0. 实验环境准备 RabbitMQ 集群

使用 docker-comose 实现

目录结构:

```
[root@prometheus rabbitmqCluster]# ls -l
total 12
drwxr-xr-x 2 root root  127 May 29 11:55 conf.d
drwxr-xr-x 2 root root   29 May 27 16:27 config
-rw-r--r-- 1 root root  840 May 27 16:18 create-cluster.sh
drwxr-xr-x 3  100 root   63 May 27 17:17 data1
drwxr-xr-x 3  100 root   42 May 27 16:39 data2
drwxr-xr-x 3  100 root   42 May 27 16:39 data3
-rw-r--r-- 1 root root 1327 May 27 16:18 docker-compose.yml
```


docker-compose.yml

```yaml
version: "3.9"
services:
  rabbit1:
    environment:
      RABBITMQ_DEFAULT_USER: admin
      RABBITMQ_DEFAULT_PASS: Pass_123shark
    image: rabbitmq:3.9.11-alpine
    restart: always
    hostname: rabbit1
    ports:
      - "5671:5672"
      - "15671:15672"
    volumes:
      - "/etc/localtime:/etc/localtime"
      - "./data1:/var/lib/rabbitmq"
      - "./config/enabled_plugins:/etc/rabbitmq/enabled_plugins"
      - "./conf.d:/etc/rabbitmq/conf.d"
  rabbit2:
    environment:
      RABBITMQ_DEFAULT_USER: admin
      RABBITMQ_DEFAULT_PASS: Pass_123shark
    image: rabbitmq:3.9.11-alpine
    restart: always
    hostname: rabbit2
    ports:
      - "5672:5672"
      - "15672:15672"
    volumes:
      - "/etc/localtime:/etc/localtime"
      - "./data2:/var/lib/rabbitmq"
      - "./config/enabled_plugins:/etc/rabbitmq/enabled_plugins"
      - "./conf.d:/etc/rabbitmq/conf.d"
  rabbit3:
    environment:
      RABBITMQ_DEFAULT_USER: admin
      RABBITMQ_DEFAULT_PASS: Pass_123shark
    image: rabbitmq:3.9.11-alpine
    restart: always
    hostname: rabbit3
    ports:
      - "5673:5672"
      - "15673:15672"
    volumes:
      - "/etc/localtime:/etc/localtime"
      - "./data3:/var/lib/rabbitmq"
      - "./config/enabled_plugins:/etc/rabbitmq/enabled_plugins"
```

```
      - "./conf.d:/etc/rabbitmq/conf.d"
```

conf.d/10-default-guest-user.conf

```
loopback_users.guest = false
```

conf.d/management_agent.disable_metrics_collector.conf

```
management_agent.disable_metrics_collector = true
```

config/enabled_plugins

```
[rabbitmq_federation_management,rabbitmq_management,rabbitmq_mqtt,rabbitmq_stomp
,rabbitmq_prometheus].
```

create-cluster.sh

```
#!/bin/bash
cd /apps/middle/rabbitmqCluster

for i in {1..3}
do
    docker-compose exec rabbit${i} chown rabbitmq.rabbitmq
/var/lib/rabbitmq/.erlang.cookie
    docker-compose exec rabbit${i} rabbitmqctl set_permissions -p "/" admin ".*"
".*" ".*"
done
docker-compose restart
sleep 10
for i in {1..3}
do
    docker-compose exec rabbit${i} rabbitmqctl set_permissions -p "/" admin ".*"
".*" ".*"
done

for i in {2..3}
do
    docker-compose exec rabbit${i} rabbitmqctl stop_app
    sleep 2
    docker-compose exec rabbit${i} rabbitmqctl join_cluster --ram rabbit@rabbit1
    sleep 2
    docker-compose exec rabbit${i} rabbitmqctl start_app
    sleep 2
    docker-compose exec rabbit${i} rabbitmqctl set_policy ha-all "^" '{"ha-
mode":"all" , "ha-sync-mode":"automatic"}'
    sleep 2
done
docker-compose exec rabbit1 rabbitmqctl cluster_status
```

1. 介绍

RabbitMQ 的 Prometheus 监控方案有两种：

- RabbitMQ (3.8版本以上)自带的监控插件 `rabbitmq_prometheus`
- Prometheus 官方推荐的 `rabbitmq_exporter`

两个插件的区别
官方插件

- 具有 runtime/erlang 的监控指标
- 有聚合的或每个对象的指标
- 无法禁用某些不需要的监控指标，但是 Prometheus 可以单独获取某些监控指标数据

rabbitmq_exporter

- 适用于 Rabbitmq 的新旧版本
- 具有更多配置选项/筛选对象

可能最好的解决方案是同时使用两个：

- 官方的普罗米修斯插件不能跳过收集一些指标的能力。在官方插件中，您可以选择聚合指标或每个对象，但是不能跳过不需要的监控指标。
- 另一方面，官方插件提供erlang 虚拟机的监控指标，这也很重要。

所以建议两者这样用。在官方插件中，禁用每个对象的指标，并且只使用它来收集与erlang相关的指标，使用非官方插件收集其他所有监控指标。

> 对于以上建议，仅仅是个人建议。如果想简单实现，使用官方的插件即可。

2. 配置使用官方插件 rabbitmq_prometheus

2.1. 配置文件方式启用

重启 RabbitMQ 服务生效
`/etc/rabbitmq/enabled_plugins` 文件中增加如下内容

```
[rabbitmq_prometheus].
```

2.2. 命令行方式启用

立即生效

```
rabbitmq-plugins enable rabbitmq_prometheus
```

> 如果非首次部署后开启监控插件，可以同时使用两者结合，这样无需重启服务，并且在下次服务重启后，插件也是永久生效可用的。

2.3 配置启用指标收集器

确保指标收集器是启用的。

> docker 容器方式，默认是禁用的

在配置文件 `/etc/rabbitmq/conf.d/management_agent.disable_metrics_collector.conf` 添加如下内容

```
management_agent.disable_metrics_collector = false
```

