查询路径中，当直接查询 Fleet 指标时，Fleet Monitoring 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收窄到 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 Monitoring 的第一个版本有以下限制：

- Fleet Monitoring 不提供专用的多集群告警页面。
- Fleet Monitoring 数据可供现有告警机制使用，但告警规则仍通过现有告警工作流进行管理。
- Fleet Monitoring 不提供供普通用户自助配置采集指标或录制规则的页面。
- Fleet Monitoring 不回填历史数据。仅在启用 Fleet Monitoring 之后才会采集数据。
- Fleet Monitoring 不适用于二级故障排查。请使用单集群监控进行详细排查。

探针管理

目录

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 监控数据的备份与

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

为 Monitoring 规划 Infra 节点

概述
支持的配置方式
配置放置规则前
在控制台中配置放置规则
在 YAML 中配置放置规则
故障排查
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
配置采集间隔
配置自定义指标和录制规则
验证数据新鲜度
故障排查
了解更多

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

Prometheus 监控数据的备份与恢复

目录

功能概述

Prometheus 监控数据以 TSDB（Time Series Database）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' # Prometheus 容器中用于存储监控数据的目录
```

使用场景

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

前提条件

- 已安装 ACP Monitoring with Prometheus 插件（计算组件名称为 `prometheus-kube-prometheus-0`，类型为 `StatefulSet`）
- 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

在开始备份之前，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。以下备份方法使用存储目录作为数据源，因此不会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 `/cpaas/monitoring/prometheus` 目录复制
- **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
   ```

   添加如下配置：

   ```
   spec:
     enableAdminAPI: true
   ```

   注意：更新并保存配置后，Prometheus Pod（Pod 名称：prometheus-kube-prometheus-0-0）将重新启动。请等待所有 Pod 处于 Running 状态后，再继续后续操作。

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 容器中：

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

2. 将数据移动到指定目录：

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

快捷方式：在插件安装时，如果存储类型为 **LocalVolume**，只需将备份数据直接复制到安装了该插件的节点上的 `/cpaas/monitoring/prometheus/prometheus-db/` 目录即可。

操作结果

- 备份完成后，可在目标存储路径中看到完整的 TSDB 格式监控数据
- 恢复完成后，Prometheus 会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

- 备份数据不包含备份时缓存中的数据
- 建议定期进行数据备份
- 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，可选择禁用 Admin API

VictoriaMetrics 监控数据的备份与恢复

目录

功能概述

VictoriaMetrics 监控数据的备份与恢复功能允许您定期备份监控数据，并在必要时恢复数据，确保监控数据的安全性和可用性。

使用场景

- 定期备份监控数据以防止数据丢失
- 系统迁移时的数据迁移
- 灾难恢复
- 重建测试环境数据

前提条件

- 集群中已安装 ACP Monitoring with VictoriaMetrics 插件
- 确保有足够的存储空间用于备份
- 拥有访问 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指定路径中，该路径由 `-storageDataPath` 参数指定，默认值为 `/vm-data`。

配置示例：

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

注意：ACP Monitoring with VictoriaMetrics 插件中计算组件的名称为 `vmstorage-cluster`，类型为 `StatefulSet`。

2. 执行数据备份

使用 vmbackup 工具进行数据备份；详细操作请参考 vmbackup 官方文档↗。

3. 执行数据恢复

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

操作结果

完成备份后，您将获得一份完整的监控数据备份文件。执行恢复操作后，您的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↗
- 数据恢复故障排查↗

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

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

为 Monitoring 规划 Infra 节点

目录

概述

为 Monitoring 插件规划并配置 infra 节点。使用插件配置将 Monitoring 工作负载放置到 infra 节点上，而不是在安装后对生成的工作负载进行 patch。

支持的配置方式

- 对于 **ACP Monitoring with Prometheus**，可在控制台中通过 **Advanced Configuration** 进行配置，也可在 YAML 中通过 Installation 里的 `spec.config.components.nodeSelector` 和 `spec.config.components.tolerations` 进行配置。
- 对于 **ACP Monitoring with VictoriaMetrics**，可在控制台中通过 **Advanced Configuration** 进行配置，也可在 YAML 中通过 Installation 里的 `spec.config.components.nodeSelector` 和 `spec.config.components.tolerations` 进行配置。

不要将对生成的 Deployments、StatefulSets 或其他由插件管理的工作负载进行 patch 作为将 Monitoring 工作负载放置到 infra 节点上的标准方式。

