# 应用运维管理

最佳实践

文档版本01发布日期2024-05-27





#### 版权所有 © 华为云计算技术有限公司 2024。保留一切权利。

非经本公司书面许可,任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部,并不得以任何形式传播。

#### 商标声明

NUAWE和其他华为商标均为华为技术有限公司的商标。 本文档提及的其他所有商标或注册商标,由各自的所有人拥有。

#### 注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束,本文档中描述的全部或部 分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定,华为云计算技术有限公司对本文 档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因,本文档内容会不定期进行更新。除非另有约定,本文档仅作为使用指导,本文 档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

### 华为云计算技术有限公司

地址: 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编: 550029

网址: <u>https://www.huaweicloud.com/</u>



目录

1 最佳实践概述	1
2 建设完整指标体系,实现立体化监控	2
3 通过告警降噪清除告警风暴	16
4 通过多账号聚合 Prometheus 实例实现指标数据统一监控	21
5 CCE 容器场景自建中间件接入	30
5.1 PostgreSQL Exporter 接入	30
5.2 MySQL Exporter 接入	35
5.3 Kafka Exporter 接入	40
5.4 Memcached Exporter 接入	43
5.5 MongoDB Exporter 接入	
5.6 ElasticSearch Exporter 接入	52
5.7 Redis Exporter 接入	55
5.8 其他 Exporter 接入	60



本章汇总呈现应用运维管理 AOM的最佳实践。

- 建设完整指标体系,实现立体化监控
- 通过告警降噪清除告警风暴
- 通过多账号聚合Prometheus实例实现指标数据统一监控
- 自建中间件接入(CCE容器场景)
  - PostgreSQL Exporter接入
  - MySQL Exporter接入
  - Kafka Exporter接入
  - Memcached Exporter接入
  - MongoDB Exporter接入
  - ElasticSearch Exporter接入
  - Redis Exporter接入
  - 其他Exporter接入

# 2 建设完整指标体系,实现立体化监控

本文档介绍如何建设完整的指标体系和统一监控大盘,实现资源和应用的全方位、立体化、可视化监控。

#### 实践场景

用户体验至上的互联网时代,页面的响应速度、访问时延和页面的访问成功率常常会 影响用户的体验,如果无法及时获知,就会导致流失大量用户,某商城的运维人员使 用开源的监控软件,虽然能采集很多指标,但却分散在各处,无法统一展示。

#### 解决方案

AOM能够实现云上应用的一站式立体化运维管理,在接入中心中可以接入需要监控的 业务层、应用层、中间件层、基础设施层指标,在仪表盘中实现个性化监控,以及通 过统一告警入口配置告警规则,实现业务的日常巡检,保障业务的正常运行。

AOM提供多场景、多层次、多维度指标数据的监控能力,建立了从基础设施层指标、 中间件层指标、应用层指标到业务层指标的四层指标体系,将1000+种指标数据全方 位呈现,数据丰富全面。

表 2-1 AOM 支持的四层指标体系

类型	来源	指标举例	如何接入	
业务层指 标	通常来源于端侧日志 SDK、提取的ELB日 志。	访问UV、访问PV、访问 延时、访问失败率、访 问流量情况等	接入业务层指标	
	通常来源于事务监控 或上报的自定义指 标。	URL的调用次数、URL 的最大并发数、URL的 最大响应时间等		
应用层指 标	通常来源于组件性能 图表或接口性能数 据。	接口调用次数、请求平 均时延、错误调用次 数、请求吞吐量等	接入应用层指标	
中间件指 标	通常来源于原生中间 件或云中间件数据。	文件系统容量、文件系 统使用率等	接入中间件指标	

类型	来源	指标举例	如何接入
基础设施 层指标	通常来源于容器或云 服务相关数据,例如 计算、存储、网络、 数据库等。	CPU使用率、内存使用 率、健康状态等	接入基础设施层指标 • 接入容器指标 • 接入云服务指标

#### 图 2-1 AOM 四层指标体系



#### 前提条件

- 已将ELB日志接入LTS。
- 已为环境关联ECS资源。

#### 步骤一:建设四层指标体系

步骤1 接入业务层指标。

- 1. 登录AOM 2.0控制台。
- 2. 在左侧导航栏中选择"接入中心"。
- 3. 在右侧"业务层"面板单击需要接入的指标卡片。
  - 接入ELB 日志指标
     系统可自动接入,无需用户手动操作。

在左侧导航栏,选择已创建的仪表盘,单击页面右上角的<sup>444</sup>,输入对应 SQL语句,即可在仪表盘中查看该日志指标。以查看流量指标为例,输入对 应SQL语句,单击"查询",如<mark>图2-2</mark>所示。

#### 图 2-2 查看流量指标

★ 图表ł	示题 流量		
指标源	日志源		
Ē	志组: phoenixap ▼	活流: elbStreamL ・	曲 15分钟(相对)▼ □ ▼
7	1 SELECT * *TIME_FORMAT(_	ime_'yyyy-MM-dd HH:mm:ss','+08:00)AS_time_round(CASS WHEN"人注量" > 0 THEN	© 🛛 🔿
	-9 统计李段 搜索字段 Q	協会 ダメ 应用子役未直 ELB7层放技中心_phoenkapp.orderingcomp.phoenka 新 回目代表: 由車精論	建保存 另存为 下载 展开图表
	time	しが新聞のなどの意思である。	通用配置
	mme msec    access_log_topic_id     time_iso8601     log_ver	L:         → ARB         → URB           O         #00         0.0           B	E M K C B
	remote_addr remote_port remote_port	& 02 0.15	<ul> <li>标准配置</li> <li>查询分析设置</li> </ul>
	T request_method T scheme	0.1	▼ 图例配置
	T host T router_request_uri	0 2022- <u>2</u> 2022- <u>2</u> 2022- <u>2</u> 2022- <u>2</u> 2022- <u>2</u> 2022- <u>2</u>	<ul> <li>■ B2/7/HC/展</li> <li>Tooltip用ご籤</li> </ul>
	T server_protocol request_length	20.00.00 20:50:00 21:40:00 22:30:00 23:20:00 00:10:00	▼ X轴

- 接入APM事务指标
  - i. 为工作负载安装APM探针,具体操作请参见安装APM探针。
  - ii. 安装完成后,请登录安装探针的服务对应的控制台界面,执行操作触发 APM事务指标的采集。以本实践场景中的商城服务为例,可以在商城操 作界面将对应商品添加到购物车。
  - iii. 登录AOM 2.0控制台。
  - iv. 在左侧导航栏选择"指标浏览"。在右侧区域通过选择指标的方式查看 接入的APM指标。
- 步骤2 接入应用层指标。
  - 1. 为工作负载安装APM探针,具体操作如下:
    - a. 登录CCE控制台,单击集群名称进入集群。
    - b. 在左侧导航栏中选择"工作负载",选择需要上报到AOM的工作负载类型。
    - c. 单击工作负载名称,选择"性能管理配置",单击右下角"编辑",修改 "性能管理配置"相关信息。
    - d. 选择"APM 2.0探针",设置"探针版本"为"latest-x86","APM环境" 为"phoenixenv1",从"APM应用"的下拉列表中选择创建的 "phoenixapp1"应用。
    - e. 设置完成后,单击"保存"。
  - 安装完成后,请登录安装探针的服务对应的控制台界面,执行操作触发应用层指 标的采集。以本实践场景中的商城服务为例,可以在商城操作界面将对应商品添 加到购物车。
  - 3. 登录AOM 2.0控制台。
  - 在左侧导航栏选择"指标浏览"。在右侧区域通过选择指标的方式查看接入的应用层指标。

步骤3 接入中间件指标。

- 1. 将数据上传到ECS服务器。
  - a. 下载mysqld\_exporter-0.14.0.linux-amd64.tar.gz软件包,下载地址: https://prometheus.io/download/。
  - b. 以root用户登录ECS服务器,将下载的Exporter软件包上传到ECS服务器并解 压。
  - c. 登录RDS 控制台,在"实例管理"界面实例列表中单击一个RDS实例名。在 "基本信息"界面查看RDS安全组。

#### 图 2-3 查看 RDS 安全组

< rds-aom 🧿 正常 🔞					
基本信息 1	数据库信息				
备份恢复	实例名称	rds-aom 🖉 🗇			
连接管理	实例备注	🖉	管理安全组		
秋号管理	可維护时间段 (?)	02:00 - 06:00 (GMT+08:00) 修改	实例名称	rds-aom	
数据库管理			选择安全组		
日志管理	性能规格	rds.mysql.c6.large.2   2 vCPUs   4 G			
SQL审计	管理员帐户名	root 重置密码	aom-rds-sg 🔮		
参数修改	±/// ~~~~ @		顺序	安全组 3	操作
高级运维	爭忤定时 辭 🕐		1	aom-rds-sg	上移   7
智能DBA助手 🔻			<ol> <li>安全组规则</li> </ol>	先根据绑定安全组的顺序生效。	再根据组内规则的优先级生效
	连接信息				
	内网地址	192.168.0.198 🗇 修改			是否
	虚拟私有云	vpc-aom		如此手術口	5500 E ()
	子网	subnet-aom(192.168.0.0/24)		建议最大连接数	1,500
	安全组	2 1个安全组 管理			

d. 检查RDS的安全组是否已开放3306端口。

#### 图 2-4 检查 RDS 端口是否开放

<	< aom-rds-sg								
基本	信息 入方向规则	出方向规则 关联实例							
[	安全组规则对不同规	格的云服务器生效情况不同,如果您的安	全组规则未生效,请查看 安全组规则限制	J.					
	添加规则 快速器 通过指定属性的关键字搜	<b>森加規則                                     </b>	入方向规则:8 教我设置						
	□ 优先级 ⑦	▽ 策略 ⑦	〒 协议端口 ⑦	类型	源地址 ⑦				
	1	允许	TCP : 3306	IPv4	0.0.0.0/0 ⑦				
	1	允许	TCP : 20-21	IPv4	0.0.0.0/0 ⑦				
	1	允许	TCP : 443	IPv4	0.0.0.0/0 ⑦				
	1	允许	TCP : 3389	IPv4	0.0.0.0/0 ⑦				
	1	允许	全部	IPv4	aom-rds-sg 🕐				
	1	允许	TCP : 22	IPv4	0.0.0.0/0 ⑦				
	1	允许	ICMP:全部	IPv4	0.0.0.0/0 ⑦				
	1	允许	TCP : 80	IPv4	0.0.0.0/0 ⑦				

e. 执行以下命令,进入解压文件夹,并在ECS服务器上配置mysql.cnf文件。 cd mysqld\_exporter-0.14.0.linux-amd64 vi mysql.cnf

例如,在mysql.cnf文件中添加如下内容:

[client]

user=*root(rds用户名)* password=*\*\*\*\*(rds密码)* host=*192.168.0.198(rds公网IP)* 

port=*3306(端口)* 

- f. 执行以下命令,启动mysqld\_exporter工具。 nohup ./mysqld\_exporter --config.my-cnf="mysql.cnf" --collect.global\_status -collect.global\_variables &
- g. 执行以下命令,确认工具是否正常启动。 curl http://127.0.0.1:9104/metrics

如果回显信息如<mark>图</mark>2-5所示,能够查看到指标则说明工具启动正常。

#### **图 2-5** 查看指标

curt. (7) Faited to connect to 127.0.0.1 port 9105. connection refused
[root@aom-elb mysqld_exporter-0.14.0.linux-amd64]# curl <u>http://127.0.0.1</u> :9104/metrics
# HELP go_gc_cycles_automatic_gc_cycles_total Count of completed GC cycles generated by the Go runtime.
# TYPE go_gc_cycles_automatic_gc_cycles_total counter
go_gc_cycles_automatic_gc_cycles_total 0
# HELP go_gc_cycles_forced_gc_cycles_total Count of completed GC cycles forced by the application.
# TYPE go_gc_cycles_forced_gc_cycles_total counter
go_gc_cycles_forced_gc_cycles_total 0
# HELP go_gc_cycles_total_gc_cycles_total Count of all completed GC cycles.
# TYPE go gc cycles total gc cycles total counter
go_gc_cycles_total_gc_cycles_total 0
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0
go_gc_duration_seconds{quantile="0.25"} 0
go_gc_duration_seconds{quantile="0.5"} 0
go_gc_duration_seconds{quantile="0.75"} 0
go_gc_duration_seconds{quantile="1"} 0
go_gc_duration_seconds_sum 0
go_gc_duration_seconds_count 0
# HELP go_gc_heap_allocs_by_size_bytes_total Distribution of heap allocations by approximate size. Note that this does not include tiny objects
allocs:objects, only tiny blocks.
# TYPE go_gc_heap_allocs_by_size_bytes_total histogram
go_gc_heap_allocs_by_size_bytes_total_bucket{le="8.999999999999999999999999999999999999
go_gc_heap_allocs_by_size_bytes_total_bucket{le="24.9999999999999999999"} 6825
go_gc_heap_allocs_by_size_bytes_total_bucket{le="64.999999999999999999"} 9509
go_gc_heap_allocs_by_size_bytes_total_bucket{le="144.99999999999999999"} 11832
go_gc_heap_allocs_by_size_bytes_total_buckettle="320.999999999999999994"} 13069
go_gc_heap_allocs_by_size_bytes_total_bucket{le="704.9999999999999999999"} 13539

- 2. 通过虚机接入方式接入中间件指标。
  - a. 登录AOM 2.0控制台。
  - b. 在左侧导航栏中选择"接入中心",在右侧"中间件"面板单击需要接入的 指标卡片。
  - c. 在"虚机接入"界面为ECS服务器安装UniAgent采集工具,具体操作请参见 **手动安装UniAgent**。
  - d. 为创建的"phoenixenv1"环境创建中间件采集任务。在左侧导航栏中,选择 "采集任务",单击"新增采集任务"。

#### 图 2-6 创建采集任务

采集管理	2 中间件 自定义	+ 新增采集任务 Q 请输入搜索条件
UniAgent管理 🔹 🔻		
插件市场	采集任务	采集插件
		REDIS

e. 在创建采集任务页面中,配置相关参数信息。

#### 🛄 说明

关键配置项说明如下:

- Exporter地址/REDIS\_Exporter地址/MySQL\_Exporter地址:安装Exporter/ REDIS\_Exporter/MySQL\_Exporter的主机IP地址和端口号。格式为IP:Port,例如: 127.0.0.1:9104。
- 中间件地址/REDIS 地址/MySQL 地址: Exporter/REDIS\_Exporter/ MySQL\_Exporter启动实例监控的主机/REDIS/MySQL,通常填写主机/REDIS/ MySQL的IP地址。
- 指标:待采集的指标。默认为:"(单引号)表示输出原本的所有指标。如果需对 采集指标进行过滤,则按如下格式进行填写,例如:'metric1, metric2'。
- 指标维度:新增采集指标维度。单击 + Tag ,输入指标维度名和指标维度 值,字符长度不超过20个字符。最多可添加10个标签。例如:指标维度名为 label1,指标维度值为label2,则添加成功后为label1:"label2"。

#### **图 2-7** 配置采集参数-1

#### 选择实例

* Prometheus实例 ⑦	
uniagent-default *	○ 创建普罗实例
插件配置	
操作系统	
O ∆ Linux O . Windows	
* 采集插件	
MySal COMP_MYSQL_EXPORTER	
* 插件版本	
_ 1.0.0 ·	Ø
采集任务	
* 采集任务名	
mysql_metrics	
* 主机	
温馨提示: 主机需要安装UniAgent。	
<ul> <li>添加主机</li> </ul>	

插件采集配参		
• *MySQL_Exporter地址	0	
127.0.0.1:9104		
Prometheus Exporter		
•*MySQL地址 ②		
192.168.0.198		
•*指标 ⑦		
н		
指标维度		
job namespace	exporter instance target + Tag	
言語記書 へ		
★ 采集周期(秒)		
600		_
005		•
▲ 控制市门(利))		
× 204343163(422)		
60s		*
★ 执行用户		
root		

f. 完成后,单击"立即创建"。

**图 2-8** 配置采集参数-2

- 接入完成后,在左侧导航栏,选择"指标浏览"。在右侧区域通过选择指标的方 式查看接入的中间件指标。
- 步骤4 接入基础设施层指标。
  - 1. 登录AOM 2.0控制台。
  - 2. 在左侧导航栏中选择"接入中心"。
  - 3. 在右侧"运行环境"与"云服务"面板单击需要接入的指标卡片。
    - 选择容器指标卡片:

以选择"云容器引擎CCE"卡片为例,具体操作如下:

i. 在"插件市场"界面搜索找到云原生监控插件,单击"安装"。

云容器引擎	插件市场
集群管理	
权限管理	
插件市场	
镜像仓库 🖸	插件类型: 全部 容器调度与弹性 云原生可观测性 云原生异构计算
监控中心 🖸	插件来源: 全部 CCE 开源
	未安装
	云原生监控插件 CCE Monitoring
	kube-prometheus-stack通过使用Prometheus-operator和
	▲ Prometheus,提供简单易用的端到端Kubernetes集群监控能力…

#### 图 2-9 查找云原生监控插件

- ii. 设置集群名称、规格信息。
- iii. 开启对接第三方开关,将普罗数据上报至AOM,填写上报AOM的地址 及token,并跳过证书认证。

图 2-10 设置对接第三方参数



#### 参数说明如下:

 数据上报地址: https://aom-internal-access. {region\_name}.myhuaweicloud.com:8443/v1/{project\_id}/push, 其中region\_name为指定承载REST服务端点的服务器域名或IP, project\_id 为项目的ID,您可以单击右上方的用户名称,在下拉列 表中选择"我的凭证"。在"我的凭证"页面中选择"API凭证"页 签。从右侧项目列表的"项目"和"项目ID"信息中获取。例如 AOM服务在"华北-北京一"区域名称为"cn-north-1"。

- Token: 登录AOM 2.0控制台,在左侧导航栏选择"设置",单击 "认证管理",从右侧AccessCode列表的"ID"信息中获取。
- iv. 设置完成后,单击"安装"。安装完后单击插件查看安装状态,当状态 都为"运行中"则表示插件安装成功。

已安装插件				实制名称	秋志 🏹	命名空间	实例IP	所在节点 🏹	重启	E CPU 申请值限制值使用	内存 申请值,限制值,使用:	创建时间	操作	
kube-prometheus-stack			prometheus-s	erve 🛛 运行中	monitoring	. 172.16.0.27	192.168.0.229	0	0.6 Cores 2.2 Cores 1.04%	2.05 GIB 8.1 GIB 16.62%	13 小时前	监控 事件 更多 ▼		
	版本 状态 更新时间		3.4.25 ● 运行中 13 小时前		custom-metric	s-ap 🐠 运行中	monitoring	. 172.16.0.38	192.168.0.174	0	0.4 Cores 2 Cores 0.12%	400 MiB 1 GiB 3.87%	13 小时前	监控 事件 更多 ▼
	術组	1	Ditt	C	kube-state-m	itrics 🛛 运行中	monitoring	. 172.16.0.36	192.168.0.174	0	0.2 Cores 0.5 Cores 0.05%	200 MiB 500 MiB 4.70%	13 小时前	监控 事件 更多 ▼
<b>可安装器件</b>		C	node-exporte 主机用格	-18w • 运行中	monitoring	. 192.168.0.174	192.168.0.174	0	0.2 Cores 0.5 Cores 0.01%	200 MIB 1 GIB 2.24%	13 小时前	监控 事件 更多 ▼		
			node-exporte 主机网络	-xxg • 运行中	monitoring	. 192.168.0.229	192, 168, 0, 229	0	0.2 Cores 0.5 Cores 0.56%	200 MIB 1 GIB 0.40%	13 小时前	监控 事件 更多 ▼		
<b>空</b> 接				prometheus-c	pera 🛛 透行中	monitoring	172.16.0.37	192.168.0.174	0	0.2 Cores 0.5 Cores	200 MIB 500 MIB	13 小时间	监控 事件 更多 ▼	