2.4. 监听地址和端口

此插件开启成功后，默认监听 TCP 端口 `15692`，监听地址是服务器上的所有地址。

此端口可以使用如下配置修改监听的端口：
`/etc/rabbitmq/conf.d/rabbitmq_prometheus.conf`

```
prometheus.tcp.port = 15692
```

可以使用如下方式修改监听的地址:
`/etc/rabbitmq/conf.d/rabbitmq_prometheus.conf`

```
prometheus.tcp.ip = 0.0.0.0
```

可以使用如下命令检查每个节点监听的端口

```
rabbitmq-diagnostics -s listeners
```

配置成功后，重启 RabbiMQ。

确认是否成功开启

```
curl -s localhost:15692/metrics | head -n 3
```

3. 配置 RabbitMQ 集群名称

检查每个节点的机群名称是否一致

```
rabbitmq-diagnostics -q cluster_status
```

如果不一致，修改为一致，并且这样也可以和其他被监控的集群区别开来.
只需下集群中的任意一个节点上执行一次即可。

```
rabbitmqctl -q set_cluster_name sharkyun-rabbitmq
```

4. RabbitMQ 插件获取指标的频率

默认情况下，rabbitmq_prometheus 插件每隔 `5000` 毫秒（5 秒）抓取一次监控指标数据。
生产环境，建议修改为 `10000`(10秒)，并且将 Prometheus 的抓取间隔设置为 15秒。

查询获取指标的频率(是一个环境变量)

```
rabbitmq-diagnostics environment | grep collect_statistics_interval
```

当使用 RabbitMQ 的管理 UI 默认5秒自动刷新时，保持默认的 `collect_statistics_interval` 设置是最佳的。由于这个原因，两个时间间隔默认都是 `5000` 毫秒（5秒）。

5. 配置到 Prometheus

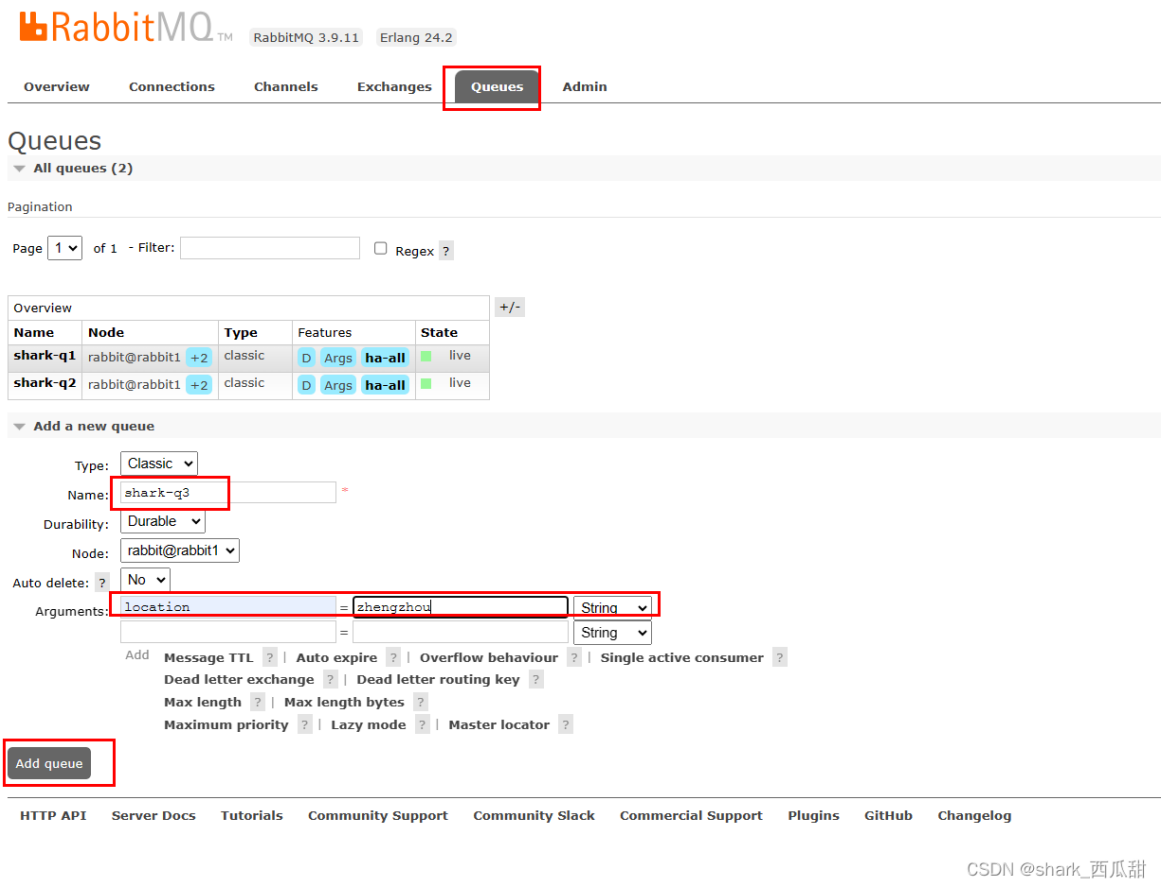
添加如下内容到 `prometheus.yml` 文件中

```
scrape_configs:
  - job_name: 'rabbitmq-exporter'
    static_configs:
      - targets:
        - rabbitmq1-host:15692
        - rabbitmq2-host:15692
        - rabbitmq3-host:15692
```

更新配置

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

6. 创建测试数据



7. 禁用统计信息和指标收集

如果使用普罗米修斯外部监控解决方案，可以禁用UI和HTTP API中的统计信息，以便这些统计信息仅用于操作。如果以以下任何方式禁用统计信息，则所有图表和详细统计信息都将隐藏在UI中。