配置放置规则前

在配置放置规则之前，请确保满足以下条件：

- 根据 Cluster Node Planning 规划 infra 节点。
- 确认你的存储是否使用 `LocalVolume`，或是否使用带有 `spec.nodeAffinity` 的其他持久卷。
- 确保所选的 infra 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规则

Prometheus

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

控制台字段	说明
Node Selectors	为 Prometheus 工作负载设置插件级 node selector 规则。
Node Tolerations	为 Prometheus 工作负载设置插件级 toleration 规则。

VictoriaMetrics

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

控制台字段	说明
Node Selectors	为 VictoriaMetrics 工作负载设置插件级 node selector 规则。
Node Tolerations	为 VictoriaMetrics 工作负载设置插件级 toleration 规则。

在 **YAML** 中配置放置规则

Prometheus

如果你希望 Prometheus 插件工作负载运行在专用的 infra 节点上，请在安装或升级期间配置插件级调度规则。

示例：

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

VictoriaMetrics

如果你希望 VictoriaMetrics 插件工作负载运行在专用的 infra 节点上，请在安装或升级期间配置插件级调度规则。

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，你可以选择一个或多个节点。请确保每个所选的存储节点也都匹配已配置的 node selector 规则。

故障排查

Monitoring 工作负载仍然调度到通用节点上

检查以下项：

- 目标节点具有预期的标签。
- 已配置的 tolerations 与 infra 节点上的 taints 匹配。
- 插件已使用最新的调度配置完成升级或重新应用。

Monitoring 工作负载无法调度到所选的 infra 节点上

此问题通常表示所选节点不满足一项或多项调度或存储约束。

常见原因：

- infra 节点没有 `nodeSelector` 引用的标签。
- infra 节点存在未被已配置 tolerations 覆盖的 taints。
- 所选的 `LocalVolume` 节点或 PV `nodeAffinity` 规则指向了 infra 节点组之外的节点。

了解更多

- Installation
- Monitor Component Capacity Planning

后续操作

- 验证 Monitoring 工作负载是否运行在预期的 infra 节点上。
- 评估所选的 infra 节点是否仍然满足容量规划目标。

配置 Fleet Monitoring

目录

了解更多

概述

Fleet Monitoring 使用两个服务：

- **Alauda Container Platform Fleet Monitoring Central Service** 运行在 Global 集群上，用于启用 Fleet Monitoring 的 Global 侧。
- **Alauda Container Platform Fleet Monitoring Cluster Service** 运行在你希望纳入 Fleet Monitoring 的每个集群上。

启用 Fleet Monitoring 后，已连接的集群会将 fleet 级别的指标写入 Global VictoriaMetrics 存储。内置的 Fleet Monitoring dashboard 使用这些指标来展示 fleet 健康状况、资源使用情况、数据新鲜度以及项目配额使用情况。

开始之前

在配置 Fleet Monitoring 之前，请确保满足以下要求：

- 你拥有平台管理员权限，或者拥有安装 Operator 和创建 Fleet Monitoring 资源所需的权限。
- Global 集群具有可用于 Fleet Monitoring 数据的可用 remote write endpoint。支持带有 write endpoint 的 VictoriaMetrics。如果 Monitoring 功能暴露的 endpoint 支持 remote write 流量，也可以使用基于 Prometheus 或 Thanos 的 Monitoring。
- 你希望连接的每个集群都由 Global 集群管理。
- 你希望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 Operators 形式提供，默认不包含在平台安装中。
- New Web Console 插件部署能力可用。Fleet Monitoring Operator 通过 New Web Console OperatorHub 工作流部署。有关更多信息，请参见 安装 New Web Console。

