

|  |  |
| --- | --- |
| 基调听云运维手册 | |
| 手册版本 | V1.0 |

前言

**文档修订记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **标题** | -基调听云运维手册 | | | |
| **项目名称** |  | | | |
| **类别** | 管理文档 □ 设计方案 □ 实施文档 ☑  配置文档 □ 测试文档 □ 其他 □ | | | |
| **创建日期** | 2022/04/01 | | | |
| **一线人员** | 刘孟凯 | | | |
|  |  | | | |
| **版本号** | **日期** | **修改人** | **审阅人** | **摘要** |
| V1.0 | 2022/04/01 | 刘孟凯 |  | 编写 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

版权说明

本文档中出现的任何文字叙述、文档格式、插图、照片、方法、代码等内容，除由特别注明，版权均属于以基调网络公司所有，受到有关产权及版权法保护。任何个人、机构未经基调网络公司的书面授权许可，不得以任何方式复制或引用本文档的任何片断。

产品版本

与本手册相对应的产品版本如下所示。

| 产品名称 | 简称 | 产品版本 |
| --- | --- | --- |
| 听云悟空 | 悟空 | 3.6.2.x |

内容介绍

本手册主要介绍了听云悟空私有化平台的在运行过程中，巡检的相关任务和故障处理的方法及步骤以及其他。

读者对象

本手册适用于以下人员：

* 维护工程师
* 监控工程师

目录

1 日常维护命令 7

1.1 概述 7

1.2 应用维护命令 7

1.2.1 管理脚本 7

1.2.2 launch.sh 8

spring组件的launch.sh 8

非spring组件的launch.sh 8

2 日常维护工具 9

2.1 自监控prometheus 9

2.2 自监控grafana 9

2.3 Druid router 10

2.3.1 概览页 10

2.3.2 DataSource页 10

2.3.3 Task页 11

2.3.4 Servers页 11

2.4 配置中心Nacos 12

3 巡检 13

3.1 日常巡检 13

3.2 无数据问题排查 13

3.2.1 messages in为0 13

3.2.2 messages out为0 14

3.2.3 messages out不为0，但明显小于in ，且有积压。 14

3.2.4 各产线重点topic 14

3.2.5 消费正常 14

4 磁盘空间清理 15

4.1 组件日志清理 15

4.2 es数据清理 15

4.3 nbfs数据清理 16

4.4 Druid/hadoop数据清理 16

4.5 Kafka数据清理 17

5 组件扩容&迁移 18

5.1 组件说明 18

5.2 可直接扩容的自有组件举例 21

5.3 需要修改配置的组件扩容举例 22

5.4 es节点扩容 22

5.5 Common-nbfs-server节点扩容 22

5.6 Druid节点扩容 23

5.7 result.list调整 23

5.8 自监控调整 23

6 附件：Grafana监控指标详解 24

6.1 应用整体维度视图 24

6.1.1 Application Overview概述 24

6.1.2 HTTP视图 25

6.1.3 Database 视图 25

6.1.4 Redis 视图 26

6.1.5 Dubbo 视图 26

6.1.6 Elasticsearch 视图 26

6.1.7 Druid 视图 26

6.1.8 Log Event 视图 27

6.1.9 JVM 视图 27

6.2 应用侧组件视图 27

6.2.1 HTTP Request 27

6.2.2 MySQL 28

6.2.3 Redis 28

6.2.4 Dubbo 29

6.2.5 ES 29

6.2.6 Druid 30

6.3 操作系统维度视图 31

6.3.1 操作系统监控指标详解 31

6.3.2 系统摘要 31

6.3.3 CPU负载 32

6.3.4 内存使用 32

6.3.5 CPU使用率 32

6.3.6 进程信息 33

6.3.7 上下文切换 33

6.3.8 中断次数 33

6.3.9 网络吞吐量 33

6.3.10 网卡每小时的收发量 34

6.3.11 SWAP的使用 34

6.3.12 SWAP的使用活跃度 35

6.3.13 打开文件数 35

6.3.14 挂载点磁盘空间 35

6.3.15 磁盘延迟 36

6.3.16 磁盘IOPS 36

6.3.17 磁盘吞吐 36

6.3.18 IO util 36

6.4 组件监控 37

6.4.1 Zookeeper 监控 37

6.4.2 Kafka监控 39

6.4.3 Redis 监控 41

6.4.4 MySQL监控 43

6.4.5 ElasticSearch 监控 46

6.4.6 Druid监控 47

6.5 JVM监控 47

6.5.1 JVM启动时间 47

6.5.2 当前线程数/deadlock线程数 48

6.5.3 JDK版本 48

6.5.4 Heap区内存使用情况 48

6.5.5 Non Heap区内存使用情况 48

6.5.6 Eden/Survivor/Old区使用情况 49

6.5.7 平均每秒GC次数 49

6.5.8 平均每次GC时间 50

6.5.9 进程负载 50

概述

# 日常维护命令

## 概述

APM整套系统是一套复杂的分布式系统，组件较多且复杂，故设定管理节点概念。管理节点的service\_maneger.sh可基于ssh免密对集群全部组件进行管理。

## 应用维护命令

### 管理脚本

管理脚本service\_manager.sh在管理节点的安装目录下，比如/data/tingyun/service\_manager.sh

* 全部应用的停止命令：

sh service\_manager.sh stop all

因实际安装组件较多，stop all命令是逐个停止，建议可多开启几个窗口，同时执行某一类组件的停止。

* 全部应用的状态检查命令：

sh service\_manager.sh status all

* 全部应用的启动命令：

因组件有依赖关系，应用启动有先后顺序。需按照下面顺序进行启动：

sh service\_manager.sh start base

sh service\_manager.sh status base (验证都是running继续下一步)

sh service\_manager.sh start common

sh service\_manager.sh status common (验证都是running继续下一步)

sh service\_manager.sh start application

sh service\_manager.sh start self-monitor

sh service\_manager.sh status all (最后验证全部组件启动是否成功)

* 某一类组件启停

sh service\_manager.sh start/status/stop base/apm ..... 后面参数可以带某一类组件

Base - 基础组件 / apm - apm产品线全部组件 / common- 全部公共组件等

* 某一个组件全部节点启停

sh service\_manager.sh start/status/stop base-mysql/apm-api....后面参数跟具体组件名，可以管理该组件全部节点

* 某节点上的某个组件启停

sh service\_manager.sh start/status/stop apm-api apm-01 后面跟第二个参数加上主机hostname，可以管理对应节点上的具体应用

### launch.sh

#### spring组件的launch.sh

自有组件的launch.sh在每台主机/data/tingyun/bin目录下，可使用该脚本管理改节点的spring进程。

/data/tingyun/bin/launch.sh start/status/stop apm-dc-wrap

#### 非spring组件的launch.sh

非spring的比如base-mysql base-kafka等。在对应的组件目录下会有launch.sh脚本维护对应组件。

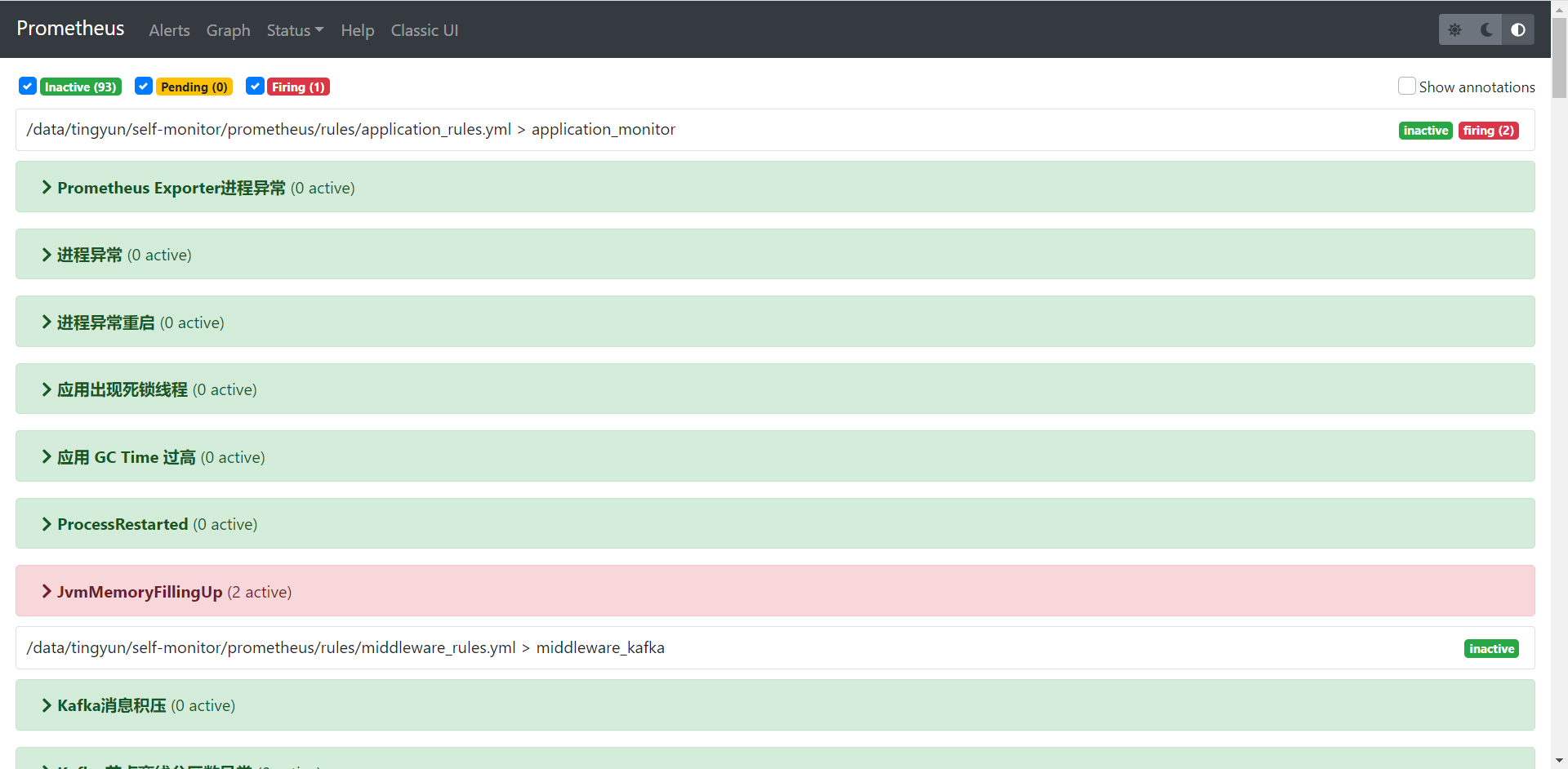
例如：/data/tingyun/base/base-nginx/launch.sh stop 就是停止该nginx节点。

# 日常维护工具

## 自监控prometheus

* 登陆地址：<http://$nginx-ip:8080/prometheus>

tingyun/Tingyun2\_18



* 访问监控告警页面查看是否有异常告警，可监控类型有：

a服务器负载、内存、磁盘io、磁盘空间、磁盘带宽等基础信息。

b进程异常、进程重启、进程日志错误数、进程jvm等进程信息

c基础组件mysql redis 进程状态、连接数、qps、 slave状态等

d消息队列kafka消息积压、副本数异常、进程异常等信息

e存储es、druid、hadoop等集群状态、内存使用率等

可根据实际内容进行问题定位。

## 自监控grafana

* 登陆地址：[http://$nginx-ip:8080/grafana](http://$nginx-ip:8080/prometheus)

admin/nEtop2o18tingyun

Grafana可以查看平台各个组件运行的具体情况。主要分为3类：Application Group（应用整体维度视图）、Middleware Group（中间件维度视图）、System Group（操作系统维度视图）

## Druid router

需要先启动druid-router节点

Druid集群01节点：

source /data/tingyun/tingyun-env.sh

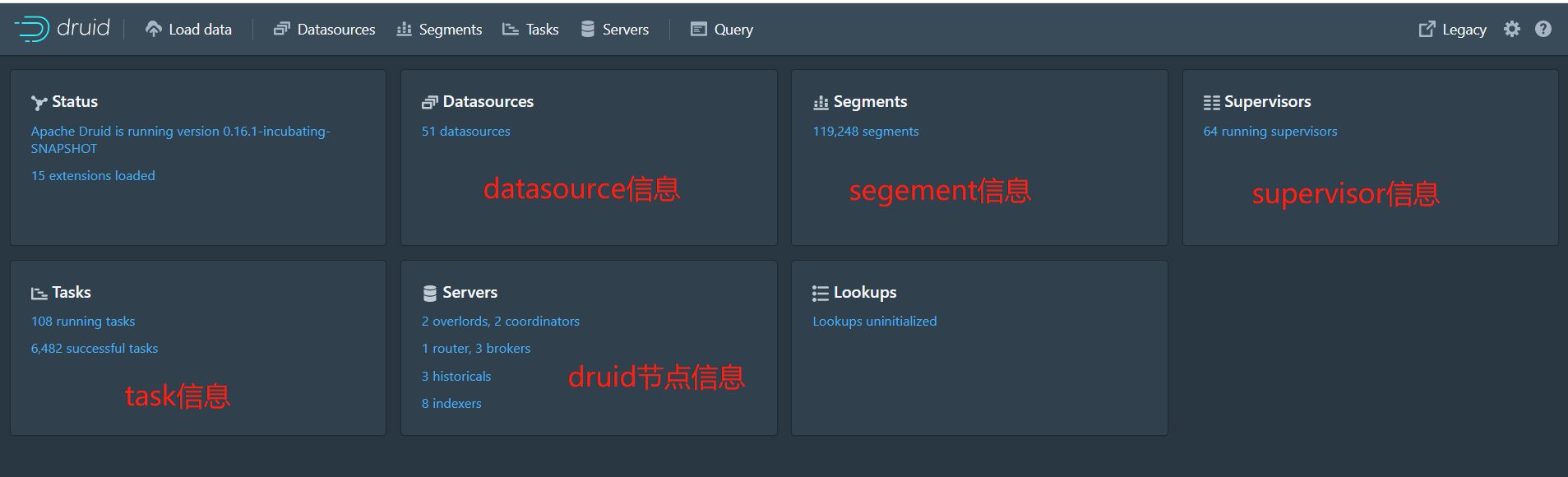
/data/tingyun/base/base-druid/bin/router.sh start