1. 为了完全禁用内部度量集合，必须在 **rabbitmq_management_agent** 插件中设置 `disable_metrics_collector` 标志。即使收集被禁用，普罗米修斯插件仍然可以工作。就是 RabbitMQ 自身不需要再主动的收集监控数据，并存储这些监控数据，同时又支持 HTTP API 的调用（被动的，有请求就返回对应的监控指标数据）。
   `etc/rabbitmq/conf.d/rabbitmq_prometheus.conf`

   ```
   management_agent.disable_metrics_collector = true
   ```

   如果与外部监控系统一起使用，禁用指标收集是首选选项，因为这减少了统计信息收集和聚合在代理中造成的开销。

2. 由于目前Prometheus插件无法报告单个队列总数，因此有一个配置选项允许在队列端点中列出消息： `messages_ready` 和 `messages_unaknowledge`。
   以下是一个配置示例，它禁用统计信息，但在队列页面中返回单个队列总数：

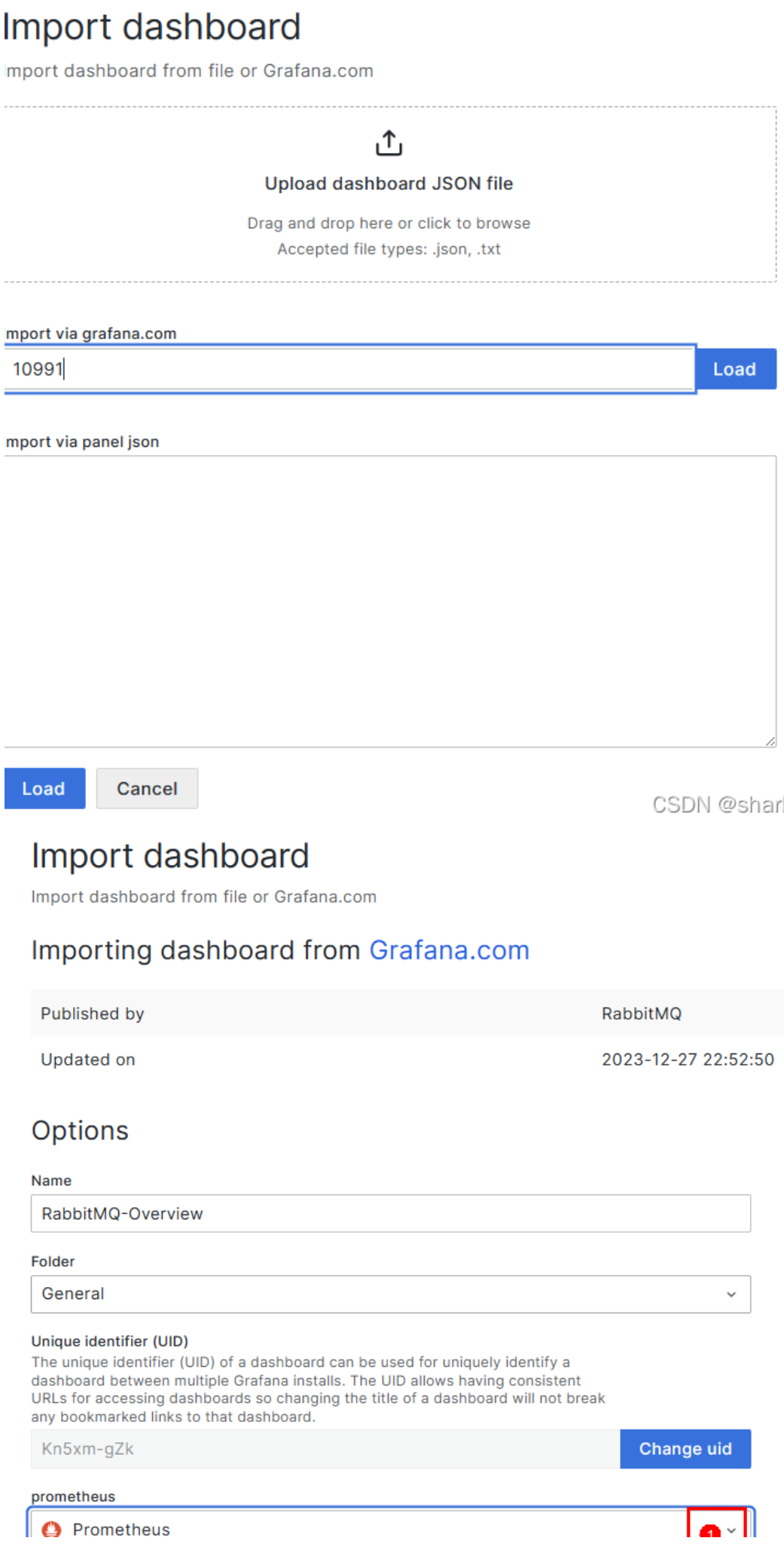
   ```
   management.disable_stats = true
   management.enable_queue_totals = true
   ```