推送 **Fleet Monitoring Operator** 软件包

在从 OperatorHub 安装 Fleet Monitoring 之前，请先使用 `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** 推送到将要安装 Cluster Service 的每个集群。如果还需要将 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>"
```

软件包推送完成后，对应的 Operator 会出现在所选集群的 **Marketplace** > **OperatorHub** 中。有关 `violet push` 的更多信息，请参见 Upload Packages。

在 Global 集群上启用 Fleet Monitoring

1. 进入 **Administrator**。
2. 在左侧导航栏中，点击 **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**，请点击 **Install**，并保持默认安装配置，除非你的环境需要不同的 channel、namespace 或升级策略。

参数	推荐配置
Channel	使用 Operator 软件包提供的默认 channel。

参数	推荐配置
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`，除非你的平台升级策略要求自动升级。

6. 验证 **Alauda Container Platform Fleet Monitoring Central Service** 在 OperatorHub 中的状态为 **Installed**。
   如果你选择 `Manual` 升级策略，并且 OperatorHub 显示待处理的 install plan，请批准该 install plan 以完成安装。
7. 验证 Global 集群具有可用的 VictoriaMetrics write endpoint。
   Fleet Monitoring 使用 Global VictoriaMetrics write endpoint 接收来自已连接集群的数据。如果 Global 集群仅提供 Prometheus 或 Thanos Query，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 cluster-scoped 资源。不要设置 `metadata.namespace`。

字段	必需	描述
`metadata.name`	是	必须为 `fleet-monitoring-hub`。
`metadata.namespace`	否	不要设置此字段。`FleetMonitoringHub` 是 cluster-scoped。
`spec`	是	使用空对象 `{}`。创建该资源会启用 Fleet Monitoring 的 Global 侧。

9. 验证 `FleetMonitoringHub` 的状态。

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

检查以下条件是否就绪：

条件	描述
`ConfigurationReady`	Global 存储和数据库信息可用。
`DashboardsReady`	已应用内置 Fleet Monitoring dashboard。
`GlobalRulesReady`	已应用 Global 录制规则。

你也可以检查 phase：

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

预期的 phase 为 `Ready`。

将集群连接到 **Fleet Monitoring**

对你希望纳入 Fleet Monitoring 的每个集群重复以下步骤。

1. 进入 **Administrator**。
2. 在左侧导航栏中，点击 **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**，请点击 **Install**，并保持默认安装配置，除非你的环境需要不同的 channel、namespace 或升级策略。

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`，除非你的平台升级策略要求自动升级。

6. 验证 **Alauda Container Platform Fleet Monitoring Cluster Service** 在 OperatorHub 中的状态为 **Installed**。
   如果你选择 `Manual` 升级策略，并且 OperatorHub 显示待处理的 install plan，请批准该 install plan 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 cluster-scoped 资源。不要设置 `metadata.namespace`。

字段	必需	描述
`metadata.name`	是	必须为 `fleet-monitoring-agent`。
`metadata.namespace`	否	不要设置此字段。`FleetMonitoringAgent` 是 cluster-scoped。
`spec.interval`	否	采集间隔。支持的值为 `5m`、`10m`、`15m` 和 `30m`。默认值为 `5m`。

如果要将 Global 集群本身也纳入 Fleet Monitoring 数据，请同时在 Global 集群上安装 **Alauda Container Platform Fleet Monitoring Cluster Service**，并在该集群上创建

`FleetMonitoringAgent` 资源。

8. 验证 `FleetMonitoringAgent` 的状态。

```
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，或者由于受支持的原因该集群被有意跳过。

你也可以检查 phase 和检测到的集群名称：

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

预期的 phase 为 `Ready`。常见的 `Ready` 原因包括：

- `WorkloadReady`：集群已部署 VMAgent，并已准备好写入 Fleet Monitoring 数据。
- `SkippedForGlobal`：该集群是 Global 集群，因此 Cluster Service 会跳过向同一个 Global 存储再次部署 VMAgent。
- `SkippedBackendReuse`：该集群复用了 Global VictoriaMetrics 后端，因此 Cluster Service 会跳过部署 Fleet Monitoring VMAgent，以避免写入循环。

如果目标集群将数据写入 Global 存储，请验证数据库信息是否可用：

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

将 `<fleet-monitoring-namespace>` 替换为 **Alauda Container Platform Fleet Monitoring Cluster Service** 所在的 namespace。

9. 打开 **Platform** > **Observe** > **Fleet Monitoring**，并验证该集群是否出现在 dashboard 数据中。

配置采集间隔

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

支持的值：

- `5m`
- `10m`
- `15m`
- `30m`

示例：

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

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

在部署了 Fleet Monitoring VMAgent 的集群上，VMAgent 的采集间隔遵循 `spec.interval`，而 Fleet Monitoring recording rules 仍然以用于 federation 的系统管理间隔进行评估。在 Global 集群以及复用 Global VictoriaMetrics 后端且未部署 Fleet Monitoring VMAgent 的集群上，本地 Fleet Monitoring 规则遵循 `spec.interval`。

配置自定义指标和录制规则

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

当你希望将用户定义的指标报告到 Fleet Monitoring 时，请使用以下流程：

1. 在已连接的 workload 集群上，定义一个本地 Fleet recording rule，将源指标转换为一个 Fleet 指标名称。
2. 将该录制后的指标名称添加到 Fleet allowlist，以便 Fleet Monitoring VMAgent 将其 federate 并 remote-write 到 Global 集群。
3. 如果你需要 fleet 级别的汇总，例如 1 小时聚合，请在 Global 集群上单独添加一个自定义 Hub 侧 `PrometheusRule`。
4. 使用 Fleet Monitoring 查询或 dashboard 验证录制指标和汇总指标。