Url: http://$druid-01节点ip:8888

用户名：admin

密码：nEtben@2\_19

### 概览页



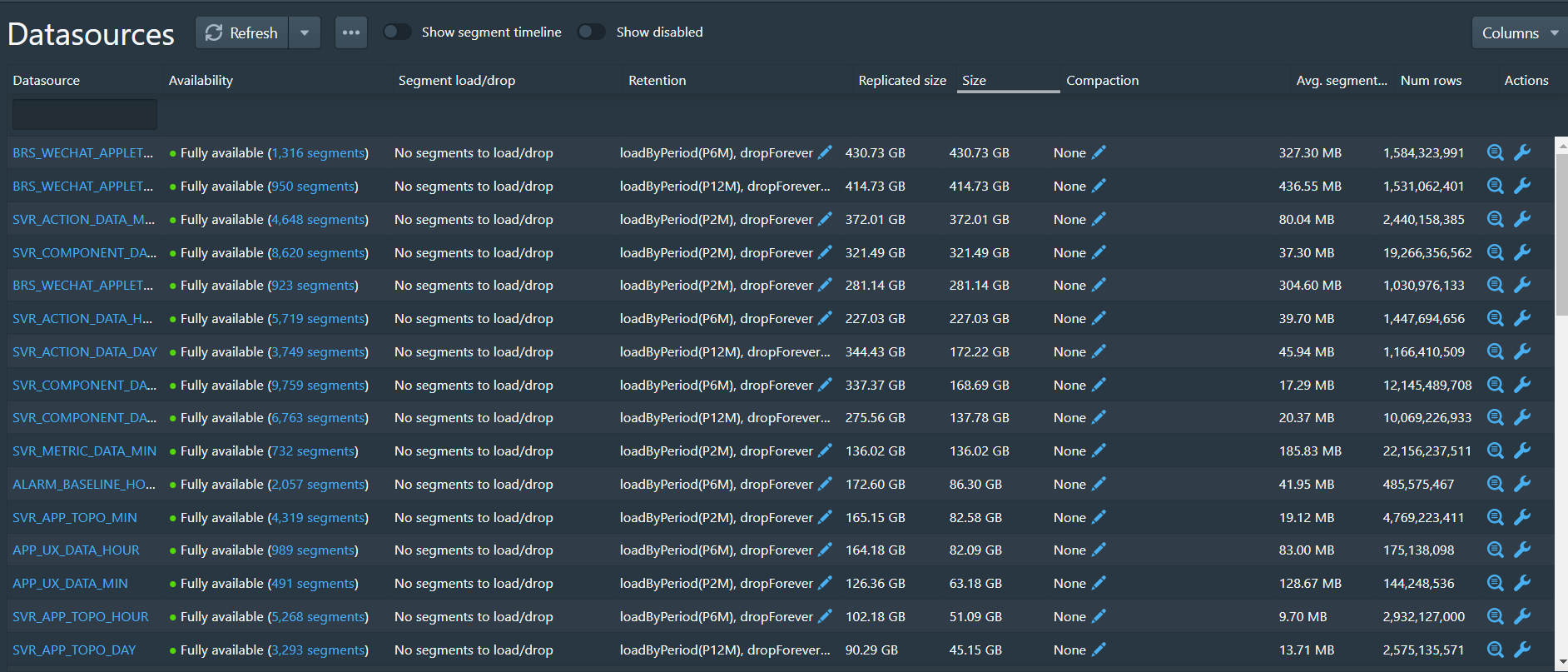
Task信息：展示task的状态，如果出现failed task，需要关注。

Segements：展示数据源文件信息，如果出现Unavailable segement 需要关注

Servers：展示节点个数，如果节点个数和实际个数不同，需要关注

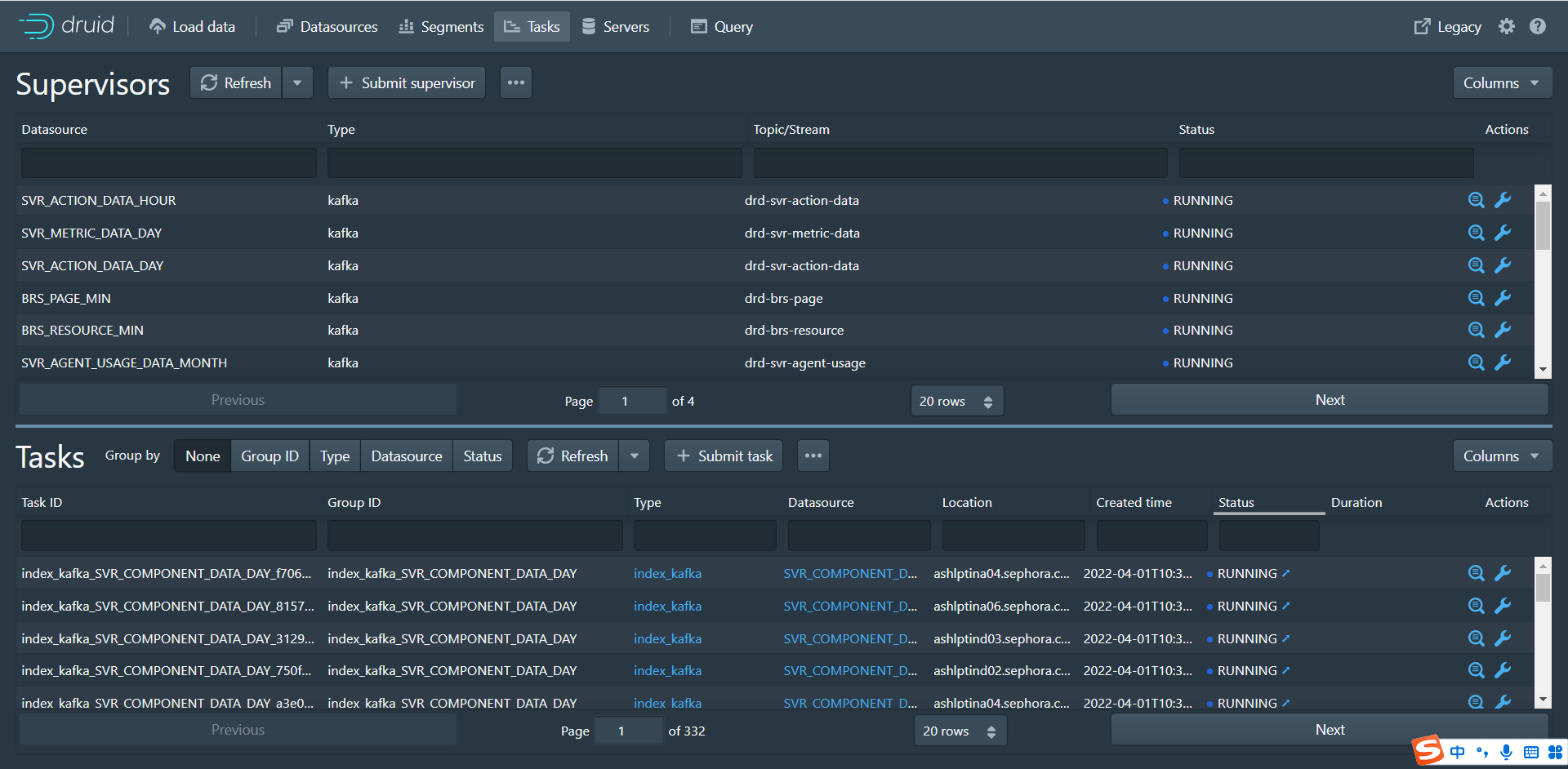
### DataSource页

展示全部datasource的状态、清理策略、空间大小等信息。点击size可按照磁盘大小排序，在做druid机器磁盘清理时可根据该项值，针对调整单独DataSource的数据保存周期。