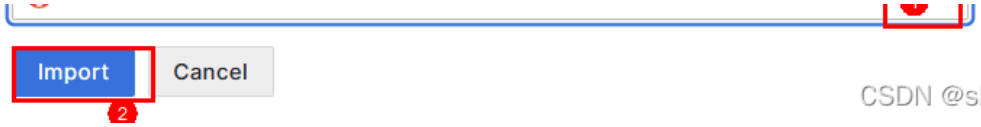
修改后，重启 RabbitMQ 服务。

8. dashboard

RabbitMQ和Erlang的Grafana仪表板是开源的，并从 RabbitMQ-server GitHub存储库公开。

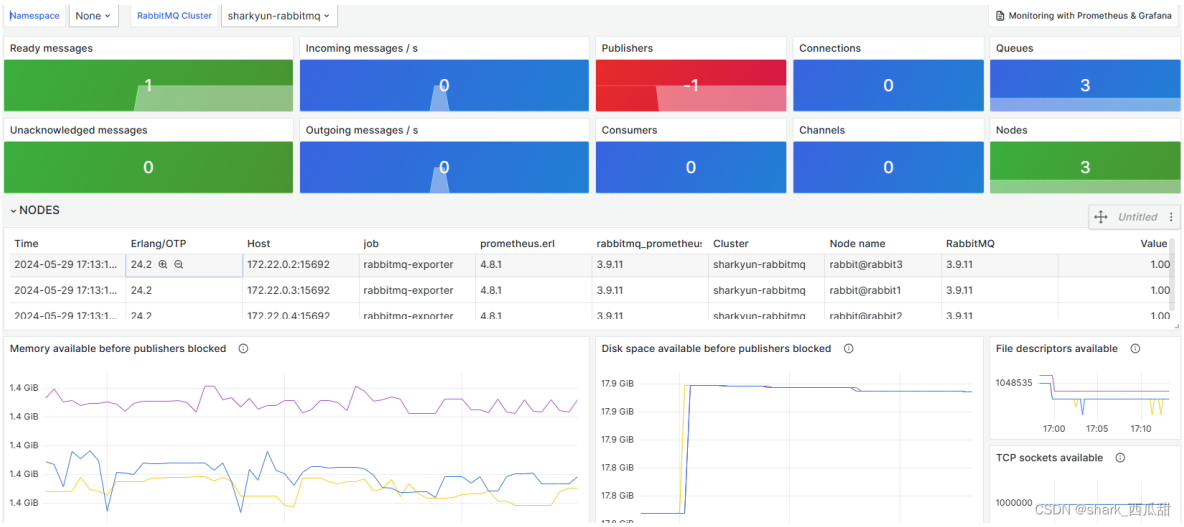
主要的是 RabbitMQ-Overview：**ID 10991**





管理 UI 中提供的所有指标概述页面在 Grafana 仪表板 `RabbitMQ-Overview` 中可用。它们按对象类型分组， 重点关注 RabbitMQ nodes 和 message rates.。

示例图：



5.1. 健康指标

仪表板顶部的单个统计信息指标捕获单个统计信息的运行状况 RabbitMQ 集群。
所有 RabbitMQ Grafana 仪表板上的面板都使用不同的颜色来捕获以下内容 指标状态：

- **绿色** 表示指标值在正常范围内
- **蓝色** 表示利用率不足或某种形式的退化
- **红色** 表示指标值低于或高于被视为正常的范围

单个统计信息指标的默认范围并非适用于所有 RabbitMQ 部署的最佳范围。
可以轻松调整默认阈值以适应当前的工作负载和系统。鼓励用户重新访问这些范围并进行调整 他们认为适合其工作负载、监控和操作实践以及宽容度 对于误报。

5.2. 指标和图表

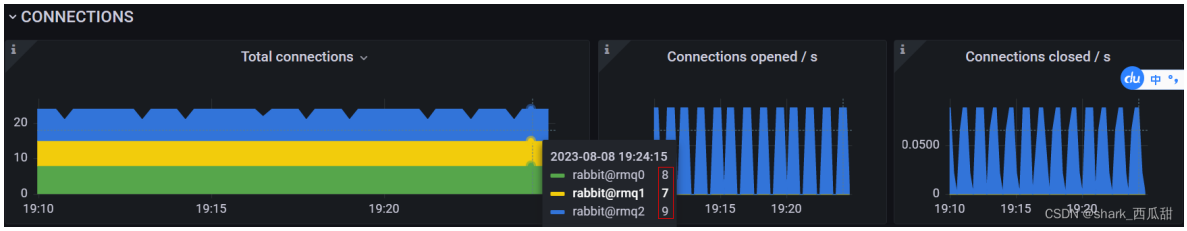
大多数 RabbitMQ 和运行时指标在 Grafana 中表示为图形：它们是随时间变化的值。 这是可视化系统某些方面如何变化的最简单、最清晰的方法。

基于时间的图表可以轻松理解关键指标的变化：消息速率、每个节点使用的内存 群集或并发连接数。除运行状况之外的所有指标 指标是特定于节点的，也就是说，它们表示单个节点上的指标值。

一些指标，例如在 **CONNECTIONS** 下分组的面板，被堆叠以捕获整个集群的状态。这些指标是在各个节点上收集的，并以可视方式分组，这使得人们很容易注意到，例如，当一个节点服务的连接数量不成比例时。

我们将这样的 RabbitMQ 集群称为不平衡集群，这意味着至少在某些方面， 少数节点执行了大部分工作。

在下面的示例中，连接大部分时间均匀分布在所有节点上：



5.3. 许多指标没有“正确”的阈值

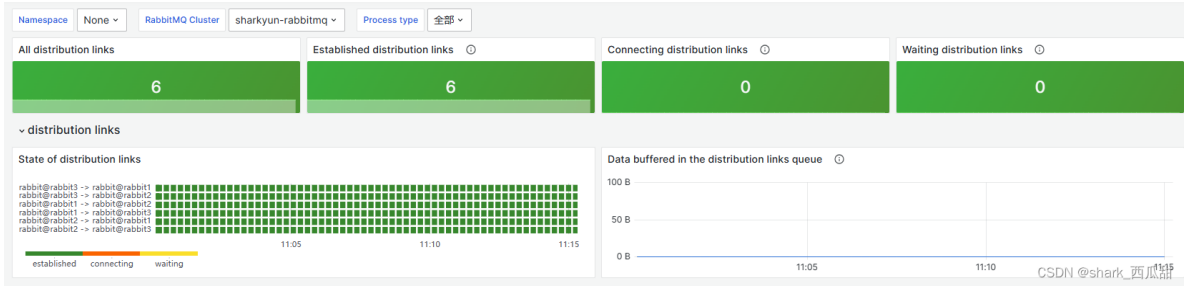
请注意，Grafana 仪表板使用的阈值必须具有默认值。不管怎样 仪表板开发人员选择什么值，它们将不适合所有环境和工作负载。

