

1. 部署

- 1.1. 二进制方式部署
 - 1.1.1. 下载
 - 1.1.2. 部署
 - 1.1.3. MySQL 服务的配置
- 1.2 docker-compose 方式
- 1.3 配置 Prometheus
- 1.4 测试
- 1.5 mysqld_exporter 命令行运行参数
 - 1.5.1. 配置格式
 - 1.5.2. 运行参数详解
 - 1.5.3. 监控不同的集群

2. Dashboard

- 2.1. 在线导入
- 2.2. 修改主题为浅色
- 2.3. 离线下载自动导入

3. 告警规则实战

- 3.1. MySQL 宕机
- 3.2. 实例连接数过多
 - 3.2.1. 规则拆解分析
 - 3.2.2. 完整规则
- 3.3. MySQL高线程运行
 - 3.3.1 规则拆解分析
 - 3.3.2. 完整规则
- 3.4. MySQL 从服务器 IO 线程没有运行
 - 3.4.1 规则拆解分析
 - 3.4.2. 完整规则
- 3.5. MySQL 从服务器 SQL 线程没有运行
 - 3.5.1 规则拆解分析
 - 3.5.2. 完整规则
- 3.6. MySQL复制滞后
 - 3.6.1 规则拆解分析
 - 3.6.2. 完整规则
- 3.7. 慢查询
 - 3.7.1 规则拆解分析
 - 3.7.2. 完整规则
- 3.8. innodb 日志等待
 - 3.8.1 规则拆解分析
 - 3.8.2. 完整规则
- 3.9. MySQL 实例 1 分钟内重启过
 - 3.9.1 规则拆解分析
 - 3.9.2. 完整规则
- 3.10. MySQL 组复制成员丢失
 - 3.10.1 规则拆解分析
 - 3.10.2. 完整规则

1. 部署

1.1. 二进制方式部署

1.1.1. 下载

```
curl -o mysqld_exporter-0.15.1.linux-amd64.tar.gz -L
https://github.com/prometheus/mysqld_exporter/releases/download/v0.15.1/mysqld_e
xporter-0.15.1.linux-amd64.tar.gz
```

1.1.2. 部署

解压二进制部署包到指定位置

```
tar -xf mysqld_exporter-0.15.1.linux-amd64.tar.gz mysqld_exporter-0.15.1.linux-
amd64/mysqld_exporter
mv mysqld_exporter-0.15.1.linux-amd64 /usr/local/mysqld_exporter
```

配置服务管理文件

/etc/systemd/system/mysqld-exporter.service

```
[Unit]
Description=The mysqld exporter server
After=network-online.target remote-fs.target nss-lookup.target
Wants=network-online.target

[Service]
ExecStart=/usr/local/mysqld_exporter/mysqld_exporter --config.my-
cnf=/root/.my.cnf

KillSignal=SIGQUIT

Restart=always

RestartPreventExitStatus=1 6 SIGABRT

TimeoutStopSec=5
KillMode=process
PrivateTmp=true
LimitNOFILE=1048576
LimitNPROC=1048576

[Install]
WantedBy=multi-user.target
```

1.1.3. MySQL 服务的配置

配置认证文件

创建 /root/.my.cnf 文件，参考如下内容填充文件

这个文件路径可以放在其他地方，启动的时候使用 --config.my-cnf=<path> 指定即可。

.my.cnf

```
# client 是 mysqld_exporter 程序启动的时候默认连接的,如果不设置 host 的值,将会使用
127.0.0.1 进行连接
[client]
host = MySQL服务的IP
user = exporter
```

```
password = Exporter+110
# 默认端口会使用 3306
port = 3306
# 被监控的其他 MySQL 服务实例的认证信息，在这里分别配置上，建议授权专门用于监控的相同的授权用户
[client.mysql2]
host = MySQL服务的IP
user = exporter
password = Exporter+110
[client.mysql3]
host = MySQL服务的IP
user = monitor
password = Exporter+110
```

如果使用的是 **docker-compose** 部署的，`.my.cnf` 文件放到 `docker-compose.yml` 目录下，并挂载到容器中就好。

MySQL授权

```
CREATE USER 'exporter'@'mysqld-exporter部署所在服务器的IP' IDENTIFIED BY
'Exporter+110';
GRANT PROCESS, REPLICATION CLIENT, SELECT ON *.* TO 'exporter'@'mysqld-exporter
部署所在服务器的IP';
```

注意：

如果部署的 **mysqld_exporter** 和 MySQLserver 不再同一个网段，授权的IP可能需要是 MySQL server 所在网段的网关IP，特别是MySQLserver使用 docker 容器访问实现时。

启动服务

```
systemctl enable --now mysqld-exporter.service
```

1.2 docker-compose 方式

```
docker pull prom/mysqld-exporter:v0.15.1
```

创建项目目录 `mysqld-exporter`

```
mkdir mysqld-exporter
```

将 `.my.cnf` 文件也放到 `mysql-exporter` 目录下

在项目目录 `mysql-exporter` 下创建 `docker-compose.yml` 文件，内容如下：

```

version: "3.9"
services:
  mysql_exporter:
    image: prom/mysql-d-exporter:v0.15.1
    command: --collect.perf_schema.replication_group_members
    restart: always
    volumes:
      - "/etc/localtime:/etc/localtime"
      - ".my.cnf:/my.cnf"
    network_mode: "host"
    expose:
      - "9104"

```

启动服务

```
docker-compose up -d
```

1.3 配置 Prometheus

将如下内容添加到 `prometheus.yml` 中

```

- job_name: mysql
  metrics_path: /probe
  params:
    auth_module:
      # 这里需要对应 .my.cnf 中的配置
      # 当获取监控指标的时候，会依次使用这里的认证信息尝试连接认证
      # 直到有一个认证成功，如果都没认证成功将会报错。
      - client # 需要和 .my.cnf 文件中配置的一致
      - client.mysql2 # 需要和 .my.cnf 文件中配置的一致
      - client.mysql3 # 需要和 .my.cnf 文件中配置的一致
  static_configs:
    - targets:
      # 被监控的 mysql 服务器IP和端口
      - "168.12.15.195:3306"
      - "168.12.15.197:3306"
      labels:
        location: "南京"
        project: "项目二"
        app: mysqld
    - targets:
      - "192.166.0.95:3307"
      - "192.166.0.10:3308"
      labels:
        location: "北京"
        project: "项目一"
        app: mysqld
  relabel_configs:
    - source_labels: [__address__]
      target_label: __param_target
    - source_labels: [__param_target]
      target_label: instance
    - target_label: __address__
      # mysqld-exporter的IP和端口