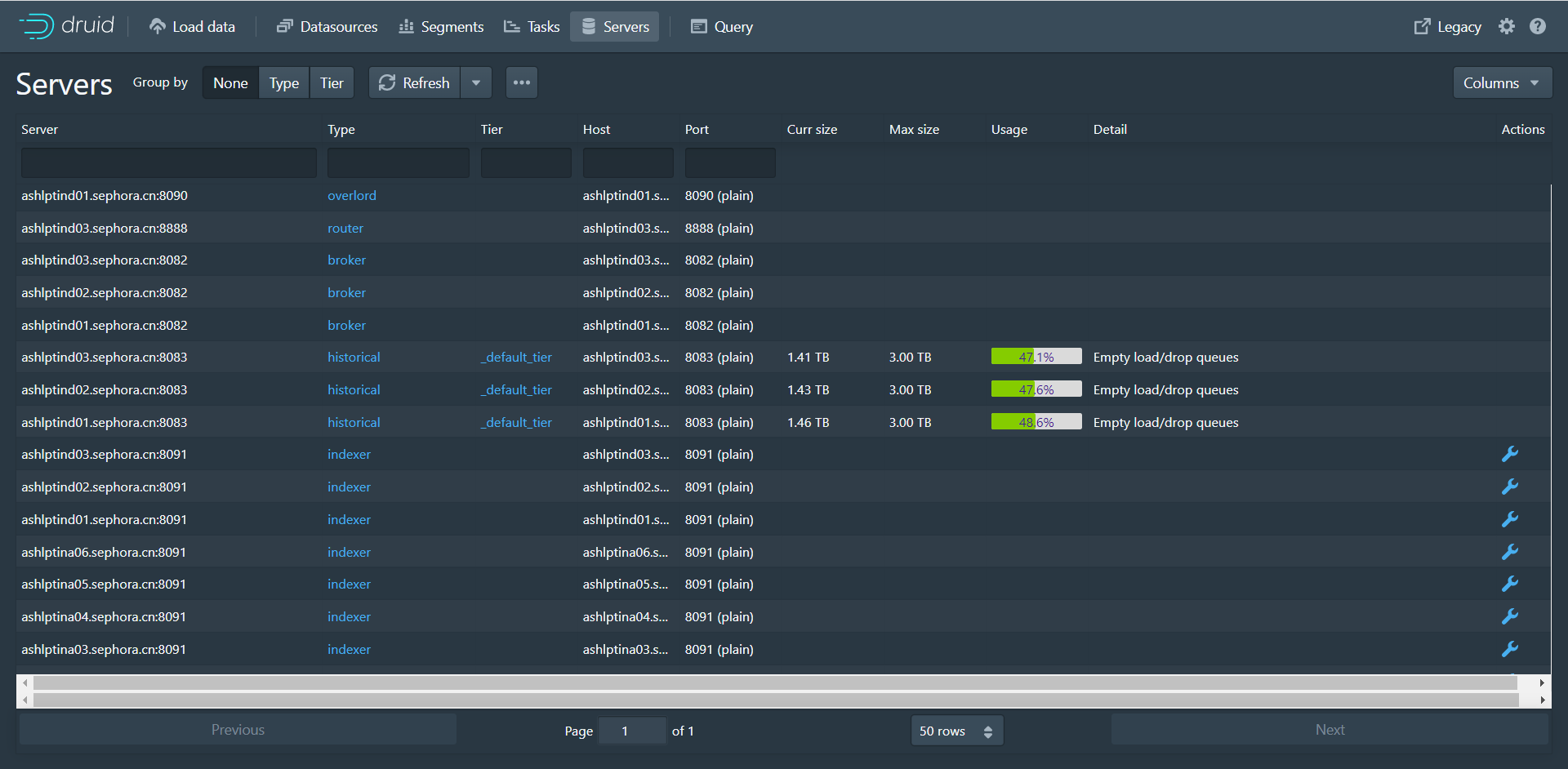
### Task页

展示supervisor和task运行状态，若druid摄入有积压，不进行摄入，需要关注该页面对应的supervisor和task状态。



### Servers页

展示全部节点的信息。



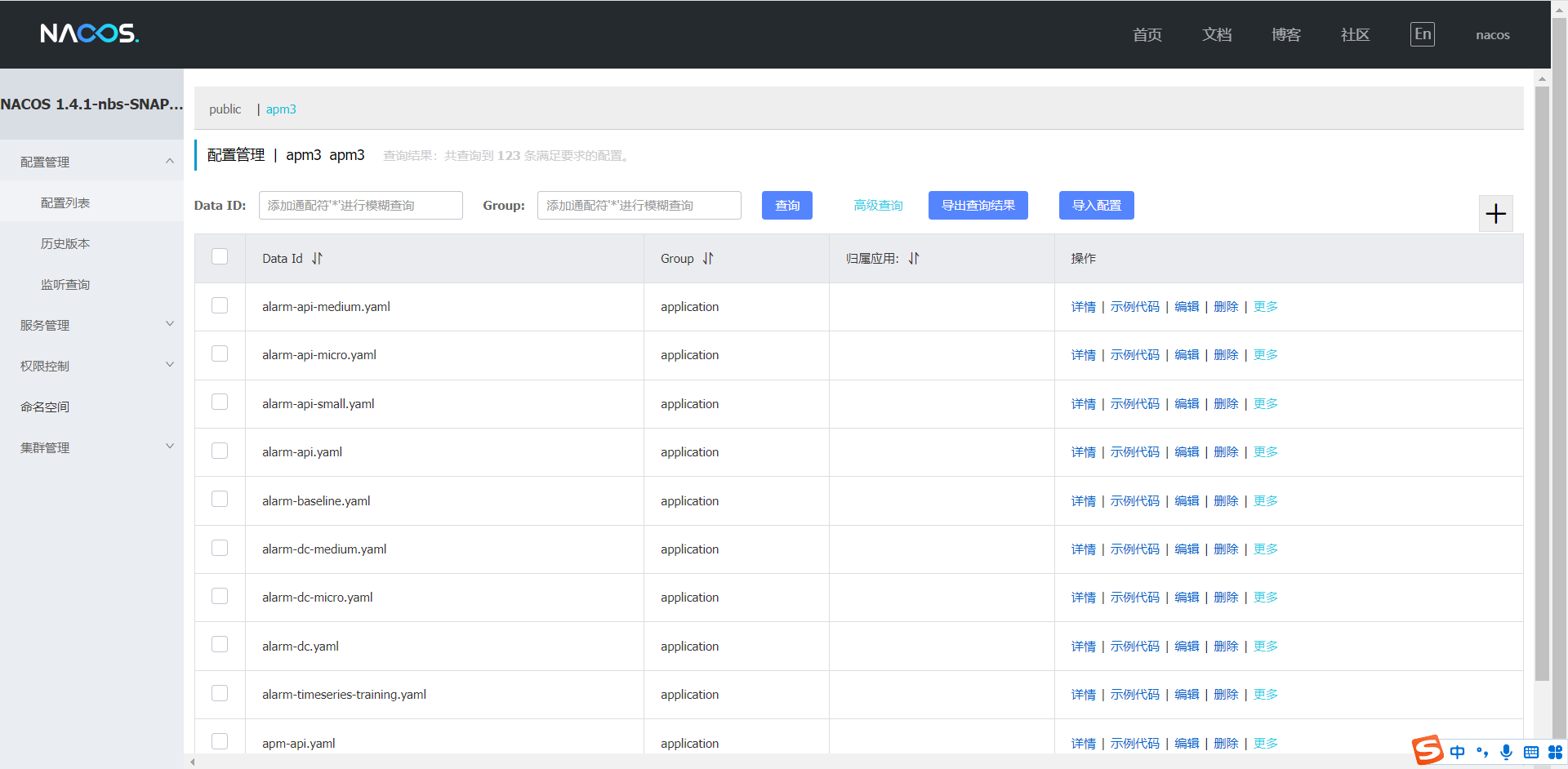
## 配置中心Nacos

Url: http://$reportip:18848/nacos

用户名：nacos

密码：RzTuW0QLnHXQ9MTr

配置管理中心



# 巡检

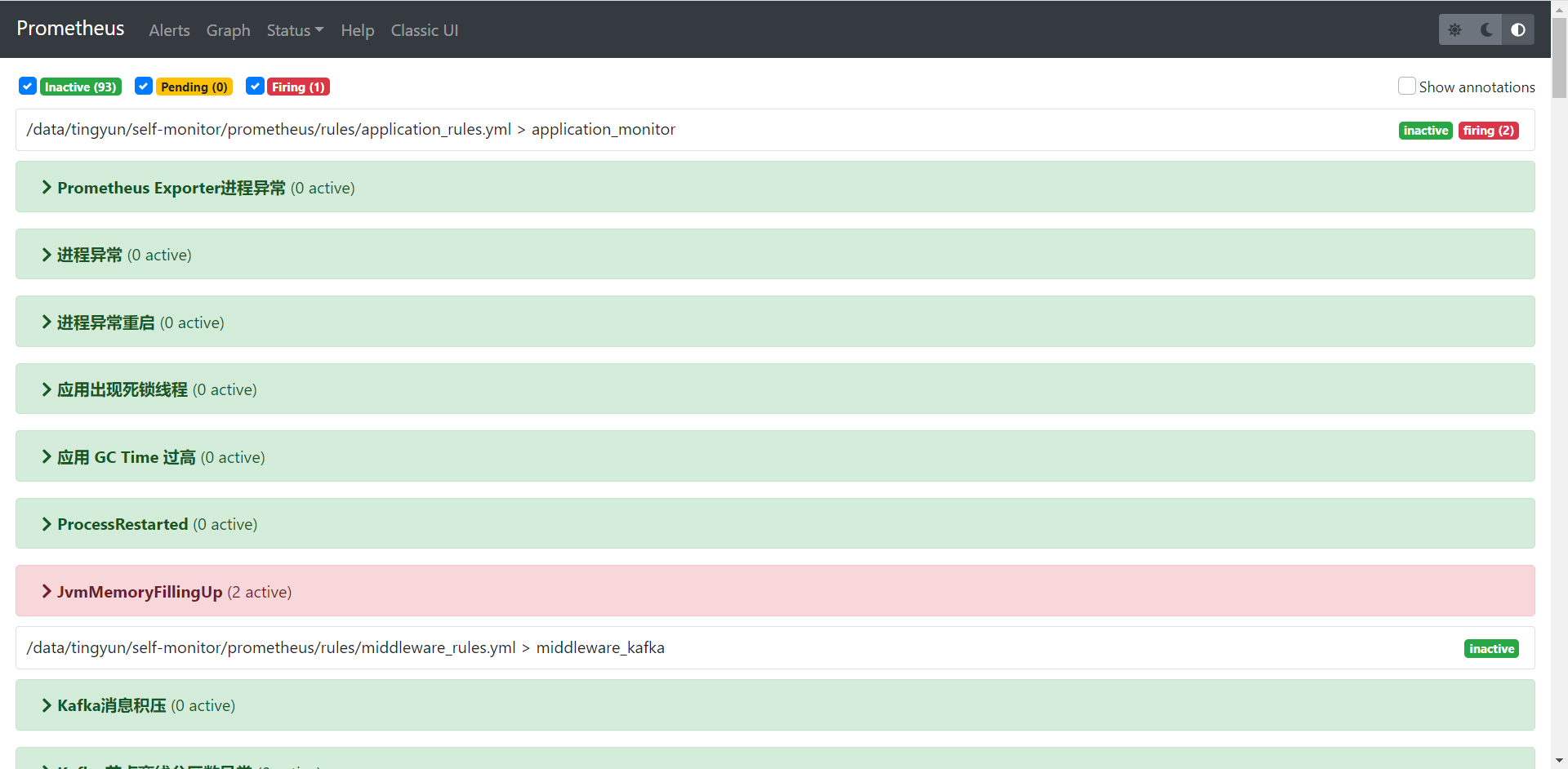
## 日常巡检

平台数据无异常的情况下。日常巡检只需要查看prometheus告警信息即可：<http://$nginx-ip:8080/prometheus>

tingyun/Tingyun2\_18

## 无数据问题排查

针对出现的平台无数据，先通过prometheus判断是否有异常告警。



如果无异常告警信息。可先从kafka消息消费监控判断是生产者问题还是消费者问题。

Middleware Group/Kafka Overview

### messages in为0

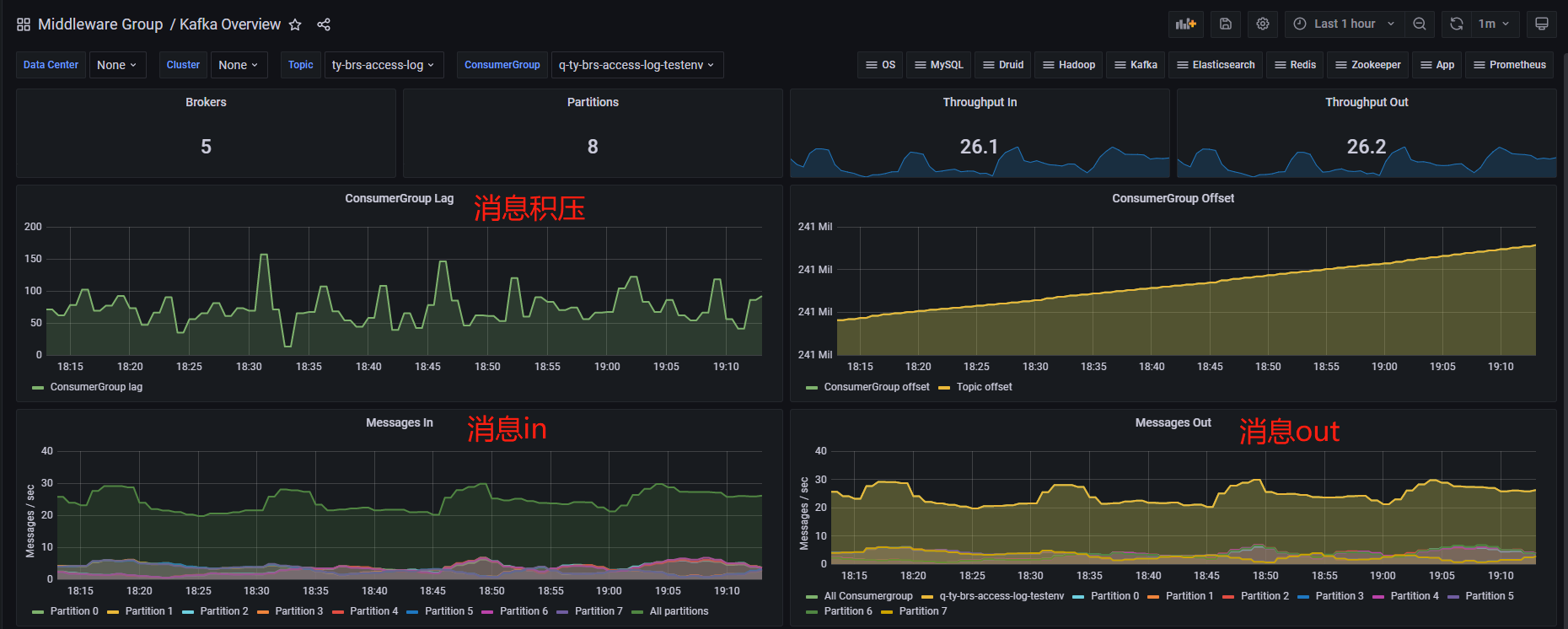
则说明是对应的产品线对应的生产者有问题，apm和app就需要排查对应dc-frontend日志（browser就需要排查openresty），根据access日志判断agent sdk是否有正常数据上传。通过error日志判断是否数据推送或数据处理有问题。

### messages out为0

则排查对应wrap日志判断不消费原因。

### messages out不为0，但明显小于in ，且有积压。

则说明是消费能力不足，需要通过优化wrap内存、线程、或者扩容节点的方式来提高消费能力。



### 各产线重点topic

Apm产品线主要关注topic：q-app-perf q-app-trace，其中健康度问题要关注topic：q-kstream-health-data

App产品线主要关注topic：q-mob-network-request q-mob-network-device

Browser页面产品线主要关注topic：ty-brs-access-log

Mp小程序产品线主要关注topic：ty-mp-access-log

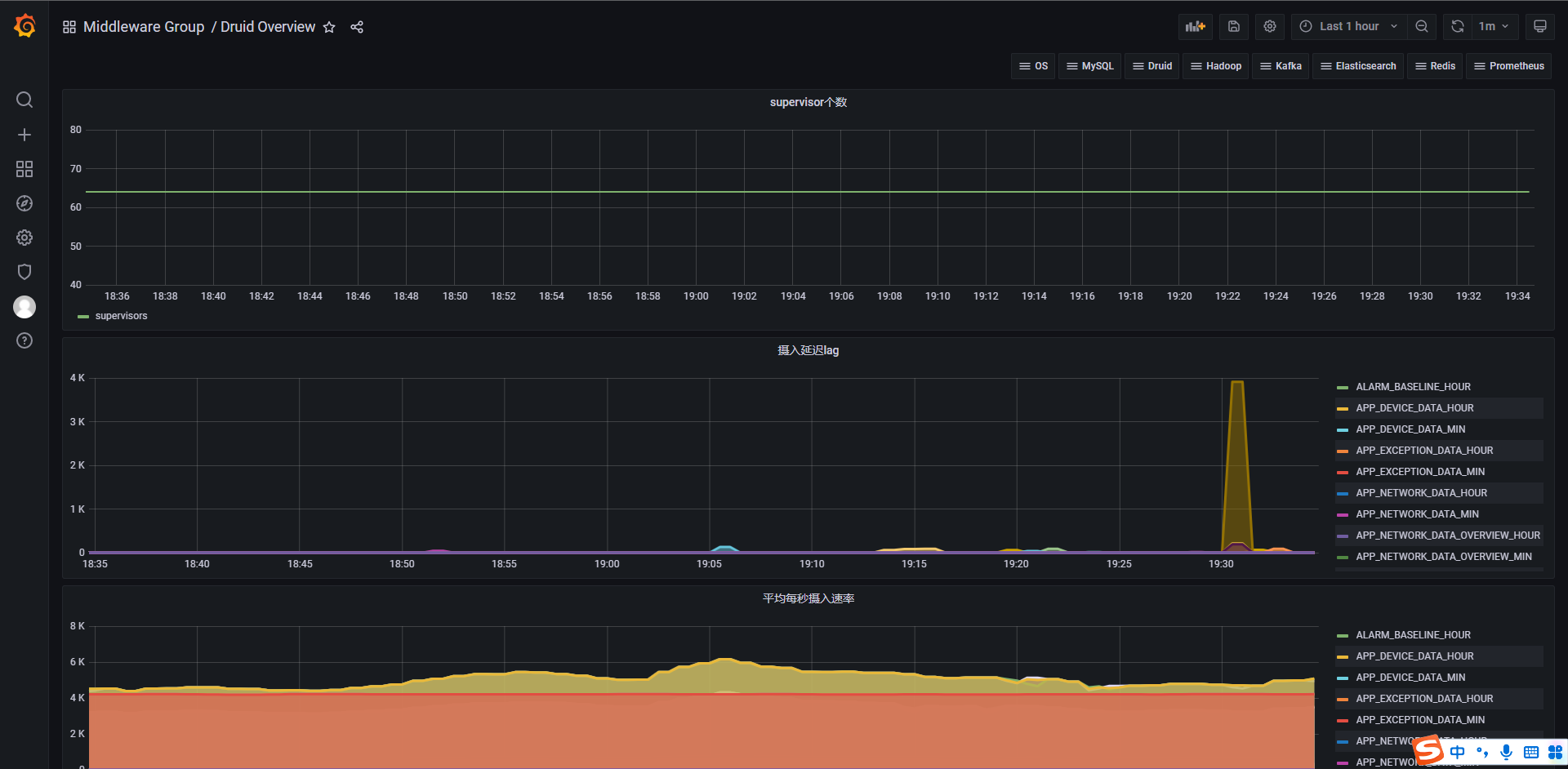
Infra基础组件监控主要关注topic:sys-metrics

Bpi业务分析主要关注：brs-bpi-message svr-bpi-message

### 消费正常

若messages消费in out均正常，切无数据积压。则根据异常的数据类型排查druid或者es情况。

如果是指标类型数据，就通过Middleware Group/Druid Overview判断是否有正常摄入。如果无supervisor，说明druid集群有问题，检查druid集群各节点运行状态。如果延迟较大，且无摄入，则说明是indexer节点异常。可通过druid router页面排查supervisor和task信息。如果有延迟，但摄入也正常，但是摄入速率慢。则说明是druid indexer摄入能力不足。可通过增加task count，减少数据提交周期，优化内存等手段进行优化。建议联系听云运维工程师进行处理。



# 磁盘空间清理

## 组件日志清理

听云自有组件日志均存储再所在主机安装目录 logs目录下，程序自带清理策略，无需手动配置。

## es数据清理

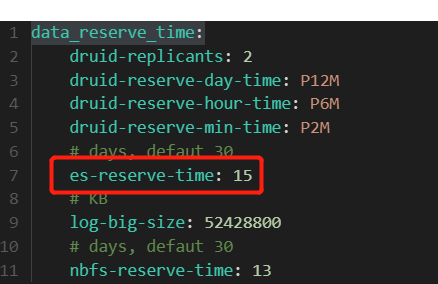
清理脚本：es节点所在机器：/data/tingyun/bin/clean\_log/data\_es\_delete.sh

安装完成后默认已配置crontab定时：

10 \*/1 \* \* \* /data/tingyun/bin/clean\_log/data\_es\_delete.sh

数据保存周期配置：

Nacos:environment.yaml > data\_reserve\_time:es-reserve-time: 15



## nbfs数据清理

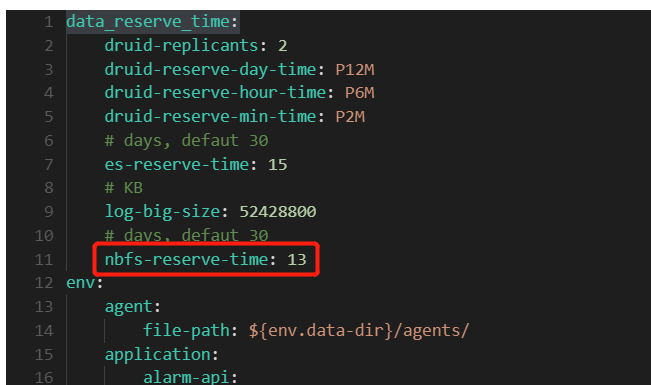
清理脚本：nbfs节点所在机器：/data/tingyun/bin/clean\_log/data\_nbfs\_delete.sh