某些工作负载可能需要更高的阈值，而其他工作负载可能会选择降低阈值。虽然默认值应该是 在许多情况下，操作员必须审查和调整阈值以适应其特定 要求。

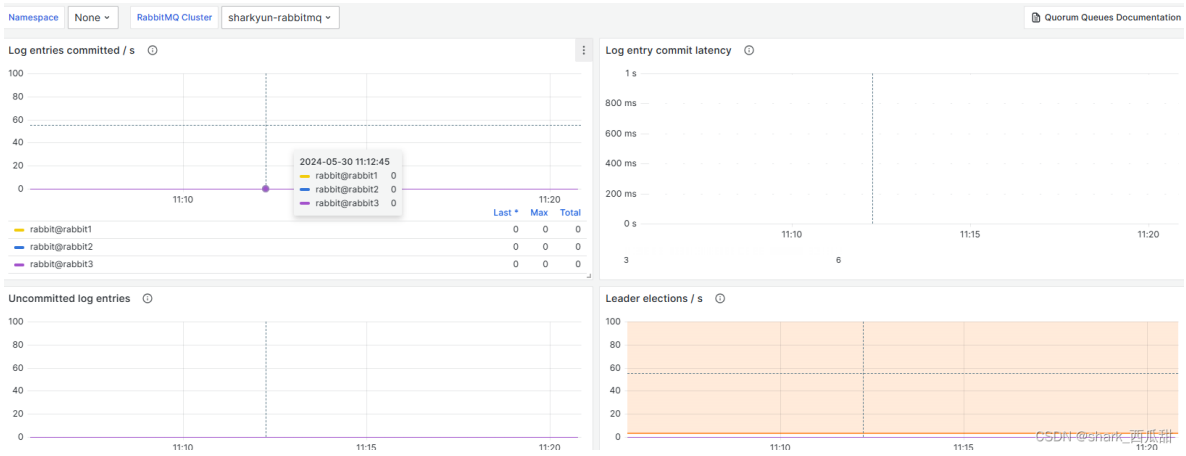
5.4. 更多仪表板：Raft 和 Erlang 运行时

还有两个Grafana面板可用：

Erlang-Distribution: ID 11352。



RabbitMQ-Raft : ID 11340

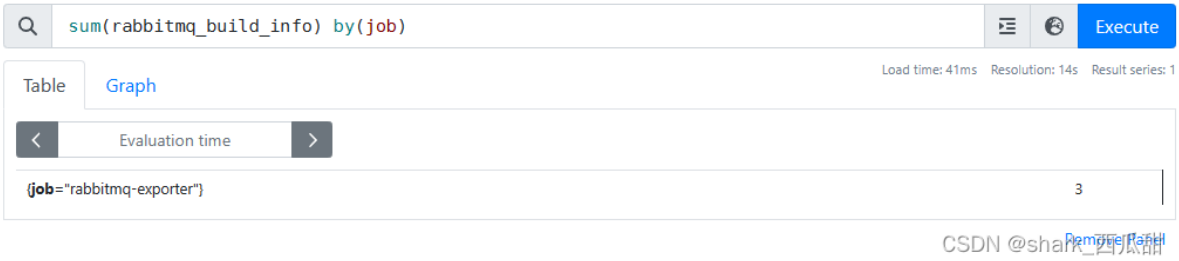


他们收集并可视化与Raft一致性算法（由Quorum Queues和其他功能使用）相关的指标，以及更具体的运行时指标，如节点间通信缓冲区。

9. 告警规则

9.1 Rabbitmq node down

指标 `rabbitmq_build_info` 返回每个实例的 rabbitmq 程序的构建信息。
如果监控多个集群，建议每个集群配置一个 `job_name`，统计的时候，使用 `by(job)` 子句进行聚合运算。




```
groups:
- name: MySQL Rules
  interval: 5s
  rules:
  - alert: RabbitmqNodeDown
    expr: sum(rabbitmq_build_info)  by(job) < 3
    for: 0m
    labels:
      severity: critical
    annotations:
      summary: Rabbitmq node down (instance {{ $labels.instance }})
      description: "RabbitMQ cluster 当前节点数量少于 3 个\n  VALUE = {{ $value
}}\n  LABELS = {{ $labels }}"
```

9.2 Rabbitmq node not distributed

分发链接状态不是 "up"
这个值应该是集群的节点数。

```
  - alert: RabbitmqNodeNotDistributed
    expr: erlang_vm_dist_node_state < 3
    for: 0m
    labels:
      severity: critical
    annotations:
      summary: Rabbitmq node not distributed (instance {{ $labels.instance }})
      description: "Erlang 部分节点状态异常\n  当前值: {{ $value }}\n  标签:  {{
$labels }}"
```

9.3 Rabbitmq instances different versions

Running different version of Rabbitmq in the same cluster, can lead to failure.
在同一个集群中运行不同版本的Rabbitmq可能会导致故障。

一个集群中节点的版本必须一致。

```
  - alert: RabbitmqInstancesDifferentVersions
    expr: count(count(rabbitmq_build_info) by (job,rabbitmq_version)) > 1
    for: 1h
    labels:
      severity: warning
    annotations:
      summary: Rabbitmq instances different versions (instance {{
$labels.instance }})
      description: "R在同一个集群中运行不同版本的Rabbitmq可能会导致故障。\n  当前值: {{
$value }}\n  标签:  {{ $labels }}"
```

9.4 Rabbitmq memory high

A node use more than 90% of allocated RAM
一个节点使用90%以上的分配 RAM

指标 `rabbitmq_process_resident_memory_bytes` 统计的是每个实例进程的常驻内存大小
指标 `rabbitmq_resident_memory_limit_bytes` 统计的是每个实例常驻内存限制大小

```
  - alert: RabbitmqMemoryHigh
    expr: rabbitmq_process_resident_memory_bytes /
rabbitmq_resident_memory_limit_bytes * 100 > 90
    for: 2m
    labels:
      severity: warning
    annotations:
      summary: Rabbitmq memory high (instance {{ $labels.instance }})
      description: "此节点使用的内存达到给其分配的 90%以上\n  当前值: {{ $value }}\n
标签:  {{ $labels }}"
```