决定只需要 **Agent** 侧规则，还是同时需要 **Agent** 侧和 **Hub** 侧规则

请选择以下模式之一：

- 当你只需要在 Global 集群上使用原始 Fleet 指标，并且可以直接查询它时，仅使用已连接集群的 ConfigMap。
- 当你还需要 fleet 级别的汇总，例如用于 dashboard 或长时间范围视图的 1 小时聚合时，同时使用已连接集群的 ConfigMap 和 Global 集群上的自定义 `PrometheusRule`。

示例目标：

- 已连接集群上的源指标：`node_load15`
- 已连接集群上录制的 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，创建或更新位于 **Alauda Container Platform Fleet Monitoring Cluster Service** 所在 namespace 中的 `fleet-monitoring-custom-metrics` ConfigMap。

该 ConfigMap 有两个作用：

- `metrics.yaml` 会将指标名称添加到 Fleet Monitoring VMAgent 的 federate allowlist。

- `recording-rules-prometheus.yaml` 或 `recording-rules-victoriametrics.yaml` 定义生成 Fleet 指标的本地 recording rule。

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 **Alauda Container Platform Fleet Monitoring Cluster Service** 所在的 namespace。

`metrics.yaml` 会将指标名称追加到内置 allowlist 中。

对于 recording rules，Cluster Service 只会加载与本地 Monitoring 栈匹配的 key：

- 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 workload 集群上，Agent reconciler 会将渲染后的 Fleet Monitoring recording-rule group interval 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出的本地 Fleet Monitoring 规则间隔。

对于自定义 Fleet 指标，请为录制后的指标使用 Fleet 命名约定，例如 `fleet:node:node_load15:avg`。这样可以使该指标与 Fleet Monitoring dashboard、汇总和查询模式保持兼容。

Fleet Monitoring 的查询和 dashboard 要求录制后的时间序列带有 `cluster` 标签。如果规则尚未定义该标签，Agent reconciler 会自动为渲染后的 Fleet Monitoring recording rules 添加 `cluster=<local-cluster-name>`。

验证已连接集群侧的渲染结果

更新 ConfigMap 后，Fleet Monitoring Agent 会自动重新协调本地规则和 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
```

确认以下内容：

- 自定义 group 出现在 `spec.groups` 中
- 自定义录制指标出现在规则列表中
- 渲染后的规则包含 `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
```

确认自定义录制指标出现在 `data.prometheus.yml` 的 `params.match[]` 下。

添加自定义 **Hub** 汇总规则

如果你希望某个自定义 Fleet 指标具有 fleet 级别的汇总，例如 1 小时聚合，请在 Global 集群上、**Alauda Container Platform Fleet Monitoring Central Service** 所在的 namespace 中创建一个单独的 `PrometheusRule`。

示例：

```yaml
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 **Alauda Container Platform Fleet Monitoring Central Service** 所在的 namespace。

这个自定义 `PrometheusRule` 是附加性的。它不需要复制内置的 Hub 规则，并且不应使用 Fleet Monitoring operator 的所有权元数据创建。

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
```

确认以下内容：

- 该规则存在于 Global 集群上
- 该规则使用预期的 Fleet 指标作为输入
- 汇总输出指标名称与计划使用的 dashboard 或查询表达式匹配

查询自定义 **Fleet** 指标

通过平台 Monitoring API 或 Fleet Monitoring dashboard 查询 Fleet 指标时，请在选择器中显式包含 `vmcluster=~".*"`. 在当前平台查询路径中，省略该选择器可能导致查询代理将查询范围缩小到 Global monitoring 后端，并且不会返回已连接集群的 Fleet 数据。

示例查询：

- 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

请留意以下问题：

- 在 Fleet Monitoring 安装于其他 namespace（例如 `fleet-monitoring`）时，却在 `cpaas-system` 中创建 `fleet-monitoring-custom-metrics`
- 在 `metrics.yaml` 中添加源指标名称，而不是录制后的 Fleet 指标名称
- 定义了本地 recording rule，但没有将录制后的 Fleet 指标名称添加到 `metrics.yaml`
- 认为已连接集群 ConfigMap 中的自定义间隔在渲染后仍然生效
- 查询 Fleet 指标时，选择器中没有包含 `vmcluster=~".*"`
- 在创建对应的自定义 Hub 侧规则之前，就期望看到 Global Fleet 汇总指标

验证数据新鲜度

集群连接完成后，打开 **Platform** > **Observe** > **Fleet Monitoring**，并在概览 dashboard 上检查以下信息：