安装完成后默认已配置crontab定时：

10 \*/1 \* \* \* /data/tingyun/bin/clean\_log/data\_nbfs\_delete.sh

数据保存周期配置：

Nacos:environment.yaml > data\_reserve\_time:nbfs-reserve-time: 15



## Druid/hadoop数据清理

通过druid所在节点脚本进行清理规则提交：

/data/tingyun/bin/clean\_log/data\_druid\_delete.sh

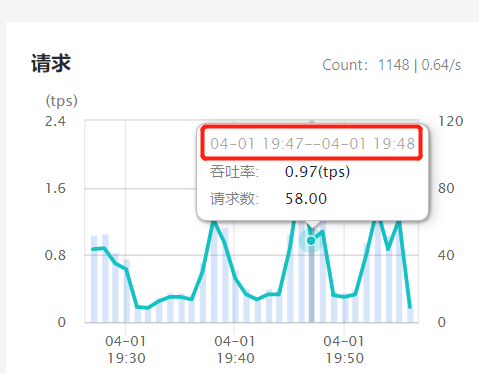
提交后，druid会自动进行清理。

数据保存周期配置：

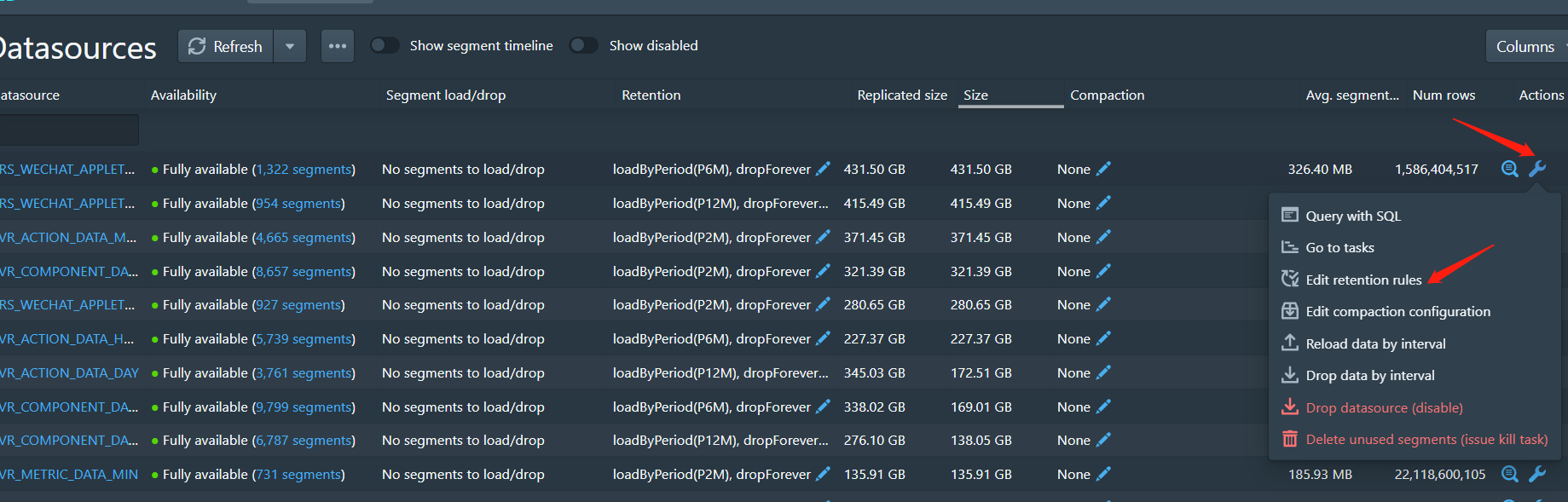
Nacos:environment.yaml

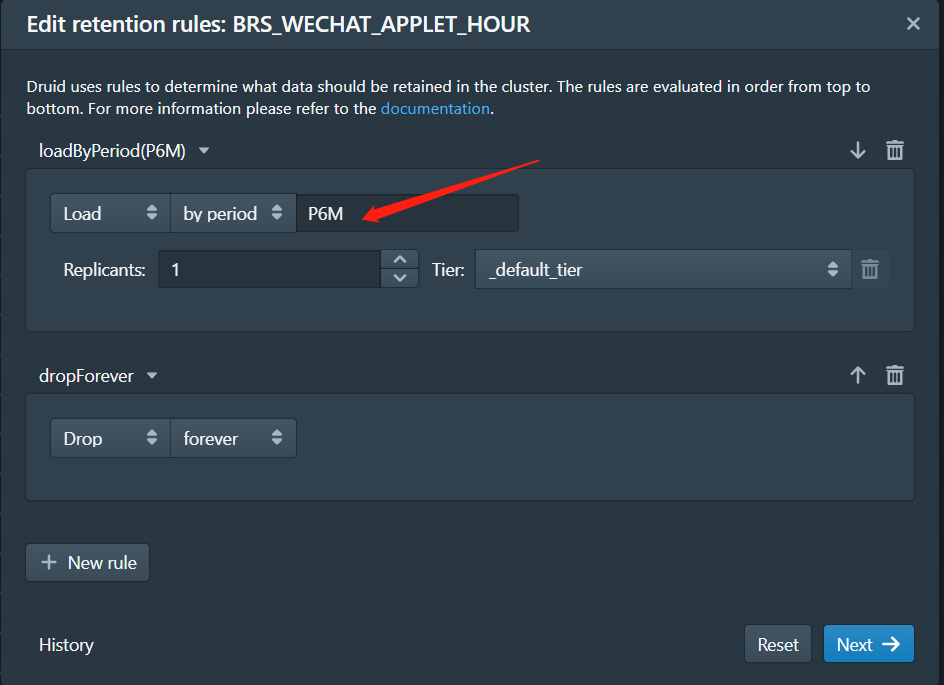


分别为：副本数，天表保存周期、小时表保存周期、分钟表保存周期。（不同的表数据对应的是平台上不同时间颗粒度的数据，比如曲线图颗粒度为1分钟，30分钟均是取自分钟表。1小时 6小时 均是取自小时表。主要是曲线图当个点的颗粒度。不是数据查询周期！！！）



若通过druid-router判断是个别datasource占用空间过大，可在druid-router单独修改个别datasource的数据保存周期。





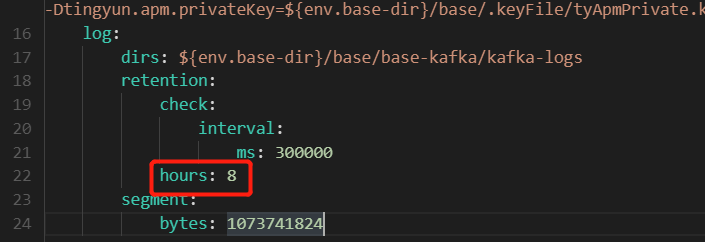
## Kafka数据清理

清理kafka数据要根据消费情况。若经常出现积压，则kafka消息保存周期不能小于消息积压被处理的延迟时间。Kafka消息保存周期默认为8小时，

配置项路径为：

Nacos:base-kafka.yaml:log.retention.hours

！！！修改后，需要重启kafka节点才能生效。



# 组件扩容&迁移

日常个别节点消费能力不足，或内存、磁盘等资源紧张需要进行节点横向扩容或者迁移可参考该章节。

如果是新增机器，注意需要进行相关机器初始化配置（系统参数、/etc/hosts、ssh免密、用户、目录等...），如果是扩容到集群内资源宽松机器，无需初始化。

新增机器除初始化外，需要copy集群原节点如下目录以及文件：

base/base-java base/base-java base/base-nbencoder base/.keyFile

bin/ conf/ optools/ package\_tmp/ self-monitor/ service\_manager.sh tingyun-env.sh

！！！所有组件扩容后，都需要进行维护脚本读取的result.list的调整和自监控调整

## 组件说明



|  |  |  |
| --- | --- | --- |
| 组件名称 | 组件描述 | 扩容方式 |
| base-druid | 基础组件 druid 时序数据库 | 集群，可直接添加新节点（indexer/historical） |
| base-elasticsearch | 基础组件 elasticsearch 搜索数据库 | 集群，可直接添加新节点(datanode) |
| base-hadoop | 基础组件 hadoop 大数据处理/分布式存储 | 集群，可直接添加新节点(datanode) |
| base-kafka | 基础组件 kafka 分布式发布订阅消息系统 | 可直接添加新的集群，涉及配置修改以及相关组件重启 |
| base-mysql | 基础组件 mysql 数据库（配置数据） | 不可扩容 |
| base-nacos-server | 基础组件nacos配置中心服务 | 多活方式，一般无需扩容，若扩容涉及配置修改 |
| base-nginx | 基础组件nginx静态web服务/反向代理服务/负载均衡服务 | 双活，无需扩容 |
| base-redis | 基础组件redis数据缓存服务 | 不可扩容 |
| base-zookeeper | 基础组件zookeeper分布式应用程序协调服务 | 集群，添加新节点后，涉及配置修改以及相关组件重启 |
| common-api-gateway | common组件路由转发服务 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |
| common-auth-api | common组件auth认证服务 | 多活，可直接扩容 |
| common-cluster-server | common组件url聚合等服务 | 多活，可直接扩容 |
| common-es-data-ingestion | common组件es数据入库服务 | 多活，可直接扩容 |
| common-ipdb-server | common组件ipdb服务 | 多活，可直接扩容 |
| common-license-server | common组件授权服务 | 多活，可直接扩容，若原节点迁移，会导致授权失效需要重新申请授权 |
| common-nbfs-server | common组件nbfs服务（类对象存储服务） | 多活，可直接扩容 |
| common-openresty | common组件openresty服务 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |
| common-platform-api | common组件报表平台api | 多活，可直接扩容 |
| common-platform-web | common组件报表静态页面 | 多活，可直接扩容，但只能扩容在nginx服务器，一般无需扩容 |
| common-redirect-server | common组件redirect服务 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |
| common-notification-api | common组件告警推送 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |
| apm-api | apm组件报表查询api | 多活，可直接扩容 |
| apm-config | apm组件报表配置api | 多活，可直接扩容 |
| apm-dc-frontend | apm组件数据接收服务 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |
| apm-dc-wrap | apm组件数据处理服务 | 多活，可直接扩容 |
| apm-health | apm组件健康度服务 | 多活，可直接扩容 |
| apm-web | apm组件报表静态页面 | 多活，可直接扩容 |
| app-api | app组件报表查询api | 多活，可直接扩容，但只能扩容在nginx服务器，一般无需扩容 |
| app-dc-frontend | app组件数据接收服务 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |
| app-dc-wrap | app组件数据处理服务 | 多活，可直接扩容 |
| app-deobfuscator-dubbo | app组件提供dubbo服务 | 多活，可直接扩容 |
| app-statistics-elasticjob | app组件提供dubbo服务 | 多活，可直接扩容 |
| app-url-clustering-dubbo | app组件提供dubbo服务 | 多活，可直接扩容 |
| app-web | app组件报表静态页面 | 多活，可直接扩容，但只能扩容在nginx服务器，一般无需扩容 |
| browser-api | browser组件报表查询api | 多活，可直接扩容 |
| browser-url-normalize | browser组件url解析服务 | 多活，可直接扩容 |
| browser-web | browser组件报表静态页面 | 多活，可直接扩容，但只能扩容在nginx服务器，一般无需扩容 |
| browser-wrap | browser组件数据处理服 | 多活，可直接扩容 |
| infra-api | infra组件报表查询api | 多活，可直接扩容 |
| infra-dc-wrap | infra组件数据接收处理服务 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |
| infra-metrics-correlator-aggregation | infra组件 | 多活，可直接扩容 |
| infra-metrics-correlator-compression | infra组件 | 多活，可直接扩容 |
| infra-web | infra组件报表静态页面 | 多活，可直接扩容，但只能扩容在nginx服务器，一般无需扩容 |
| alarm-api | alarm组件报表查询api | 多活，可直接扩容 |
| alarm-baseline | alarm组件后端基线服务 | 多活，可直接扩容 |
| alarm-dc | alarm组件后端数据处理服务 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |
| alarm-timeseries-training | alarm组件后端时间序列训练服务 | 多活，可直接扩容 |
| alarm-web | alarm组件报表静态页面 | 多活，可直接扩容，但只能扩容在nginx服务器，一般无需扩容 |
| mp-web | mp组件报表静态页面 | 多活，可直接扩容，但只能扩容在nginx服务器，一般无需扩容 |
| mp-api | mp组件报表查询api | 多活，可直接扩容 |
| mp-wrap | mp组件数据处理服务 | 多活，可直接扩容 |
| mp-url-normalize-service | mp组件提供dubbo服务 | 多活，可直接扩容 |
| mp-url-normalize | mp组件提供dubbo服务 | 多活，可直接扩容 |
| visboard-sense-mock | visboard组件 | 多活，可直接扩容 |
| visboard-sense-api | visboard组件报表查询api | 多活，可直接扩容 |
| visboard-dashboard-api | visboard组件报表查询api | 多活，可直接扩容 |
| visboard-data-report-api | visboard组件报表查询api | 多活，可直接扩容 |
| visboard-sense-web | visboard组件报表静态页面 | 多活，可直接扩容，但只能扩容在nginx服务器，一般无需扩容 |
| bpi-dc-wrap | bpi组件后端数据处理服务 | 多活，可直接扩容 |
| bpi-api | bpi组件报表查询api | 多活，可直接扩容 |
| bpi-task | bpi组件后端数据处理服务 | 多活，可直接扩容 |
| bpi-grafana | bpi组件报表dashboard | 多活，可直接扩容 |
| bpi-adhoc-store | bpi组件后端数据处理服务 | 多活，可直接扩容 |
| bpi-adhoc-olap | bpi组件后端数据处理服务 | 多活，可直接扩容 |
| neuralert-api | neuralert组件报表查询api | 多活，可直接扩容 |
| neuralert-dc | neuralert组件数据接收处理服务 | 多活，可直接扩容，扩容后，需要新增前端nginx upstream节点 |

## 可直接扩容的自有组件举例

以apm-dc-wrap为例：

打包现有组件目录:

cd /data/tingyun/apm/

tar -zcvf apm-dc-wrap.tgz apm-dc-wrap

推送到新节点并解压到 /data/tingyun/apm/目录(若无，需要手动创建)下即可使用该节点的/data/tingyun/bin/launch.sh 脚本进行启停。

## 需要修改配置的组件扩容举例

以apm-dc-frontend为例，安装上述apm-dc-wrap方式扩容节点后，需要修改base-nginx/conf/nginx.conf中的apm-dc-frontend对应的upstream



## es节点扩容

将原es节点除base/base-elasticsearch下除了data log外的其他全部文件推送到新节点base/base-elasticsearch目录下。

新es节点/data/tingyun/conf/wukong-local.conf文件中（若无，则新增）添加如下配置：其中name设置为新节点主机名称。Master设置为false，data设置为true。然后启动即可。



## Common-nbfs-server节点扩容

和直接扩容spring项目方式外，多了一步/data/tingyun/conf/wukong-local.conf的配置：id递增，bucket-name的最后数字和id一致即可。