9.5 Rabbitmq file descriptors usage

A node use more than 90% of file descriptors
一个节点使用了 90% 以上的文件描述符

```
  - alert: RabbitmqFileDescriptorsUsage
    expr: rabbitmq_process_open_fds / rabbitmq_process_max_fds * 100 > 90
    for: 2m
    labels:
      severity: warning
    annotations:
      summary: Rabbitmq file descriptors usage (instance {{ $labels.instance }})
      description: "这个节点使用的文件描述符超过了给其分配的 90% 了\n  当前值: {{ $value
}}\n  标签:  {{ $labels }}"
```

9.6 Rabbitmq too many unack messages

Too many unacknowledged messages
未确认的消息太多

```
  - alert: RabbitmqTooManyUnackMessages
    expr: sum(rabbitmq_queue_messages_unacked) BY (queue) > 1000
    for: 1m
    labels:
      severity: warning
    annotations:
      summary: Rabbitmq too many unack messages (instance {{ $labels.instance
}})
      description: "此节点未确认的消息太多了 \n  当前值: {{ $value }}\n  标签:  {{
$labels }}"
```

9.7 Rabbitmq too many connections

The total connections of a node is too high
节点的总连接数过高

```
  - alert: RabbitmqTooManyConnections
    expr: rabbitmq_connections > 1000
    for: 2m
    labels:
      severity: warning
    annotations:
      summary: Rabbitmq too many connections (instance {{ $labels.instance }})
      description: "节点的总连接数过高\n  当前值: {{ $value }}\n  标签:  {{ $labels
}}"
```

9.8 Rabbitmq no queue consumer

A queue has less than 1 consumer
队列的使用者少于1个。

集群在初建成，没有投入使用的时候，值会是 `0`，请根据情况处理告警。

```
  - alert: RabbitmqNoQueueConsumer
    expr: rabbitmq_queue_consumers < 1
    for: 1m
    labels:
      severity: warning
    annotations:
      summary: Rabbitmq no queue consumer (instance {{ $labels.instance }})
      description: "队列的使用者少于1个\n  当前值: {{ $value }}\n  标签:  {{ $labels
}}"
```

9.9 Rabbitmq unroutable messages

通道中不可路由的消息数量，如果在一段时间内值在增加，说明有问题，需要运维人员关注并处理。

```
  - alert: RabbitmqUnroutableMessages
    expr: increase(rabbitmq_channel_messages_unroutable_returned_total[1m]) > 0
or increase(rabbitmq_channel_messages_unroutable_dropped_total[1m]) > 0
    for: 2m
    labels:
      severity: warning
    annotations:
      summary: Rabbitmq unroutable messages (instance {{ $labels.instance }})
      description: "队列包含不可路由的消息\n  当前值: {{ $value }}\n  标签:  {{
$labels }}"
```

10. 指标

10.1 全局

Metric 指标	Description 描述
rabbitmq_consumer_prefetch	每个消费者的未确认消息的限制
rabbitmq_channel_prefetch	通道上所有消费者的未确认消息的总限制

Overview

Metric	Description
rabbitmq_connections_opened_total	打开的连接总数
rabbitmq_connections_closed_total	关闭或终止的连接总数
rabbitmq_channels_opened_total	打开的通道总数
rabbitmq_channels_closed_total	关闭的通道总数
rabbitmq_queues_declared_total	声明的队列总数
rabbitmq_queues_created_total	创建的队列总数
rabbitmq_queues_deleted_total	删除的队列总数
rabbitmq_process_open_fds	打开文件描述符
rabbitmq_process_open_tcp_sockets	打开TCP套接字
rabbitmq_process_resident_memory_bytes	使用的内存（字节）
rabbitmq_disk_space_available_bytes	可用磁盘空间（字节）
rabbitmq_process_max_fds	打开文件描述符限制
rabbitmq_process_max_tcp_sockets	打开TCP套接字限制
rabbitmq_resident_memory_limit_bytes	内存高水印（以字节为单位）
rabbitmq_disk_space_available_limit_bytes	可用磁盘空间低水位线（以字节为单位）
rabbitmq_connections	当前打开的连接
rabbitmq_channels	当前打开的频道
rabbitmq_consumers	当前连接的消费者
rabbitmq_queues	可用队列
rabbitmq_build_info	RabbitMQ和Erlang/OTP版本信息
rabbitmq_identity_info	RabbitMQ节点和集群标识信息

Connections

Metric	Description
rabbitmq_connection_incoming_bytes_total	连接上接收的字节总数
rabbitmq_connection_outgoing_bytes_total	在连接上发送的字节总数
rabbitmq_connection_process_reductions_total	Total number of connection process reductions 连接进程减少的总数
rabbitmq_connection_incoming_packets_total	在连接上接收的数据包总数
rabbitmq_connection_outgoing_packets_total	在连接上发送的数据包总数
rabbitmq_connection_pending_packets	在连接上等待发送的数据包数
rabbitmq_connection_channels	连接上的通道

Channels

Metric	Description
rabbitmq_channel_consumers	通道上的消费者
rabbitmq_channel_messages_unacked	已发送但尚未确认的消息
rabbitmq_channel_messages_unconfirmed	已发布但尚未确认的消息
rabbitmq_channel_messages_uncommitted	在事务中收到但尚未提交的消息
rabbitmq_channel_acks_uncommitted	事务中尚未提交的消息确认
rabbitmq_channel_messages_published_total	在通道上发布到exchange的邮件总数
rabbitmq_channel_messages_confirmed_total	发布到exchange并在通道上确认的消息总数
rabbitmq_channel_messages_unroutable_returned_total	作为强制发布到exchange并作为不可更改返回到发布者的邮件总数
rabbitmq_channel_messages_unroutable_dropped_total	以非强制方式发布到exchange并以不可更改方式丢弃的邮件总数
rabbitmq_channel_process_reductions_total	通道进程减少的总数
rabbitmq_channel_get_ack_total	在手动确认模式下使用basic.get获取的消息总数
rabbitmq_channel_get_total	在自动确认模式下使用basic.get提取的消息总数
rabbitmq_channel_messages_delivered_ack_total	以手动确认模式传递给消费者的消息总数
rabbitmq_channel_messages_delivered_total	以自动确认模式传递给消费者的消息总数
rabbitmq_channel_messages_redelivered_total	重新传递给消费者的邮件总数
rabbitmq_channel_messages_acked_total	消费者确认的邮件总数
rabbitmq_channel_get_empty_total	`basic.get` 操作未提取消息的总次数