```

```
replacement: localhost:9104
```

更新配置

```
curl -X POST --user 'shark:123456' localhost:9090/-/reload
```

1.4 测试

启动服务后，执行如下 linux 命令测试。

```
curl http://localhost:9104/probe?target=168.12.15.195:3306  
curl http://localhost:9104/probe?target=192.166.0.10:3308
```

1.5 mysqld_exporter 命令行运行参数

运行参数是配置在启动命令后面的参数。

配置不同的参数，会影响最终返回结果中指标的多少。

1.5.1. 配置格式

-- 参数名称

例如：

```
--collect.auto_increment.columns  
--no-collect.auto_increment.columns
```

用法：

```
/mysqld_exporter --collect.auto_increment.columns
```

1.5.2. 运行参数详解

获取最新的运行参数，和帮助信息，可以通过命令 `mysqld_exporter --help` 获取。

命令行参数表：

	Name	MySQL Version	Description
collect.auto_increment.columns	5.1	从information_schema中收集auto_increment列和最大值。	
collect.binlog_size	5.1	收集所有已注册binlog文件的当前大小	
collect.engine_innodb_status	5.1	从SHOW engine innodb status收集。	
collect.engine_tokudb_status	5.6	从SHOW engine tokudb status收集。	
collect.global_status	5.1	从SHOW global status收集（默认情况下已启用）	
collect.global_variables	5.1	从SHOW global variables收集（默认情况下启用）	
collect.heartbeat	5.1	启用collect.heartbeat后，mysqld_exporter将抓取由心跳机制测量的复制延迟。。	
collect.heartbeat.database	5.1	从何处收集心跳数据的数据库。（默认值：heartbeat）	
collect.heartbeat.table	5.1	从何处收集心跳数据的表。（默认值：heartbeat）	
collect.heartbeat.utc	5.1	使用utc表示当前服务器的时间戳（pt-heartbeat 用~utc）。（默认值：false）	
collect.info_schema.clientstats	5.5	如果在userstat=1的情况下运行，则设置为true以收集客户端	
collect.info_schema.innodb_metrics	5.6	从information_schema.innodb_metrics收集度量。	
collect.info_schema.innodb_tablespace	5.7	从information_schema.innodb_sys_tablespace收集度量。	
collect.info_schema.innodb_cmp	5.5	从information_schema.innodb_cmp收集innodb压缩表度量。	
collect.info_schema.innodb_compmem	5.5	从information_schema.innodb_compmem收集innodb缓冲池压缩度量。	
collect.info_schema.processlist	5.1	从information_schema.rocesslist中收集线程状态计数。	
collect.info_schema.processlist.min_time	5.1	线程在每个状态下的最短计数时间。（默认值：0）	
collect.info_schema.query_response_time	5.5	如果query_responce_time_stats为ON，则收集查询响应时间分布。	
collect.info_schema.replica_host	5.6	从information_schema.replica_host_status收集度量。	
collect.info_schema.tables	5.1	从information_schema.table中收集度量。	
collect.info_schema.tables.databases	5.1	要为其收集表统计信息的数据库列表，或为所有数据库收集“*”。	
collect.info_schema.tablestats	5.1	如果在userstat=1的情况下运行，请设置为true以收集表统计信息。	
collect.info_schema.schemastats	5.1	如果在userstat=1的情况下运行，请设置为true以收集架构统计信息	
collect.info_schema.userstats	5.1	如果在userstat=1的情况下运行，请设置为true以收集用户统计信息。	
collect.mysql.user	5.5	从mysql.user表中收集数据	
collect.perf_schema.eventsstatements	5.6	从performance_schema.vents_statements_summary_by_digest中收集度量。	
collect.perf_schema.eventsstatements.digest_text_limit	5.6	规范化语句文本的最大长度。（默认值：120）	
collect.perf_schema.eventsstatements.limit	5.6	按响应时间限制事件语句摘要的数量。（默认值：2）	50)
collect.perf_schema.eventsstatements.timelimit	5.6	限制“last_seen”事件语句的存在时间（以秒为单位）。（默认值：86400）	
collect.perf_schema.eventsstatementssum	5.7	从performance_schema.vents_statements_summary_by_digest sumed中收集度量。	
collect.perf_schema.eventswaits	5.5	从performance_schema.events_waits_summary_global_by_event_name收集指标。	
collect.perf_schema.file_events	5.6	从performance_schema.files_summary_by_event_name收集度量。	
collect.perf_schema.file_instances	5.5	从performance_schema.file_summary_by_instance收集度量。	
collect.perf_schema.file_instances.remove_prefix	5.5	删除performance_schema.files_summary_by_instance中的路径前缀。	
collect.perf_schema.indexiowaits	5.6	从performance_schema.table_io_waits_summary_by_index_usage收集指标。	
collect.perf_schema.memory_events	5.7	从performance_schema.memory_summary_global_by_event_name收集度量。	
collect.perf_schema.memory_events.remove_prefix	5.7	删除performance_schema.memory_summary_global_by_event_name中的仪器前缀。	
collect.perf_schema.tableiowaits	5.6	从performance_schema.table_io_waits_summary_by_table收集度量。	
collect.perf_schema.tablelocks	5.6	从performance_schema.table_lock_waits_summary_by_table收集度量。	
collect.perf_schema.replication_group_members	5.7	从 performance_schema.replication_group_members 收集度量，针对组复制集群。	
collect.perf_schema.replication_group_member_stats	5.7	从 performance_schemareplication_group_member_stats 收集度量，针对组复制集群。	
collect.perf_schema.replication_applier_status_by_worker	5.7	从 performance_schema.replication_applier_status_by_worker 收集指标，针对组复制集群。	
collect.slave_status	5.1	从 SHOW SLAVE STATUS 收集（默认启用）	
collect.slave_hosts	5.1	从 SHOW SLAVE HOSTS 收集	
collect.sys.user_summary	5.7	从sys.x\$user_summary收集度量（默认情况下禁用）。	