## Druid节点扩容

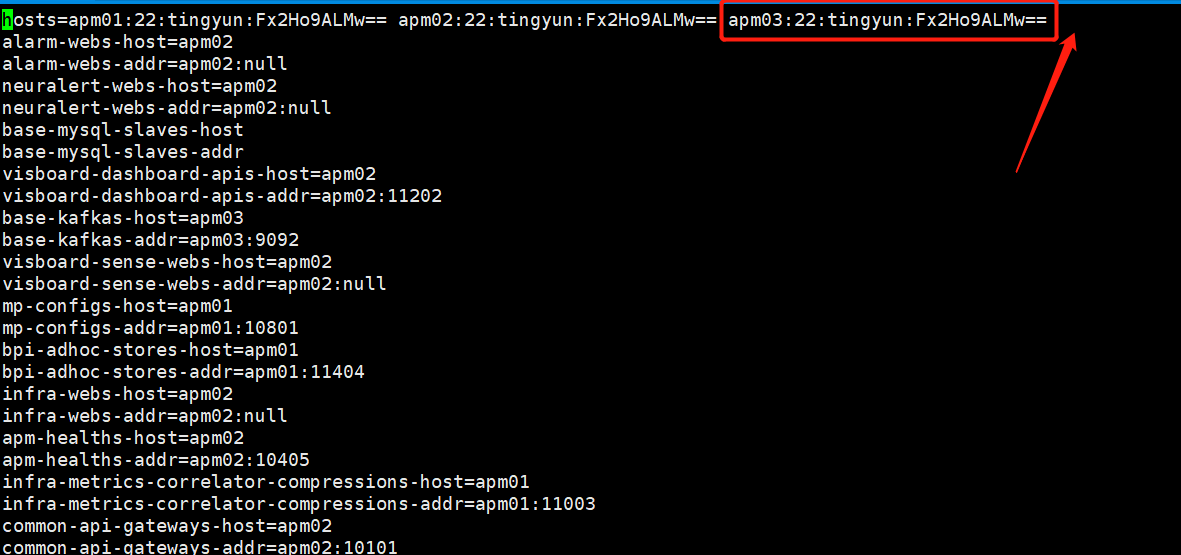
Druid节点一般只需要扩容indexer和historical节点。

Copy原druid节点base/base-druid目录下除了var log目录外的全部文件到新的机器。然后使用base/base-druid/launch.sh start/stop/status indexer historical即可维护该节点的服务。