图 2-11 安装 kube-prometheus-stack 插件

- 选择云服务监控指标卡片:
  - i. 在弹出的"云服务接入"对话框中选择需要监控的云服务。例如RDS或 DCS服务。
  - ii. 单击"确定"完成接入。
     接入完成后,系统自动跳转至"云服务监控"页面,即可查看已选择的 云服务运行状态等信息。
- 接入完成后,在左侧导航栏选择"指标浏览"。在右侧区域通过选择指标的方式 查看接入的基础设施层指标。

----结束

#### 步骤二: 配置统一监控大盘

步骤1 创建指标告警规则。

通过指标告警规则可对资源的指标设置阈值条件。当指标数据满足阈值条件时产生阈 值告警,当没有指标数据上报时产生数据不足事件。

按照配置方式的不同,创建指标告警规则可分为三种:**按资源类型创建、按全量指标** 创建和按Prometheus命令创建。下面的操作以按资源类型创建为例说明。

- 1. 登录AOM 2.0控制台。
- 2. 在左侧导航栏中选择"告警管理 > 告警规则"。
- 3. 单击"创建告警规则"。
- 4. 设置告警规则的规则名称等基本信息。
- 5. 设置告警规则的详细信息。
  - a. 选择"规则类型"为"指标告警规则"。
  - b. 选择指标配置方式为"资源类型",设置资源类型和监控对象信息。 选择监控对象时,如果开启了"应用到所有"开关,将会针对应用或服务下的所有该类型指标创建一条告警规则。例如选择了"CCE/主机/主机/CPU 使用率"指标,开启应用到所有开关,则会为CCE服务下所有主机创建一条 告警规则。
  - c. 设置告警规则详情。

指标告警的检测规则,由统计方式(平均值、最小值、最大值、总计、样本 个数)、判断条件(>=、<=、>、<)和阈值组成。例如,检测规则设置为 "平均值>1",表示指标的平均值大于已设置的阈值1时,生成指标告警。

#### 图 2-12 设置告警详细信息



- d. 单击"高级设置",设置检查频率、告警恢复等信息。
- 6. 设置告警通知策略。告警通知策略有两种方式,此处选择直接告警方式。
   直接告警:满足告警条件,直接发送告警。选择直接告警方式,需要设置通知频率和是否启用告警行动规则。
  - a. 设置发送告警通知的频率,请根据需要从下拉列表中选择。
  - b. 设置是否启用告警行动规则。启用告警行动规则后,系统根据关联SMN主题 与消息模板来发送告警通知。
  - c. 启用告警行动规则后,需要设置是否开启告警恢复通知。开启告警恢复通知
     后,当满足告警条件中设置的告警恢复条件,则按照选择的告警行动规则发送告警恢复通知。

#### 图 2-13 设置直接告警方式

十敏	: 留知
台号	迎州

通知场景



告警方式

通知频率 只告警─次 * 行动规则 ●	直接告警	告警降噪		
只告警─次 ▼	通知频率			
行动规则	只告警一次		•	
	行动规则			
Monitor_host • S 🗟	Monitor_host		•	S E

9. 单击"立即创建",完成创建。创建完成后,单击"返回告警规则列表"可查看已创建的告警规则。

如下图所示,创建了一条指标告警规则,单击规则名称前的之,可查看该告警规则的详细信息。

在展开的列表中,只要某个主机的CPU使用率满足设置的告警条件时,在告警界 面就会生成一条指标类告警,您可在左侧导航栏中选择"告警管理 > 告警列 表",在告警列表中查看该告警。只要某个主机满足已设的通知策略,系统就会 以邮件、短信或企业微信等方式发送告警通知给指定人员。

#### 图 2-14 创建指标告警规则

	报则名称与类型	规则状态	监控对象	告留条件	行动规则	关联Prometheus实例	启伊秋念	操作
•	Monitor_host 指标告答	正常	CCE-主机 1 个监控对象	CPU使用率。连续3次(每次统计最近1				/ 0 8
基本信息 1	监控对象 <b>告警条件</b> 触发告答							
告警条件	告留条件					告密级别		
	CPU使用率. 连续3次(每次统计最近1分	计种数据)平均值大于18	th*生			0 東急告留		
检查质率	国地间隔1分钟							
舌醫恢复	当监控对象在最近1个监控周期内不满足触》	线条件时,已产生的告誓	将自动恢复。					
无数据处理	关闭							

#### 步骤2 创建仪表盘。

- 1. 新建仪表盘。
  - a. 登录AOM 2.0控制台。
  - b. 在左侧导航栏选择"仪表盘"。
  - c. 单击列表左上角的"创建仪表盘"。
  - d. 在弹出的"新建仪表盘"对话框中,设置相关参数。
     将仪表盘绑定到事先创建的应用,后续可以在"应用监控"页面可视化监控 应用的关键指标。

#### **图 2-15** 新建仪表盘

	* 仪表盘名称 ②	四层指标-凤凰商城
	* 企业项目	default
	绑定到应用	phoenixapp 🔹
	分组类型	○ 已有分组 ○ 新建分组
	选择仪表盘分组	Promehteus •
		创建取消
	e. 设置完成,单击 "仓	]建"。
2.	为仪表盘添加可视化图表	ξ <sub>o</sub>

- a. 在仪表盘列表中,单击已创建的仪表盘。
- b. 进入对应仪表盘页面,单击页面右上角的 <sup>山</sup>,为该仪表盘添加图表。请根据需要,选择合适的图表。

表 2-2 添加图表

添加图表类型	数据来源	使用场景
请添加指标图表	指标数据	监控基础设施层、中间件、应用层和业 务层指标。
请添加日志图表	日志数据	监控业务指标或其他日志指标,如基于 ELB日志清洗出来的接口黄金指标(时 延、吞吐和错误)。

下面以添加"CPU使用率"的指标图表和"延迟"的日志图表为例说明。

添加"CPU使用率"的指标图表。
 选择"CPU使用率"指标,设置完成后,添加的指标图表如图2-16所示。

#### 图 2-16 添加指标图表



添加"延迟"的日志图表。单击"日志源",设置日志图表的相关参数。

可直接从图表中获取SQL查询语句:

1) 在图表展示区右上方单击"展开图表"。

#### 图 2-17 展开图表

🛄 1593th(1883) * 🔅 *
0 I 0 i
新雄 保存 男存为 下紙 展开選択

在"可视化图表"列表中选择需要监控的日志指标,例如"延迟",如图2-18所示。

**图 2-18** 选择日志指标



3) 该指标对应的查询语句会自动填充到SQL语句设置区。

参数设置完成后,单击"添加至仪表盘"。添加的日志图表如<mark>图2-19</mark>所示。

#### 图 2-19 添加日志图表





c. 可重复上面的操作为仪表盘添加多个可视化图表。添加完成后,单击动,保存仪表盘,如图2-20所示。

#### 图 2-20 添加可视化图表



----结束



本文档介绍如何为告警规则配置告警降噪功能,在发送告警通知前按告警降噪规则对 告警进行处理,处理完成后再发送通知,避免产生告警风暴。

#### 实践场景

应用运维管理

最佳实践

某电商运维人员在定位分析应用、资源及业务的实时运行状况时,发现系统上报的告 警数量过大,重复性告警过多,需要从众多告警中快速及时发现故障,全面掌握应 用。

#### 解决方案

AOM通过设置告警规则,实时监控环境中主机、组件等资源使用情况。当产品自身或 外部服务存在异常情况时,立即触发告警。并提供告警降噪功能,支持发送告警通知 前按告警降噪规则对告警进行处理,处理完成后再发送通知,帮助用户快速识别重点 问题,避免产生告警风暴。

告警降噪功能分为分组、去重、抑制、静默四部分:

- 使用分组规则,您可以从告警中筛选出满足条件的告警子集,然后按分组条件对告警子集分组,告警触发时同组告警会被汇聚在一起发送一条通知。
- 使用抑制规则,您可以抑制或阻止与某些特定告警相关的其他告警通知。例如: 当严重级别的告警产生时,可以抑制与其相关的低级别的告警。或当节点故障发 生时,抑制节点上的进程或者容器的所有其他告警。
- 使用静默规则,您可以在指定时间段屏蔽告警通知,静默规则一旦创建完成,即 刻生效。
- 去重为内置策略,服务后台会自动检验告警内容是否一致实现去重的效果,用户 无需手动创建规则。

下面以监控ELB业务层全量指标为例说明。

#### 前提条件

已创建告警行动规则。

#### 步骤一: 创建分组规则

创建一个分组规则,当产生AOM的紧急、重要告警时,触发"Monitor\_host"行动规则,且告警按照告警源合并分组。

步骤1 登录AOM 2.0控制台。

步骤2 在左侧导航栏中选择"告警管理 > 告警降噪"。

步骤3 在"分组规则"页签下单击"创建分组规则",设置规则名称、分组条件等信息。

#### 图 3-1 创建分组规则

* 规则名称	aom_rule					
* 企业项目	default	•				
描述	D					
牛勢分須抑	81					
百言力组成	20					
分组条件	告警级别	event_severity	等于	紧急 💿	重要 🔕	•
	告警察	resource_provider	等于	AOM 🛞		Ū
	◎ 添加串行条件					
	行动规则 ⑦					
	Monitor_host	▼ C 创建告警行动规则	1)   查看告警行动规则			
			<ul> <li>添加并行条件</li> </ul>			
告警合并规	则					
* 通知合并规	则 ⑦ 按告警源	•				

告警合并规则			
* 通知合并规则 🕜	按告警源	•	
* 首次等待 ?	30	秒 •	取值范围是0s-10min
* 变化等待	30	秒 •	取值范围是5s-30min
* 重复等待 ?	1	分钟 🔻	取值范围是0min-15day

#### 表 3-1 告警合并规则说明

通知合并 方式	根据指定字段对分组后的告警合并 。合并在一组的告警会被汇聚在一 起发送一条通知。 合并方式包括:
	• 按告警源: 由相同告警源触发的告警,合并为一组发送告警通知。
	<ul> <li>按告警源 + 严重度:由相同告警源触发的告警,且其严重度相同时,合并为一组发送告警通知。</li> </ul>
	<ul> <li>按告警源 + 所有标签:由相同告警源触发的告警,且其标签相同时,合并为一组发送告警通知。</li> </ul>
首次等待	首次创建告警合并集合后,等待多久发送第一次告警通知。通常设置 为秒级别的时间,便于告警合并后再发送,避免告警风暴。 取值范围:0s-10min,推荐设置为 15s。
变化等待	合并集合内的告警数据发生变化后,等待多久发送告警通知。通常设 置为分钟级别的时间。如果您需要尽快收到告警通知,也可设置为秒 级时间。
	此处的变化是指新增告警或告警状态改变。
	取值范围:5s-30min,推荐设置为60s。

重复等待	合并集合内的告警数据重复后,等待多久发送告警通知。通常设置为 小时级别的时间。
此处的重复是指无新增告警和状态变化,仅其他属性(例如 容等)改变。	
	取值范围:0min-15day,推荐设置为1h。

#### ----结束

#### 步骤二: 创建全量指标告警规则

通过指标告警规则可对资源的指标设置阈值条件。当指标数据满足阈值条件时产生阈 值告警,当没有指标数据上报时产生数据不足事件。

按照配置方式的不同,创建指标告警规则可分为三种:**按资源类型创建、按全量指标** 创建和按Prometheus命令创建。下面的操作以按全量指标创建为例说明,创建一个监 控ELB业务层全量指标的告警规则。

- 步骤1 登录AOM 2.0控制台。
- 步骤2 在左侧导航栏中选择"告警管理 > 告警规则"。
- 步骤3 在"指标或事件告警规则"页签下单击"创建告警规则"。
- 步骤4 设置告警规则的规则名称等基本信息。
- 步骤5 设置告警规则的详细信息。
  - 1. 选择"规则类型"为"指标告警规则","配置方式"为"按全量指标"。
  - 2. 设置指标、环境、检查频率等告警条件参数。

#### 图 3-2 设置告警规则详细信息

告警规则设置		
规则类型		
指标告警规则 事件告警规则		
配置方式		
全量指标 PromQL	资源类型	
推荐使用: 配置多种类型资源的告警条件; 也可以使用 正则方式 动态配置	资源告警。	
* Prometheus 实例		
O Prometheus_AOM_Default *		
告警规则详情		
多指标 混合运算		
胂位: Byte/s		
2k		
2k		
2k		
1k		
900		
600		
200		
0		
15:49 15:52 15:55 15:58 16:01 16:04 16:07 16:10 16:13 16:16	16:19 16:22 16:25 16:28 16:31 16:34 16:37 16:40 16:43 16:46 16:49 16:52	16:55 16:58 17:01 17:04 17:07 17:10
指标维度		当前值 最大值 平均值
◆ 集群ID: 00000000-0000-0000-000000000000000000	514fc2b3fc03b347151235   上行Bps(Byte/s)	1716.74 1822.94 1731.25
a 描标 aom_cluster_network_transmit_bytes	统计局期 1分钟 * 条件 ⑧ 请选择维度名称 * = 请选择维度值	+ 不分組 ◎ 小 回 首
松別规则 平均值 ▼ > 1 Byte/s 触发条件 谨	▲ 新聞 3 告警報知 🚫 派急 👻	
新増指标		

文档版本 01 (2024-05-27)

 根据需要设置告警标签和告警标注信息,为告警匹配分组,后续可关联告警降噪 策略来发送告警通知。步骤5.2选择的是业务层指标,所以此处标签设置为 "aom\_monitor\_level:business"。

#### 图 3-3 自定义标签信息

告警标签 💿	
aom_monitor_level:business	+ Tag
告警标注 🔘	
+ Tag	

#### 🗀 说明

全量指标的标签为key:value键值对格式,key通常设置为 "aom\_monitor\_level",value 的设置说明如下:

- 全量指标为基础设施层指标: infrastructure
- 全量指标为中间件指标: middleware
- 全量指标为应用层指标: application
- 全量指标为业务层指标: business
- 步骤6 设置告警通知策略。告警通知策略有两种方式,此处选择告警降噪方式。

告警降噪:对告警信息自动匹配告警降噪分组规则后再发送告警,防止产生告警风 暴。

#### 图 3-4 设置告警降噪方式

告警通知	
通知场景 ✔ 告警触发时 🛛 🖌 告警恢复时	
告警方式	
日接音響	
scl_test	• C E

**步骤7** 单击"立即创建",完成创建。创建完成后,单击"返回告警规则列表"可查看已创建的告警规则。

如下图所示,创建了一条指标告警规则,单击规则名称前的~,可查看该告警规则的 详细信息。

#### 图 3-5 创建指标告警规则

	規则名称与类型	规则状态	监控对象	告警条件	行动规则	关联Prometheus实例	启停状态	操作	
•	Monitor_host 描标告答	正常		监控对象连续3次(每次统计最近1分钟		Prometheus_AOM_ Default		/ 0 0	
基本信息 监	拉对象 告 <b>警条件</b> 触发告警								
告警条件	告警条件					告警摄别			
	监控对象连续3次(每次统计最近1分钟数	欧据)平均值大于1时产的	±			◎ 紧急告答			
检查频率	固定间隔1分钟								
告警恢复	8 当监理刘家在最近1个监控周期约不满已被发展作时,已产生的智慧冲自动恢复。								
无数据处理	关闭								

在展开的列表中,只要指标数据满足设置的告警条件时,在告警界面就会生成一条指 标类告警,您可在左侧导航栏中选择"告警管理 > 告警列表",在告警列表中查看该 告警。

只要该告警满足已设的通知策略,系统就会以邮件、短信或企业微信等方式发送告警 通知给指定人员。

#### 图 3-6 接收告警通知

 2022/12/5 (周-) 20:37

 pmail\_smnhw

 [紧急]华为云AOM服务通知:[华: -]于2022-12-05 20:28:05 GMT+08:00[新增]运维通知。

 0/4人

 ① 如果显示此邮件的方式有问题:请单击此处以在 Web 浏览器中查看读邮件。

 尊敬的 华为云用户 , 您好!

放在 华: 区域设置的告誓行动规则 Monitor\_host 新增 5 条运维通知(最多显示最新 20 条),分组规则方 aom\_rule。更多信息请登录 AOM。
详细信息如下,请您查阅:
华为云账号:
通知类型: 告警:
事件级别:紧急:
事件名称: Monitor\_business;
发生时间: 2022-12-05 20:28:05 GMT+08:00;
事体森称: Monitor\_business;
资源类型: CustomResource;
资源标识:
alarmName:Monitor\_business;
namespace;;

-----结束

# 4 通过多账号聚合 Prometheus 实例实现指标数据统一监控

本文档介绍通过配置统一监控告警,同时监控不同账号下的指标数据。

#### 实践场景

某电商平台运维人员在监控指标时,只能实时监控一个账号下的指标数据,无法同时 监控其他账号。

#### 解决方案

AOM通过Prometheus监控功能,创建多账号聚合实例,并接入账号、云服务与云服务 相关指标,支持在"指标浏览"界面同时监控多个成员账号的指标数据并为这些指标 设置告警规则。当指标存在异常情况时,立即触发告警,发送通知。

#### 前提条件

- 监控账号与被监控账号均已加入组织。监控账号需为组织管理员,非组织管理员 的组织成员需进行步骤二,授权委托管理员身份。
- 被监控账号当前支持汇聚的包括"Prometheus for 云服务"可接入的18个云服务 指标(FunctionGraph, EVS, CBR, OBS, VPC, ELB, DC, NAT, DMS, DCS, RDS, DDS, DRS, LakeFormation, MRS, GaussDB DWS, CSS, WAF)以及ICAgent采集的CCE和ECS指标。

#### 步骤一: 被监控账号接入云服务资源

下面的操作以接入FunctionGraph、DCS、ECS为例说明。接入CCE与接入ECS类似,但当前CCE购买时默认自动安装ICAgent。接入其他云服务资源的操作与接入 FunctionGraph、DCS类似。

- 接入FunctionGraph、DCS云服务资源。
  - a. 登录AOM 2.0控制台。
  - b. 在左侧导航栏中选择"接入中心"。
  - c. 在"云服务"下单击需要接入的云服务资源。
  - d. 选择云服务FunctionGraph、DCS后,单击"确定",即可接入。

图 4-1	云服务接入
-------	-------



**确**定 取消

- 接入ECS资源。
  - 将鼠标移动到右上方的用户名称,并在下拉列表中选择"我的凭证"。 a.