Queues

Metric	Description
rabbitmq_queue_messages_published_total	发布到队列的邮件总数
rabbitmq_queue_messages_ready	准备发送给消费者的消息
rabbitmq_queue_messages_unacked	已发送给消费者但尚未确认的消息
rabbitmq_queue_messages	就绪消息和未确认消息的总和-队列总深度
rabbitmq_queue_process_reductions_total	队列进程减少的总数
rabbitmq_queue_consumers	排队的消费者
rabbitmq_queue_consumer_utilisation	消费者利用率
rabbitmq_queue_process_memory_bytes	Erlang队列进程使用的内存（以字节为单位）
rabbitmq_queue_messages_ram	存储在内存中的就绪和未确认信息
rabbitmq_queue_messages_ram_bytes	存储在内存中的就绪和未确认消息的大小
rabbitmq_queue_messages_ready_ram	存储在存储器中的就绪信息
rabbitmq_queue_messages_unacked_ram	存储在内存中的未确认消息
rabbitmq_queue_messages_persistent	持久消息
rabbitmq_queue_messages_persistent_bytes	持久消息的大小（以字节为单位）
rabbitmq_queue_messages_bytes	就绪消息和未确认消息的大小（以字节为单位）
rabbitmq_queue_messages_ready_bytes	就绪消息的大小（以字节为单位）
rabbitmq_queue_messages_unacked_bytes	所有未确认消息的大小（以字节为单位）
rabbitmq_queue_messages_paged_out	消息分页到磁盘
rabbitmq_queue_messages_paged_out_bytes	分页到磁盘的消息大小（以字节为单位）
rabbitmq_queue_disk_reads_total	队列从磁盘读取消息的总次数
rabbitmq_queue_disk_writes_total	队列将消息写入磁盘的总次数

Erlang via RabbitMQ

Metric	Description
rabbitmq_erlang_processes_used	使用的Erlang进程
rabbitmq_erlang_gc_runs_total	Erlang垃圾收集器运行的总数
rabbitmq_erlang_gc_reclaimed_bytes_totalTotal	Erlang垃圾收集器回收的内存字节数
rabbitmq_erlang_scheduler_context_switches_total	Erlang调度程序上下文交换的总数
rabbitmq_erlang_processes_limit	Erlang进程限制
rabbitmq_erlang_scheduler_run_queue	Erlang调度程序运行队列
rabbitmq_erlang_net_ticktime_seconds	节点间检测信号间隔（秒）
rabbitmq_erlang_uptime_seconds	节点正常运行时间

Disk IO

Metric	Description
rabbitmq_io_read_ops_total	I/O 读取操作总数
rabbitmq_io_read_bytes_total	读取的I/O字节总数
rabbitmq_io_write_ops_total	I/O写入操作总数
rabbitmq_io_write_bytes_total	写入的I/O字节总数
rabbitmq_io_sync_ops_total	I/O同步操作总数
rabbitmq_io_seek_ops_total	I/O寻道操作总数
rabbitmq_io_open_attempt_ops_total	文件打开尝试的总数
rabbitmq_io_reopen_ops_total	重新打开文件的总次数
rabbitmq_schema_db_ram_tx_total	Total number of Schema DB memory transactions
rabbitmq_schema_db_disk_tx_total	Total number of Schema DB disk transactions
rabbitmq_msg_store_read_total	Total number of Message Store read operations
rabbitmq_msg_store_write_total	Total number of Message Store write operations
rabbitmq_queue_index_read_ops_total	Total number of Queue Index read operations
rabbitmq_queue_index_write_ops_total	Total number of Queue Index write operations
rabbitmq_queue_index_journal_write_ops_total	Total number of Queue Index Journal write operations
rabbitmq_io_read_time_seconds_total	总I/O读取时间
rabbitmq_io_write_time_seconds_total	T总I/O写入时间
rabbitmq_io_sync_time_seconds_total	总I/O同步时间
rabbitmq_io_seek_time_seconds_total	总I/O寻道时间
rabbitmq_io_open_attempt_time_seconds_total	文件打开尝试的总时间

Raft

Metric	Description
rabbitmq_raft_term_total	当前 Raft 编号
rabbitmq_raft_log_snapshot_index	Raft log 日志快照索引
rabbitmq_raft_log_last_applied_index	Raft log 最新的应用的索引
rabbitmq_raft_log_commit_index	Raft log 提交索引
rabbitmq_raft_log_last_written_index	Raft log 最新写入索引
rabbitmq_raft_entry_commit_latency_seconds	提交条目所花费的时间

Erlang

Mnesia

Metric	Description
erlang_mnesia_held_locks	持有锁的数量
erlang_mnesia_lock_queue	等待锁定的事务数
erlang_mnesia_transaction_participants	参与者交易数量
erlang_mnesia_transaction_coordinators	协调员事务数
erlang_mnesia_failed_transactions	失败（即中止）的事务数
erlang_mnesia_committed_transactions	已提交交易的数量
erlang_mnesia_logged_transactions	记录的事务数
erlang_mnesia_restarted_transactions	事务重新启动的总数