## result.list调整

扩容的节点，都需要调整/data/tingyun/self-monitor/hsttool/resule.list

如果涉及到新增机器首先新增host列表：



然后找到对应组件的host和addr按照原格式 新增节点的信息即可。主节点修改后，可通过ssh推送到其他全部节点。

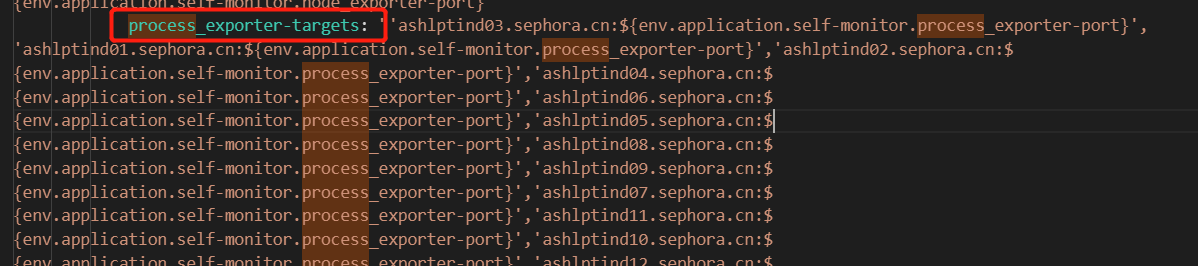
## 自监控调整

新增节点需要在Nacos self-monitor.yaml中添加对应的jvmtargets即可：



如果涉及到新增机器，要注意node\_expoter和process\_exporter的节点信息：



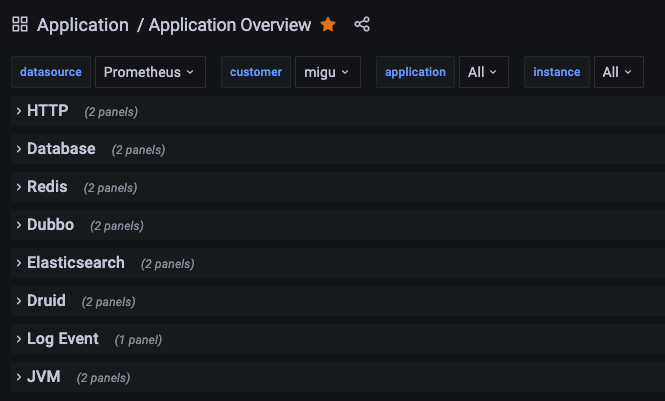


# 附件：Grafana监控指标详解

## 应用整体维度视图

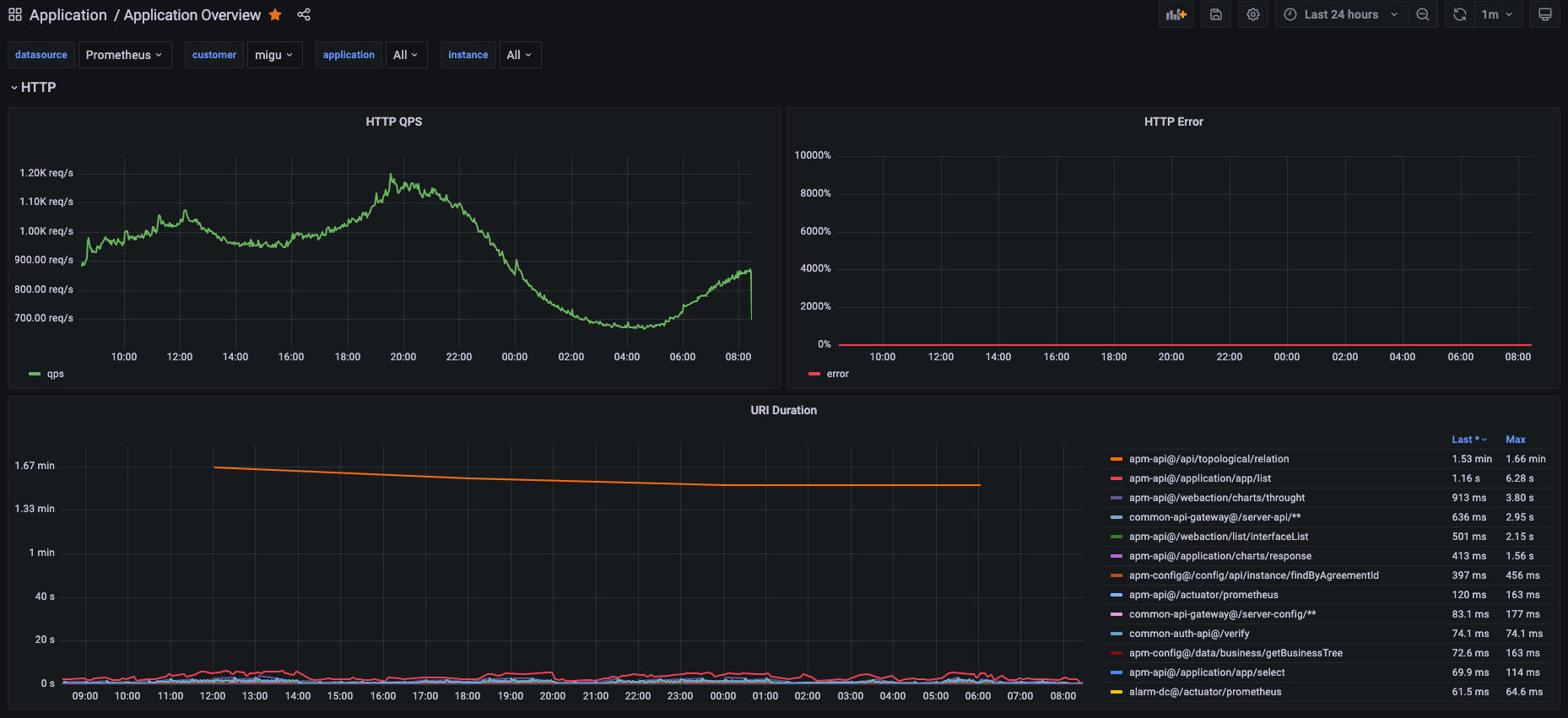
### Application Overview概述

展示应用整体维度以下类型数据。同时也支持按照应用、实例维度进行过滤查询。



### HTTP视图

展示Springboot Tomcat/Netty/Jetty HTTP接口的每秒钟平均请求次数以及错误率。

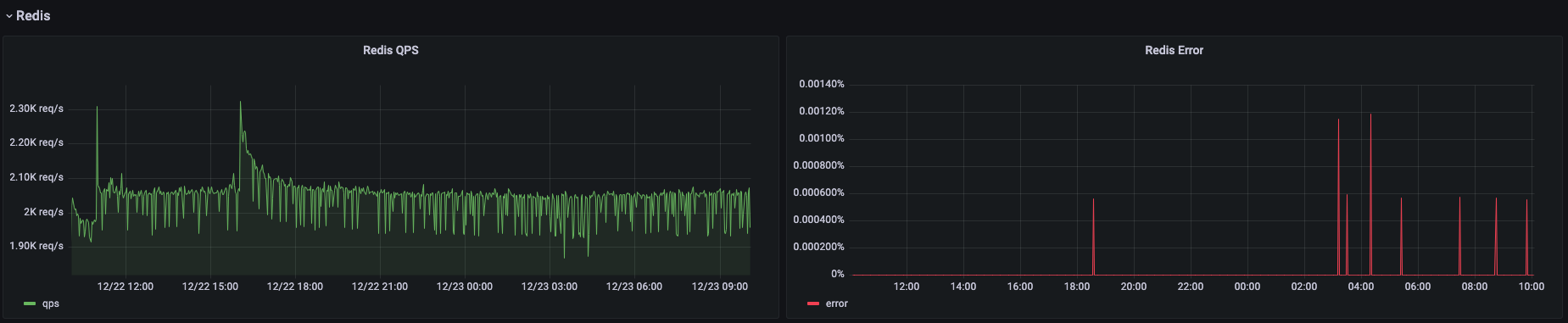


### Database 视图

展示Springboot JPA/JDBC Template/JDA 组件访问数据库请求次数以及错误率。



### Redis 视图



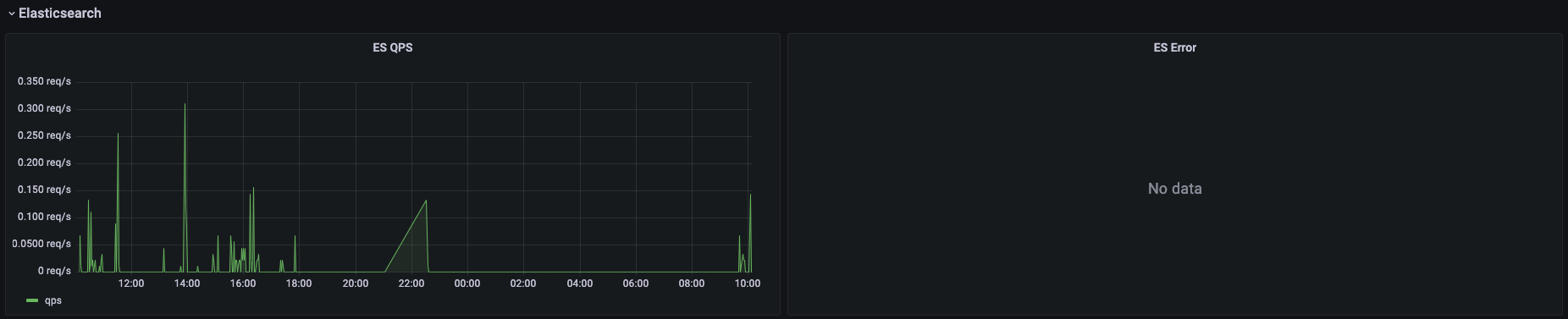
### Dubbo 视图

展示dubbo组件的调用次数以及错误率。



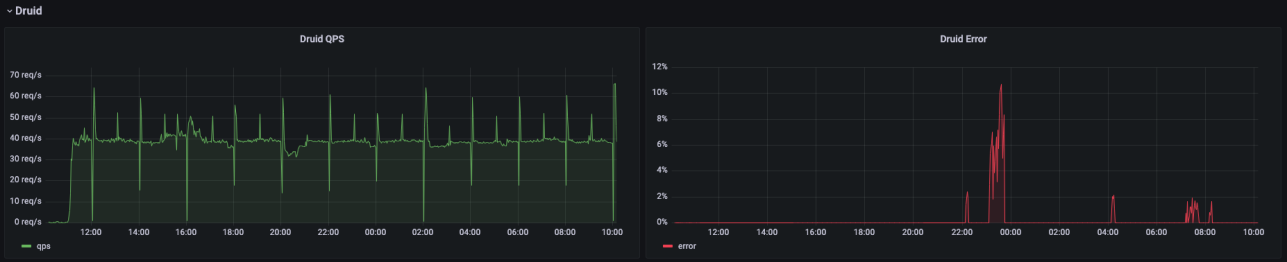
### Elasticsearch 视图

展示 report tingyun-grpah-es 组件请求 ES 的次数与错误率。（不支持 ES HighLevelClient 组件）



### Druid 视图

展示 report tingyun-grpah-druid 组件请求 druid 的次数与错误率。不支持其它方式，如SQL方式。



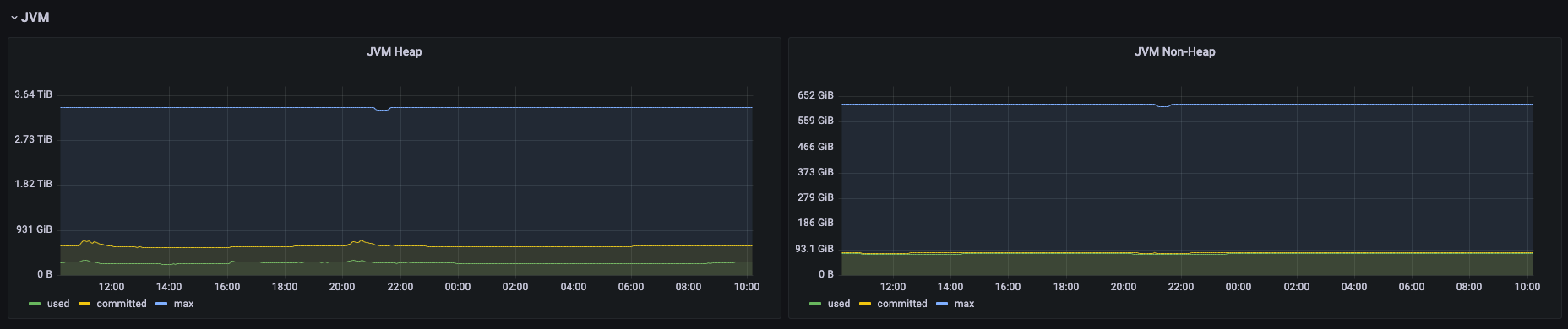
### Log Event 视图

展示Springboot Log 打印次数，支持按照 Level 维度分开展示。



### JVM 视图

展示JVM内存占用情况。



## 应用侧组件视图

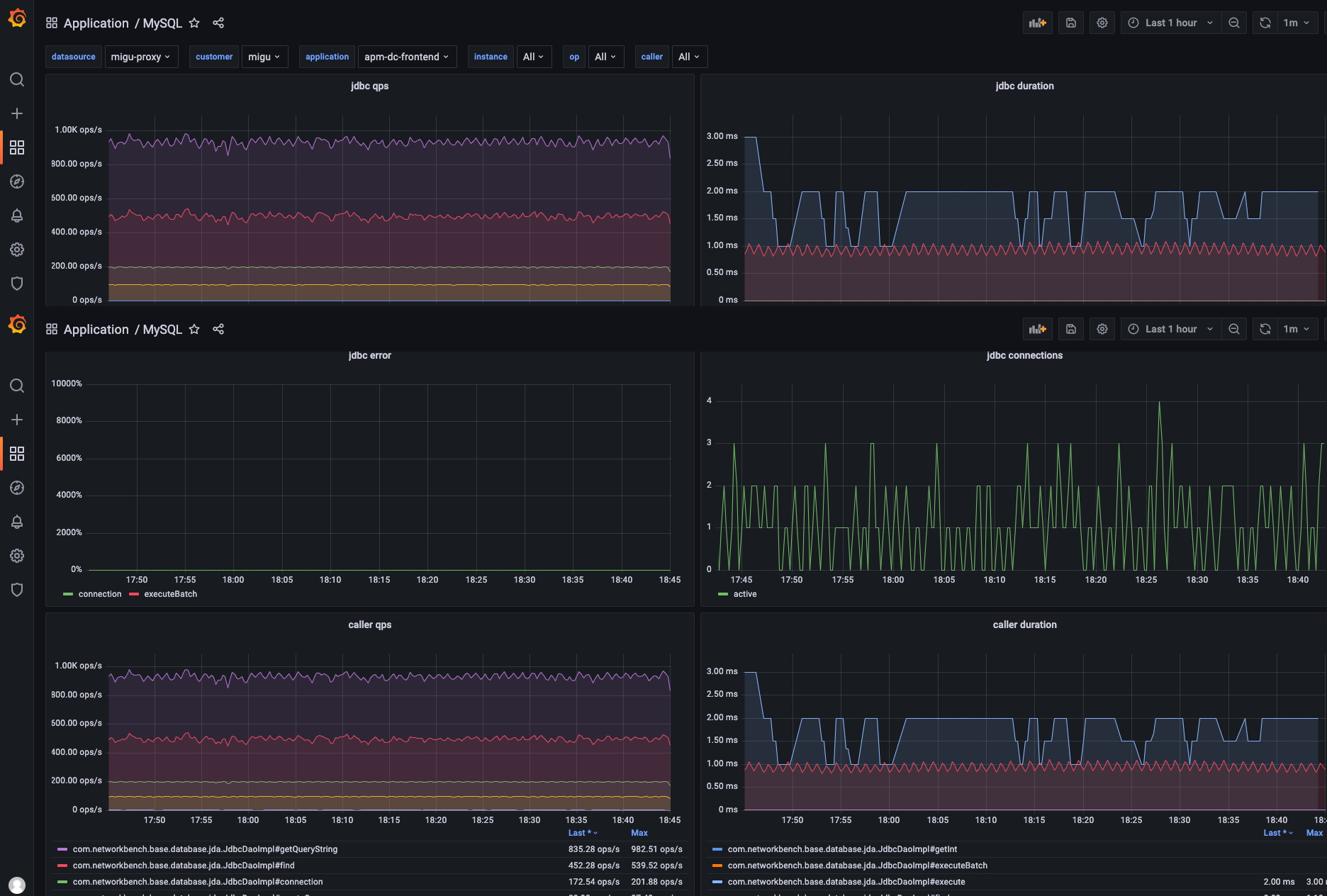
### HTTP Request

支持展示 URI 级别的请求次数，响应时间，错误率。以及 Tomcat / Jetty 并发线程数。



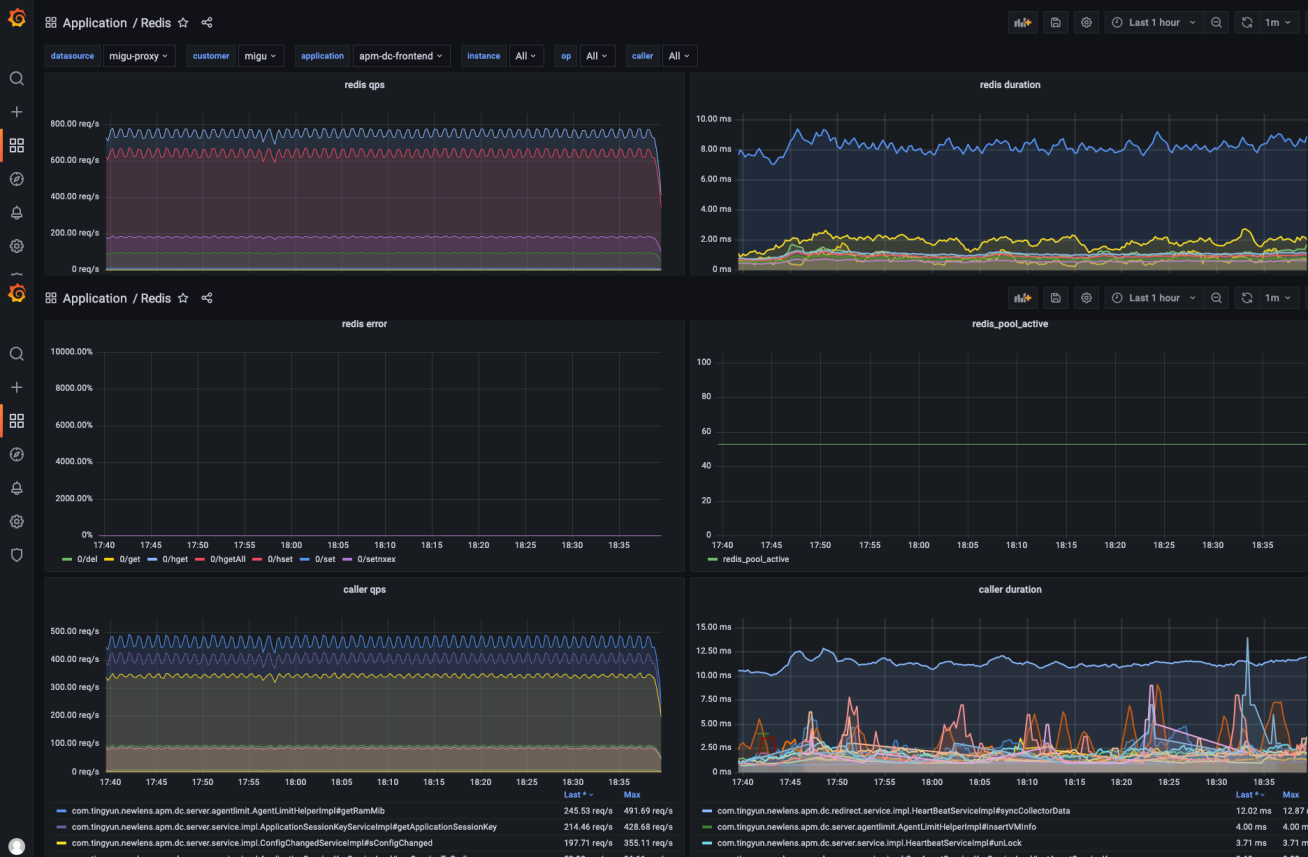
### MySQL

支持展示应用程序 method 级别请求MySQL的请求次数，响应时间，错误率，并发请求数。其中 method 定义为：JPA/JDBC/JDA 调用数据库请求的业务逻辑方法，如 execute，findObject 等。



### Redis

支持展示应用程序 method 级别请求 Redis 的请求次数，响应时间，错误率，并发请求数。其中method定义为 RedsiTemplate/RedisClient 调用redis server的业务逻辑方法，如 get，set，hget 等。



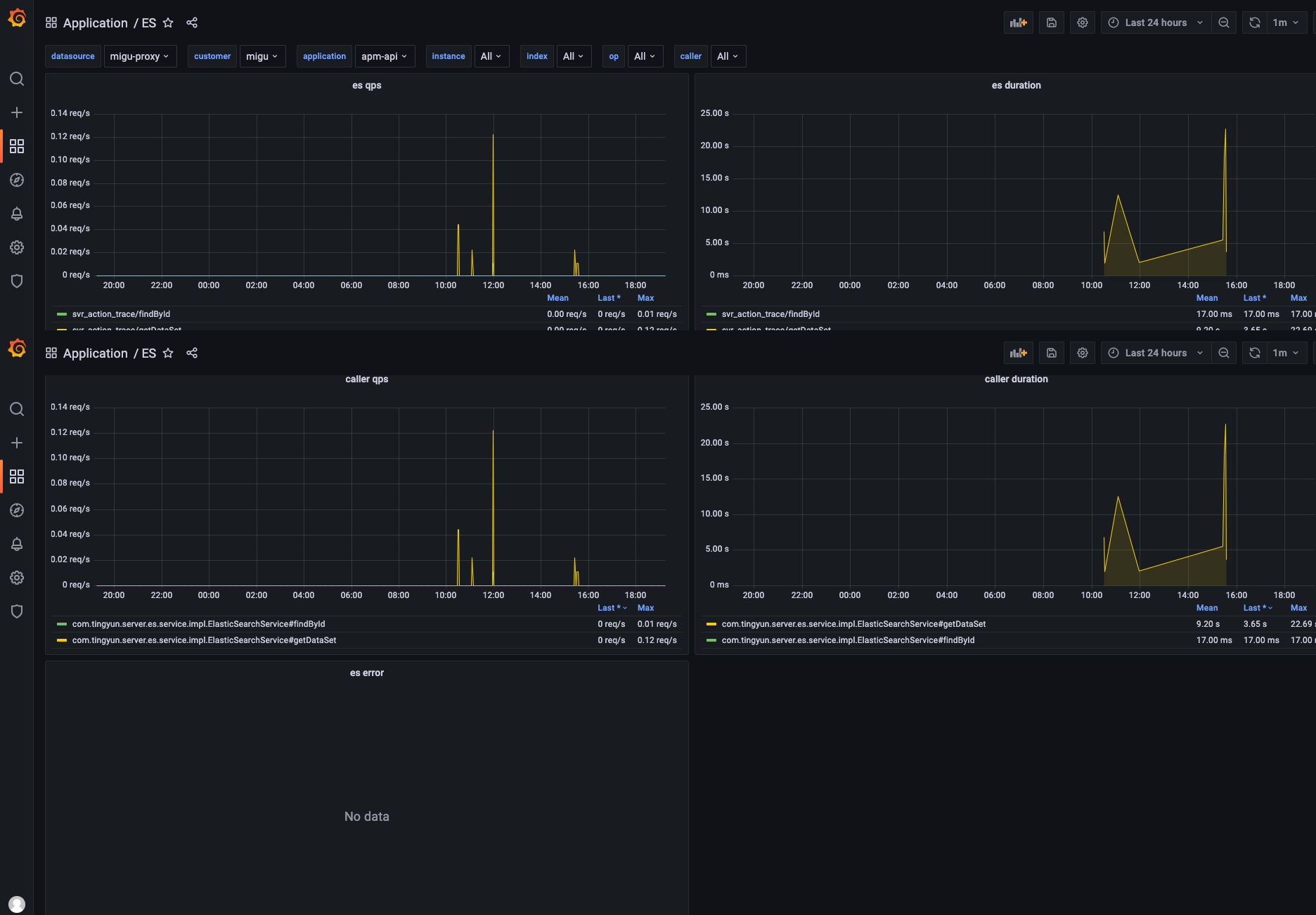
### Dubbo

支持展示 dubbo method 级别请求次数，响应时间，错误率，并发请求次数。其中 method 定义为 dubbo 接口的业务逻辑方法。



### ES

支持展示 index 维度 report tingyun-graph-es 访问ES的请求次数，响应时间，错误率。



### Druid

支持展示 table 维度 report tingyun-graph-druid 访问Druid的请求次数，响应时间，错误率。



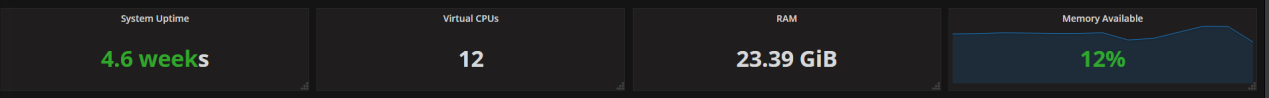
## 操作系统维度视图

### 操作系统监控指标详解

众所周知，操作系统层面的监控是监控的基本监控，许多的故障和问题都发生在操作系统本身上，如cpu负载过高，内存不够等。接下来我们会对系统层面的所有指标逐一进行说明。