企业	工具	工单	۶.	Д <b>°</b>	3	⊕ 简体	p; k;		01
							基本信息	已实名认证	
							安全设置		
							我的凭证		
							统一身份认	、证	
							切换角色		>
							标签管理		
							操作日志		
							<u>i</u> l	副登录	

**图 4-2** 我的凭证

- b. 在"我的凭证"页面中选择"访问密钥"页签。
- 在列表上方单击"新增访问密钥",输入验证码或密码。 C.

版权所有 © 华为云计算技术有限公司

#### 图 4-3 新增访问密钥

我的凭证	访问密钥 ②	
API凭证   访问密钥	① 如果访问密钥泄器,会带来数据泄漏风险,且每个访问密钥仅解下载一次,为了帐号安全性,建议您定明	更换并妥善保存访问密钥。
	1 若您的访问密钥已丢失,您可创建新的访问密钥并停用原有的访问密钥。	
	② ●新增访问密钥 员还可以添加1个访问密钥。	
	访问密钥ID 💠 描述 👙	状态 ≑
	2 test	◎ 启用

- d. 单击"确定",生成并下载AK/SK。
   创建访问密钥成功后,您可以在访问密钥列表中查看访问密钥ID(AK),在
   下载的.csv文件中查看秘密访问密钥(SK)。
- e. 返回AOM 2.0控制台页面,在左侧导航栏中选择"采集管理",进入"采集管理"界面。
- f. 在左侧导航栏中,选择"UniAgent管理 > 虚机接入"。
- g. 在虚机接入中,选择待安装ICAgent的主机,单击"插件批量操作"。

#### 图 4-4 安装 ICAgent

UniAgent批量操作 ▼	ICAgent批量操作 ▼	插件批量操作 Q 默认按照	主机名称搜索
主机名称及IP地	址 接入方式 ▽	UniAgent状态 了	操作系统
ecs-wjt	直连接入	○ 异常	LINUX
	直连接入	○ 异常	LINUX
	直连接入	○ 异常	LINUX

h. 在弹出的对话框中,操作类型选择"安装",选择插件为"ICAgent",插件版本选择"5.12.163",在"ak"、"sk"中输入d获取的AK/SK。
 i. 设置完成后,单击"确认",安装ICAgent。

#### 步骤二:开启 AOM 可信服务并设置委托管理员(若进行监控的账号为组织管理员, 可跳过此步骤 )

- 步骤1 使用组织中的管理员账号登录组织Organizations控制台。
- 步骤2 在左侧导航栏选择"可信服务"。
- **步骤3** 在可信服务列表中,单击"应用运维管理服务(AOM)"操作列的"启用",开启 AOM可信服务。
- **步骤4** 单击"应用运维管理服务(AOM)"操作列的"设置委托管理员",选择需要设置为 委托管理员的账号,单击"确定"。如<mark>图</mark>4-5所示,将paas\_aom设置为委托管理员。

 $\times$ 

#### **图 4-5** 设置委托管理员

#### 设置委托管理员 可信服务 应用运维管理服务 (AOM) 选择委托管理员 请输入账号名称 Q 账号 ID 加入时间 paas\_apm\_ 1d26cc8c86a84... 2024/01/15 16:23:35 GMT+08:00 paas\_cts\_ 2306579dc99f4c... 2023/08/07 09:34:43 GMT+08:00 2024/01/02 19:21:32 GMT+08:00 paas\_aom\_ 78ac2cb7c0be4... $\bigcirc$ paas\_lts\_ f9b56cfb0a4a49... 2024/01/16 19:36:32 GMT+08:00

确定		取消
----	--	----

----结束

#### 步骤三: 配置多账号聚合实例

- 步骤1 使用组织中身份为管理员或委托管理员的监控账号登录AOM 2.0 控制台。
- **步骤2** 在左侧菜单栏中选择"Prometheus监控 > 实例列表",单击"创建Prometheus实例"。
- 步骤3 填写实例名称,选择实例类型为"Prometheus for 多账号聚合实例"。
- **步骤4** 单击"确定"完成创建。如图4-6所示,创建了一个名为"test-aom"的多账号聚合实例。

#### 图 4-6 Prometheus 实例列表

Prometheus 实例	实例类型	企业项目
test-aom	Prometheus for 多账号聚合实例	default
	O Prometheus for CCE	default
	🗈 Prometheus for 多账号聚合实例	default
	⊘ Prometheus for 云服务	default
	O Prometheus for Remote Write	default
	O Prometheus for CCE	default

**步骤5** 在"Prometheus实例"列表中单击创建的多账号聚合实例的名称,进入多账号聚合实例的"账号接入"页面,选择需要接入的账号,云服务及云服务指标。

例如,成员账号接入"paas\_apm、paas\_aom"。云服务选择接入"函数工作流 FunctionGraph、分布式缓存 DCS、弹性云服务器 ECS"。在云服务列表中选择云服 务后,单击"新增指标",可以在新增指标弹框里勾选任意需要接入的指标。

#### **图 4-7** 账号接入

← test-som

联号接入设置	成员账号 2 staintere paas_apm	о раа	as_aomI O			•					
	EBS 3 BEITHE Funderland O Structure Co S HeleBase Co O BEITHE ADEBS										
	数据存储 □ 组织内子林号编标数编辑 Q 输输入指标名称	入Promethe	zus Fol预会实例后,子称号保属数据。								
	<ul> <li>● 函数工作流 Function</li> <li>◎ 分布式跛行 DCS</li> </ul>	8 85	+ 864281855	1865 S 45	单位	握作					
		35	aom_node_network_receive_bytes	下行Bps	Bytes/Second	Θ					
			aom_node_network_receive_packets	下行Pps	Packets/Second	Θ					
			aom_node_network_receive_error_packets	下行編包率	Packets/Second	Θ					
			aom_node_network_transmit_bytes	上行Bps	Bytes/Second	Θ					
			aom_node_network_transmit_error_packets	上行續包率	Packets/Second	Θ					

接入后,等待2-3min在指标浏览处即可查看接入的指标数据。

----结束

#### 步骤四:监控账号配置统一监控告警

步骤1 验证多账号聚合实例的指标是否接入。

- 1. 在左侧导航栏选择"指标浏览",在"Prometheus实例"下拉列表中选择<mark>步骤三</mark> 配置的多账号聚合实例"test-aom"。
- 2. 单击"全量指标",选择一个指标并复制指标名称。
- 3. 单击"按普罗语句添加",输入普罗表达式: sum(指标名称) by (aom\_source\_account\_name),即可查看指标是否接入。

		统计方式:半均值	• <sup>①</sup> 近	30分钟 •	
10:14 10:15 10:16 10:17 10:18 10:19 10:20 10:2	1 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034	↓ 10:35 10:36 10:37 10:	38 10:39 10:40	10:41 10:42 10	10:4
10:14 10:15 10:16 10:17 10:18 10:19 10:20 10:2 開始構成	1 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034	1 10:35 10:36 10:37 10: 当前庙	38 10:39 10:40 最大值	- 10:41 10:42 10 平均值	10:4
10:14 10:15 10:16 10:17 10:18 10:19 10:20 10:2 指标规定 aom_source_account_name:paas_aom_	1 10.22 10.23 10.24 10.25 10.26 10.27 10.28 10.29 10.30 10.31 10.32 10.33 10.34 [sum(aom_node_cpu_usage) by (aom_source_account_name)	t 10:35 10:36 10:37 10: 当前值 52:9	38 10:39 10:40 最大值 53.70	10:41 10:42 10 平均值 51.86	10:4
10:14 10:15 10:16 10:17 10:18 10:19 10:20 10:2 Hilfstätä aom_source_account_name.paas_aom_ aom_source_account_name.paas_apm_	I 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 [sum(aom_node_cpu_usage) by (aom_source_account_name) [sum(aom_node_cpu_usage) by (aom_source_account_name)	1 10:35 10:36 10:37 10: 当時値 52.9 239.1	38 10:39 10:40 <b>最大值</b> 53.70 244.10	10:41 10:42 10 <b>平均值</b> 51.86 238.65	10:4
10:14 10:15 10:16 10:17 10:18 10:19 10:20 10:2 ###### aom_source_account_name: paas_aom_ aom_source_account_name: paas_apm_	1 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 [sum(aom_node_cpu_usage) by (aom_source_account_name) [sum(aom_node_cpu_usage) by (aom_source_account_name)	3 10:35 10:36 10:37 10: 当时值 52:9 239.1	38 10:39 10:40 <b>最大值</b> 53.70 244.10	10:41 10:42 10 <b>平均值</b> 51.86 238.65	10:

**图 4-8** 查看指标

**步骤2** 单击"全量指标",选择需要监控的指标,即可查看该账号下的指标。如<mark>图4-9</mark>所示, 选择指标"aom\_node\_cpu\_usage",即可在图表中实时监控"paas\_apm"与 "paas\_aom"账号下该指标的指标值与趋势。 **图 4-9** 查看指标



**步骤3** 单击指标列表右上角的<sup>(1)</sup>,为选择的指标新增告警规则。

- 1. 设置告警规则的规则名称等基本信息。
- 2. 设置告警规则的详细信息。
  - a. 告警规则设置中的规则类型、配置方式、Prometheus 实例默认选择为指标浏览处的配置。
  - b. 设置告警规则详情。监控的指标自动选择为指标浏览处选择的指标。 指标的详细设置由统计周期、条件、检测规则、触发条件以及告警级别组成。指标告警的检测规则,由统计方式(平均值、最小值、最大值、总计、 样本个数)、判断条件(>=、<=、>、<)和阈值组成。例如,统计周期为 "1分钟",检测规则设置为"平均值>1",触发条件为连续周期"3",告 警级别为"紧急",表示连续三个统计周期,指标的平均值大于已设置的阈 值1时,生成紧急告警。

#### **图 4-10** 设置告警规则

指标 aon	_node_virtual_i	memory_usag	e			シビンエ 日月日	1万钟		ASIT V	1910/04:2012-02			1月12月半月年1日、	12	TIT	为祖	900
							4/1.84		au+ @	100000000000000000000000000000000000000	g0+		200220-07240a mitra	14r		110	@ 4 A
aom_sc	urce_account_r	ame: paas_a	om.	ao	m_source	_project_	id: a0a12	b069ab4	491185d7	cf26c3e86ada	集群ID:	189cf01f-	6e4c-11e	20.1	20.9	10	19.73
aom_sc	urce_account_r	ame: paas_a	om	ao	m_source	_project_	id: a0a12	b069ab4	491185d7	cf26c3e86ada	集群ID:	0000000	0-0000-0	45.6	45.9	10	44.35
aom_sc	urce_account_r	ame: paas_a	om.	ao	m_source	_project_	id: a0a12	b069ab4	491185d7	cf26c3e86ada	焦群ID:	00000000	0-0000-0	4.7	4.80		4.74
指标维度														当前值	最大	偵	平均值
5:34 15:	36 15:38	15:40	15:42	15:44	15:46	15%	18 1	5:50	15:52	15:54	15:56	15:58	16:00	16:02	16:04	16:06	16:08
/																	
<u>D</u> : %																	
	19101		/8日/兰州														
			油合注意														
规则详情																	
test-aom					*												
netheus 实例																	
推荐使用: 酉	25000000000000000000000000000000000000	题的告警条件;	也可以使用	正则方式;	动态配置资	源告警。											
全症	增标		PromQL			街	渡类型										
方式																	
指标普	警规则		事件告警规	则													
		_															
作用																	

- c. 单击"高级设置",设置检查频率、告警恢复等信息。
- d. 设置告警通知策略。告警通知策略有两种方式,如<mark>图</mark>4-11所示,此处选择直接告警方式。

直接告警:满足告警条件,直接发送告警。选择直接告警方式,需要设置通知频率和是否启用告警行动规则。

- i. 设置发送告警通知的频率,请根据需要从下拉列表中选择。
- ii. 设置是否启用告警行动规则。启用告警行动规则后,系统根据关联SMN 主题与消息模板来发送告警通知。

**图 4-11** 告警通知

告警通知				
通知场景				
🖌 告警触发时 🛛 🖌 告警恢复时				
告警方式				
直接告警	告警降噪			
通知频率				
只告警—次		-		
行动规则				
aomtest		*	S	E

e. 单击"立即创建",完成创建。创建完成后,单击"返回告警规则列表"可 查看已创建的告警规则。

如图4-12所示,单击规则名称前的,可查看该告警规则的详细信息。 在展开的列表中,只要监控对象满足设置的告警条件时,在告警界面就会生 成一条指标类告警,您可在左侧导航栏中选择"告警管理 > 告警列表",在 告警列表中查看该告警。只要某个主机满足已设的通知策略,系统就会以邮 件、短信或企业微信等方式发送告警通知给指定人员。

#### 图 4-12 告警规则

规则名称与类型	规则状态	监控对象	告智条件	行动规则	关键Prometheus实例	扇停状态	操作
Mon_aom 指标告警	正常	**	监控对象连续3次(每次统计最近1分钟		test-aom		/ 0 8
的象 告望条件 触发告誓							
告警条件					告答级别		
监控对象连续3次(每次统计最近1分钟数据)平均值大于1时产生 O 素色色馨							
固定间隔1分钟							
当监控对象在最近1个监控周期内不满足触发	这条件时,已产生的告替	将自动恢复。					
关闭							
				ARRIVAD         ARRIVAD	取取れたり支空         加取加           Monom         回答         回答         ロ         回びが加速         回びが加速         ロ         回びが加速         ロ	支援的体力型         支援防体         支援防体         支援合体         支援防体         支援防          支援防          支援防          支援防          支援防          支援防          支援防          支援         <	取取化合         取取化合         加加合         和加合         和加合         和和化合         和化合         和和化合         和和化合         和和化合         和和化合         和和化合         和化合         和化合         和化合         和化合         和化合 </td

- 步骤4 单击指标列表右上角的<sup>4</sup>,将图表添加至仪表盘。
  - 在下拉列表中选择仪表盘并输入图表名称。如果现有列表中的仪表盘无法满足需要,可单击"创建新的仪表盘",新建仪表盘的操作详见创建仪表盘。

<b>图 4-13</b> 添加到仪表盘
----------------------

添加到仪表盘				×
*选择仪表盘	aom	Ŧ	C 创建新的仪表盘	
* 图表名称	CPU使用率			
	确认	取消		

2. 单击"确定",自动跳转至仪表盘界面查看创建的图表。如<mark>图4-14</mark>所示,在仪表 盘"aom"下创建了"CPU使用率"的图表,可以实时监控"paas\_apm"与 "paas\_aom"账号下"CPU使用率"的指标值与趋势。

#### 图 4-14 查看图表



----结束

# 5 CCE 容器场景自建中间件接入

# 5.1 PostgreSQL Exporter 接入

#### 操作场景

使用PostgreSQL过程中需要对PostgreSQL运行状态进行监控,以便了解PostgreSQL服务是否运行正常,及时排查PostgreSQL故障问题原因。Prometheus监控服务提供了CCE容器场景下基于Exporter的方式来监控PostgreSQL运行状态。本文介绍如何部署Exporter以及实现PostgreSQL Exporter告警接入等操作。

#### 前提条件

- CCE服务已拥有CCE集群并已安装PostgreSQL。
- 服务已接入可观测Prometheus监控并接入CCE集群,具体请参见Prometheus实例 for CCE。
- 已将postgres\_exporter镜像上传到SWR,具体操作请参见使用容器引擎客户端上 传镜像。

#### PostgreSQL Exporter 部署

- 步骤1 登录CCE控制台。
- 步骤2 单击已接入的集群名称,进入该集群的管理页面。
- 步骤3 执行以下操作完成Exporter部署。
  - 1. 使用Secret管理PostgreSQL密码。

在左侧导航栏中选择"工作负载",在右上角单击"YAML创建"完成YAML配置。YAML配置说明:使用Kubernetes的Secret来管理密码并对密码进行加密处理,在启动PostgreSQL Exporter的时候直接使用Secret Key,需要调整对应的password。

YAML 配置示例如下:

apiVersion: v1 kind: Secret metadata: name: postgres-test type: Opaque stringData: username: postgres password: you-guess #对应 PostgreSQL 密码 2. 部署PostgreSQL Exporter。

在左侧导航栏中选择"工作负载",在右上角单击"YAML创建",以YAML的方式部署Exporter。

#### YAML配置示例如下(请直接复制下面的内容,根据实际业务调整相应的参数):

apiVersion: apps/v1 kind: Deployment metadata: name: postgres-test # 根据业务需要调整成对应的名称,建议加上 PG 实例的信息 namespace: default #需要和 postgres 的 service 在同一命名空间 labels: app: postgres app.kubernetes.io/name: postgresql spec: replicas: 1 selector: matchLabels: app: postgres app.kubernetes.io/name: postgresql template: metadata: labels: app: postgres app.kubernetes.io/name: postgresql spec: containers: - name: postgres-exporter image: swr.cn-north-4.myhuaweicloud.com/aom-exporter/postgres-exporter:v0.8.0 # 上传至 SWR 的 postgres-exporter 镜像 args: - "--web.listen-address=:9187" # Exporter 开启的端口 - "--log.level=debug" # 日志级别 env: - name: DATA\_SOURCE\_USER valueFrom: secretKeyRef: name: postgres-test # 对应上一步中的 Secret 的名称 key: username # 对应上一步中的 Secret Key - name: DATA\_SOURCE\_PASS valueFrom: secretKeyRef: name: postgres-test # 对应上一步中的 Secret 的名称 key: password # 对应上一步中的 Secret Key - name: DATA\_SOURCE\_URI value: "x.x.x.x:5432/postgres?sslmode=disable" # 对应的连接信息 ports: - name: http-metrics containerPort: 9187

3. 获取指标。

通过"curl http://exporter:9187/metrics"无法获取Postgres实例运行时间,可以 通过自定义一个queries.yaml来获取该指标。