1.5.3. 监控不同的集群

可以针对不同的MySQL集群模式，配置不同的运行参数。

- 监控主从复制的集群可以添加如下运行参数。
 - `--collect.slave_hosts`

- 监控组复制或者 InnoDB Cluster 集群可以添加如下运行参数
 - `--collect.perf_schema.replication_group_members`
对应的指标名称: `mysql_perf_schema_replication_group_member_info`, 可以获取到组复制成员信息。

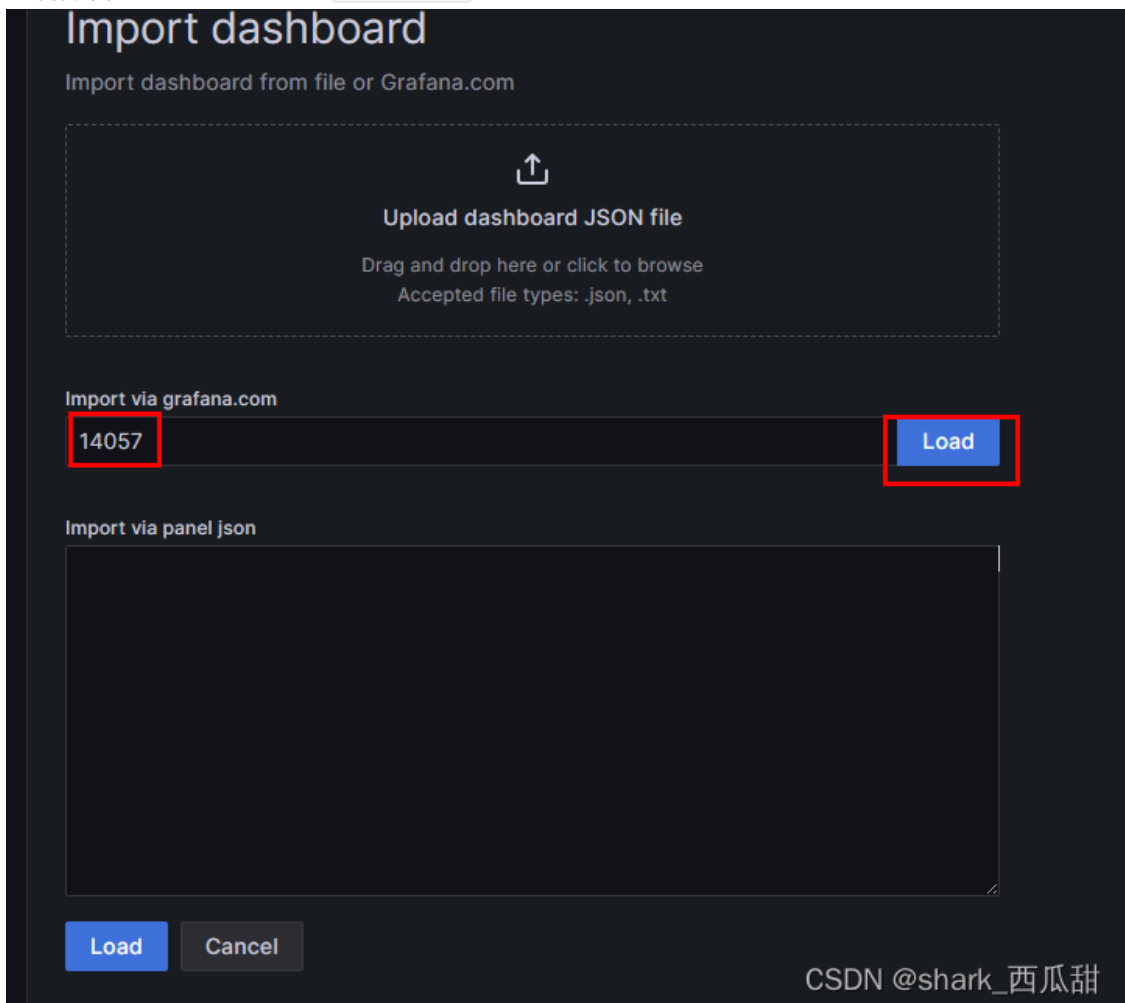
2. Dashboard

2.1. 在线导入

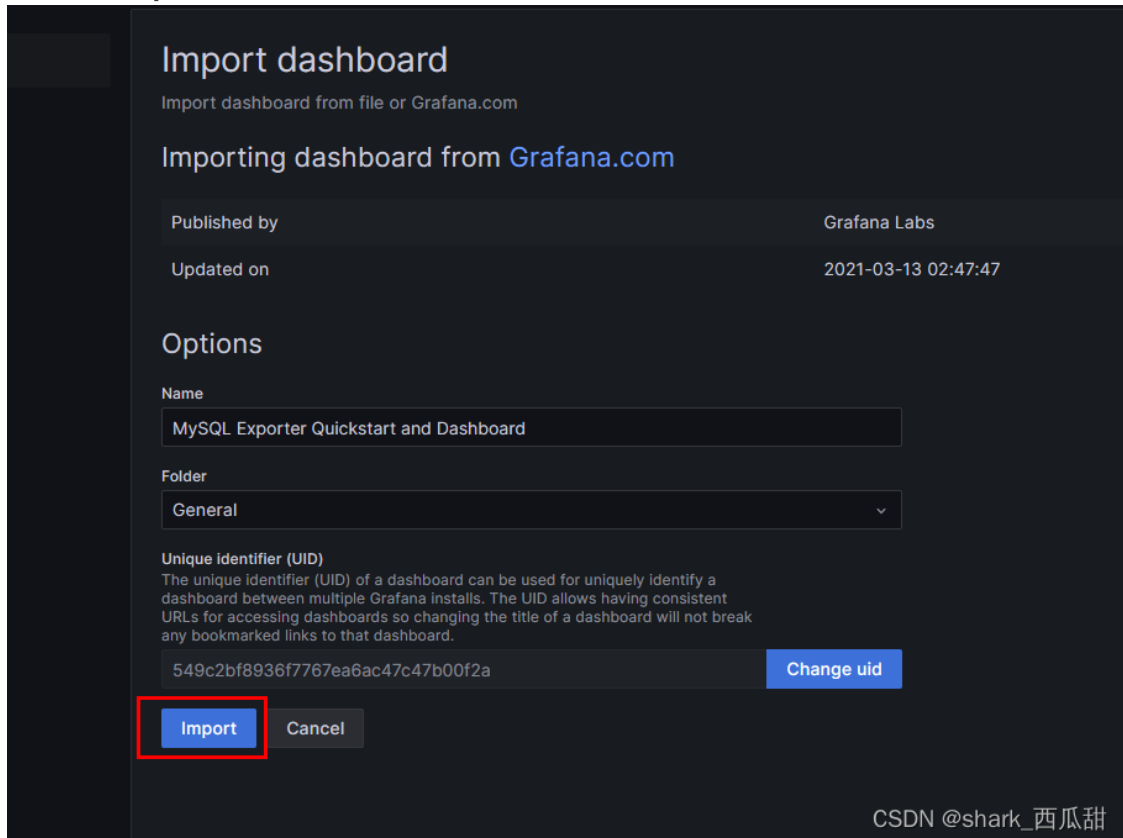
1. 首先, 找到仪表板的新建--》导入页面



2. 之后再填入 Dashboard ID ID: 14057



最后点击 **Import**



The image shows the 'Import dashboard' interface in Grafana. It has a dark theme. At the top, it says 'Import dashboard from file or Grafana.com'. Below that, it says 'Importing dashboard from Grafana.com'. There are two rows of information: 'Published by' (Grafana Labs) and 'Updated on' (2021-03-13 02:47:47). Under the 'Options' section, there is a 'Name' field with the text 'MySQL Exporter Quickstart and Dashboard', a 'Folder' dropdown menu set to 'General', and a 'Unique identifier (UID)' field with the text '549c2bf8936f7767ea6ac47c47b00f2a'. There is a 'Change uid' button next to the UID field. At the bottom, there are two buttons: 'Import' (highlighted with a red box) and 'Cancel'.

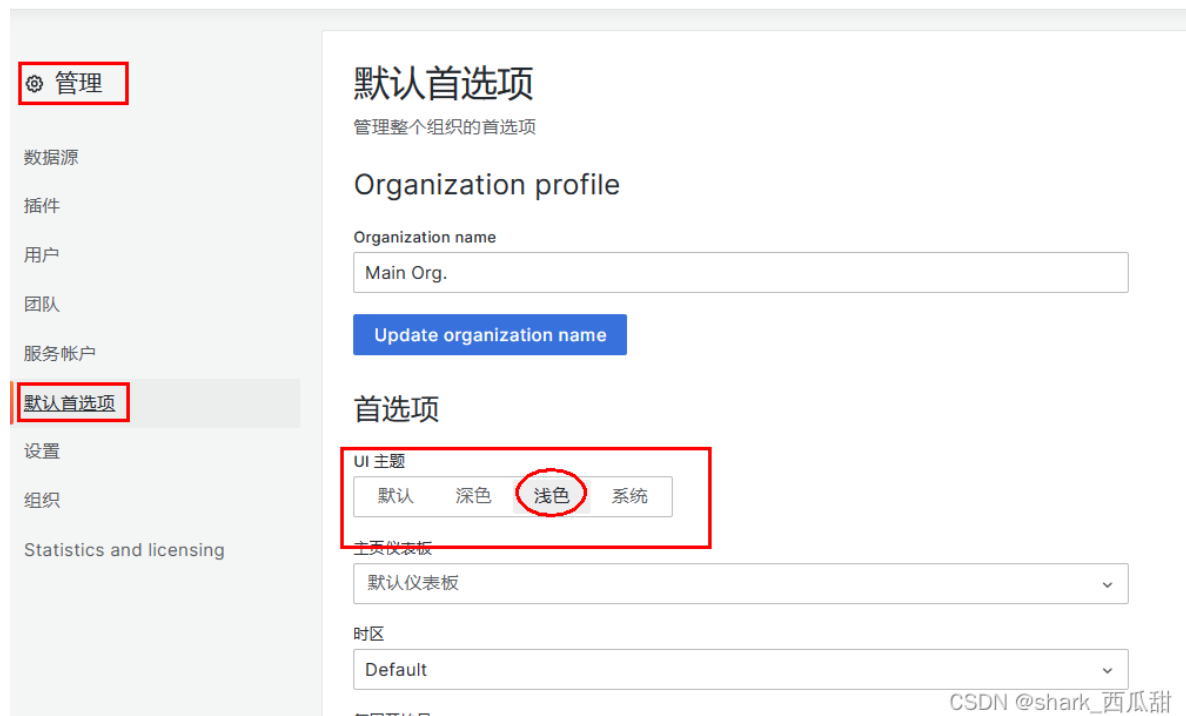
3. 配置数据源

导入成功后，有可能需要选择一下正确的数据源。

2.2. 修改主题为浅色

为了截图显示更好的效果，我修改了页面主题为浅色。

≡ 首页 > 管理 > 默认首选项