### 系统摘要

主要包括机器启动时间、cpu核心数、内存总容量和可用内存容量。



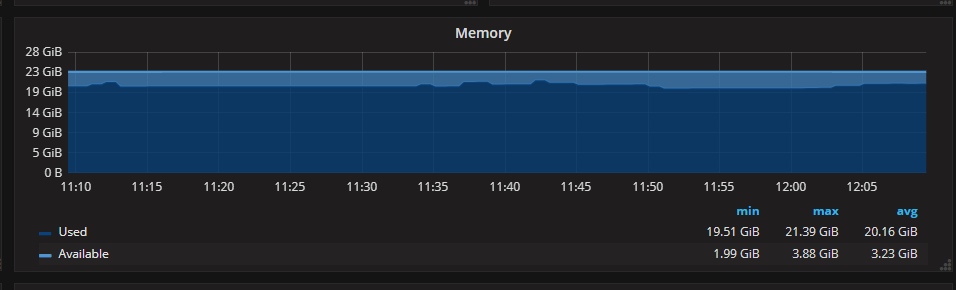
### CPU负载

用于展示cpu的负载、注意观察负载是否有超过核心数，如负载较高，需要扩容CPU或者进行应用迁移。



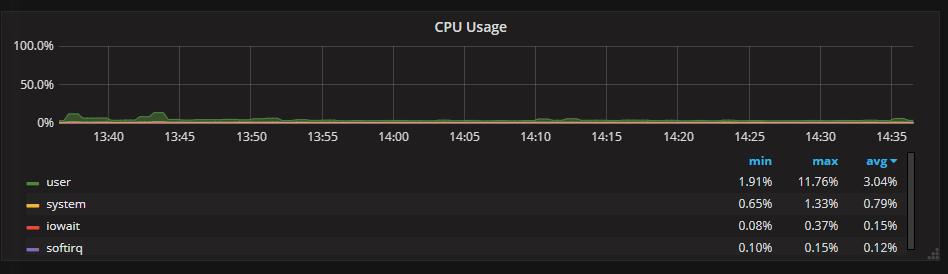
### 内存使用

用于展示已用和可用内存，建议可用内存至少2G



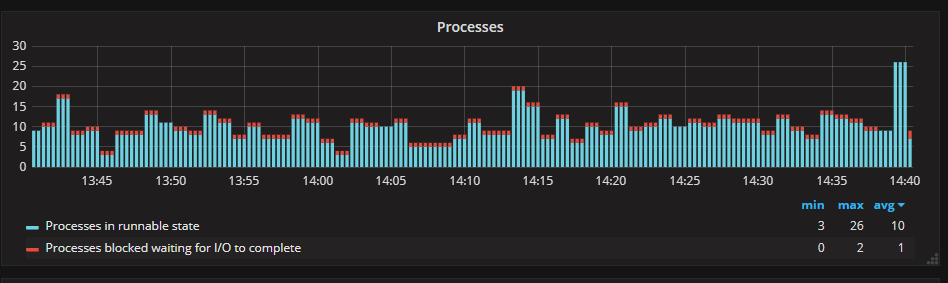
### CPU使用率

展示user、syster、iowait、softirq cpu占比，建议全部加起来不超过60%



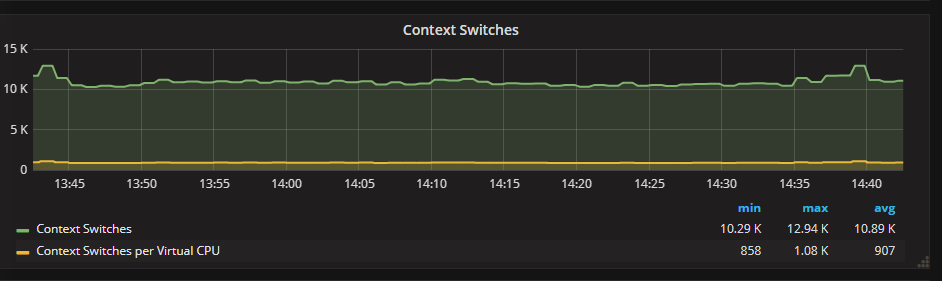
### 进程信息

展示当前处于Runnable状态的进程以及阻塞等待的进程数，需要注意是否有突然暴涨或暴跌。



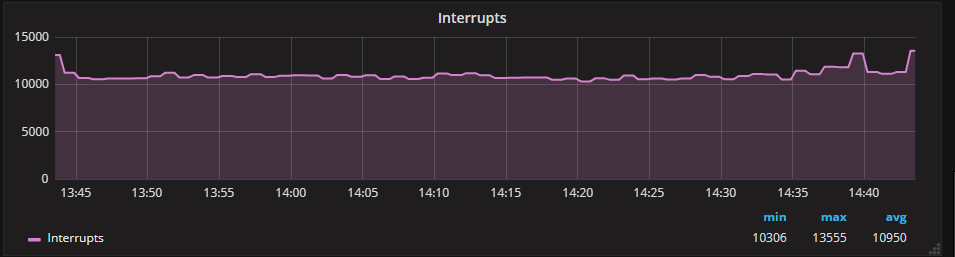
### 上下文切换

用于展示上下文切换次数，需注意是否有暴涨或暴跌。



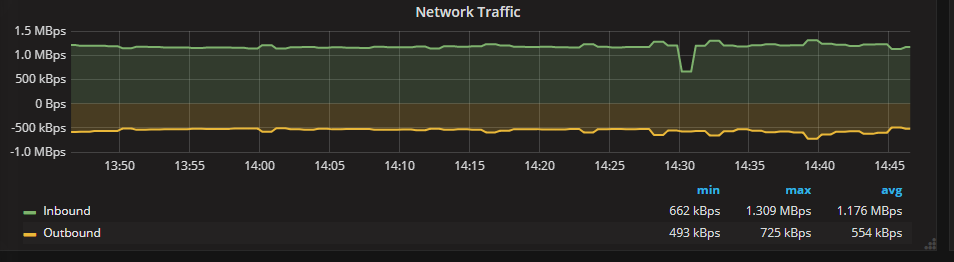
### 中断次数

用于展示系统中断次数，需注意是否有暴涨或暴跌。



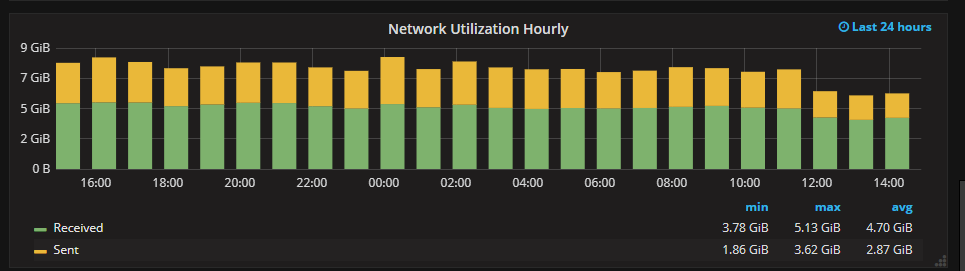
### 网络吞吐量

用于展示出入的网络吞吐量，注意不要超过网卡和交换机的上限，如当前大部分为千兆网卡，则网络吞吐量建议不要超过600Mbps



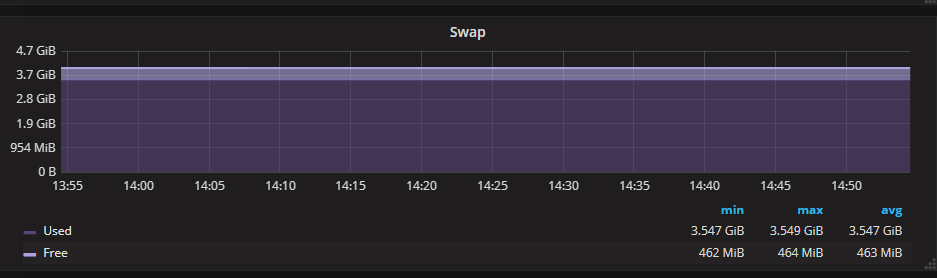
### 网卡每小时的收发量

用于展示网卡每小时的收发量，此数据与网络吞吐量正相关，注意如有暴涨暴跌要分析其原因是否合理。



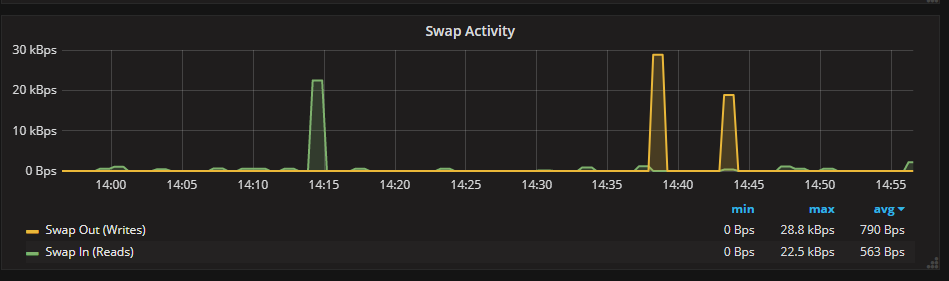
### SWAP的使用

用于展示swap的使用，因为Linux系统会尽量使用内存，偶尔使用swap，所以使用一点swap没关系，但是如果使用较多，占比超过60%,建议对内存进行扩容。



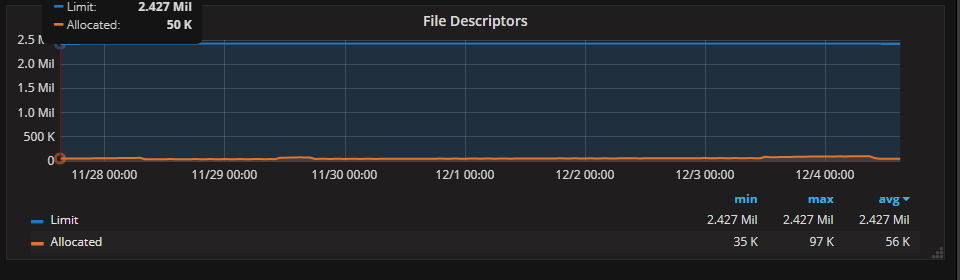
### SWAP的使用活跃度

用于展示swap换进换出的频率，此指标和SWAP使用空间对应着看，如换进换出频繁，且占比超过60%建议对内存进行扩容。如换进换出不频繁有可能是某一时刻swap使用量过多，可以考虑将SWAP内容清掉再进行观察。



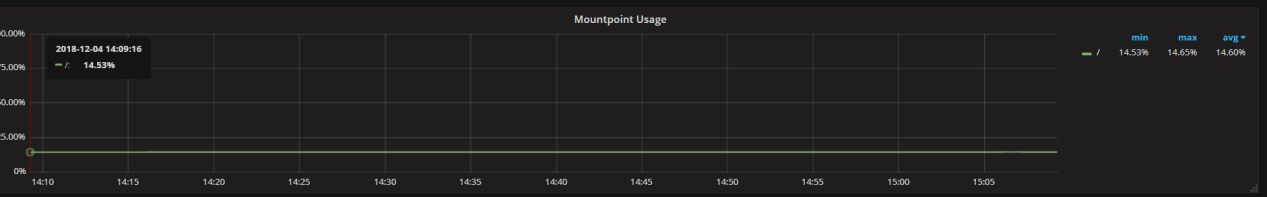
### 打开文件数

用于展示系统的最大打开文件数和当前打开文件数，注意如果使用率超过60%需要调整系统参数或迁移部分应用。

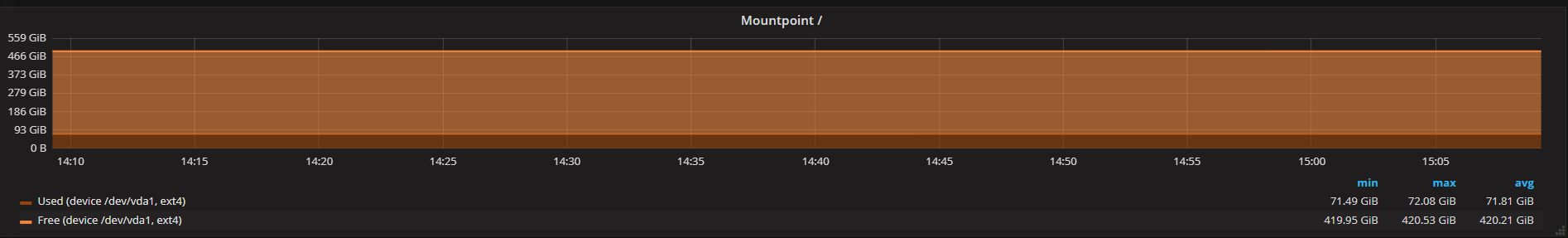


### 挂载点磁盘空间

用于展示挂载点磁盘空间的使用情况，超过80%需要清理磁盘或进行磁盘扩容

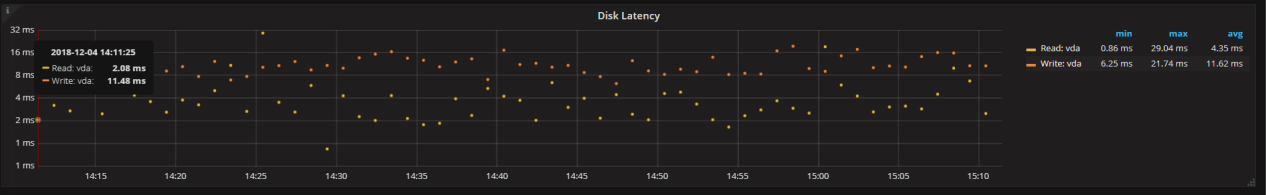


下图可展示各挂载点的磁盘的具体容量信息



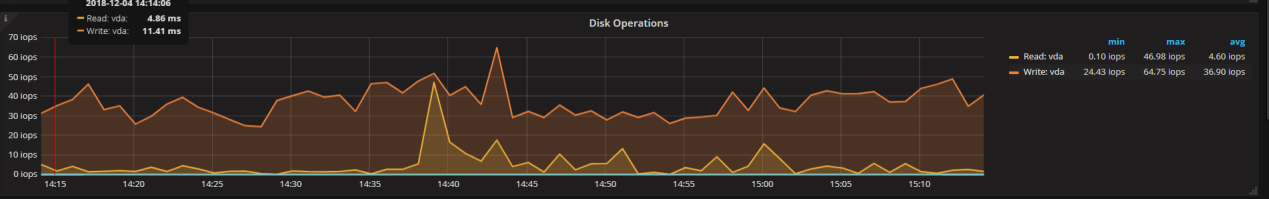
### 磁盘延迟

用于展示磁盘读写的延迟，注意如果有延迟大幅增高，需要确认是否为业务正常



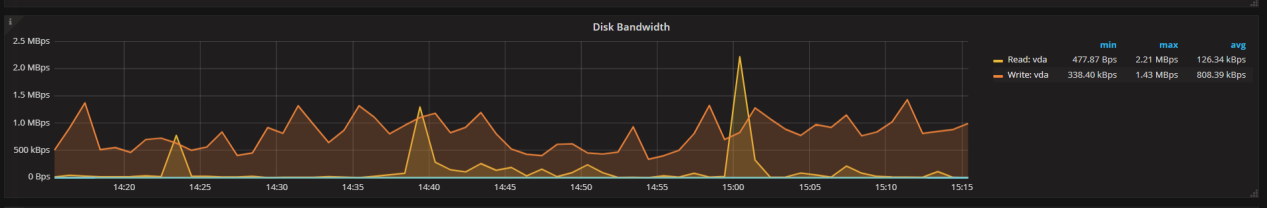
### 磁盘IOPS

用于展示各磁盘读写IOPS，可用于查看磁盘读写频率，如有暴涨或暴跌需注意是否异常，此指标要结合吞吐和io util来判断磁盘是否存在瓶颈



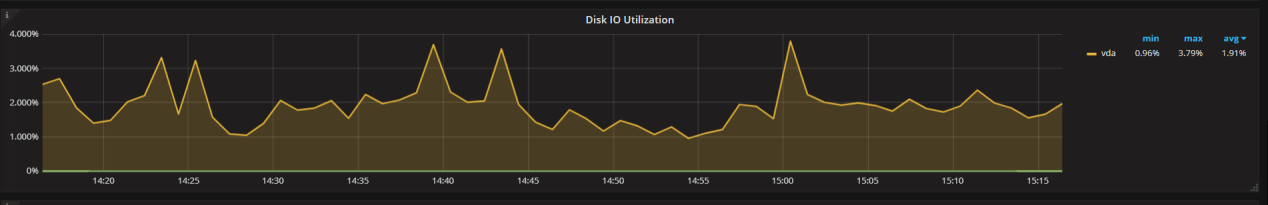
### 磁盘吞吐

用于展示各磁盘的读写吞吐量，如果有暴涨或暴跌需注意是否异常，此指标要结婚iops和io uti来判断是否磁盘存在瓶颈。



### IO util

用于展示各磁盘的繁忙程度，如超过60%且iops不高，磁盘吞吐也不高则说明磁盘读写存在严重瓶颈，建议升级更快的磁盘。注意顺序读写会大量占用磁盘吞吐，离散读写会大量占用IOPS。