- a. 创建一个包含queries.yaml的配置。
- b. 将配置作为Volume挂载到Exporter某个目录下。
- c. 通过extend.query-path来使用配置,将上述的Secret以及Deployment进行汇 总,汇总后的YAML如下所示:

```
# 以下 document 创建一个包含自定义指标的 queries.yaml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
name: postgres-test-configmap
namespace: default
data:
queries.yaml: |
```

```
pg_postmaster:
   query: "SELECT pg_postmaster_start_time as start_time_seconds from
pg_postmaster_start_time()"
   master: true
   metrics:
     - start_time_seconds:
       usage: "GAUGE"
       description: "Time at which postmaster started"
# 以下 document 挂载了 Secret 和 ConfigMap ,定义了部署 Exporter 相关的镜像等参数
apiVersion: apps/v1
kind: Deployment
metadata:
 name: postgres-test
 namespace: default
 labels:
  app: postgres
  app.kubernetes.io/name: postgresql
spec:
 replicas: 1
 selector:
  matchLabels:
   app: postgres
   app.kubernetes.io/name: postgresql
 template:
  metadata:
   labels:
     app: postgres
    app.kubernetes.io/name: postgresql
  spec:
   .
containers:
     - name: postgres-exporter
      image: wrouesnel/postgres_exporter:latest
      args:
       - "--web.listen-address=:9187"
        - "--extend.query-path=/etc/config/queries.yaml"
       - "--log.level=debug"
      env:
        - name: DATA_SOURCE_USER
         valueFrom:
          secretKeyRef:
           name: postgres-test-secret
           key: username
        - name: DATA_SOURCE_PASS
         valueFrom:
          secretKeyRef:
           name: postgres-test-secret
           key: password
        - name: DATA_SOURCE_URI
        value: "x.x.x.x:5432/postgres?sslmode=disable"
      ports:
        name: http-metrics
         containerPort: 9187
      volumeMounts:
        - name: config-volume
         mountPath: /etc/config
   volumes:
     - name: config-volume
      configMap:
       name: postgres-test-configmap
apiVersion: v1
kind: Service
metadata:
 name: postgres
spec:
 type: NodePort
 selector:
```

app: postgres app.kubernetes.io/name: postgresql ports: - protocol: TCP nodePort: 30433 port: 9187 targetPort: 9187

d. 访问地址:

http://{集群任意节点的公网IP}:30433/metrics,即可通过自定义的 queries.yaml查询到Postgres实例启动时间指标。

#### **图 5-1** 访问地址

← → C :30433/metrics	
# TYPE go_memstats_stack_inuse_bytes gauge	
go_menstats_stack_inuse_bytes b24288	
# HELF go_memstats_stack_sys_bytes wumber of bytes obtained from system for stack allocator.	
# IIFE go_memstats_stack_sys_bytes gauge	
go_menstats_stack_sys_bytes b24288	
# HELP go_memstats_sys_bytes Number of bytes obtained from system.	
W IIPE go_memstats_sys_bytes gauge	
go_nemstats_sys_bytes 7.04b12e+07	
# HELP go_threads Number of OS threads created.	
# TYPE go_threads gauge	
go_threads 6	
# HELP pg_exporter_last_scrape_duration_seconds Duration of the last scrape of metrics from PostgresSQL.	
# TYPE pg_exporter_last_scrape_duration_seconds gauge	
pg_exporter_last_scrape_duration_seconds 0.016062949	
# HELP pg_exporter_last_scrape_error Whether the last scrape of metrics from PostgreSQL resulted in an error (	1 for error, 0 for success).
# TYPE pg_exporter_last_scrape_error gauge	
pg_exporter_last_scrape_error 0	
# HELP pg_exporter_scrapes_total Total number of times PostgresSQL was scraped for metrics.	
# TYPE pg_exporter_scrapes_total counter	
pg_exporter_scrapes_total 2	
# HELP pg_locks_count Number of locks	
# IYPE pg_locks_count gauge	
pg_locks_count{datname="aa", mode="accessexclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="aa",mode="accesssharelock",server="192.168.0.205:30432"} 0	
pg_locks_count {datname="aa", mode="exclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="aa", mode="rowexclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="aa", mode="rowsharelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="aa",mode="sharelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="aa",mode="sharerowexclusivelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="aa",mode="shareupdateexclusivelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="postgres", mode="accessexclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="postgres", mode="accesssharelock", server="192.168.0.205:30432"} 1	
pg_locks_count{datname="postgres", mode="exclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="postgres", mode="rowexclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="postgres", mode="rowsharelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="postgres", mode="sharelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="postgres", mode="sharerovexclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="postgres", mode="shareupdateexclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template0",mode="accessexclusivelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template0",mode="accesssharelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template0",mode="exclusivelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template0",mode="rowexclusivelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template0",mode="rowsharelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template0",mode="sharelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template0",mode="sharerowexclusivelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template0",mode="shareupdateexclusivelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template1",mode="accessexclusivelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template1",mode="accesssharelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template1", mode="exclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template1", mode="rowexclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template1",mode="rowsharelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template1",mode="sharelock",server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template1", mode="sharerowexclusivelock", server="192.168.0.205:30432"} 0	
pg_locks_count{datname="template1",mode="shareupdateexclusivelock",server="192.168.0.205:30432"} 0	
# HELP pg_settings_allow_system_table_mods Allows modifications of the structure of system tables.	
# TYPE pg_settings_allow_system_table_mods_gauge	
pg_settings_allow_system_table_mods{server="192.168.0.205:30432"} 0	
# HELP pg_settings_archive_timeout_seconds Forces a switch to the next WAL file if a new file has not been sta	rted within N seconds. [Units converted to seconds.]
# TYPE pg_settings_archive_timeout_seconds gauge	

----结束

#### 添加采集任务

通过<mark>新增PodMonitor</mark>方式为应用配置可观测监控Prometheus版的采集规则,监控部 署在CCE集群内的应用的业务数据。

#### 🛄 说明

如下指标采集的周期是30秒,所以等待大概30秒后才能在AOM的界面上查看到上报的指标。

apiVersion: monitoring.coreos.com/v1 kind: PodMonitor metadata: name: postgres-exporter namespace: default spec: namespaceSelector: matchNames: - default # exporter 所在的命名空间 podMetricsEndpoints: - interval: 30s path: /metrics port: http-metrics selector: matchLabels: app: postgres

#### 验证指标上报到 AOM

- 步骤1 登录AOM 2.0控制台。
- 步骤2 在左侧菜单栏中选择 "Prometheus监控 > 实例列表"。
- 步骤3 单击接入了该CCE集群的"prometheus for CCE"实例名称,进入实例详情页面。
- 步骤4 在"服务发现"页面的"指标"页签下,选择对应集群。
- 步骤5 选择Job: {namespace}/postgres-exporter,可以查询到pg开头的postgresql指标。

#### **图 5-2** 搜索指标

<b>()</b>	R有在CCE "插件管理" 页面或AOM "集成中心" 页面安装的3.9.0以上的云原生监控插作	+ (kube-prometheus-stack)	上报的指标可以被废弃;	且只有kube-prometheus-stack	处于"运行中"状态时,	废弃指核
集群	aomtest • Job default/postgres-exporter • 新增調标	Q pg				
	期标名	指标类型		1	指标量 (最近10分钟)	
	pg_settings_plan_cache_mem_bytes	自定义指标		2	2	
	pg_settings_max_standby_streaming_delay_seconds	自定义指标		1	1	
	pg_settings_log_duration	自定义指标		1	1	
	pg_settings_krb_caseins_users	自定义指标		2	2	
	pg_stat_database_xact_commit	自定义指标		4	4	
	pg_settings_auto_explain_log_nested_statements	自定义指标		2	2	
	pg_settings_vacuum_multixact_freeze_min_age	自定义指标		1	1	
	pg_settings_autovacuum_work_mem_bytes	自定义指标		1	1	
	pg_settings_track_commit_timestamp	自定义指标		1	1	
	pg_settings_pgaudit_log_statement_once	自定义指标		1	1	
10 -	<b>劉</b> 顶, 共 177 祭 〈 <mark>1</mark> 2 3 4 5 6 7 8 18 →					

#### ----结束

#### 在 AOM 上配置仪表盘和告警

通过仪表盘功能可视化监控CCE集群数据,通过告警规则功能,在集群发生故障时能够 及时发现并预警。

- 配置仪表盘图表
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"仪表盘",单击"创建仪表盘"新建一个仪表盘,详 情可参见创建仪表盘。
  - c. 在仪表盘页面选择实例类型为 "Prometheus for CCE"的实例并单击 "添加 图表",详情请参见添加图表至仪表盘。
- 配置告警
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"告警管理 > 告警规则"。
  - c. 单击"创建告警规则"配置告警,详情请参见创建指标告警规则。

# 5.2 MySQL Exporter 接入

#### 操作场景

MySQL Exporter专门为采集MySQL数据库监控指标而设计开发,通过Exporter上报核 心的数据库指标,用于异常报警和监控大盘展示。目前,Exporter支持5.6版本或以上 版本的MySQL。在MySQL低于5.6版本时,部分监控指标可能无法被采集。

#### 🗀 说明

为了方便安装管理Exporter,推荐使用CCE进行统一管理。

#### 前提条件

- CCE服务已拥有CCE集群并已安装MySQL。
- 服务已接入可观测Prometheus监控并接入CCE集群,具体请参见Prometheus实例 for CCE。
- 已将对应mysql\_exporter镜像上传到SWR,具体操作请参见使用容器引擎客户端 上传镜像。

#### 数据库授权

步骤1 登录集群执行以下命令:

kubectl exec -it \${mysql\_podname} bash mysql -u root -p

#### **图 5-3**执行命令



#### 步骤2 登录数据库,执行以下命令:

CREATE USER 'exporter'@'x.x.x.(hostip)' IDENTIFIED BY 'xxxx(password)' WITH MAX\_USER\_CONNECTIONS 3; GRANT PROCESS, REPLICATION CLIENT, SELECT ON \*.\* TO 'exporter'@'x.x.x.x(hostip)';

步骤3 验证授权是否成功。

输入以下命令查询sql,查看是否有exporter的数据,host为mysql所在节点的IP。

select user,host from mysql.user;

#### **图 5-4** 查询 sql

<pre>mysql&gt; select use</pre>	r,host from mys	ql.user;
++		+
user	host	
++		+
root	%	1
exporter	192.168.0.205	1
mysql.session	localhost	1
mysql.sys	localhost	1
root	localhost	1
++		+
5 rows in set (0.	00 sec)	
mysql>		

----结束

#### MySQL Exporter 部署

- 步骤1 登录CCE控制台。
- 步骤2 单击已接入的集群名称,进入该集群的管理页面。
- 步骤3 执行以下操作完成Exporter部署。
  - 使用Secret管理MySQL连接串: 在左侧导航栏中选择"配置与密钥",在右上角单击"YAML创建",输入以下 yml文件,密码是按照Opaque加密过的。

```
apiVersion: v1
kind: Secret
metadata:
name: mysql-secret
namespace: default
type: Opaque
stringData:
datasource: "user:password@tcp(ip:port)/" #对应 MySQL 连接串信息,需要加密
```

#### 🗀 说明

配置密钥的详细操作参见创建密钥。

2. 部署MySQL Exporter。

```
在左侧导航栏中选择"工作负载",在右上角单击"创建负载",选择"负载类型"为无状态工作负载Deployment,选择需要的命名空间部署MySQL
Exporter。如果以YAML的方式部署Exporter,YAML配置示例如下:
```

apiVersion: apps/v1

kind: Deployment

metadata: labels:

k8s-app: mysql-exporter # 根据业务需要调整成对应的名称,建议加上MySQL实例的信息, 如 ckafka-2vrgx9fd-mysql-exporter

name: mysql-exporter # 根据业务需要调整成对应的名称,建议加上MySQL实例的信息, 如 ckafka-2vrgx9fd-mysql-exporter

namespace: default #需要和CCE集群中安装的MySQL命名空间一致

spec:

- replicas: 1
- selector: matchLabels:

k8s-app: mysql-exporter # 根据业务需要调整成对应的名称,建议加上MySQL实例的信息, 如

ckafka-2vrgx9fd-mysql-exporter template: metadata: labels: k8s-app: mysql-exporter # 根据业务需要调整成对应的名称,建议加上MySQL实例的信息, 如 ckafka-2vrgx9fd-mysql-exporter spec: containers: - env: - name: DATA\_SOURCE\_NAME valueFrom: secretKeyRef: name: mysgl-secret key: datasource image: swr.cn-north-4.myhuaweicloud.com/aom-exporter/mysqld-exporter:v0.12.1 imagePullPolicy: IfNotPresent name: mysql-exporter ports: - containerPort: 9104 name: metric-port terminationMessagePath: /dev/termination-log terminationMessagePolicy: File dnsPolicy: ClusterFirst imagePullSecrets: - name: default-secret restartPolicy: Always schedulerName: default-scheduler securityContext: {} terminationGracePeriodSeconds: 30 apiVersion: v1 kind: Service metadata: name: mysql-exporter spec: type: NodePort selector: k8s-app: mysql-exporter ports: - protocol: TCP nodePort: 30337 port: 9104 targetPort: 9104

🛄 说明

更多Exporter详细参数介绍请参见mysql-exporter。

- 3. 验证MySQL Exporter是否部署成功。
  - a. 在工作负载列表中"无状态负载"页签下,单击<mark>步骤3.2</mark>创建的无状态工作负载的名称,在实例列表中单击操作列下的"更多 > 日志",可以查看到 Exporter成功启动并暴露对应的访问地址。

#### **图 5-5** 查看日志

#### 日志

0	当前显示的日志内容为容器标准输出日志,不具	具备持久化和高阶运维能力,	如需使用更完善的日志能力,	可使用 <mark>日志中心</mark> 区功能。	如用户开启了 AOM 采集标准输出的
	功能,可前往 AOM 查阅更多的负载日志。查看	AOM日志搜索I			

实例 mysql-exporter-b65/6cfb8 ▼ 容器 mysql-exporter ▼ 行数 200 ▼ 下载 Q 定时局新 305
time="2023-12-18T02:27:452" level=info msg="Starting mysqld_exporter (version=0.12.1, branch=HEAD, revision=48667bf7c3b438b5e93b25
9f3d17b70a7c9aff96)
time="2023-12-18T02:27:452" level=info msg="Build context (go=go1.12.7, user=root@0b3e56a7bc0a, date=20190729-12:35:58)" source="m
ysqld_exporter.go:258 <sup>°</sup>
time="2023-12-18T02:27:452" level=info msg="Enabled scrapers:" source="mysqld_exporter.go:269"
time="2023-12-18T02:27:452" level=info msg=" —collect.global_status" source="mysqld_exporter.go:279"
time="2023-12-18T02:27:452" level=info msg="collect.global_variables" source="mysqld_exporter.go:273"
time="2023-12-18T02:27:452" level=info msg=" —collect.slave_status" source="mysqld_exporter.go:273"
time="2023-12-18T02:27:452" level=info msg="collect.info_schema.innodb_cmp" source="mysqld_exporter.go:273"
time="2023-12-18T02:27:452" level=info msg="collect.info_schema.innodb_cmpmem" source="mysqld_exporter.go:273"
time="2023-12-18T02:27:452" level=info msg="collect.info_schema.query_response_time" source="mysqld_exporter.go:273"
t me=~2023-12-13T02:27:452″ level=info msg=~Listening on :9104″ source=~mysqld_exporter.go:283″

- 验证。有以下三种方法进行验证: b.
  - 登录集群节点执行如下任意一种命令: curl http://{集群IP}:9104/metrics curl http://{集群任意节点私有IP}:30337/metrics
  - 在实例列表中单击操作列下的"更多 > 远程登录",执行如下命令: curl http://localhost:9104/metric
  - 访问: http://{集群任意节点的公网IP}:30337/metrics。

#### 图 5-6 访问地址

B -> 0 JUJUE 30337/metrics ← → C ▲

----结束

#### 采集 CCE 集群的业务数据

通过<mark>新增PodMonitor</mark>方式为应用配置可观测监控Prometheus版的采集规则,监控部 署在CCE集群内的应用的业务数据。

#### 配置信息如下:

apiVersion: monitoring.coreos.com/v1 kind: PodMonitor metadata: name: mysql-exporter namespace: default spec: namespaceSelector: matchNames: - default # exporter 所在的命名空间 podMetricsEndpoints: - interval: 30s path: /metrics port: metric-port selector: matchLabels: k8s-app: mysql-exporter

#### 🛄 说明

指标采集的周期是30秒,所以等待大概30秒后才能在AOM的界面上查看到上报的指标。

#### 验证指标上报到 AOM

- 步骤1 登录AOM 2.0控制台。
- 步骤2 在左侧菜单栏中选择 "Prometheus监控 > 实例列表"。
- 步骤3 单击接入了该CCE集群的"prometheus for CCE"实例名称,进入实例详情页面。
- 步骤4 在"服务发现"页面的"指标"页签下,选择对应集群。
- **步骤5** 选择Job: {namespace}/mysql-exporter,可以查询到mysql开头的自定义指标。 ----结束

#### 在 AOM 上配置仪表盘和告警

通过仪表盘功能可视化监控CCE集群数据,通过告警规则功能,在集群发生故障时能够 及时发现并预警。

- 配置仪表盘图表
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"仪表盘",单击"创建仪表盘"新建一个仪表盘,详 情可参见创建仪表盘。
  - c. 在仪表盘页面选择实例类型为 "Prometheus for CCE"的实例并单击 "添加 图表",详情请参见<mark>添加图表至仪表盘</mark>。
- 配置告警
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"告警管理 > 告警规则"。
  - c. 单击"创建告警规则"配置告警,详情请参见创建指标告警规则。

# 5.3 Kafka Exporter 接入

#### 操作场景

使用Kafka过程中需要对Kafka运行状态进行监控,例如集群状态、消息消费情况是否 有积压等。Prometheus监控服务提供了CCE容器场景下基于Exporter的方式来监控 Kafka运行状态。本文介绍如何部署Kafka Exporter以及实现Kafka Exporter告警接入 等操作。

#### 🛄 说明

为了方便安装管理Exporter,推荐使用CCE进行统一管理。

#### 前提条件

- CCE服务已拥有CCE集群并已安装Kafka。
- 服务已接入可观测Prometheus监控并接入CCE集群,具体请参见Prometheus实例 for CCE。
- 已将对应kafka\_exporter镜像上传到SWR,具体操作请参见使用容器引擎客户端 上传镜像。

#### Kafka Exporter 部署

- 步骤1 登录CCE控制台。
- 步骤2 单击已接入的集群名称,进入该集群的管理页面。
- 步骤3 执行以下操作完成Exporter部署。
  - 1. 部署Kafka Exporter。

```
在左侧导航栏中选择"工作负载",在右上角单击"创建负载",选择"负载类型"为无状态工作负载Deployment,选择需要的命名空间部署Kafka Exporter。如果以YAML的方式部署Exporter,YAML配置示例如下:
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
  k8s-app: kafka-exporter # 根据业务需要调整成对应的名称,建议加上Kafka实例的信息, 如
ckafka-2vrgx9fd-kafka-exporter
name: kafka-exporter # 根据业务需要调整成对应的名称,建议加上Kafka实例的信息, 如
ckafka-2vrgx9fd-kafka-exporter
namespace: default #已存在集群的namespace
spec:
replicas: 1
selector:
  matchLabels:
   k8s-app: kafka-exporter # 根据业务需要调整成对应的名称,建议加上Kafka实例的信息, 如
ckafka-2vrgx9fd-kafka-exporter
template:
  metadata:
   labels:
    k8s-app: kafka-exporter # 根据业务需要调整成对应的名称,建议加上Kafka实例的信息, 如
ckafka-2vrgx9fd-kafka-exporter
  spec:
  containers:
   - args:
    - --kafka.server=120.46.215.4:30092 # 对应Kafka实例的地址信息
```

image: swr.cn-north-4.myhuaweicloud.com/mall-swarm-demo/kafka-exporter:latest imagePullPolicy: IfNotPresent name: kafka-exporter ports: - containerPort: 9308 name: metric-port # 这个名称在配置抓取任务的时候需要 securityContext: privileged: false terminationMessagePath: /dev/termination-log terminationMessagePolicy: File dnsPolicy: ClusterFirst imagePullSecrets: - name: default-secret restartPolicy: Always schedulerName: default-scheduler securityContext: {} terminationGracePeriodSeconds: 30 apiVersion: v1 kind: Service metadata: name: kafka-exporter spec: type: NodePort selector: k8s-app: kafka-exporter ports: - protocol: TCP nodePort: 30091 port: 9308 targetPort: 9308