The image shows the 'Default Preferences' (默认首选项) settings page in Grafana. The left sidebar has a '管理' (Management) menu item highlighted with a red box. The main content area is titled '默认首选项' (Default Preferences) and 'Organization profile'. It has a form for 'Organization name' with the text 'Main Org.' and an 'Update organization name' button. Below that, there is a '首选项' (Preferences) section. In this section, the 'UI 主题' (UI Theme) is set to '浅色' (Light), which is highlighted with a red circle. Other options in the 'UI 主题' section are '默认' (Default), '深色' (Dark), and '系统' (System). Below the 'UI 主题' section, there are dropdown menus for '主仪表盘' (Main Dashboard) set to '默认仪表盘' (Default Dashboard) and '时区' (Timezone) set to 'Default'.

2.3. 离线下载自动导入

MySQL Exporter Quickstart and Dashboard

添加Last 1 hour

datasourcePrometheusjob全部instance全部

UptimeCurrent QPSInnoDB Buffer Pool

Variables

ConnectionsSettings

GeneralAnnotationsVariablesLinksVersionsPermissionsJSON Model

Variables

Variables can make your dashboard more dynamic and act as global filters.

Variable	Definition	
datasource	prometheus	
job	label_values(mysql_up, job)	
instance	label_values(mysql_up, instance)	

Show dependenciesNew variable

Renamed or missing variables

Settings

GeneralAnnotationsVariablesLinksVersionsPermissionsJSON Model

Description

Descriptive text

Show on dashboardLabel and valueValueNothing

Data source options

TypePrometheus

Instance name filter

Regex filter for which data source instances to choose from in the variable value list. Leave empty for all.