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

如果某个集群出现在 **Cluster** 变量中，但未被计为已连接，则说明平台可能已识别该集群，但它尚未写入 Fleet Monitoring 数据。请检查该集群是否已通过 OperatorHub 安装 **Alauda Container Platform Fleet Monitoring Cluster Service**，并且 `FleetMonitoringAgent` 是否处于就绪状态。

故障排查

未显示任何 **Fleet Monitoring** 数据

检查以下项目：

- **Alauda Container Platform Fleet Monitoring Central Service** 是否已安装在 Global 集群上。
- `FleetMonitoringHub` 资源是否存在并且条件已就绪。
- Global 集群是否具有可用的 VictoriaMetrics 存储和 write endpoint。仅使用 Prometheus 的 Monitoring 无法接收 Fleet Monitoring remote write 数据。
- 是否已应用内置 dashboard 和 Global 规则。

某个集群未出现在 **Connected Clusters** 中

在目标集群上检查以下项目：

- 是否已安装 **Alauda Container Platform Fleet Monitoring Cluster Service**。

- `FleetMonitoringAgent` 资源是否存在。
- 该集群是否由 Global 集群管理。
- 该集群是否已启用 Monitoring 功能。
- `FleetMonitoringAgent` 状态是否未报告缺失数据库信息或无效的 Monitoring 功能信息。
- `fleet-monitoring-database` Secret 中是否包含指向 Global VictoriaMetrics write endpoint 的 `remoteWriteURL`。
- `fleet-monitoring-vmagent` 日志是否报告 remote write 错误。如果日志显示 `405 Method Not Allowed`，则 `remoteWriteURL` 可能指向了 Prometheus 或 Thanos Query endpoint，而不是 VictoriaMetrics write endpoint。

自定义 **dashboard** 未出现在 **Fleet Monitoring** 中

检查 dashboard 是否创建在 Global 集群上，以及 `cpaas-system` namespace 中的 dashboard 资源是否包含以下标签：

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

没有此标签的 dashboard 不会带有 `fleet-monitoring` 标签，也不会列在 Fleet Monitoring 的 **Switch** 列表中。

数据新鲜度异常

检查以下项目：

- 已连接集群是否正在运行。
- Fleet Monitoring Cluster Service 的 pod 是否健康。
- 已连接集群上的本地 Monitoring 组件是否健康。
- 已连接集群是否可以将数据写入 Global VictoriaMetrics 存储。
- `FleetMonitoringAgent` 状态是否未报告配置或资源应用错误。

了解更多

将 MonitorDashboard 迁移到 Perses

目录

概述

从 Alauda Container Platform 4.4 开始，Perses 监控面板与现有的 MonitorDashboard 功能并行提供。您可以将现有的 MonitorDashboard 资源迁移为 PersesDashboard 资源，并从 **Operations Center** > **Monitor** > **Dashboards (Perses)** 中进行查看。

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

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

前提条件

在迁移监控面板之前，请确保满足以下要求：

- 目标集群已安装 **Alauda Container Platform Dashboard for Perses**。
- `global` 集群已安装 **Alauda Container Platform Dashboard Essentials**。
- 您对源 MonitorDashboard 资源具有读取权限。
- 您对目标 PersesDashboard 资源具有 create 或 write 权限。

有关组件安装，请参见 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)**，并确认已列出迁移后的监控面板且可以打开。

INFO

- 如需迁移多个监控面板，请对每个监控面板分别运行标签命令。您可以批量迁移监控面板。

- 迁移不会删除或修改源 MonitorDashboard。原始监控面板和入口仍然可用。
- 运行该命令的用户必须具有读取 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	说明
图表类型	Grafana 风格的图表类型会映射到 Perses 图表类型，例如 Time Series Chart、Stat Chart、Gauge Chart、Bar Chart、Pie Chart 和 Table。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`。该面板不暴露数据源选择。
显示名称和元数据	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 监控面板 中导入。打开 创建 下拉菜单，选择 导入监控面板，然后提供 JSON 内容。

Grafana JSON 导入使用有损的 Grafana-to-Perses 转换。仅保留当前平台版本支持的图表类型和函数。

迁移后

- 生成的 Perses 监控面板是可编辑的自定义监控面板。
- 源 MonitorDashboard 仍可从现有的监控面板入口访问。

- 如果源面板不包含 `cpaas.io/display-name` 注解，迁移后的监控面板可能会在 Perses 监控面板列表中显示 Kubernetes 资源名称。这不会影响渲染或指标查询。