VM

Metric 指标	Description 描述
erlang_vm_dist_recv_bytes	套接字接收的字节数。
erlang_vm_dist_recv_cnt	套接字接收的数据包数。
erlang_vm_dist_recv_max_bytes	套接字接收的最大数据包的大小（以字节为单位）。
erlang_vm_dist_recv_avg_bytes	套接字接收的数据包的平均大小（以字节为单位）。
erlang_vm_dist_recv_dvi_bytes	套接字接收的平均数据包大小偏差（以字节为单位）。
erlang_vm_dist_send_bytes	从套接字发送的字节数。
erlang_vm_dist_send_cnt	从套接字发送的数据包数。
erlang_vm_dist_send_max_bytes	从套接字发送的最大数据包的大小（以字节为单位）。
erlang_vm_dist_send_avg_bytes	从套接字发送的数据包的平均大小（以字节为单位）。
erlang_vm_dist_send_pend_bytes	等待套接字发送的字节数
erlang_vm_dist_port_input_bytes	从端口读取的字节总数
erlang_vm_dist_port_output_bytes	写入端口的总字节数
erlang_vm_dist_port_memory_bytes	运行时系统为此端口分配的字节总数。端口本身可能已分配未包含的内存
erlang_vm_dist_port_queue_size_bytes	使用ERTS驱动程序队列实现的端口排队的字节总数
erlang_vm_dist_proc_memory_bytes	进程的大小（以字节为单位）。这包括调用堆栈、堆和内部结构
erlang_vm_dist_proc_heap_size_words	进程最年轻堆生成的大小（以字为单位）。此生成包括进程堆栈。这些信息高度依赖于实现，如果实现发生变化，这些信息可能会发生变化
erlang_vm_dist_proc_min_heap_size_words	进程的最小堆大小
erlang_vm_dist_proc_min_bin_vheap_size_words	进程的最小二进制虚拟堆大小
erlang_vm_dist_proc_stack_size_words	进程的堆栈大小（以字为单位）
erlang_vm_dist_proc_total_heap_size_words	进程的所有堆碎片的总大小（以字为单位）。这包括进程堆栈和被认为是堆的一部分的任何未接收消息
erlang_vm_dist_proc_message_queue_len	进程的消息队列中当前的消息数
erlang_vm_dist_procreductions	进程执行的缩减数
erlang_vm_dist_proc_status	分发进程的当前状态。状态表示为一个数值，其中 `exiting=1`, `suspended=2`, `runnable=3`, `garbage_collecting=4`, `running=5'` and `waiting=6`
erlang_vm_dist_node_state	分发链接的当前状态。状态表示为一个数值，其中 `pending=1`、`up_pending=2` 和 `up=3`
erlang_vm_dist_node_queue_size_bytes	输出分发队列中的字节数。这个队列位于Erlang代码和端口驱动程序之间
erlang_vm_memory_atom_bytes_total	当前分配给原子的内存总量。该内存是作为系统内存呈现的内存的一部分
erlang_vm_memory_bytes_total	当前分配的内存总量。这与进程和系统的内存大小之和相同
erlang_vm_memory_dets_tables	Erlang VM DETS表计数。
erlang_vm_memory_ets_tables	Erlang VM ETS表计数。
erlang_vm_memory_process_bytes_total	当前分配给erlang进程的内存总量
erlang_vm_memory_system_bytes_total	当前分配给与任何erlang进程没有直接关系的模拟器的内存总量。以进程形式呈现的内存不包括在此内存中
erlang_vm_statistics_bytes_output_total	输出到端口的字节总数
erlang_vm_statistics_bytes_received_total	通过端口接收的字节总数

Metric 指标	Description 描述
erlang_vm_statistics_context_switches	自系统启动以来上下文开关的总数
erlang_vm_statistics_dirty_cpu_run_queue_length	脏cpu运行队列的长度
erlang_vm_statistics_dirty_io_run_queue_length	脏io运行队列的长度
erlang_vm_statistics_garbage_collection_number_of_gcs	垃圾收集：GC的数量
erlang_vm_statistics_garbage_collection_bytes_reclaid	垃圾回收：回收的字节数
erlang_vm_statistics_garbage_collection_words_retaind	垃圾回收：回收的单词
erlang_vm_statistics_reductions_total	总减少量
erlang_vm_statistics_run_queues_length_total	正常运行队列的长度
erlang_vm_statistics_wallclock_time_milliseconds	关于挂钟的信息。与 `erlang_vm_statistics_runtime_milliseconds` 相同，但测量的是实时
erlang_vm_statistics_runtime_milliseconds	erlang运行时系统中所有线程的运行时总和。可以大于墙上的时钟时间
erlang_vm_statistics_wallclock_time_milliseconds	关于挂钟的信息。与 `erlang_vm_statistics_runtime_milliseconds` 相同，但测量的是实时
erlang_vm_dirty_cpu_schedulers	模拟器使用的调度程序脏cpu调度程序线程数
erlang_vm_dirty_cpu_schedulers_online	联机的脏cpu调度程序线程数
erlang_vm_dirty_io_schedulers	模拟器使用的调度程序脏I/O调度程序线程数
erlang_vm_ets_limit	允许的最大ets表数
erlang_vm_logical_processors	检测到的系统中配置的逻辑处理器数
erlang_vm_logical_processors_available	检测到的erlang运行时系统可用的逻辑处理器数
erlang_vm_logical_processors_online	检测到的系统上联机的逻辑处理器数
erlang_vm_port_count	本地节点上当前存在的端口数
erlang_vm_port_limit	本地节点上同时存在的最大端口数
erlang_vm_process_count	本地节点上当前存在的进程数
erlang_vm_process_limit	本地节点上同时存在的进程的最大数目
erlang_vm_schedulers	模拟器使用的调度程序线程数
erlang_vm_schedulers_online	联机的调度程序数
erlang_vm_smp_support	1（如果模拟器是使用smp支持编译的），否则为0
erlang_vm_threads	1（如果模拟器是使用线程支持编译的），否则为0
erlang_vm_thread_pool_size	异步线程池中用于异步驱动程序调用的异步线程数
erlang_vm_time_correction	1如果启用了时间校正，则为0
erlang_vm_atom_count	本地节点上当前存在的原子数
erlang_vm_atom_limit	本地节点上同时存在的原子的最大数量
erlang_vm_allocaters	为vm中的不同分配器分配的（carriers_size）和使用的（blocks_size）内存。

#