Example: /¹^prod/

/Prom¹e.*/

这里写上正则表达式

Selection options

Multi-value

Includes multiple values to be selected at the same time

Include All option

Enables an option to include all variables

Preview of values²

Prometheus

这里会即时显示正则匹配的结果

DeleteRun queryApply³

CSDN @shark_西瓜甜

er Quickstart and Dashboard > Variables

CloseSave asSave dashboard

Variables

Variables can make your dashboard more dynamic and act as global filters.

Variable	Definition	
datasource	prometheus	✓🔗📄🗑️⋮
job	label_values(mysql_up, job)	✓🔗📄🗑️⋮
instance	label_values(mysql_up, instance)	✓🔗📄🗑️⋮

Show dependencies+ New variable

CSDN @shark_西瓜甜

global filter

Save dashboard

MySQL Exporter Quickstart and Dashboard

DetailsChanges 7

☒ Save current variable values as dashboard default

添加了数据源的正则表达式，以便匹配正确的数据源。|

CancelSave

CSDN @shark_西瓜甜

之后就可以重新点击 这个 Dashboard，以便重新加载。

首页 > 仪表盘 > MySQL Exporter Quickstart and Dashboard > General

CloseSave asSave dashboard

SettingsGeneralAnnotationsVariablesLinksVersionsPermissions

General

NameMySQL Exporter Quickstart and Dashboard

DescriptionA quickstart to setup the Prometheus MySQL Exporter with preconfigured dashboards, i

TagsNew tag (enter key to add)Add

Folder

MySQL Exporter Quickstart and Dashboard ☆🔗

添加📄🔗🕒 Last 1 hour🔍🔄 10s

datasourcePrometheusjob全部instance全部

Uptime

datasource	job	instance	Uptime
prometheus	mysql	mysql-01	43.7min
prometheus	mysql	mysql-02	43.7min
prometheus	mysql	mysql-03	43.7min

Current QPS

datasource	job	instance	Current QPS
prometheus	mysql	mysql-01	13.46
prometheus	mysql	mysql-02	1.06
prometheus	mysql	mysql-03	1.07

InnoDB Buffer Pool

datasource	job	instance	InnoDB Buffer Pool
prometheus	mysql	mysql-01	91.9M
prometheus	mysql	mysql-02	91.9M
prometheus	mysql	mysql-03	91.9M

Connections

MySQL Connections

	min	max	avg
Max Connections	6 K	6 K	6 K
Max Used Connections	11	11	11

MySQL Client Thread Activity

	min	max	avg
Peak Threads Running	6.00	7.00	6.08
Avg Threads Running	6.00	6.25	6.02

CSDN @shark_西瓜甜

3. 告警规则实战

3.1. MySQL 宕机

```
- alert: MysqlDown
  expr: mysql_up == 0
  for: 0m
  labels:
    severity: critical
  annotations:
    summary: MySQL ({{ $labels.instance }}) is down
    description: "MySQL 挂了: {{ $labels.instance }}\n 当前值: {{ $value }}\n
  标签: {{ $labels }}"
```

3.2. 实例连接数过多

3.2.1. 规则拆解分析

MySQL实例上使用了超过80%的MySQL连接。

MySQL 实例实际连接率=当前连接数 / 全局变量最大连接数 *100

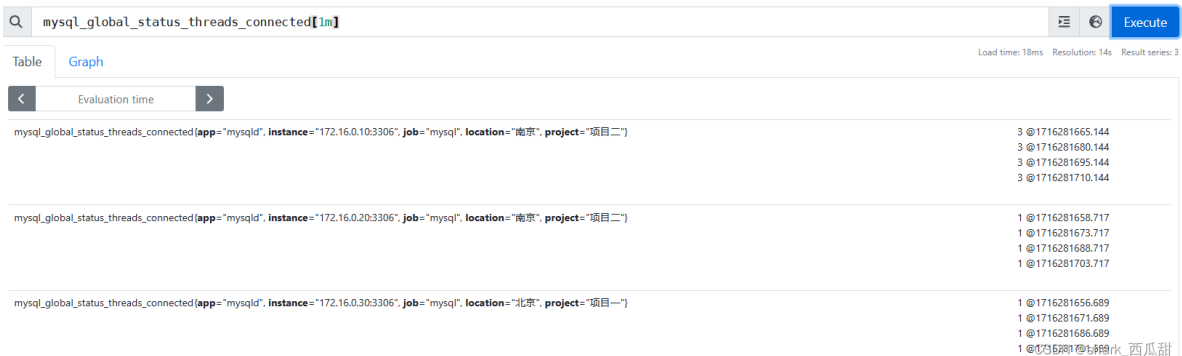
当前连接数通过指标 `mysql_global_status_threads_connected` 获取。



mysql_global_status_threads_connected	
mysql_global_status_threads_connected{app="mysqld", instance="172.16.0.10:3306", job="mysql", location="南京", project="项目二"}	3
mysql_global_status_threads_connected{app="mysqld", instance="172.16.0.20:3306", job="mysql", location="南京", project="项目二"}	1
mysql_global_status_threads_connected{app="mysqld", instance="172.16.0.30:3306", job="mysql", location="北京", project="项目一"}	1

瞬时的连接过多，不足以影响性能指标，所以我们通常统计 1 分钟内的。

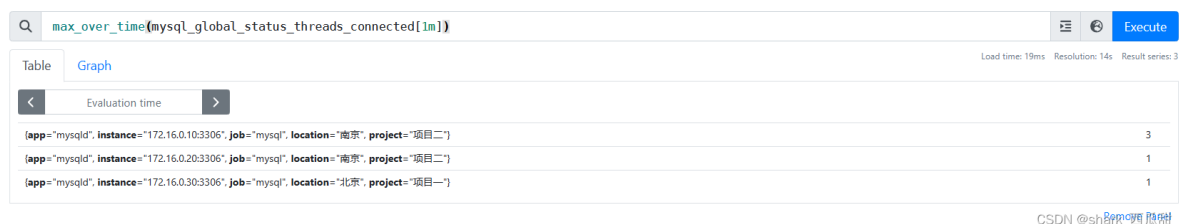
表达式: `mysql_global_status_threads_connected[1m]`



mysql_global_status_threads_connected	
mysql_global_status_threads_connected{app="mysqld", instance="172.16.0.10:3306", job="mysql", location="南京", project="项目二"}	3 @1716281665.144 3 @1716281690.144 3 @1716281695.144 3 @1716281710.144
mysql_global_status_threads_connected{app="mysqld", instance="172.16.0.20:3306", job="mysql", location="南京", project="项目二"}	1 @1716281658.717 1 @1716281673.717 1 @1716281688.717 1 @1716281703.717
mysql_global_status_threads_connected{app="mysqld", instance="172.16.0.30:3306", job="mysql", location="北京", project="项目一"}	1 @1716281656.689 1 @1716281671.689 1 @1716281686.689 1 @1716281701.689