相关操作

- Perses 监控面板
- 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

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

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

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

Inspection

Execute Inspection

Inspection Configuration

Component Health Status

Procedures to Operate

Inspection Report Explanation

介绍

Inspection 模块是 ACP 平台可观测性套件的核心组件，提供自动化的检查和评估能力，实现对资源的全面监控和风险管理。

该模块提供四项关键的检查功能：

- 资源检查，对集群、节点、Pod、证书及其他平台资源进行自动评估，以识别风险和使用模式
- 实时监控，实时跟踪检查任务进度，立即了解资源的运行状态
- 可视化报告，直观展示检查结果，包括资源风险、使用信息和运行洞察
- 报告生成，支持以 PDF 或 Excel 格式下载检查报告，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评估算法相结合，帮助组织降低人工检查成本，主动识别资源异常，规避业务风险，并通过系统化的健康评估保持平台的最佳性能。

目录

使用限制

- 平台部分检查项依赖于集群已安装监控组件。请确保每个集群事先安装了 ACP Monitoring with Prometheus 插件或 ACP Monitoring with VictoriaMetrics 插件之一。

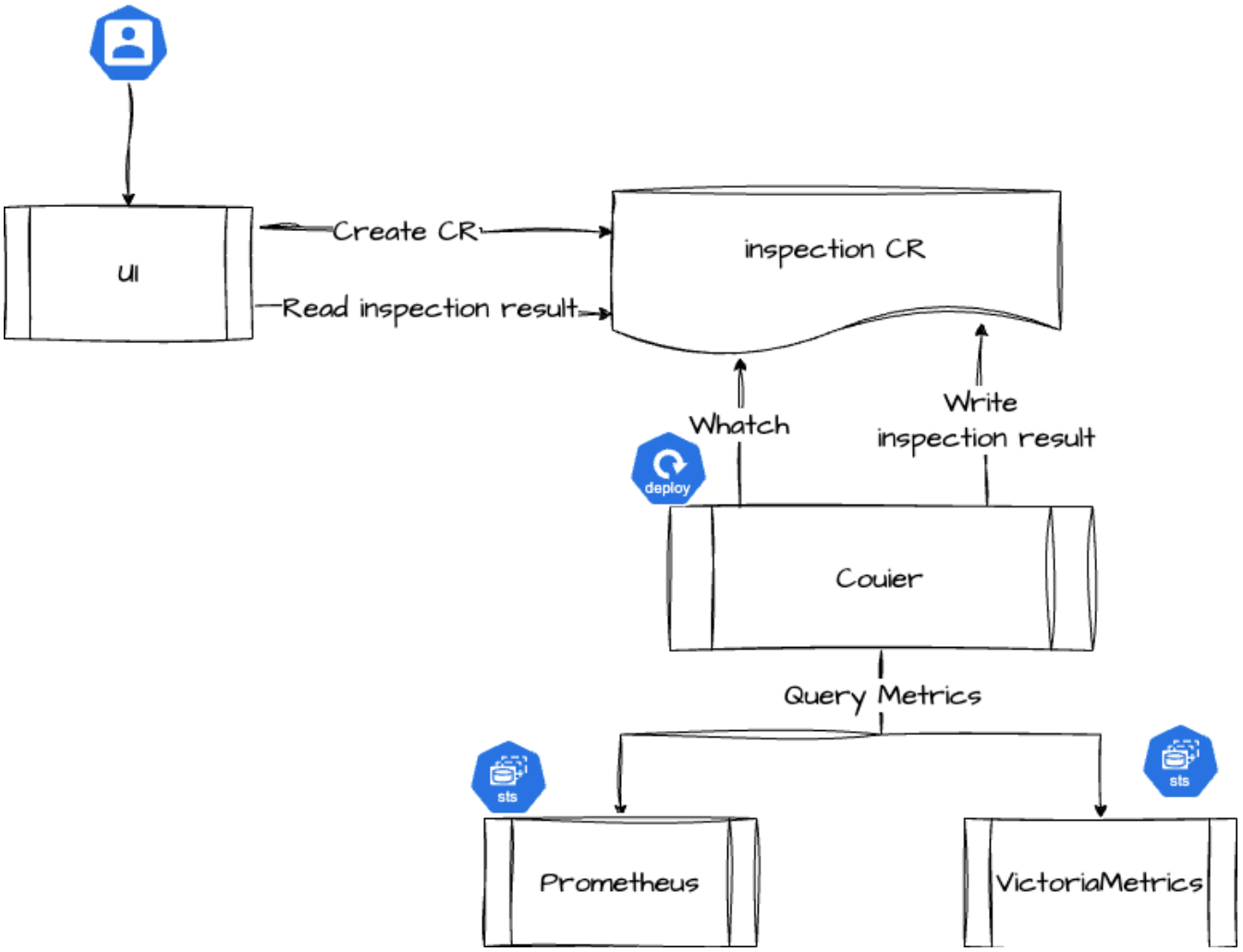
- 平台检查支持通过邮件发送检查结果。请确保邮件通知服务器配置已提前完成。

借助容器平台的检查功能，用户能够更高效地管理和维护容器环境，提升系统的稳定性和安全性。

架构

目录

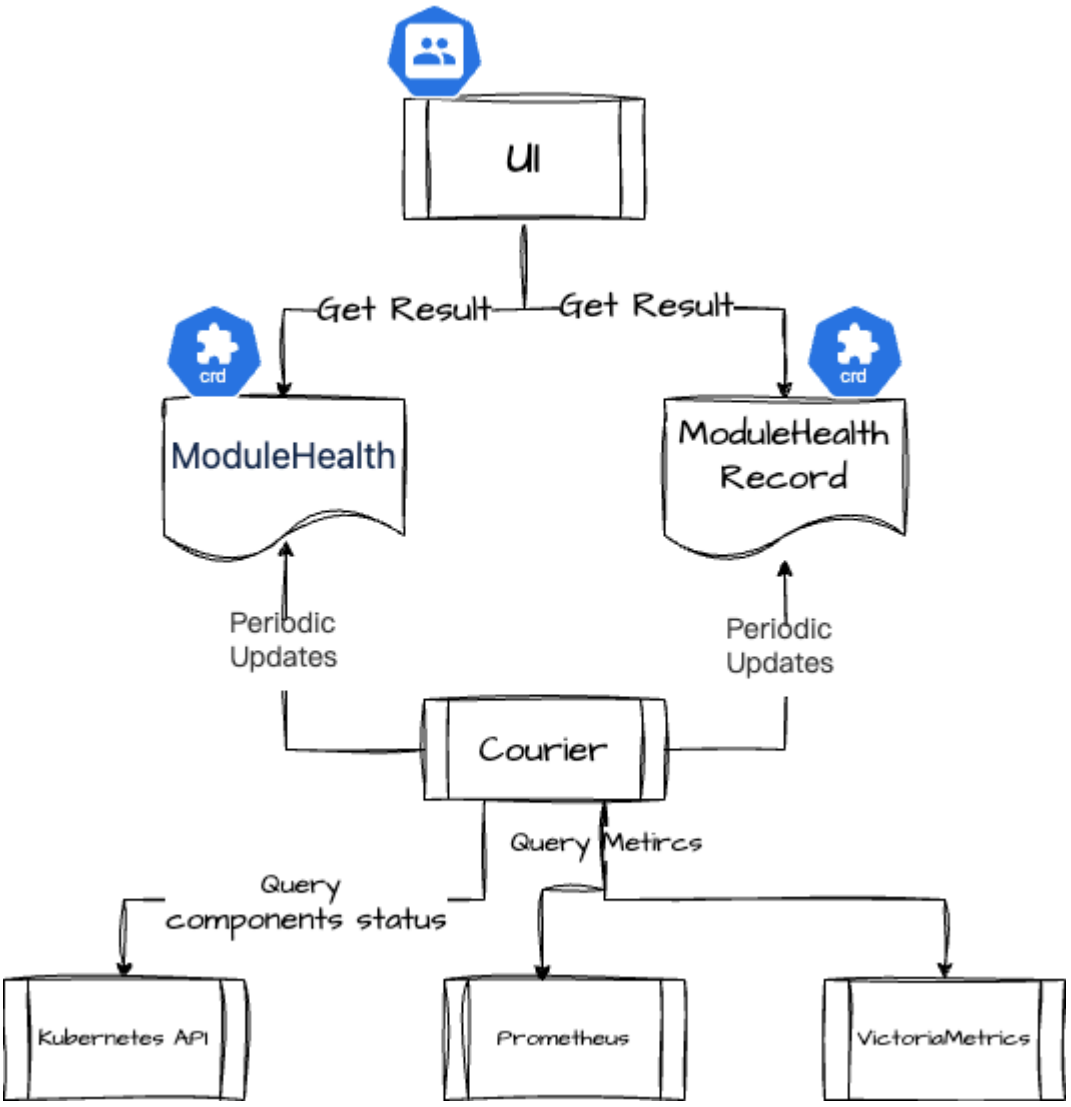
Inspection



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

Inspection

Execute Inspection

Inspection Configuration

Inspection Report Explanation

Component Health Status

Procedures to Operate

Inspection

目录

Execute Inspection

1. 点击左侧导航栏中的 **Operation Center** > **Inspection** > **Basic Inspection**。
   提示：检查页面展示最近一次检查的检查数据信息。检查过程中，可实时查看已完成检查的资源数据。
2. 在 Basic Inspection 页面，支持以下操作：