#### 🛄 说明

更多 Exporter详细参数介绍请参见 kafka-exporter。

- 2. 验证Kafka Exporter是否部署成功。
  - a. 在工作负载列表中"无状态负载"页签下,单击<mark>步骤3.1</mark>创建的无状态工作负载,在实例列表中单击操作列下的"更多 > 日志",可以查看到Exporter成功启动并暴露对应的访问地址。

#### **图 5-7** 查看日志

日志	>
● 当前显示的日志内容为容器标准输出日志,不具备持久化和高阶运维能力,如需使用更完备的日志能力,可使用日志中心 ⑦功能,如用户开启了 AOM 采集标准输出的功能,可能在 AOM 查阅更多的负载日志,查看 AOM 日志搜索 ⑦	
案例 kafak-exporter-7b9847b ▼ 容器 kafka-exporter ▼ 行数 200 ▼ 下戦 Q 定时開新 305	
11214 U/:20:51.191/33 I katka_exporter.go:7/4] Starting katka_exporter (version=1.4.2, branch=HEAD, revision=10e4adDa9ead203 135d4b974e825f22e31c750e5) 11214 07:20:51.198432 I kafka_exporter.go:934] Listening on HTT: :9308	

- b. 验证。有以下三种方法进行验证:
  - 登录集群节点执行如下任意一种命令: curl http://(集群IP):9308/metrics curl http://(集群任意节点私有IP):30091/metrics
  - 在实例列表中单击操作列下的"更多 > 远程登录",执行如下命令: curl http://localhost:9308/metric
  - 访问: http://{集群任意节点的公网IP}:30091/metrics。

#### **图 5-8** 访问地址

<pre>p. sametary_provide_income_terms_letter 10500 HEEP 0, masters_mach_gry_bytes (maker of bytes und for nonche structures obtained from system. # TFF 0, masters_mach_gry_bytes (maker of bytes in use by maps structures) # DEP 0, masters_maps_limits_bytes (Maker of bytes in use by maps structures) # DEP 0, masters_maps_limits_bytes (Maker of bytes used for maps structures) # DEP 0, masters_maps_limits_bytes (Maker of bytes used for maps structures) # DEP 0, masters_maps_limits_bytes (Maker of bytes used for maps structures) # DEP 0, masters_maps_limits_bytes (Maker of bytes used for other system structures) # DEP 0, masters_maps_limits_bytes (Maker of bytes used for other system allocations. # DEP 0, masters_maps_limits_bytes (Maker of bytes used for other system allocations. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes in use by the stack allocator. # DEP 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system allocations. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system for stack allocator. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system for stack allocator. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_bytes (L) (JAEG 000 of bytes obtained from system. # TFF 0, masters_there.grv_b</pre>	← → C ▲	30091/metrics
<pre># Hilf 70.meatata_scata_sys_brss Number of bytes in use by napan structures obtained from system. 1TFS f0.meatata_space_inuse_brss square 0.meatata_space_inuse_brss square 0.meatata_space_brss squares brss square 0.meatata_space_brss squares brss square 0.meatata_space_brss squares brss square 0.meatata_space_brss squares brss squares 0.meatata_space_brss squares brss squares brss squares brss squares brss squares brss squares brss squares 0</pre>	go_menstats_mcache_inus	e_bytes 19200
<pre>n use vicual state provide State EllPr 0.mestate specific State EllPr 0.mestate specific</pre>	# HELP go_memstats_mcac.	he_sys_bytes Number of bytes used for mcache structures obtained from system.
<pre>a TED (or</pre>	# life go_menstats_mcache	ne_sys_bytes gauge
<pre>H TYPE (o _sentit_spap.insc.byte squee Co.mentit_spap.insc.byte 6420 H HEL (o _sentit_spap.insc.byte 6420 H HEL (o _sentit_spap.insc.byte) for a gave Co.mentit_space 4.17924er06 H TYPE (o _sentit_space 4.17924er07 H TYPE (o _sentit_space 4.17924er07) H TYPE (o</pre>	go_menscats_mcache_sys_ # HELP go memotate meno	uytes Jaroon n inuse Nutae Number of butes in use by menen structures
<pre>co.martiz_prop.im.s.g.tytes 40:40 HEF po_mentstr_appa_vytes (40:40 HEF po_mentst_appa_vytes (40:40 HEF po_mentst_appa_vst_appa_vytes (40:40 HEF po_mentst_appa_vst_appa_v)tes (40:40 HEF po_mentst_appa_vst_appa_vytes (40:40 HEF po_mentst_appa_vst_appa_vytes (40:40 HEF po_mentst_appa_vst_appa_vst_appa_vytes (40:40 HEF po_mentst_appa_vst_</pre>	# TYPE go memstats msna	n inize bytes same
<pre>f HED po_sentit_angyet_bytes Number of bytes used for mapan structures obtained from system. HT Fire_son_struct_angyet_bytes gauge so_martit_mart_protytes Multiple so_martit_inst_protytes Number of bytes used for other system allocations. HT Fire_son_struct_clusters 4.473924er06 HT Fire_son_struct_cluster_sys_bytes [0.4965er06 Commartit_cluster_sys_bytes [0.4965er06 HT Fire_son_struct_cluster_sys_bytes [0.4965er06 Commartit_cluster_sys_bytes [0.4965er06 HT Fire_son_struct_cluster_sys_bytes [0.4965er06 HT Fire_son_struct_cluster_sys_bytes [0.4965er06 Commartit_cluster_sys_bytes [0.4965er06 HT Fire_son_struct_cluster_sys_bytes [0.4965er06 HT Fire_son_struct_cluster_sys_bytes [0.4965er06 HT Fire_son_struct_sys_bytes [0.4965er07 HT Fire_son_struct_sys_bytes [0.4976er07 HT Fire_son_struct_sys_bytes [0.4976er07 HT Fire_son_struct_sys_bytes [0.4976er07 HT Fire_son_struct_sys_bytes [0.4976er07 HT Fire_son_struct_sys_bytes [0.4977er07 HT Fire_son_struct_sys_bytes [0.4977er07 HT Fire_son_struct_sys_bytes [0.4977er07 HT Fire_son_struct_sys_bytes [0.4977er07 HT Fire_son_struct_sys_bytes [0.4977er07 HT Fire_son_struct_sys_byte</pre>	co menstats mspan inuse	bytes 45240
<pre>H TFF [:</pre>	# HELP go memstats mspa	n sys bytes Number of bytes used for mspan structures obtained from system.
go_mentit_maps_py_bytes 40152 ' HEF go_mentit_maps_py_bytes Number of heap bytes when next sarbage collection will take place. HITE go_mentit_maps_py_bytes Number of bytes used for other system allocations. HITE go_mentit_collect, graphytes (1998)er06 HEE go_mentit_collect, graphytes gauge go_mentit_collect, graphytes (1998)er06 HEE go_mentit_collect, graphytes gauge go_mentit_collect, graphytes (1998)er06 HEE go_mentit_collect, graphytes (1998)er06 HEE go_mentit_collect, graphytes (1998)er06 HEE go_mentit_collect, graphytes (1998)er06 HEE go_mentit_graphytes (1998)er06 HEE go_threads Number of Dickers in the Kafka Cluster. HITE fraght_mention (HeE) (1998) HEE for (1998)er06 HEE go_threads gauge HEE hafka_koporter_build_info fauge HEE hafka_koporter_build_info fauge HEE hafka_koporter_build_info fauge HEE for (1998) er06 HEE go_mentit_graphytes (1998) HEE for (1998) er06 HEE go_mentit_graphytes (1998) HEE for (1998) er06 HEE for (1998) er0	# TYPE go_memstats_mspa	n_sys_bytes gauge
<pre># HELP (p.e.mentst.next_protypes Runder of heap bytes when next parbage collection will take place. # IFE (p.emetst.next_platts.next.next.next.next.next.next.next.next</pre>	go_menstats_mspan_sys_b	ytes 49152
<pre>H TFE (p_exection_prot_c_byte f algorithm) H TFE (p_exection_prot_byte f Norder Of bytes used for other system allocations. H TFE (p_exection_prot_byte f NorderMed) H TFE (proces_prot_byte f NorderMed) H TFE (proces_prot_byte f NorderMed) H TFE (proces_prot_f NorderMed) H TFE (</pre>	# HELP go_memstats_next	_gc_bytes Number of heap bytes when next garbage collection will take place.
go_mantist_ment_cole_prote 4.473924006 HTPF go_mantist_coler_grs_byces Rubber of bytes used for other system allocations. HTPF go_mantist_coler_grs_byces Rubber of bytes in use by the stack allocator. HTPF go_mantist_stack_inse_bytes Stage co_mantist_stack_inse_bytes Stage HEF go_mantist_stack_grs_bytes Rubber of bytes obtained from system for stack allocator. HTPF go_mantist_stack_grs_bytes Rubber of bytes obtained from system for stack allocator. HTPF go_mantist_stack_grs_bytes Rubber of bytes obtained from system for stack allocator. HTPF go_mantist_stack_grs_bytes Rubber of bytes obtained from system. HTPF go_mantist_grs_bytes Rubber of bytes obtained from system. HTPF solution Rubber of Brokers in the Kafka Cluster. HTPF stafa_brokers Rubber of Brokers in the Kafka Cluster. HTPF stafa_apporter_built_info & string with a constant '1' value labeled by version, revision, branch, and goversion from which kafka_exporter was built. HTPF spreamantist of the stafa apporter_built_info & string with a constant '1' value labeled by version, revision, branch, and goversion from which kafka_exporter was built. HTPF proces_gou_second_total counter HTPF proces_gou_second_total counter HTPF proces_gou_second_total counter HTPF proces_gou_did Rubber of open file descriptors. HTPF proces_gou_did Rubber of Dytes Rubber of bytes in bytes. HTPF proces_gou_did Rubber of Dytes Rubber of bytes in bytes. HTPF proces_gou_did Rubber of Dytes Rubber of bytes in bytes. HTPF proces_gou_did Rubber of Dytes Rubber appendix and the process since unix epoch in seconds. HTPF proces_statt_time_seconds Start time of the process since unix epoch in seconds. HTPF proces_statt_time_seconds Rubber appendi	# TYPE go_memstats_next	_gc_bytes gauge
<pre>H HEL (r</pre>	go_memstats_next_gc_byt	es 4.473924e+06
<pre>H TFE (or Amentate, other, syr, bytes [suge commentate, therm, syr, bytes [1]/4586+46 H TFE (or Amentate, state, syr, bytes for of bytes in use by the stack allocator. H TFE (or Amentate, state, syr, bytes Musher of bytes obtained from system for stack allocator. H TFE (or Amentate, state, syr, bytes Musher of bytes obtained from system for stack allocator. H TFE (or Amentate, state, syr, bytes Musher of bytes obtained from system for stack allocator. H TFE (or Amentate, state, syr, bytes Musher of bytes obtained from system. H TFE (or Amentate, state, syr, bytes Musher of bytes obtained from system. H TFE (or Amentate, syr, bytes 1.5166489+07 H TFE (or Amentate, syr, bytes 1.51670+07) H TFE (or Amentate, second, 5170+07) H TFE (or Amentate, 5170+07) H TFE</pre>	# HELP go_memstats_othe	r_sys_bytes Number of bytes used for other system allocations.
<pre>co.matti_cite_cytes 1.0/4886+06 HEF co.matti_cite_cite_cite co.matti_cite_cite_cite co.matti_cite_cite_cite co.matti_cite_cite_cite_cite co.matti_cite_cite_cite_cite_cite co.matti_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite domatti_cite_cite_cite_cite_cite_cite_cite_c</pre>	# TYPE go_memstats_othe	r_sys_bytes gauge
<pre>Hilf go_exects_ts_ts_twice_invise_types number of bytes in use by the stack allocator. TTPE go_exects_ts_tack_invice_types gauge go_exects_ts_tack_ys_bytes Number of bytes obtained from system for stack allocator. HIEF go_exects_ts_tack_ys_bytes S4208 HIEF go_exects_ts_tack_ys_bytes S4208 HIEF go_exects_ts_types to 503 threads of bytes obtained from system. TTPE go_exects_ts_types to 503 threads of type obtained from system. TTPE go_exects_ts_types to 503 threads or type obtained from system. TTPE go_exects_ts_types to 503 threads or type obtained from system. TTPE to exects_ts_types to 503 threads or type obtained from system. TTPE to exects_types to 503 threads or type obtained from system. TTPE to the type of the type of type obtained from system. HIEF to the type of the type of type of type obtained from system. TTPE to the type of type of type of type of type obtained from system. HIEF to type of type o</pre>	<pre>go_menstats_other_sys_b</pre>	ytes 1.074585e+06
<pre>1 IPE growtst, stat, stat</pre>	# HELP go_memstats_stac	k_inuse_bytes Number of bytes in use by the stack allocator.
<pre>Construct_risk_risk_risk_risk_risk_risk_risk_risk</pre>	7 INPL go_memstats_stac.	k_inuse_bytes gauge
<pre>n HDL = [0] statist_princ.princ prince number of pyter obtained from system for frank allocator. generating track, princ the SQL state of the system of the system for frank allocator. generating tracks prince the system of the system of the system of frank allocator. generating tracks prince the system of the system of the system of frank allocator. generating tracks prince the system of the system of the system of frank allocator. HE Fr (n_tension Humber of OS hreads created. generating tracks prince the system of th</pre>	go_menstats_stack_inuse	_Dypes 024208
<pre>n if a judge is a second by the second by the obtained from system. H TTPE journeests, yrs, bytes maker of byte obtained from system. H TTPE journeests, yrs, bytes is 151654880070 H EEE journeests, yrs, bytes is 10000000000000000000000000000000000</pre>	<pre># HELF go_memstats_stac.</pre>	R_sys_Dytes Munder of Dytes Obtained from system for stack allocator.
<pre>HED (p</pre>	# HIFE g0_memstats_stat.	A_SYS_DY-085 gauge
<pre>H TYPE (o</pre>	go_menscats_state_sys_b	Just 24400
ro_marrit_proces_types1.51564080e+07 HEF polytecks Number of Direktors to the Marka Cluster. HTFE polytecks page HTFE kata_bockers gauge HTFE proces_gauge-cond_total organs HTFE proces_gauge-cond_total could HTFE proces_gauge-cond_total could even HTFE proces_gauge-cond_total outer Proces_gauge-cond_total outer HTFE proces_gauge.tog HTFE proces	TYPE do memotate eve	bytes manuel of bytes obtained from system.
FEEF co. threads Yumber of 05 threads created. TFFE poltreads gauge To threads fo HEEP katha brokers Number of Brokers in the Kafka Cluster. TFFE katha prokers Number of Brokers in the Kafka Cluster. TFFE katha prokers Number of Brokers in the Kafka Cluster. TFFE katha exporter_build_info for an entrie with a constant 'l' value labeled by version, revision, branch, and goversion from which kafka exporter vas built. TFFE kafka exporter_build_info for an entrie with a constant 'l' value labeled by version, revision, branch, and goversion from which kafka exporter vas built. TFFE kafka, exporter_build_info for an entrie with a constant 'l' value labeled by version, revision, branch, and goversion from which kafka exporter vas built. TFFE process_gau_second_total counter TFFE process_gau_second_total counter TFFE process_gau_second_total counter TFFE process_gau_fide Kariaan number of open file descriptors. TFFE process_gau_fide Kariaan number of open file descriptors. TFFE process_gau_fide Kariaan number of open file descriptors. TFFE process_gau_fide for the anory, bytes Resident memory size in bytes. TFFE process_gau_attic for and anory, bytes Resident memory size in bytes. TFFE process_gau_attic for an expony. TFFE process_gau_attic for an exporter build_total gauge process_cautatt_tas_seconds Start time of the process since unix epoch in seconds. TFFE process_gau_attic tas_seconds Start time of the process since unix epoch in seconds. TFFE process_pauter_tast_tast_seconds Start time of the process since unix epoch in seconds. TFFE process_paut_tast_tast_seconds Start time of the process since unix epoch in seconds. TFFE process_paut_tast_tast_seconds Start time of the process since unix epoch in seconds. TFFE process_paut_tast_tast_seconds 1.7037782609er00 TFFE process_paut_tast_tast_seconds 1.7037782609er00 TFFE process_paut_tast_tast_seconds 1.7037782609er00 TFFE process_paut_tast_tast_seconds 1.7037782609er00 TFFE process_paut_tast_tast_secondseconds 1.7037782609er00 TFFE process_paut_tast_tast_tast_seco	o menstats sys bytes 1	5156498+47
<pre>H TYE [0] threads gauge H THE [0] threads gauge H TEL Kafta_brokers Mumber of Brokers in the Kafta Cluster. H TEL Kafta_brokers gauge Kafta_brokers [ HEL Kafta_seports_build_info &amp; metric with a constant '1' value labeled by version. revision. branch. and goversion from which kafta_exporter was built. HEL Kafta_seports_build_info &amp; metric with a constant '1' value labeled by version. revision. branch. and goversion from which kafta_exporter was built. Hell Kafta_seports_build_info &amp; metric with a constant '1' value labeled by version. revision. branch. and goversion from which kafta_exporter was built. Hell Process_qu_second_total counter process_qu_second_total counter process_qu_second_total counter process_qu_second_total counter process_qu_second_total counter process_qu_second_total outer process_quescond_total outer process_quescond_total outer process_quescond_total 0.02 H HEL Process_quescond_total counter process_quescond_total 0.02 H HEL Process_quescond_total counter process_quescond_total 0.02 H HEL Process_gent_fds gauge process_quescond_total 0.04 H HEL Process_testiont_meancy_bytes for of open file descriptors. H TYF process_gent_fds 10 H HEL Process_testiont_meancy_bytes Resident memory size in bytes. H TYF process_testiont_meancy_bytes for 1.03472e+07 H HEL Process_testiont_meancy_bytes for 1.03472e+07 H HEL Process_testiont_meancy_bytes for 1.03472e+07 H HEL Process_virtual_meancy_bytes for status H HEL Process_virtual_meancy_max_bytes for status H HEL Process_virtual_meancy_m</pre>	# HELP go threads Numbe	r of OS threads created.
eg, threads 6 The La kfa, brokers wher of Brokers in the Kafka Cluster. H TFE kafka, brokers wage fafka brokers term build, info a metric with a constant 'l' value labeled by version. revision. branch. and goversion from which kafka, exporter yould, info farmand "EMD", goversion="ledsd6a9es0201305d4074e026f22c3lc760e6", version="l.4.2"] I H TFE kafka, exporter_yould, info farmator "EMD", goversion="ledsd6a9es0201305d4074e026f22c3lc760e6", version="l.4.2"] I H TEP process_opu_second_total counter TFE process_opu_second_total counter I TFE process_opu_second_total counter I TFE process_opu_second_total counter H TFE process_opu_second_total counter TFE process_opu_second_total counter I TFE process_opu_second_total counter I TFE process_opu_second_total counter I TFE process_opu_second_total counter H TFE process_opu_second_total counter I TFE process_total counter I TFE process_total counter I TFE process_total counter I TFE process_total counter success for the process since unix epoch in seconds. I TFE process_total counter success for the process since unix epoch in seconds. I TFE process_total counter success for the process since unix epoch in seconds. I TFE process_total counter success for the process since unix epoch in seconds. I TFE process_total counter success for the process since unix epoch in seconds. I TFE process_total counter success for the success for the process since unix epoch in seconds. I TFE process_total counter success for the process since unix epoch	# TYPE go threads gauge	
# EUP kafta_brokers Number of Brokers in the Kafta Cluster. # FEE kafta_brokers gauge Kafta_brokers gauge Kafta_brokers gauge Kafta_keporter_build_info fastric with a constant '1' value labeled by version. revision. branch. and goversion from which kafta_exporter vas built. I TFE kafta_keporter_build_info fastric I TFE process_que_scond_total counter I TFE process_que_scond_total gates I TFE proce	zo threads 6	
<pre>H TFE kafa_brokers gauge Kafa_brokers   H TFE kafa_brokers gauge Kafa_brokers   H TFE kafa_brokers   H H H L kafa_brokers   H</pre>	HELP kafka brokers Nu	mber of Brokers in the Kafka Cluster.
<pre>kaffa_cporter_build_info A metric with a constant '1' value labeled by version, revision, branch, and goversion from which kaffa_exporter vas built. N TTE kaffa_exporter_build_info fauge Maffa_exporter_build_info fauge N TTE process_pup.scond_total counter and system CFU the spent in seconds. nocess_pup.expond_total counter N TTE process_pup.scond_total counter N TTE process_cond_total .042 N TTE process_cond_total .042 N TTE process_cond_total .042 N TTE process_cond_total .043 N TTE process_cond_total .044 N TTE process_cond_total .044</pre>	# TYPE kafka_brokers ga	uge
HED kafa_exporter_build_info & metric with a constant '1' value labeled by version, revision, branch, and goversion from which kafk_exporter vas built. HTPE kafa_exporter_build_info Grauge safka_exporter_build_info Dranch="HEM".goversion="gol17.3", revision="1564ad5a9e3803135d40974e825f22e31c750e5",version="1.4.2"] 1 HTPE process_gnu_second_total counter HTPE process_gnu_second_total counter HTPE process_gnu_second_total counter TTPE process_gnu_second_total counter HTPE process_gnu_fds full momber of open file descriptors. TTPE process_gnu_fds full full momber of open file descriptors. TTPE process_gnu_fds full momber of open file descriptors. TTPE process_gnu_fds full HTEP process_gnu_fds full HTEP process_gnu_fds full momess_testidant_memory_bytes is 13172e+075 TTPE process_gnu_fds full HTEP process_testidant_memory_bytes full TTPE process_testidant_memory_bytes is 13172e+075 TTPE process_testidant_memory_bytes is 13472e+075 TTPE process_testidant_memory_bytes is 13472e+075 T	afka_brokers 1	
<pre>W TFE kafa_exporter_build_info gauge W TFE kafa_exporter_build_info gauge W TFE kafa_exporter_build_info gauge HELP process_pu_cload_total Total user and system CFU time spent in seconds. process_pu_cload_total Total user and system CFU time spent in seconds. process_pu_cload_total Total user and system CFU time spent in seconds. W HELP process_pu_cload_total FTE p</pre>	# HELP kafka_exporter_b	uild_info A metric with a constant '1' value labeled by version, revision, branch, and goversion from which kafka_exporter was built.
<pre>kafka_exportsr_build_infelbranch="HEAD"; goversion="gol1.7.3"; revision="16e4dd59e3820133646974e025f22e31c750e5",version="1.4.2"] 1 HTME process_pup_second_total lotal user and system CPU time spect in seconds. HTME process_pup_second_total lotal user and system CPU time spect on seconds. TOTS pup_second_total lotal user and system CPU time spect in seconds. HTME process_pup_second_total lotal user and system CPU time spect in seconds. TOTS pup_second_total lotal user and system CPU time spect on seconds. HTME process_pup_second_total lotal user and system CPU time spect on seconds. HTME process_poen_dfs tumes TTFE process_poen_dfs tumes TTFE process_poen_dfs tumes TTFE process_poen_dfs tumes TTFE process_poen_dfs tumesry_bytes Resident memory size in bytes. TTFE process_precident_memory_bytes for 150472e+00 HTME process_printed_heatory_bytes for 150472e+00 HTME process</pre>	# TYPE kafka_exporter_b	uild_info gauge
<pre>HELP process_ppu_second_total Total user and system CPU time spent in seconds. TFF process_pau_scife furnism number of open file descriptors. HEEP process_pau_ifs furnism number of open file descriptors. HEEP process_pau_ifs number of open file descriptors. HEEP process_resident_meancy_bytes functions. HEEP process_resident_meancy_bytes functions. HEEP process_resident_meancy_bytes functions. HEEP process_resident_meancy_bytes functions. HEEP process_restart_time_seconds Start time of the process since unix epoch in seconds. TFF process_restart_time_seconds funct is bytes. HEEP process_restart_time_seconds is applied to bytes.</pre>	<pre>xafka_exporter_build_in</pre>	fo{branch="HEAD",goversion="go1.17.3",revision="15e4ad6a9ea8203135d4b974e825f22e31c750e5",version="1.4.2"} 1
<pre>H TFE process_pu_seconds_total counter process_pu_seconds_total counter process_pu_seconds_total counter I TEP process_pus_file Number of open file descriptors. I TEP process_poen_fds tude I TEP process_poen_fds tude I TEP process_pen_fds fuge I TEP process_p</pre>	# HELP process_cpu_seco	nds_total Total user and system CPU time spent in seconds.
process_put_seconds_total 0.02 HET process_put_seconds_total 0.02 HET process_put_seconds_total 0.02 HET process_pen_fds Number of open file descriptors. HTFS process_pen_fds numery bytes Resident memory size in bytes. HET process_resides_memory_bytes resides HET process_resides_memory_bytes issue rocess_resides_memory_bytes issue rocess_resides_memory_bytes issue HET process_resides_memory_bytes issue HET process_rest time_seconds Start time of the process since unix epoch in seconds. HET process_rest time_seconds Start time of the process since unix epoch in seconds. HET process_rest time_seconds in 70037824089409 HET process_rest time_seconds in 70037824089409 HET process_virtual_memory_bytes into the memory size in bytes. HET process_virtual_memory_bytes into the seconds HET process_virtual_memory_mets issue in bytes. HET process_virtual_memory_mets issue in bytes. HET process_virtual_memory_mets issue in bytes. HET process_virtual_memory_mets bytes into its into the virtual memory available in bytes.	# TYPE process_cpu_seco	nds_total counter
<pre>Hilf process_max_its Maximum number of open file descriptors. HTPP process_max_its Maximum number of open file descriptors. HTPP process_open_ids number process_open_ids in use HTPP process_resident_menory_bytes Resident memory size in bytes. HTPP process_resident_memory_bytes is 15472e+07 HTPP process_resident_memory_bytes is 154770eB52e+05 HTPP process_resident_memory_bytes is 154770e</pre>	process_cpu_seconds_tot	a1 0.02
<pre>HIPE process_mat_ids gauge process_mat_ids (1048576+00) HEEP process_open_ids gauge HEEP process_open_ids gauge HEEP process_resident_memory_bytes family the same PTEP process_resident_memory_bytes family the same process_resident_memory_bytes (104876+00) HEEP process_rest_time_seconds Start time of the process since unix epoch in seconds. TTFE process_tart_time_seconds Start time of the process since unix epoch in seconds. TTFE process_tart_time_seconds Start time of the process since unix epoch in seconds. TTFE process_tart_time_seconds gauge process_tart_time_seconds family the start of the process since unix epoch in seconds. TTFE process_tart_time_seconds gauge process_virtual_memory_bytes family family the start of the same TTFE process_virtual_memory_bytes family family</pre>	# HELP process_max_fds .	Maximum number of open file descriptors.
<pre>process_Nat_rds 1.0400/00400 process_proc</pre>	W INPE process_max_tds	gauge
<pre>mil process_open_tas wander or open tile descriptors. TFF process_process_president_memory_bytes Resident memory size in bytes. HELP process_resident_memory_bytes rauge process_resident_memory_bytes is and the process since unix epoch in seconds. HTFF process_tatt_time_seconds Start take of the process since unix epoch in seconds. HTFF process_virtual_memory_bytes rauge process_virtual_memory_bytes for auge of the process since unix epoch in seconds. HTFF process_virtual_memory_bytes rauge process_virtual_memory_bytes for auge of the process since unix epoch in seconds. HTFF process_virtual_memory_bytes for auge of the process situal memory size in bytes. HTFF process_virtual_memory_bytes for auge of the process situal memory available in bytes. HTFF process_virtual_memory_max_bytes for auge process_virtual_memory_max_bytes is. 1.84674607309852+19 process_virtual_memory_max_bytes is.</pre>	process_max_tds 1.0485/	08-100
History goom file Unit History goom file Unit HIEP process, resident, menory, bytes Resident memory size in bytes. HIEP process, resident, memory, bytes Resident memory size in bytes. HIEP process, resident, memory, bytes file of the process since unix epoch in seconds. HIEP process, resident, memory, bytes file of the process since unix epoch in seconds. HIEP process, ritual, memory, bytes file of the process since unix epoch in seconds. HIEP process, ritual, memory, bytes file of the process since unix epoch in seconds. HIEP process, ritual, memory, bytes file of the process since unix epoch in seconds. HIEP process, ritual, memory, max, bytes file of the process since unix epoch in bytes. HIEP process, ritual, memory, max, bytes file since HIEP process, ritual, memory, max, bytes file since bytes HIEP process, ritual, memory max, bytes file since bytes HIEP process, ritual, memory max, bytes file since bytes HIEP process, ritual, memory, max, bytes file since bytes HIEP process, ritual, memory max, bytes file since bytes H	# HELF process_open_ids	winder of open file descriptors.
HEED process_recident_memory_hytes Resident memory size in bytes. HTEP process_recident_memory_hytes / superior function of the process since white epoch in seconds. HTEP process_retart_time_seconds Start time of the process since white epoch in seconds. HTEP process_retart_time_seconds functions Start time of the process since white epoch in seconds. HTEP process_retart_time_seconds functions for the process since white epoch in seconds. HTEP process_retart_time_seconds functions for the process since white epoch in seconds. HTEP process_retart_time_seconds functions for the process since white epoch is seconds for the process since white epoch is second for the process since is a second for the process since white epoch is second for the process since white epoch is second for the process since is a second for the process since white epoch is second for the process since is a second for the process since white epoch is a second for the process since is a second for the process since is a second for the process sis a second for the process sinc	# TIFE process_open_tus	Sansa.
H TYE process_texident_memory_bytes range more process_texident_memory_bytes is supported and bytes H HEE process_tart_time_econds Start time of the process since unix epoch in seconds. H THE process_start_time_econds is supported by the process since unix epoch in seconds. H THE process_trittime_econds is not bytes. H TYE process_trittime_econds is not bytes. H TYE process_trittime_econds bytes is not bytes. H TYE process_trittime_econd_metry_bres is not bytes. H TYE process_trittime_econd_metry_bres is not bytes.	# WELP process resident	menory bytes Resident menory size in bytes
process_resident_memory_bytes 1.5134724417 H TEP process_start_time_seconds Start tame of the process since unix epoch in seconds. H TYPE process_start_time_seconds flows H TEP process_virtual_nearcy_bytes Virtual nearcy size in bytes. H TEP process_virtual_nearcy_bytes for agge process_virtual_nearcy_bytes 7.34260444408 H TEP process_virtual_nearcy_bytes 7.3426044408 H TTP process_virtual_nearcy_mytes 7.3426044408 H TTP process_virtual_nearcy_mytes 7.3426044408 H TTP process_virtual_nearcy_matytes 1.342604408 H TTP process_virtual_nearcy_matytes 1.342604408	# TYPE process_resident	_monory_bytes resident memory size in bytes.
H HEUF process_start_time_seconds Start time of the process since waix epoch in seconds. # TWF process_start_time_seconds sums process_start_time_seconds 1.702573782499-00 # TWF process_virtual_memory_bries Virtual memory size in bytes. # TWF process_virtual_memory_bries 7.340944+10 # HEUP process_virtual_memory_max_bries Maximum amount of virtual memory available in bytes. # TWF process_virtual_memory_max_bries Maximum amount of virtual memory available in bytes. # TWF process_virtual_memory_max_bries factors for the second sec	process resident memory	hotes 1513472+407
H TYE process_ttat_time_seconds gauge process_ttat_time_seconds !Dug053782489e00 H HELP process_virtual_nearcy_bytes furtual nearcy size in bytes. H TYE process_virtual_nearcy_bytes for auge process_virtual_nearcy_bytes ?.3426944e008 H TYE process_virtual_nearcy_max_bytes Instinum amount of virtual nearcy available in bytes. H TYE process_virtual_nearcy_max_bytes Instinum amount of virtual nearcy available in bytes.	# HELP process start ti	be seconds Start time of the process since unix enoch in seconds.
process_start_time_seconds in70257326309e-00 HEMP process_virtual_memory_brief Virtual memory size in bytes. H TVE process_virtual_memory_brief Virtual memory size in bytes. H HEMP process_virtual_memory_max_brief Maximum amount of virtual memory available in bytes. H TVE process_virtual_memory_max_brief information of virtual memory available in bytes. H TVE process_virtual_memory_max_brief information of virtual memory available in bytes.	# TYPE process start ti	ne seconds gauge
H HEE process virtual menory bytes Virtual menory size in bytes. H TEE process virtual_menory bytes for agent H HEE process virtual_menory pack bytes for a strain and the strain and the strain and the strain and the strain H THE process virtual_menory max bytes for a strain and the strain and the strain and the strain and the strain H THE process virtual menory max bytes for a strain and the s	process_start_time_seco	nds 1.70253782489e+09
H TYE process virtual memory bytes gauge process virtual memory bus 7. 200044+00 H HELP process virtual memory max bytes Maximu amount of virtual memory available in bytes. H YEE process virtual memory max bytes function of the state of the state of the state of the state of the state process virtual memory max bytes 1.84474007309552+19 H HELP members wirtual memory max bytes called function with a state of source being source	# HELP process virtual :	nemory bytes Virtual memory size in bytes.
process_virtual_memory_hyptes 7.3426944+060 M TFE process_virtual_memory_max_bytes Maximum amount of virtual memory available in bytes. M TFE process_virtual_memory_max_bytes [auge process_virtual_memory_max_bytes ].844674407370552e+19 mombar of converse virtual_memory_max_bytes ].844674407370552e+19	# TYPE process_virtual_	nemory_bytes gauge
# HELP process virtual neary max bytes Haxiaus anount of virtual memory available in bytes. # TEPE process virtual neary max bytes space rocess virtual memory max bytes 1.844544003709552+19 . HERP scatter nearbot had byte converts in # 11446 Current womber of scattere being spaced	process_virtual_memory_	bytes 7. 3426944e+08
# TYPE process_virtual_memory_max_bytes gauge process_virtual_memory_max_bytes 1.8446744073709552et19 UMP provider patria budlay request is it ick four and muchar of scrape being several	# HELP process_virtual_	nemory_max_bytes Maximum amount of virtual memory available in bytes.
process virtual_memory_max_bytes 1.8446744073709552e119	# TYPE process_virtual_	nemory_max_bytes gauge
" WEIP promitty matrix handler requests in flight Current number of screenes being served	process_virtual_memory_	nax_bytes 1.8446744073709552e+19
· DELE DECOMPLETE DECOMPLETE DECOMPLEXES IN LITTERE COPERNY DE SCHOOLS DETRY SECOND.	4 HELP promhttp metric 1	handler remuests in flight Current number of scranes being served.