而 1 分钟内有可能连接数再不断变化，因此我们通过函数 `max_over_time` 获取到 1 分钟内最大的那个数据。

表达式: `max_over_time(mysql_global_status_threads_connected[1m])`



max_over_time(mysql_global_status_threads_connected[1m])	
{app="mysqld", instance="172.16.0.10:3306", job="mysql", location="南京", project="项目二"}	3
{app="mysqld", instance="172.16.0.20:3306", job="mysql", location="南京", project="项目二"}	1
{app="mysqld", instance="172.16.0.30:3306", job="mysql", location="北京", project="项目一"}	1

注意：

函数 `max` 是计算多个指标实例中值最大的。

函数 `max_over_time` 是计算每个指标实例的多个时间戳对应的值中哪个值最大。

全局最大连接数通过指标 `mysql_global_variables_max_connections` 获取。

mysql_global_variables_max_connections	2000
mysql_global_variables_max_connections	2000
mysql_global_variables_max_connections	2000

CSDN @shark 西瓜甜

因此计算 1 分钟内连接率的表达式是:

`max_over_time(mysql_global_status_threads_connected[1m]) /`
`mysql_global_variables_max_connections * 100`

max_over_time(mysql_global_status_threads_connected[1m]) / mysql_global_variables_max_connections * 100	0.15
max_over_time(mysql_global_status_threads_connected[1m]) / mysql_global_variables_max_connections * 100	0.05
max_over_time(mysql_global_status_threads_connected[1m]) / mysql_global_variables_max_connections * 100	0.05

CSDN @shark 西瓜甜

而我们通常不希望连接率大于 80%，最终的表达式是：

`max_over_time(mysql_global_status_threads_connected[1m]) /`
`mysql_global_variables_max_connections * 100 > 80`

3.2.2. 完整规则

```
group:
- name: MySQL Rules
  interval: 2s
  rules:
- alert: MysqlTooManyConnections(>80%)
  expr: max_over_time(mysql_global_status_threads_connected[1m]) /
mysql_global_variables_max_connections * 100 > 80
  for: 2m
  labels:
    severity: warning
  annotations:
    summary: MySQL too many connections (> 80%) (instance {{ $labels.instance }})
    description: "{{ $labels.proj }}MySQL 的连接数超过了允许的 80% {{ $labels.instance }}"
    value: {{ $value }}
```

3.3. MySQL高线程运行

3.3.1 规则拆解分析

超过60%的MySQL连接处于活动状态。

在所有的连接中，不是所有的连接都会正在发送请求，操作数据。有的连接操作数据完成了，但是客户端又没有即时断开，也没有达到连接空闲超时时间，所以有的连接会处于空闲状态。

这个和上个连接率表达式中使用的语法一样，区别是获取处于活动状态的连接数据使用指标：

`mysql_global_status_threads_running`

Q	mysql_global_status_threads_running	Execute
Table	Graph	Load time: 21ms Resolution: 14s Result series: 1
Evaluation time		
mysql_global_status_threads_running(app="mysql", instance="172.16.0.103306", job="mysql", location="南京", project="项目二")		
mysql_global_status_threads_running(app="mysql", instance="172.16.0.203306", job="mysql", location="南京", project="项目二")		
mysql_global_status_threads_running(app="mysql", instance="172.16.0.303306", job="mysql", location="北京", project="项目一")		

通常这个活动的连接率希望不大于 60%，因此最终的表达式是：

`max_over_time(mysql_global_status_threads_running[1m]) /
mysql_global_variables_max_connections * 100 > 60`

3.3.2. 完整规则

```
- alert: MysqlHighThreadsRunning
  expr: max_over_time(mysql_global_status_threads_running[1m]) /  
mysql_global_variables_max_connections * 100 > 60
  for: 2m
  labels:
    severity: warning
  annotations:
    summary: MySQL high threads running (instance {{ $labels.instance }})
    description: "超过60%的MySQL连接在 {{ $labels.instance }} 上处于活动状态\n 当前值: {{ $value }}\n 标签: {{ $labels }}"
```

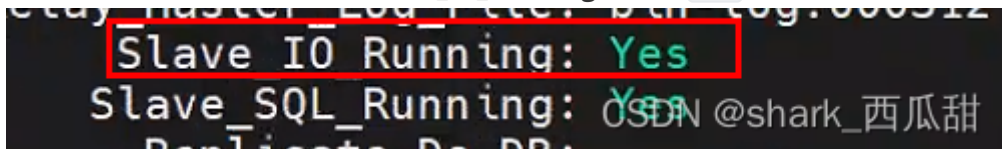
3.4. MySQL 从服务器 IO 线程没有运行

3.4.1 规则拆解分析

条件：被监控的集群是主从模式，而非组复制后者 InnoDB Cluster

确认 IO 线程处于运行状态需要满足两个条件：

- show slave status; 命令返回中 **Slave_IO_Running** 的值是 **Yes**

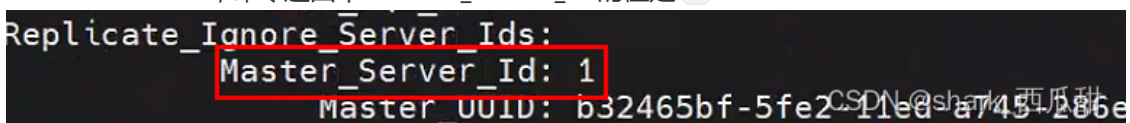


```
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
```