   - **Execute Inspection**：点击页面右上角的 **Inspection** 按钮，对平台进行检查。
   - **Download Inspection Report**：点击页面右上角的 **Download Report** 按钮，在弹出对话框中选择报告格式（PDF 和 Excel），点击下载，即可将对应格式的报告下载到本地。
     - PDF 格式的检查报告不包含资源风险详情页数据；
     - Excel 格式的检查报告包含检查的全部数据；

- 支持同时下载两种格式的报告。

Inspection Configuration

Inspection Configuration	Description
Scheduled Inspection	自动化任务执行时间规则，支持输入 Crontab 表达式。 提示：点击输入框展开平台预设的 **Trigger Rule Templates**，选择合适的模板，简单修改即可快速设置触发规则。
Inspection Record Retention	保留的检查记录数量。
Email Notification	选择邮件通知联系人。 注意：通知联系人必须配置了邮箱。
Inspection Report Name	平台内置检查通知模板用于通知联系人的名称。
Inspection Configuration Items	根据平台默认的证书、集群主机和 pod 检查项，修改告警阈值或禁用检查项。

Inspection Report Explanation

Most Recent Inspection

在 **Most Recent Inspection** 信息区域，可查看最近一次检查的相关信息：

- **Inspection Time**：最近一次检查的开始和结束时间。
- **Total Number of Inspection Resources**：最近一次检查的资源总数（集群、节点、pod、证书）。
- **Risks**：存在风险的资源数量，包括被归类为 **Fault** 和 **Warning** 的资源。

Resource Risk Inspection

在 **Resource Risk Inspection** 页面，可查看 `global` 集群、自建集群、接入集群及这些集群下所有节点、pod 和证书的风险信息概览。

点击对应资源类型（**Cluster**、**Node**、**pod**、**Certificate**）卡片上的 **Risk Details** 按钮，进入该资源类型的风险详情页。在详情页中，可查看该资源的最近一次检查信息，以及故障和告警资源列表。

- 点击资源名称可跳转至资源详情页。
- 点击列表中 **Name** 字段右侧的展开按钮，可展开故障和告警的判断条件及原因。

各资源风险状态判断标准（Fault、Warning）说明详见下表。

注意：每种资源类型的故障和告警判断条件有多个，资源的检查数据满足任一判断条件即视为一条风险数据。

Resource Type	Inspection Scope	Fault Judgment Conditions	Warning Judgment Conditions