----结束

#### 采集 CCE 集群的业务数据

通过<mark>新增PodMonitor</mark>方式为应用配置可观测监控Prometheus版的采集规则,监控部 署在CCE集群内的应用的业务数据。

#### 🛄 说明

如下指标采集的周期是30秒,所以等待大概30秒后才能在AOM的界面上查看到上报的指标。

#### 配置信息如下:

apiVersion: monitoring.coreos.com/v1 kind: PodMonitor metadata: name: kafka-exporter namespace: default spec: namespaceSelector: matchNames: - default # exporter 所在的命名空间 podMetricsEndpoints: - interval: 30s path: /metrics port: metric-port selector: matchLabels: k8s-app: kafka-exporter

#### 验证指标上报到 AOM

- 步骤1 登录AOM 2.0控制台。
- 步骤2 在左侧菜单栏中选择 "Prometheus监控 > 实例列表"。
- 步骤3 单击接入了该CCE集群的"prometheus for CCE"实例名称,进入实例详情页面。

步骤4 在"服务发现"页面的"指标"页签下,选择对应集群。

步骤5 选择Job: {namespace}/kafka-exporter,可以查询到kafka开头的自定义指标。

----结束

#### 在 AOM 上配置仪表盘和告警

通过仪表盘功能可视化监控CCE集群数据,通过告警规则功能,在集群发生故障时能够 及时发现并预警。

- 配置仪表盘图表
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"仪表盘",单击"创建仪表盘"新建一个仪表盘,详 情可参见创建仪表盘。
  - c. 在仪表盘页面选择实例类型为"Prometheus for CCE"的实例并单击"添加 图表",详情请参见<mark>添加图表至仪表盘</mark>。
- 配置告警
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"告警管理 > 告警规则"。
  - c. 单击"创建告警规则"配置告警,详情请参见创建指标告警规则。

# 5.4 Memcached Exporter 接入

#### 操作场景

使用Memcached过程中需要对Memcached运行状态进行监控,以便了解Memcached 服务是否运行正常,排查Memcached故障等。Prometheus监控服务提供了CCE容器场 景下基于Exporter的方式来监控Memcached运行状态。本文为您介绍如何使用 Prometheus监控服务Memcached。

#### 🛄 说明

为了方便安装管理Exporter,推荐使用CCE统一管理。

#### 前提条件

- CCE服务已拥有CCE集群,已安装Memcached。
- 服务已接入可观测Prometheus监控并接入CCE集群,具体请参见Prometheus实例 for CCE。
- 已将memcached\_exporter镜像上传到SWR,具体操作请参见使用容器引擎客户端上传镜像。

#### Memcached Exporter 部署

- 步骤1 登录CCE控制台。
- 步骤2 单击已接入的CCE集群名称,进入该集群的管理页面。
- 步骤3 执行以下操作完成Exporter部署。

1. 配置密钥。

在左侧导航栏中选择"配置与密钥",单击页面右上角"YAML创建"。YAML配 置示例如下:

apiVersion: v1 kind: Secret metadata: name: memcached-exporter-secret namespace: default type: Opaque stringData: memcachedURI: 120.46.215.4:11211 # Memcached地址

#### 🛄 说明

- Memcached 连接串的格式为: http://{ip}:{port}。
- 配置密钥的详细操作参见创建密钥。
- 2. 部署Memcached Exporter。

```
在左侧导航栏中选择"工作负载",选择"无状态负载"页签,单击右上角的
"YAML创建",以YAML的方式部署Exporter。
```