对应的指标 `mysql_slave_status_slave_io_running`，其值应该是 **1**

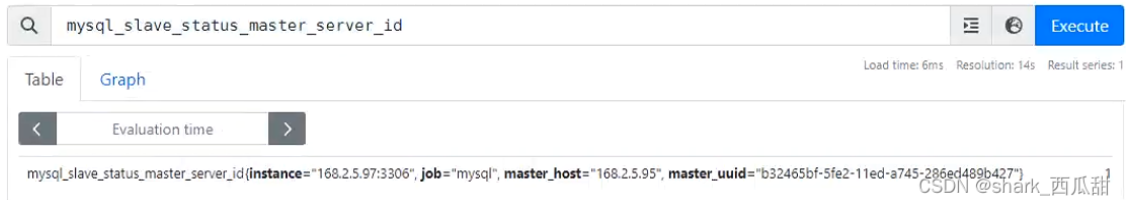
Q	mysql_slave_status_slave_io_running	Execute
Table	Graph	Load time: 3ms Resolution: 14s Result series: 1
Evaluation time		
mysql_slave_status_slave_io_running(instance="168xxx", job="mysql", master_host="168xxx", master_uuid="b32465bf-5fe2-11ed-a745-286ed489b427")		

- show slave status; 命令返回中 **Master_Server_Id** 的值是 **1**



```
Replicate_Ignore_Server_Ids:
Master_Server_Id: 1
Master_UUID: b32465bf-5fe2-11ed-a745-286e
```

对应的指标 `mysql_slave_status_master_server_id`, 其值应该大于 0



主从复制集群中可以会有多个从服务, 因此需要用逻辑运算符 `and` 进行逻辑运算, 并且使用 `ON (instance)` 子句, 指定使用标签 `instance` 进行计算, 来从两个结果集中找到分别以服务器为对应关系的值。

表达式为: `(mysql_slave_status_slave_io_running and on(instance) mysql_slave_status_master_server_id > 0)`

获取的值需要是1, 如果等于 0, 则说明不正常, 需要告警。

最终表达式为: `(mysql_slave_status_slave_io_running and ON (instance) mysql_slave_status_master_server_id > 0) == 0`

3.4.2. 完整规则

```
- alert: MysqlSlaveIoThreadNotRunning
  expr: ( mysql_slave_status_slave_io_running and ON (instance)
mysql_slave_status_master_server_id > 0 ) == 0
  for: 0m
  labels:
    severity: critical
  annotations:
    summary: MySQL slave IO thread not running (instance {{ $labels.instance
  }})
    description: "MySQL slave IO线程未在{{ $labels.instance }} 上运行 \n 当前值:
  {{ $value }}\n 标签: {{ $labels }}"
```

3.5. MySQL 从服务器 SQL 线程没有运行

3.5.1 规则拆解分析

从服务器的 SQL 线程的个规则和上面的 IO 线程的获取原理和语法一样。

3.5.2. 完整规则

```
- alert: MysqlSlaveSqlThreadNotRunning
  expr: ( mysql_slave_status_slave_sql_running and ON (instance)
mysql_slave_status_master_server_id > 0 ) == 0
  for: 0m
  labels:
    severity: critical
  annotations:
    summary: MySQL slave SQL thread not running (instance {{ $labels.instance
  }})
    description: "MySQL {{ $labels.instance }} 的 slave SQL 线程没有运行.\n 当
  前值: {{ $value }}\n 标签: {{ $labels }}"
```

3.6. MySQL复制滞后

3.6.1 规则拆解分析

就是主节点的二进制事务太多的时候，从节点复制的过慢；

或者当我们从一个之前备份的主节点的数据导入到某个从节点时候，也会出现这样的情况，因为此时从节点是从导入数据的那个时候的二进制位置开始复制的，但是此时主节点的实际二进制位置要新。

这个指标 `mysql_slave_status_seconds_behind_master` 是执行命令 `show slave status\G` 返回结果中的 `Seconds_Behind_Master` 的值；

而指标 `mysql_slave_status_sql_delay` 是 `SQL_Delay` 的值。

需要二者的值相减，获取的结果如果大于 30 秒，就认为不正常。

```
mysql> show slave status\G
***** 1. row *****
      Slave_IO_State: Waiting for source to send event
        Master_Host: db1
        Master_User: repl
        Master_Port: 3306
        Connect_Retry: 60
        Master_Log_File: bin-log.000008
        Read_Master_Log_Pos: 134071062
        Relay_Log_File: mysql-relay-bin-db1.000002
        Relay_Log_Pos: 123516
        Relay_Master_Log_File: bin-log.000008
        Slave_IO_Running: Yes
        Slave_SQL_Running: Yes
        Replicate_Do_DB:
        Replicate_Ignore_DB: mysql
        Replicate_Do_Table:
        Replicate_Ignore_Table:
        Replicate_Wild_Do_Table:
        Replicate_Wild_Ignore_Table:
          Last_Errno: 0
          Last_Error:
          Skip_Counter: 0
        Exec_Master_Log_Pos: 134066836
        Relay_Log_Space: 127948
        Until_Condition: None
        Until_Log_File:
        Until_Log_Pos: 0
        Master_SSL_Allowed: No
        Master_SSL_CA_File:
        Master_SSL_CA_Path:
        Master_SSL_Cert:
        Master_SSL_Cipher:
        Master_SSL_Key:
      Seconds_Behind_Master: 244
Master_SSL_Verify_Server_Cert: No
      Last_IO_Errno: 0
      Last_IO_Error:
      Last_SQL_Errno: 0
      Last_SQL_Error:
    Replicate_Ignore_Server_Ids:
      Master_Server_Id: 161
        Master_UUID: f6f70ce3-f453-11ed-ae15-d8bbc1b968ef
        Master_Info_File: mysql.slave_master_info
        SQL_Delay: 0
        SQL_Remaining_Delay: NULL
    Slave_SQL_Running_State: Replica has read all relay log; waiting for more updates
      Master_Retry_Count: 86400
```

CSDN @shark_西瓜甜

3.6.2. 完整规则

```
- alert: MysqlSlaveReplicationLag
  expr: ( (mysql_slave_status_seconds_behind_master -
mysql_slave_status_sql_delay) and ON (instance)
mysql_slave_status_master_server_id > 0 ) > 30
  for: 1m
  labels:
    severity: critical
  annotations:
    summary: MySQL Slave replication lag (instance {{ $labels.instance }})
    description: "MySQL 复制滞后了 \n 当前值: {{ $value }}\n 标签: {{ $labels
}}"
```