Cluster	- `global` cluster - 自建集群 - 接入集群	- 集群状态为 **Abnormal**； - apiserver 连接异常	- 集群规模（节点数/pod数/指标）增加后，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 组件中任一 pod 状态非 Running； - 集群中任一主机状态非 Ready； - 集群中任意两节点时间差超过 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-**

Resource Type	Inspection Scope	Fault Judgment Conditions	Warning Judgment Conditions
			manager 组件中任一 pod 状态非 Running； - 集群 **kube-scheduler** 组件中任一 pod 状态非 Running； - 集群 **kube-apiserver** 组件中任一 pod 状态非 Running。
Node	- 所有控制节点 - 所有计算节点	- 节点状态为 **Abnormal**； - 节点上的 **node-exporter** 组件 pod 状态非 Running； - 节点上的 **kubelet** 组件 pod 状态非 Running。	- 节点 inode 空闲数小于 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； - 过去一天内至少发生一次 NodeDeadlock（节点死锁）事件； - 过去一天内至少发生一次 NodeOOM（内存不足）事件； - 过去一天内至少发生一次 NodeTaskHung（任务挂起）事件。
pod	所有 pod	- pod 状态为 **Error**； - 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 重启次数大于等于 1。
Certificate	- Certmanager 证书 - Kubernetes 证书	证书状态为 **Expired**。	证书有效期少于 29 天。

Resource Utilization Inspection

点击 **Resource Utilization Inspection** 标签，进入 **Resource Utilization Inspection** 页面。

在 **Resource Utilization Inspection** 页面，可查看 `global` 集群、接入集群、自建集群的 CPU、内存、磁盘的总量、使用量及使用率，以及平台上集群、节点、pod、项目等资源数量。

- **Resource Usage Statistics**：可查看 global、接入和自建集群的 CPU、内存、磁盘总量及总使用率。
- **Platform Resource Quantity**：可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

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。