#### YAML配置示例如下:

apiVersion: apps/v1 kind: Deployment metadata: labels: k8s-app: memcached-exporter # 根据业务需要调整 name: memcached-exporter # 根据业务需要调整 namespace: default spec: replicas: 1 selector: matchLabels: k8s-app: memcached-exporter # 根据业务需要调整 template: metadata: labels: k8s-app: memcached-exporter # 根据业务需要调整 spec: containers: - env: - name: Memcached\_Url valueFrom: secretKeyRef: name: memcached-exporter-secret # 对应上一步中的 Secret 的名称 key: memcachedURI # 对应上一步中的 Secret Key - name: Memcached\_ALL value: "true" image: swr.cn-east-3.myhuaweicloud.com/aom-org/bitnami/memcached-exporter:0.13.0 #前提条 件中上传到swr中的镜像 imagePullPolicy: IfNotPresent name: memcached-exporter ports: - containerPort: 9150 name: metric-port securityContext: privileged: false terminationMessagePath: /dev/termination-log terminationMessagePolicy: File dnsPolicy: ClusterFirst imagePullSecrets: - name: default-secret restartPolicy: Always schedulerName: default-scheduler securityContext: {} terminationGracePeriodSeconds: 30

```
apiVersion: v1
kind: Service
metadata:
name: memcached-exporter
spec:
type: NodePort
selector:
k8s-app: memcached-exporter
ports:
- protocol: TCP
nodePort: 30122
port: 9150
targetPort: 9150
```

#### 🛄 说明

更多Exporter详细参数介绍请参见 memcached\_exporter。

- 3. 验证Memcached Exporter是否部署成功。
  - a. 在工作负载列表中"无状态负载"页签下,单击<mark>步骤3.2</mark>创建的无状态工作负载的名称,在实例列表中单击操作列下的"更多 > 日志",可以查看到 Exporter成功启动并暴露访问地址。

#### **图 5-9** 查看日志

日志

♀ 当前显示的日志内容为容器衍准输出日志,不具备持久化和高阶运维能力,如需使用更完确的日志能力,可使用日志中心应功能,如用户开启了AOM采集标准输出的功能,可前往AOM 查阅更多的负载日志, 查着 AOM 日志搜索应
案例 memcached-exporter-845 v 容器 memcached-exporter v 行数 200 v 下載 Q 定时期新 305
ts=2023-12-13T08:16:47.640Z caller=main.go:62 level=info msg="Starting memcached_exporter" version="(version=0.13.0, branch=HEAD,
revision=0a6e2f02511aefdd61d68a0ff8b6b3702af2f412)"
ts=2023-12-13T08:16:47.640Z caller=main.go:63 level=info msg="Build context" context="(go=go1.20.5, platform=linux/amd64, user=bit
nami, date=20230616-14:49:41, tags=\~netgo\~)~
ts=2023-12-13T08:16:47.641Z caller=tls_config.go:274 level=info msg="Listening on" address=[::]:9150
ts=2023-12-13T08:16:47.641Z caller=tls_config.go:277 level=info msg="TLS is disabled." http2=false address=[::]:9150

- b. 验证。有以下三种方法进行验证:
  - 登录集群节点执行如下任意一种命令: curl http://{集群IP}:9150/metrics curl http://{集群任意节点私有IP}:30122/metrics
  - 访问地址: http://{集群任意节点的公网IP}:30122/metrics。

#### 图 5-10 访问地址

← → C ▲ :30122/metrics
a III a Animatan Tanan Tanan Anata
go_menstats_mcache_inuse_bytes 19200
# HELP go_memstats_ncache_sys_bytes Number of bytes used for mcache structures obtained from system.
# TYPE go_memstats_ncache_sys_bytes gauge
go_menstats_acache_sys_bytes_31200
# HELP co memstats mspan inuse bytes Number of bytes in use by mspan structures.
# TYPE go memastats magnam inuse bytes gauge
so mensiats aspan inuse bytes 250880
# HELP so mensions are bytes Number of bytes used for manan structures obtained from system.
# TYPE so memostate memory bytes gauge
so memorals more buries 277440
# HELP so apastate part of hytes Weaker of heap bytes when part sarbage collection will take place.
# TYPE on meastats next of bytes gauge
go memorate next an hyter 5 42824e+06
# HELP so awastate other sus butes Number of bytes used for other system allocations.
# TYPE on memotate other oue hotes sense
a manetate other we have a 2008/56-06
WEIP on magning track investments of betar in use by the stark allocator
The standard of the
n name stark invas hutse () VASIGALDS
summaria_searchic_start_interior a tertar Webbar of byter obtained from rester for stark allocator
The substate of a by the base same
a nametate stack eve botes = 1.26194a+16
WIDE on magnitude such bother Number of bother obtained from system
# TYPE on manofato ovo hotao sausa
n nametale sys hybrid (2010) (200) (2010) (2
# HELP so threads Number of OS threads greated.
# TYPE to threads source
an threads 18
# HELP mencached exporter build info & metric with a constant 'l' value labeled by version, revision, branch, goversion from which mencached exporter was built, and the goos and goarch for the built
# TYPE memorahed exporter build info saure
meanached exporter build info Branch="HEAD", soarch="amd64", soar="linux", soversion="so1.20.5", revision="0a6e2f02511aefdd61d68a0ff8b6b3702af2f412", tass="\"netso\"", version="0.13.0"}
# HELP memorahed up Could the memorahed server be reached.
# TYPE penceched un saure
aeacached up 0
# HELP process cpu seconds total Total user and system CPU time spent in seconds.
# TYPE process ou seconds total counter
process cpu seconds total 10.14
# HELF process max fds Maximum number of open file descriptors.
# TYPE process max fds gauge
process max fds 1,048576e+06
# HELP process open fds Number of open file descriptors.
# TYPE process open fds gauge
process open fds 10
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 3.1174656e+07
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.70245540724e+09
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 1.949995008e+09
# HELP process virtual memory max bytes Maximum amount of virtual memory available in bytes.

 在实例列表中单击操作列下的"更多 > 远程登录",执行如下命令。 curl http://localhost:9150/metric

#### 图 5-11 执行命令

aser wangheeessepselit maenaner y cara isoarer meeraes	
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.	
# TYPE go_gc_duration_seconds summary	
go_gc_duration_seconds{quantile="0"} 0	
go_gc_duration_seconds{quantile="0.25"} 0	
go_gc_duration_seconds{quantile="0.5"} 0	
go_gc_duration_seconds{quantile="0.75"} 0	
go_gc_duration_seconds{quantile="1"} 0	
go_gc_duration_seconds_sum 0	
go_gc_duration_seconds_count 0	
# HELP go_goroutines Number of goroutines that currently exist.	
# TYPE go_goroutines gauge	
go_goroutines 9	
# HELP go_info Information about the Go environment.	
# TYPE go_info gauge	
go_info{version="go1.20.5"} 1	
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.	
# TYPE go_memstats_alloc_bytes gauge	
go_memstats_alloc_bytes 504008	
# HELP go_memstats_alloc_bytes_total lotal number of bytes allocated, even if freed.	
# TYPE Bo_memstats_alloc_bytes_total counter	
go_memstats_alloc_bytes_total 504008 M_FILD =thet_ buck hash sug butes Number of butes used by the profiling bucket back table	
# HELP go_memstats_ouck_nash_sys_bytes Number of bytes used by the profiling bucket hash table	
# TYPE go_memotats_ouck_nash_sys_bytes gauge	
go_memistals_uuck_indsin_sys_uytes 4545 H UFLD ap magnetists finans total Tatal numban of finans	
* TELP go_memotats_rrees_total routinumber of Trees.	
n momentats froes total Q	
sy_memistals_inces_total 0 # HELD on mometats or sys hytos Number of hytos used for garbage collection system metadata	
TYPE on mentates or sys bytes dauge	
n nemstats gr svs hvtas 6 74584e+06	
$y = \frac{1}{2}$	
TYPE go memorats head alloc bytes gauge	
zo menstats heap alloc bytes 504008	
# HELP go memstats heap idle bytes Number of heap bytes waiting to be used.	
# TYPE go memstats heap idle bytes gauge	
go memstats heap idle bytes 1.753088e+06	
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.	
	Γ.

----结束

#### 添加采集任务

通过<mark>新增PodMonitor</mark>方式为应用配置可观测监控Prometheus版的采集规则,监控部 署在CCE集群内的应用的业务数据。

文档版本 01 (2024-05-27)

#### 🛄 说明

如下示例中指标采集的周期是30秒,所以等待大概30秒后才能在AOM的界面上查看到上报的指标。

apiVersion: monitoring.coreos.com/v1 kind: PodMonitor metadata: name: memcached-exporter namespace: default spec: namespaceSelector: matchNames: - default # exporter所在的命名空间 podMetricsEndpoints: - interval: 30s path: /metrics port: metric-port selector: matchl abels: k8s-app: memcached-exporter

#### 验证指标上报到 AOM

- 步骤1 登录AOM 2.0控制台。
- 步骤2 在左侧菜单栏中选择 "Prometheus监控 > 实例列表"。
- 步骤3 单击接入了该CCE集群的"prometheus for CCE"实例名称,进入实例详情页面。
- 步骤4 在"服务发现"页面的"指标"页签下,选择集群。
- **步骤5** 选择Job:{namespace}/memcached-exporter,可以查询到go\_memstats开头的 memcached指标。

----结束

#### 在 AOM 上配置仪表盘和告警

通过仪表盘功能可视化监控CCE集群数据,通过告警规则功能,在集群发生故障时能够 及时发现并预警。

- 配置仪表盘图表
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"仪表盘",单击"创建仪表盘"新建一个仪表盘,详 情可参见创建仪表盘。
  - c. 在仪表盘页面选择实例类型为"Prometheus for CCE"的实例并单击"添加 图表",详情请参见<mark>添加图表至仪表盘</mark>。
- 配置告警
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"告警管理 > 告警规则"。
  - c. 单击"创建告警规则"配置告警,详情请参见创建指标告警规则。

# 5.5 MongoDB Exporter 接入

#### 操作场景

使用MongoDB过程中需要对MongoDB运行状态进行监控,以便了解MongoDB服务是 否运行正常,排查MongoDB故障问题原因。Prometheus监控服务提供了CCE容器场景 下基于Exporter的方式来监控MongoDB运行状态。本文介绍如何部署Exporter以及实 现MongoDB Exporter告警接入等操作。

#### 🗀 说明

为了方便安装管理Exporter,推荐使用CCE进行统一管理。

#### 前提条件

- CCE服务已拥有CCE集群,已安装MongoDB。
- 服务已接入可观测Prometheus监控并接入CCE集群,具体请参见Prometheus实例 for CCE。
- 已将mongodb\_exporter镜像上传到SWR,具体操作请参见使用容器引擎客户端 上传镜像。

#### MongoDB Exporter 部署

- 步骤1 登录CCE控制台。
- 步骤2 单击已接入的CCE集群名称,进入该集群的管理页面。
- 步骤3 执行以下操作完成Exporter部署。
  - 1. 配置密钥。

```
在左侧导航栏中选择"配置与密钥",在页面右上角单击"YAML创建"。YAML
配置示例如下:
```

```
apiVersion: v1
kind: Secret
metadata:
name: mongodb-secret-test
namespace: default
type: Opaque
stringData:
datasource: "mongodb://{user}:{passwd}@{host1}:{port1},{host2}:{port2},{host3}:{port3}/admin" #
对应连接URI
```

#### 🛄 说明

- 密码已按照Opaque加密。
- 配置密钥的详细操作参见<mark>创建密钥</mark>。
- 2. 部署MongoDB Exporter。

在左侧导航栏中选择"工作负载",在右上角单击"创建负载",选择"负载类 型"为无状态工作负载Deployment,选择需要的命名空间部署MongoDB Exporter。如果以YAML的方式部署Exporter,YAML配置示例如下: apiVersion: apps/v1 kind: Deployment metadata: labels:

```
k8s-app: mongodb-exporter # 根据业务需要调整,建议加上MongoDB实例的信息
 name: mongodb-exporter # 根据业务需要调整,建议加上MongoDB实例的信息
 namespace: default #需要和CCE集群中安装的MongoDB命名空间一致
spec:
 replicas: 1
 selector:
  matchLabels:
   k8s-app: mongodb-exporter # 根据业务需要调整,建议加上MongoDB实例的信息
 template:
  metadata:
   labels:
    k8s-app: mongodb-exporter # 根据业务需要调整,建议加上MongoDB实例的信息
  spec:
   containers:
     - args:
       ---collect.database # 启用数据库指标采集

    --collect.collection # 启用集合指标采集
    --collect.topmetrics # 启用数据库表头指标信息采集
    --collect.indexusage # 启用索引使用统计信息采集

       ---collect.connpoolstats # 启动MongoDB连接池统计信息采集
      env:
       - name: MONGODB_URI
        valueFrom:
         secretKeyRef:
          name: mongodb-secret-test
           key: datasource
     image: swr.cn-north-4.myhuaweicloud.com/mall-swarm-demo/mongodb-exporter:0.10.0
     imagePullPolicy: IfNotPresent
     name: mongodb-exporter
     ports:
        containerPort: 9216
        name: metric-port # 这个名称在配置抓取任务的时候需要
      securityContext:
       privileged: false
      terminationMessagePath: /dev/termination-log
     terminationMessagePolicy: File
   dnsPolicy: ClusterFirst
   imagePullSecrets:
    - name: default-secret
   restartPolicy: Always
   schedulerName: default-scheduler
   securityContext: { }
   terminationGracePeriodSeconds: 30
apiVersion: v1
kind: Service
metadata:
name: mongodb-exporter
spec:
 type: NodePort
 selector:
  k8s-app: mongodb-exporter
 ports:
  - protocol: TCP
   nodePort: 30003
   port: 9216
   targetPort: 9216
 🗀 说明
```

更多Exporter详细参数介绍请参见mongodb\_exporter。

- 3. 验证MongoDB Exporter是否部署成功。
  - a. 在工作负载列表中"无状态负载"页签下,单击<mark>步骤3.2</mark>创建的无状态工作负载的名称,在实例列表中单击操作列下的"更多 > 日志",可以查看到 Exporter成功启动并暴露访问地址。

#### 图 5-12 查看日志

日志	×
♀ 当前显示的日志内容为容器标准输出日志,不具备持久化和高阶运输能力,如需使用更先高的日志能力,可使用日志中心已功能,如用户开启了 AOM 采集标准输出的功能,可前往 AOM 查阅更多的负数日志,查看 AOM 日志提繁ピ	
発例 mongodb-exporter-7b9d9 ▼ 録録 mongodb-exporter ▼ 行数 200 ▼ 下載 Q 定时期新 0000 time="2002+12.10702-16.107" 1.001-16.5 xx=="2">1000-12.10702-12.10702-16.107" 1.001-16.5 xx=="2">1000-1000-1000-1000-1000-1000-1000-100	
<pre>time="2020-12-13T03:16:192" level=info msg="build context (go=go1.11.13, user=, date=19700101-00:00:00)" source="mongod_exporter.c")</pre>	
go:95" time="2023-12-13TU3:16:192" level=info msg="Starting HTTP server for http://:9216/metrics" source="server.go:140"	

- b. 验证。有以下三种方法进行验证:
  - 登录集群节点执行如下任意一种命令: curl http://{集群IP}:9216/metrics curl http://{集群任意节点私有IP}:30003/metrics
  - 访问地址: http://{集群任意节点的公网IP}:30003/metrics。

#### 图 5-13 访问地址

在实例列表中单击操作列下的"更多 > 远程登录",执行如下命令。 curl http://localhost:9216/metric

----结束

#### 采集 CCE 集群的业务数据

通过<mark>新增PodMonitor</mark>方式为应用配置可观测监控Prometheus版的采集规则,监控部 署在CCE集群内的应用的业务数据。

#### 🛄 说明

如下示例中指标采集的周期是30秒,所以等待大概30秒后才能在AOM的界面上查看到上报的指标。

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
 name: mongodb-exporter
 namespace: default
spec:
 namespaceSelector:
  matchNames:
    - default # exporter所在的命名空间
 podMetricsEndpoints:
 - interval: 30s
  path: /metrics
  port: metric-port
 selector:
  matchLabels:
   k8s-app: mongodb-exporter
```

#### 验证指标上报到 AOM

- 步骤1 登录AOM 2.0控制台。
- 步骤2 在左侧菜单栏中选择 "Prometheus监控 > 实例列表"。
- 步骤3 单击接入了该CCE集群的"prometheus for CCE"实例名称,进入实例详情页面。
- 步骤4 在"服务发现"页面的"指标"页签下,选择集群。
- **步骤5** 选择job: {namespace}/MongoDB-exporter,可以查询到mongodb开头的自定义指标。

#### ----结束

#### 在 AOM 上配置仪表盘和告警

通过仪表盘功能可视化监控CCE集群数据,通过告警规则功能,在集群发生故障时能够 及时发现并预警。

- 配置仪表盘图表
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"仪表盘",单击"创建仪表盘"新建一个仪表盘,详 情可参见**创建仪表盘**。
  - c. 在仪表盘页面选择实例类型为 "Prometheus for CCE"的实例并单击 "添加 图表",详情请参见**添加图表至仪表盘**。
- 配置告警
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"告警管理 > 告警规则"。
  - c. 单击"创建告警规则"配置告警,详情请参见创建指标告警规则。

# 5.6 ElasticSearch Exporter 接入

#### 操作场景

使用ElasticSearch过程中需要对ElasticSearch运行状态进行监控,例如集群及索引状态 等。Prometheus监控服务提供了CCE容器场景下基于Exporter的方式来监控 ElasticSearch运行状态。本文介绍如何部署ElasticSearch Exporter以及实现 ElasticSearch Exporter告警接入等操作。

#### 🛄 说明

为了方便安装管理Exporter,推荐使用CCE进行统一管理。

#### 前提条件

- CCE服务已拥有CCE集群,已安装ElasticSearch。
- 服务已接入可观测Prometheus监控并接入CCE集群,具体请参见Prometheus实例 for CCE。
- 已将elasticsearch\_exporter镜像上传到SWR,具体操作请参见使用容器引擎客 户端上传镜像。

#### ElasticSearch Exporter 部署

- 步骤1 登录CCE控制台。
- 步骤2 单击已接入的CCE集群名称,进入该集群的管理页面。
- 步骤3 执行以下操作完成Exporter部署。
  - 1. 配置密钥。

在左侧导航栏中选择"配置与密钥",单击页面右上角"YAML创建",YAML配 置示例如下:

apiVersion: v1 kind: Secret metadata: name: es-secret-test namespace: default type: Opaque stringData: esURI: http://124.70.14.51:30920 #对应 ElasticSearch 的 URI,IP为集群IP或集群任意节点IP

#### 门 说明

- ElasticSearch连接串的格式为 <proto>://<user>:<password>@<host>:<port>,例如 http://admin:pass@localhost:9200。也可以不设置密码,例如设置为: http:// 10.247.43.50:9200。
- 密码已按照Opaque加密。
- 配置密钥的详细操作参见<mark>创建密钥</mark>。
- 2. 部署ElasticSearch Exporter。

在左侧导航栏中选择"工作负载",在右上角单击"创建负载",选择"负载类型"为无状态工作负载Deployment,选择需要的命名空间部署ElasticSearch Exporter。如果以YAML的方式部署Exporter,YAML配置示例如下: apiVersion: apps/v1 kind: Deployment