3.7. 慢查询

MySQL服务器有新的慢速查询。

3.7.1 规则拆解分析

指标 `mysql_global_status_slow_queries` 获取慢查询的数量，通常会观察在一定时间范围内是否在持续增长。

函数 `increase` 用于获取在一定时间范围内，数据的增长量。

这个值如果在最近 1 分钟内是在增加的，说明前一分钟内有慢查询。

3.7.2. 完整规则

```
- alert: MysqlSlowQueries
  expr: increase(mysql_global_status_slow_queries[1m]) > 0
  for: 2m
  labels:
    severity: warning
  annotations:
    summary: MySQL slow queries (instance {{ $labels.instance }})
    description: "MySQL 有一些新的慢查询.\n 当前值: {{ $value }}\n 标签: {{
$labels }}"
```

3.8. innodb 日志等待

MySQL innodb日志写入暂停

3.8.1 规则拆解分析

指标 `mysql_global_status_innodb_log_waits` 是获取的是对InnoDB重做日志文件的物理写入次数。

是 `show status;` 命令结果中的一个状态变量: `Innodb_log_waits`，记录了InnoDB引擎在等待重做日志写入磁盘时出现的等待次数。

通常会关注这个值的增长率，希望 15 分钟内增长率不超过 10%。

3.8.2. 完整规则


```
- alert: MysqlInnodbLogwaits
  expr: rate(mysql_global_status_innodb_log_waits[15m]) > 10
  for: 0m
  labels:
    severity: warning
  annotations:
    summary: MySQL InnoDB 日志等待 (instance {{ $labels.instance }})
    description: "MySQL i InnoDB 日志写入暂停\n VALUE = {{ $value }}\n LABELS = {{ $labels }}"
```

3.9. MySQL 实例 1 分钟内重启过

MySQL 实例刚刚在一分钟内重启过。

3.9.1 规则拆解分析

指标 `mysql_global_status_uptime` 获取的是 MySQL 服务运行的时间，如果值小于 60 秒，则认为 MySQL 服务 1 分钟内重启过（收到警报后，根据具体情况判断。）。

3.9.2. 完整规则

```
- alert: MysqlRestarted
  expr: mysql_global_status_uptime < 60
  for: 0m
  labels:
    severity: info
  annotations:
    summary: MySQL restarted (实例: {{ $labels.instance }})
    description: "MySQL 实例 {{ $labels.instance }} 1 分钟内刚刚重启.\n 当前值: {{ $value }}\n 标签: {{ $labels }}"
```

3.10. MySQL 组复制成员丢失

3.10.1 规则拆解分析

条件：此告警规则，适用于组复制集群或者 InnoDB Cluster。

指标 `mysql_perf_schema_replication_group_member_info` 获取的是组复制成员信息。

需要在 `mysqld_exporter` 启动参数增加 `--collect.perf_schema.replication_group_members`，示例如下：

```
/usr/local/mysqld_exporter/mysqld_exporter --
collect.perf_schema.replication_group_members --config.my-cnf=/root/.my.cnf
```

mysql_perf_schema_replication_group_member_info	Load time: 19ms Resolution: 14s Result set: 9
Table	Graph
<div> <div> Evaluation time </div> <div> </div> </div>	
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.10:3306, job=mysql, location=南京, member_communication_stack=XCom, member_host=mysql11, member_id=627bcab1-0b82-11ef-9b04-0242ac10000a, member_port=3306, member_role=PRIMARY, member_state=ONLINE, member_version=8.0.28, project=项目二)	1
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.10:3306, job=mysql, location=南京, member_communication_stack=XCom, member_host=mysql12, member_id=63129dc-0b82-11ef-9b29-0242ac100014, member_port=3306, member_role=SECONDARY, member_state=ONLINE, member_version=8.0.28, project=项目二)	1
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.10:3306, job=mysql, location=南京, member_communication_stack=XCom, member_host=mysql13, member_id=6366edc-0b82-11ef-9b0c-0242ac10001e, member_port=3306, member_role=SECONDARY, member_state=ONLINE, member_version=8.0.28, project=项目二)	1
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.20:3306, job=mysql, location=南京, member_communication_stack=XCom, member_host=mysql11, member_id=627bcab1-0b82-11ef-9b04-0242ac10000a, member_port=3306, member_role=PRIMARY, member_state=ONLINE, member_version=8.0.28, project=项目二)	1
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.20:3306, job=mysql, location=南京, member_communication_stack=XCom, member_host=mysql12, member_id=63129dc-0b82-11ef-9b29-0242ac100014, member_port=3306, member_role=SECONDARY, member_state=ONLINE, member_version=8.0.28, project=项目二)	1
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.20:3306, job=mysql, location=南京, member_communication_stack=XCom, member_host=mysql13, member_id=6366edc-0b82-11ef-9b0c-0242ac10001e, member_port=3306, member_role=SECONDARY, member_state=ONLINE, member_version=8.0.28, project=项目二)	1
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.30:3306, job=mysql, location=北京, member_communication_stack=XCom, member_host=mysql11, member_id=627bcab1-0b82-11ef-9b04-0242ac10000a, member_port=3306, member_role=PRIMARY, member_state=ONLINE, member_version=8.0.28, project=项目二)	1
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.30:3306, job=mysql, location=北京, member_communication_stack=XCom, member_host=mysql12, member_id=63129dc-0b82-11ef-9b29-0242ac100014, member_port=3306, member_role=SECONDARY, member_state=ONLINE, member_version=8.0.28, project=项目一)	1
mysql_perf_schema_replication_group_member_info(app=mysqlid, channel_name=group_replication_applier, instance=172.16.0.30:3306, job=mysql, location=北京, member_communication_stack=XCom, member_host=mysql13, member_id=6366edc-0b82-11ef-9b0c-0242ac10001e, member_port=3306, member_role=SECONDARY, member_state=ONLINE, member_version=8.0.28, project=项目一)	1

count(mysql_perf_schema_replication_group_member_info) by(instance)

Table Graph

Load time: 22ms Resolution: 14s Result series: 3

Evaluation time

(instance="172.16.0.10:3306")	3
(instance="172.16.0.20:3306")	3
(instance="172.16.0.30:3306")	3

表达式: `count(mysql_perf_schema_replication_group_member_info) by(instance) < 3`

```

- alert: MysqlClusterMemberLoss
  expr: count(mysql_perf_schema_replication_group_member_info) by(instance) <
  for: 1m
  labels:
    severity: warning
  annotations:
    summary: MySQL 集群成员丢失
    description: "MySQL 集群的组复制成员有丢失"

```