```
metadata:
labels:
  k8s-app: es-exporter # 根据业务需要调整
 name: es-exporter # 根据业务需要调整
namespace: default # 选择一个适合的 namespace 来部署 Exporter, 如果没有需要新建一个
spec:
replicas: 1
 selector:
  matchLabels:
   k8s-app: es-exporter # 根据业务需要调整
 template:
  metadata:
   labels:
    k8s-app: es-exporter # 根据业务需要调整
  spec:
   containers:
   - env:
      - name: ES_URI
       valueFrom:
        secretKeyRef:
          name: es-secret-test # 对应上一步中的 Secret 的名称
          key: esURI # 对应上一步中的 Secret Key
      - name: ES_ALL
       value: "true'
    image: swr.cn-north-4.myhuaweicloud.com/mall-swarm-demo/es-exporter:1.1.0
    imagePullPolicy: IfNotPresent
    name: es-exporter
    ports:
     - containerPort: 9114
     name: metric-port
    securityContext:
      privileged: false
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
   dnsPolicy: ClusterFirst
   imagePullSecrets:
   - name: default-secret
   restartPolicy: Always
   schedulerName: default-scheduler
   securityContext: {}
   terminationGracePeriodSeconds: 30
apiVersion: v1
kind: Service
metadata:
 name: es-exporter
name-space: default # 与Exporter部署的namespace相同
spec:
type: NodePort
 selector:
  k8s-app: es-exporter
 ports:
  - protocol: TCP
   nodePort: 30921
   port: 9114
   targetPort: 9114
 □□ 说明
```

上述示例通过ES\_ALL采集了所有ElasticSearch的监控项,可以通过对应的参数进行调整, Exporter更多详细的参数请参见 **elasticsearch\_exporter**。

- 3. 验证ElasticSearch Exporter是否部署成功。
  - a. 在工作负载列表中"无状态负载"页签下,单击<mark>步骤3.2</mark>创建的无状态工作负载的名称,在实例列表中单击操作列下的"更多 > 日志",可以查看到 Exporter成功启动并暴露访问地址。

#### 图 5-14 查看日志

日志	×
♀ 当前显示的日志内容为容器标准输出日志、不具备持久化和高阶运维能力、如需使用更完善的日志能力、可使用日志中心び功能、如用户开启了 AOM 采果标准输出的 功能,可前往 AOM 查阅更多的负款日志,查着 AOM 日志搜索	
突例 es-exporter-74b88b9c97 * 容器 es-exporter * 行数 200 * 下戦 Q 定时用新 305	
level=info ts=2023-12-11T07:37:43.604733618Z caller=clusterinfo.go:200 msg="triggering initial cluster info call"	
level=info ts=2023-12-11T07:37:43.604808503Z caller=clusterinfo.go:169 msg="providing consumers with updated cluster info label"	
level=info ts=2023-12-11T07:37:43.619618189Z caller=main.go:148 msg="started cluster info retriever" interval=5mOs	
level=info ts=2023-12-11707:37:43.0198212782 caller=main.go:188 msg=´starting elasticsearch_exporter´ <mark>addr=:9114</mark>	

- b. 验证。有以下三种方法进行验证:
  - 登录集群节点执行如下任意一种命令: curl http://{集群IP}:9114/metrics curl http://{集群任意节点私有IP}:30921/metrics
  - 访问地址: http://{集群任意节点的公网IP}:30921/metrics。

#### 图 5-15 访问地址



 在实例列表中单击操作列下的"更多 > 远程登录",执行如下命令。 curl http://localhost:9114/metric

----结束

#### 采集 CCE 集群的业务数据

通过<mark>新增PodMonitor</mark>方式为应用配置可观测监控Prometheus版的采集规则,监控部 署在CCE集群内的应用的业务数据。

#### 🛄 说明

如下示例中指标采集的周期是30秒,所以等待大概30秒后才能在AOM的界面上查看到上报的指标。

apiVersion: monitoring.coreos.com/v1 kind: PodMonitor metadata:

name: elasticSearch-exporter
namespace: default
spec:
namespaceSelector: # 选择监控Exporter部署所在的namespace
matchNames:
- default # exporter所在的命名空间
podMetricsEndpoints:
- interval: 30s # 设置指标采集周期
path: /metrics # 填写Prometheus Exporter对应的Path的值,默认/metrics
port: metric-port # 填写Prometheus Exporter对应YAML的ports的name
selector: # 填写要监控Exporter Pod的Label标签,以定位目标Exporter
matchLabels:
k8s-app: elasticSearch-exporter

#### 验证指标上报到 AOM

- 步骤1 登录AOM 2.0控制台。
- 步骤2 在左侧菜单栏中选择 "Prometheus监控 > 实例列表"。
- 步骤3 单击接入了该CCE集群的"prometheus for CCE"实例名称,进入实例详情页面。
- 步骤4 在"服务发现"页面的"指标"页签下,选择集群。
- 步骤5 选择Job: {namespace}/elasticsearch-exporter,可以查询到elasticsearch开头的自定义指标。

#### ----结束

#### 在 AOM 上配置仪表盘和告警

通过仪表盘功能可视化监控CCE集群数据,通过告警规则功能,在集群发生故障时能够 及时发现并预警。

- 配置仪表盘图表
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"仪表盘",单击"创建仪表盘"新建一个仪表盘,详 情可参见**创建仪表盘**。
  - c. 在仪表盘页面选择实例类型为"Prometheus for CCE"的实例并单击"添加 图表",详情请参见<mark>添加图表至仪表盘</mark>。
- 配置告警
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"告警管理 > 告警规则"。
  - c. 单击"创建告警规则"配置告警,详情请参见<mark>创建指标告警规则</mark>。

## 5.7 Redis Exporter 接入

#### 操作场景

使用数据库Redis过程中需要对Redis运行状态进行监控,以便了解Redis服务是否运行 正常,及时排查Redis故障等。Prometheus监控服务提供了CCE容器场景下基于 Exporter的方式来监控Redis运行状态。本文为您介绍如何使用Prometheus监控 Redis。

#### 🛄 说明

为了方便安装管理Exporter,推荐使用云容器引擎CCE进行统一管理。

#### 前提条件

- CCE服务已拥有CCE集群,已安装Redis。
- 服务已接入可观测Prometheus监控并接入CCE集群,具体请参见Prometheus实例 for CCE。
- 已将redis\_exporter镜像上传到SWR,具体操作请参见使用容器引擎客户端上传 镜像。

#### Redis Exporter 部署

步骤1 登录CCE控制台。

1.

- 步骤2 单击已接入的CCE集群名称,进入该集群的管理页面。
- 步骤3 执行以下步骤完成Exporter部署。

```
在左侧导航栏中选择"配置与密钥",选择"密钥"页签,单击页面右上角
"YAML创建",YAML配置示例如下:
apiVersion: v1
kind: Secret
metadata:
name: redis-secret-test
namespace: default # 与Exporter部署的namespace相同
type: Opaque
stringData:
password: redis123 #对应 Redis 密码
```

🗋 说明

- 密码已按照Opaque加密。
- 配置密钥的详细操作参见创建密钥。
- 2. 部署Redis Exporter。

在左侧菜单栏中选择"工作负载",选择"无状态负载"页签,单击页面右上角 "YAML创建",选择命名空间来进行部署服务。可以通过控制台的方式创建,如 果以YAML的方式部署Exporter,YAML配置示例如下:

```
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
  k8s-app: redis-exporter # 根据业务需要调整,建议加上 Redis 实例的信息,如crs-66e112fp-redis-
exporter
name: redis-exporter # 根据业务需要调整,建议加上 Redis 实例的信息,如crs-66e112fp-redis-
exporter
namespace: default # 选择一个适合的 namespace 来部署 Exporter,如果没有需要新建一个
namespace
spec:
replicas: 1
selector:
 matchLabels:
   k8s-app: redis-exporter # 根据业务需要调整成对应的名称,建议加上 Redis 实例的信息,如
crs-66e112fp-redis-exporter
template:
 metadata:
   labels:
    k8s-app: redis-exporter # 根据业务需要调整成对应的名称,建议加上 Redis 实例的信息,如
crs-66e112fp-redis-exporter
 spec:
```

containers: - env: - name: REDIS\_ADDR value: 120.46.215.4:30379 # 对应 Redis 的 ip:port - name: REDIS\_PASSWORD valueFrom: secretKeyRef: name: redis-secret-test # 对应上一步的 Secret 的名称 key: password # 对应上一步中的 Secret Key image: swr.cn-north-4.myhuaweicloud.com/mall-swarm-demo/redis-exporter:v1.32.0 # 替换为您 上传到 SWR 的镜像地址 imagePullPolicy: IfNotPresent name: redis-exporter ports: - containerPort: 9121 name: metric-port # 这个名称在配置采集任务的时候需要 securityContext: privileged: false terminationMessagePath: /dev/termination-log terminationMessagePolicy: File dnsPolicy: ClusterFirst imagePullSecrets: - name: default-secret restartPolicy: Always schedulerName: default-scheduler securityContext: {} terminationGracePeriodSeconds: 30 apiVersion: v1 kind: Service metadata: name: redis-exporter name-space: default # 与Exporter部署的namespace相同 spec: type: NodePort selector: k8s-app: redis-exporter ports: - protocol: TCP nodePort: 30378 port: 9121 targetPort: 9121

🛄 说明

更多Exporter详细参数介绍请参见 redis\_exporter。

- 3. 验证Redis Exporter是否部署成功。
  - a. 在工作负载列表中"无状态负载"页签下,单击<mark>步骤3.2</mark>创建的无状态工作负载的名称,在实例列表中单击操作列下的"更多 > 日志",可以查看到 Exporter成功启动并暴露访问地址。

#### 图 5-16 查看日志

- b. 验证。有以下三种方法进行验证:
  - 登录集群节点执行如下任意一种命令: curl http://{集群IP}:9121/metrics curl http://{集群任意节点私有IP}:30378/metrics
  - 访问地址: http://{集群任意节点的公网IP}:30378/metrics 如发现未能得到数据,请检查一下部署Redis Exporter时YAML中的 REDIS\_ADDR和REDIS\_PASSWORD是否正确,示例如下:

#### **图 5-17** 访问地址

← → C ▲	30378/metrics
# HELP go_gc_duration_seconds A summary	of the pause duration of garbage collection cycles.
a durotion cocondo/cuontile="0"\ 0	
go_gc_duration_seconds/quantile= 0 ) 0	0
go_gc_duration_seconds{quantile= 0.25 /	
go_gc_duration_seconds(quantile= 0.0 / 0	0
go_gc_duration_seconds (quantile= 0.70 )	0
go_gc_duration_seconds quantife= 1 / 0	
go_gc_duration_seconds_sam 0	
UTIP as associations Number of association	a that supporting original
TYPE as governing gouge	s that currently exist.
a nin go_goroucines gauge	
WEIP so info Information about the Co.	empirement
TYPE so info gauge	
o info{version="gol, 17, 3"} 1	
# HELP go memstats alloc bytes Number of	bytes allocated and still in use.
TYPE go menstats alloc bytes gauge	
ro memstats alloc bytes 962888	
# HELP go memstats alloc bytes total Tot	al number of bytes allocated, even if freed.
# TYPE go memstats alloc bytes total cou	inter
zo memstats alloc bytes total 962888	
# HELP go memstats buck hash svs bytes M	Aumber of bytes used by the profiling bucket hash table.
# TYPE go memstats buck hash sys bytes a	auge
go memstats buck hash sys bytes 4236	· -
# HELP go memstats frees total Total num	ber of frees.
# TYPE go_memstats_frees_total counter	
go_memstats_frees_total 178	
# HELP go_memstats_gc_cpu_fraction The f	raction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge	
go_memstats_gc_cpu_fraction 0	
# HELP go_memstats_gc_sys_bytes Number o	/f bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge	
go_memstats_gc_sys_bytes 4.067e+06	
# HELP go_memstats_heap_alloc_bytes Numb	er of heap bytes allocated and still in use.
¥ TYPE go_memstats_heap_alloc_bytes gaug	,e
go_memstats_heap_alloc_bytes 962888	
# HELP go_memstats_heap_idle_bytes Numbe	or of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge	
go_memstats_heap_idle_bytes 1.769472e+06	j
# HELP go_memstats_heap_inuse_bytes Numb	er of heap bytes that are in use.
<pre># TYPE go_memstats_heap_inuse_bytes gaug</pre>	e
go_memstats_heap_inuse_bytes 1.998848e+0	
W HELF go_memstats_heap_objects Number of	n ailocated objects.
<pre># IIFE go_memstats_neap_objects gauge</pre>	
go_memstats_neap_objects 403/	terber of here better relevant to 05
<pre># nmlr go_memstats_neap_feleased_bytes P # TWDE</pre>	Number of heap bytes released to 03.
<pre>+ IIFE go_memstats_neap_released_bytes g</pre>	auge
go_memstats_neap_released_bytes 1.769472	ie+00

#### 在实例列表中单击操作列下的"更多 > 远程登录",在弹出的控制台中 执行如下命令。

curl http://localhost:9121/metrics

#### 图 5-18 执行命令

reais-exporter	NOGEPOPT		<none></none>	9121:30378/ICP	
user@an1sfqy9ulitku8-machine:~\$	curl http://	:30378/metrics			
<pre># HELP go_gc_duration_seconds A :</pre>	summary of the pause	duration of garbage	collection cycles.		
<pre># TYPE go_gc_duration_seconds su</pre>	mary				
<pre>go_gc_duration_seconds{quantile=</pre>	"0"} 0				
go_gc_duration_seconds{quantile=	"0.25"} 0				
<pre>go_gc_duration_seconds{quantile=</pre>	"0.5"} 0				
go_gc_duration_seconds{quantile=	"0.75"} 0				
<pre>go_gc_duration_seconds{quantile=</pre>	°1"} 0				
<pre>go_gc_duration_seconds_sum 0</pre>					
<pre>go_gc_duration_seconds_count 0</pre>					
# HELP go_goroutines Number of g	proutines that curre	ntly exist.			
<pre># TYPE go_goroutines gauge</pre>					
go_goroutines 8					
<pre># HELP go_info Information about</pre>	the Go environment.				
<pre># TYPE go_info gauge</pre>					
<pre>go_info{version="go1.17.3"} 1</pre>					
<pre># HELP go_memstats_alloc_bytes N</pre>	umber of bytes alloc	ated and still in us	se.		
<pre># TYPE go_memstats_alloc_bytes g</pre>	auge				
go_memstats_alloc_bytes 2.029288	2+06				
<pre># HELP go_memstats_alloc_bytes_t</pre>	otal Total number of	bytes allocated, ev	ven if freed.		
<pre># TYPE go_memstats_alloc_bytes_t</pre>	otal counter				
<pre>go_memstats_alloc_bytes_total 2.</pre>	329288e+06				
<pre># HELP go_memstats_buck_hash_sys</pre>	bytes Number of byt	es used by the profi	iling bucket hash table.		
<pre># TYPE go_memstats_buck_hash_sys</pre>	_bytes gauge				
<pre>go_memstats_buck_hash_sys_bytes </pre>	1236				
<pre># HELP go_memstats_frees_total T</pre>	otal number of frees				
<pre># TYPE go_memstats_frees_total c</pre>	ounter				
<pre>go_memstats_frees_total 304</pre>					
<pre># HELP go_memstats_gc_cpu_fracti</pre>	on The fraction of t	his program's availa	able CPU time used by the GC s:	ince the program started.	
<pre># TYPE go_memstats_gc_cpu_fraction</pre>	on gauge				
<pre>go_memstats_gc_cpu_fraction 0</pre>					
<pre># HELP go_memstats_gc_sys_bytes  </pre>	Number of bytes used	for garbage collect	tion system metadata.		
<pre># TYPE go_memstats_gc_sys_bytes</pre>	gauge				
<pre>go_memstats_gc_sys_bytes 4.09784</pre>	2+06				
<pre># HELP go_memstats_heap_alloc_by</pre>	tes Number of heap b	ytes allocated and s	still in use.		
<pre># TYPE go_memstats_heap_alloc_by</pre>	tes gauge				

----结束

#### 添加采集任务

通过<mark>新增PodMonitor</mark>方式为应用配置可观测监控Prometheus版的采集规则,监控部 署在CCE集群内的应用的业务数据。

#### 🗀 说明

如下指标采集的周期是30秒,所以等待大概30秒后才能在AOM的界面上查看到上报的指标。

apiVersion: monitoring.coreos.com/v1 kind: PodMonitor metadata: name: redis-exporter namespace: default spec: . namespaceSelector: #选择要监控 Exporter Pod 所在的namespace matchNames: - default # exporter所在的命名空间 podMetricsEndpoints: - interval: 30s # 设置指标采集周期 path: /metrics # 填写 Prometheus Exporter 对应的 path 的值, 默认/metrics port: metric-port# 填写 Prometheus Exporter 对应的 YAML 的 ports 的 name selector: # 填写要监控 Exporter Pod 的 Label 标签,以定位目标 Exporter matchLabels: k8s-app: redis-exporter

#### 验证指标上报到 AOM

- 步骤1 登录AOM 2.0控制台。
- 步骤2 在左侧菜单栏中选择 "Prometheus监控 > 实例列表"。
- 步骤3 单击接入了该CCE集群的"prometheus for CCE"实例名称,进入实例详情页面。
- 步骤4 在"服务发现"页面的"指标"页签下,选择集群。
- 步骤5 在搜索框输入redis,能够搜索出redis开头的指标,即可证明指标成功接入AOM。

#### ----结束

#### 在 AOM 上配置仪表盘和告警

通过仪表盘功能可视化监控CCE集群数据,通过告警规则功能,在集群发生故障时能够 及时发现并预警。

- 配置仪表盘图表
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"仪表盘",单击"创建仪表盘"新建一个仪表盘,详 情可参见创建仪表盘。
  - c. 在仪表盘页面选择实例类型为 "Prometheus for CCE"的实例并单击 "添加 图表",详情请参见<mark>添加图表至仪表盘</mark>。
- 配置告警
  - a. 登录AOM 2.0控制台。
  - b. 在左侧菜单栏中选择"告警管理 > 告警规则"。
  - c. 单击"创建告警规则"配置告警,详情请参见创建指标告警规则。

# 5.8 其他 Exporter 接入

#### 操作场景

Prometheus监控服务目前已经提供了常用中间件exporter接入操作指导,由于AOM兼 容原生Prometheus,所以您也可以安装社区其他的Exporter。

#### 操作方式

如果您所使用的基础组件还没有提供相应的集成方式,可以参考如下方式进行集成, 以及自定义监控大屏来满足相应的监控需求。

- 1. 开源社区Exporter列表。
- 2. 在<mark>容器场景自建中间件接入</mark>,已经提供部分常用中间件exporter接入操作指导,可以根据操作安装其他的Exporter。