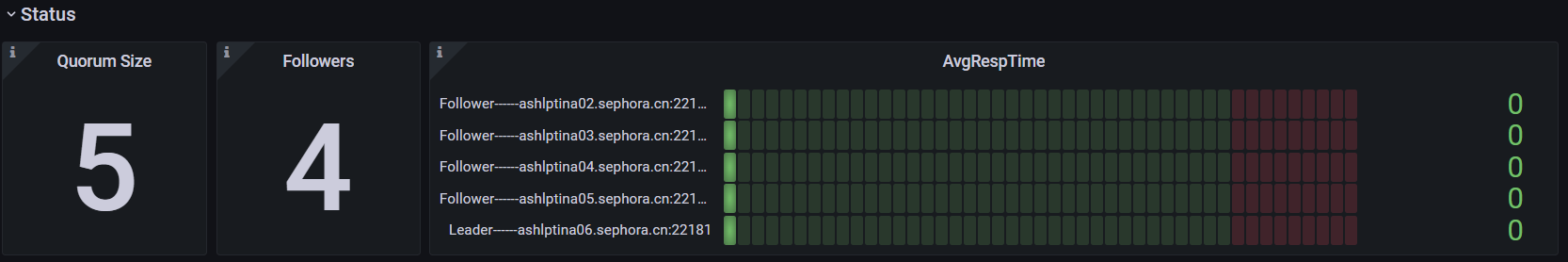


## 组件监控

### Zookeeper 监控

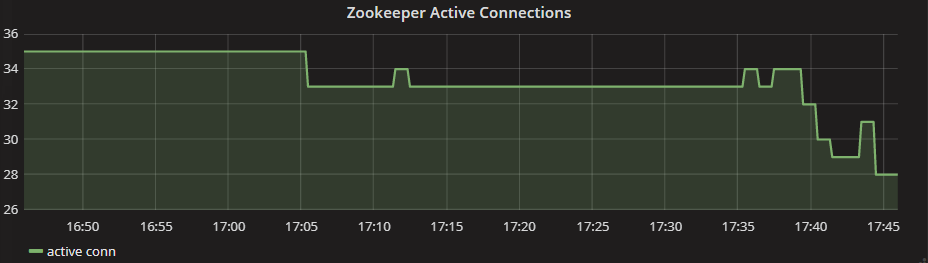
**节点角色**

用于展示Zookeeper节点类型，Leeader或Follower，注意每个节点都要看下，正常情况应只有一个Leader，剩下均为Follower



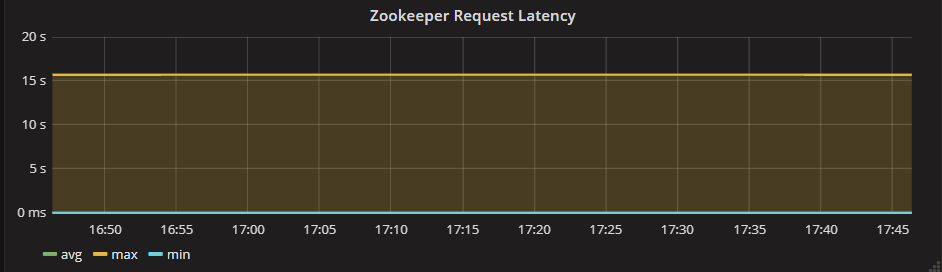
**活跃连接数**

可查看当前集群的活跃连接数，注意是否有暴涨和暴跌，从而确认业务是否正常



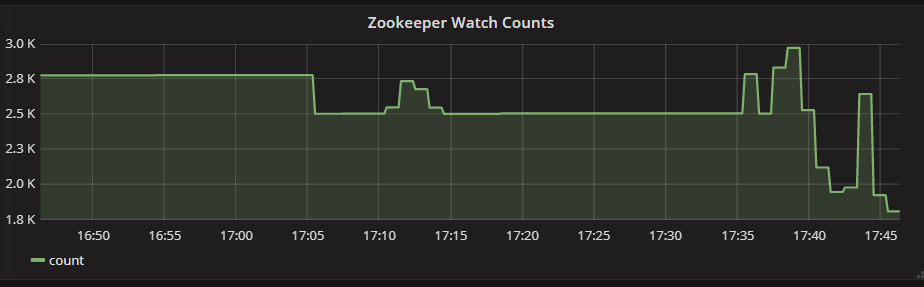
**请求延迟**

可查看请求的平均延时，如暴涨暴跌需查看集群状态是否正常，集群内网络是否正常



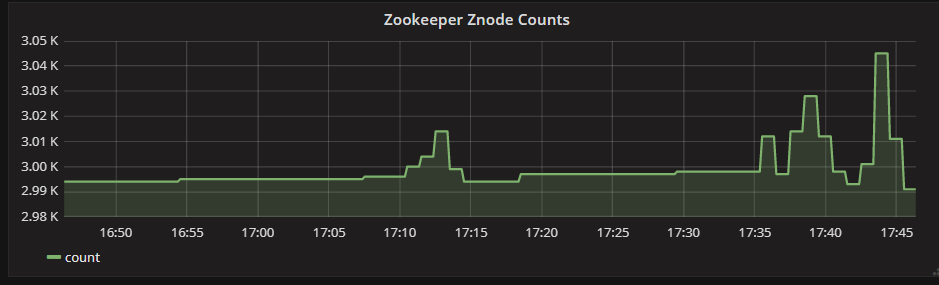
**Watch Count数**

**可查看watch事件数量**



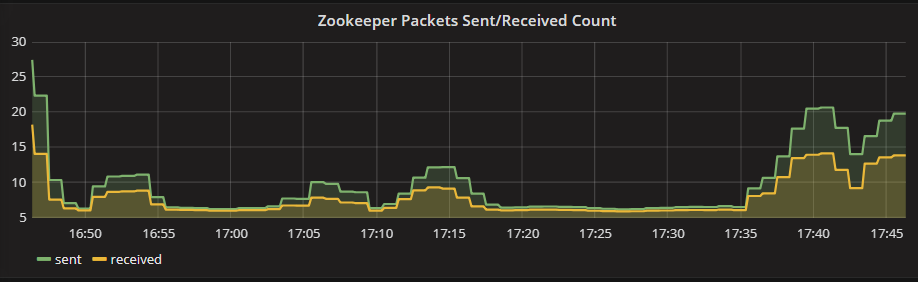
**Znode Count 数**

可用来查看znode 数量



**收发包数**

可查看zookeeper每秒收发包的数量，如暴涨暴跌需确认应用是否有异常



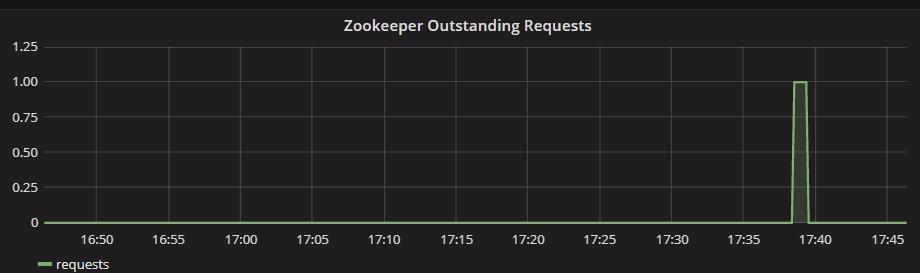
**数据大小**

可查看zookeeper储存的文件大小，用以评估是否需要扩容



**排队请求数量**

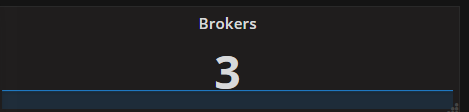
可查看排队请求的数量，用以评估zookeeper是否有性能瓶颈



### Kafka监控

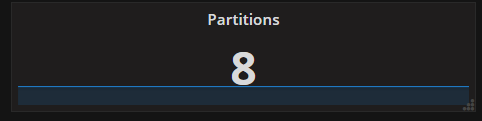
**Broker数量**

查看Broker的数量，确认是否有掉线节点



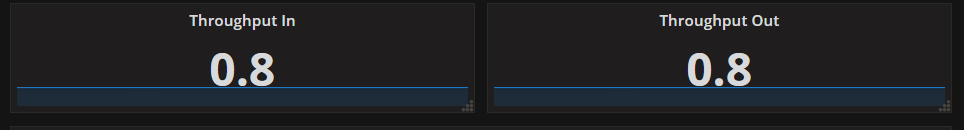
**Topic的Partittion数量**

查看某Topic的Partition数量



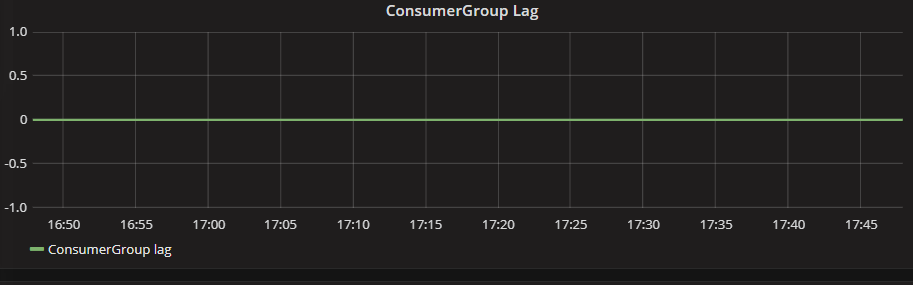
**进出消息数**

查看某个ConsumperGroup下某个Topic的进出消息个数；如果队列没积压，则两个数值应大概相等，如队列有积压，则出消息数应大于进消息数，否则队列会越积越多。



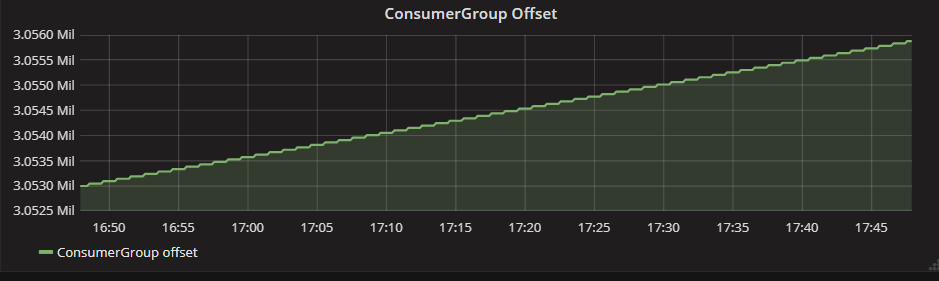
**ConsumerGroup Lag值**

查看某ConsumerGroup 积压的数量，稳定在某一个区间则没问题，如一直增长说明消费端处理不过来，需要进行排查



**ConsumerGroup Offset值**

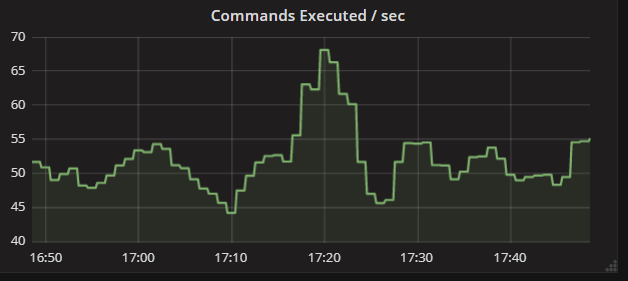
查看某ConsumerGroup Offset值，如后端一直有消费，则应一直增长，如不增长说明后端消费者已经无消费，需要检查消费者应用。



### Redis 监控

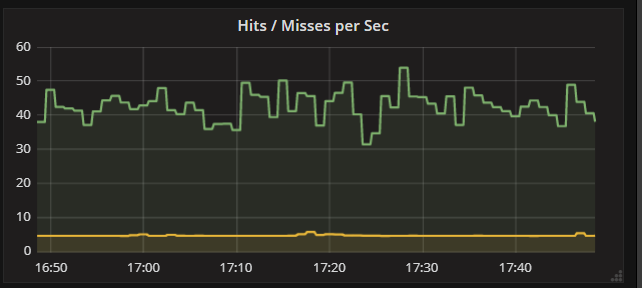
**每秒请求数**

Redis每秒请求数，用来作为Redis性能的基线数据，一般2万以下不会有问题



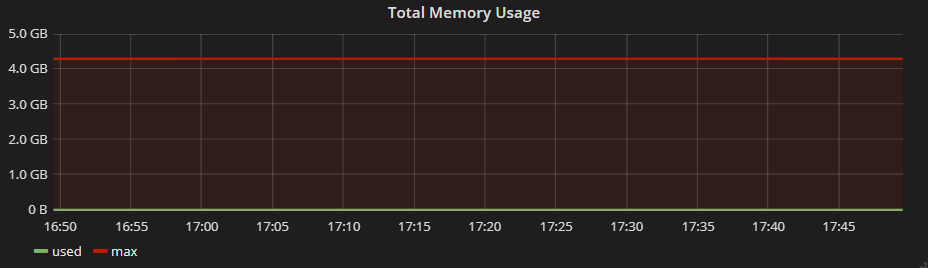
**每秒命中/未命中次数**

**用于展示命中和未命中次数，可用于查看Redis的缓存效果**



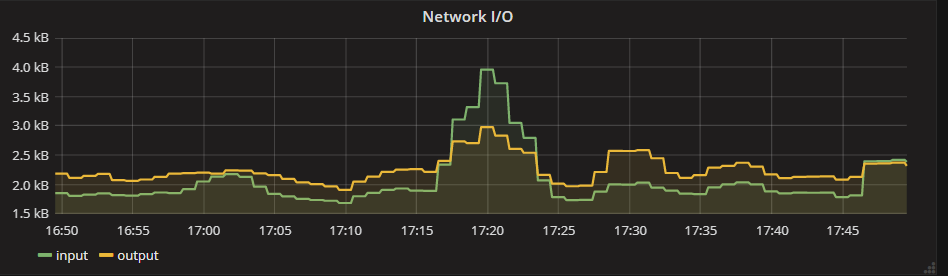
**Redis内存使用率**

查看Redis的内存使用情况，判断是否需要扩容，一般来说4G以下不会有性能瓶颈



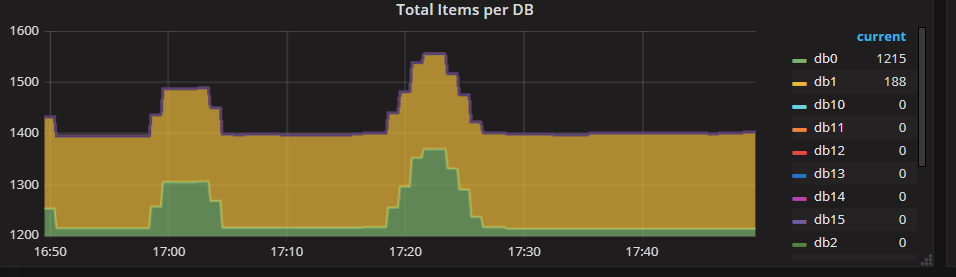
**Input/Outpu网络流量**

查看Reids网络流量，注意该流量不要超过机器的网卡带宽



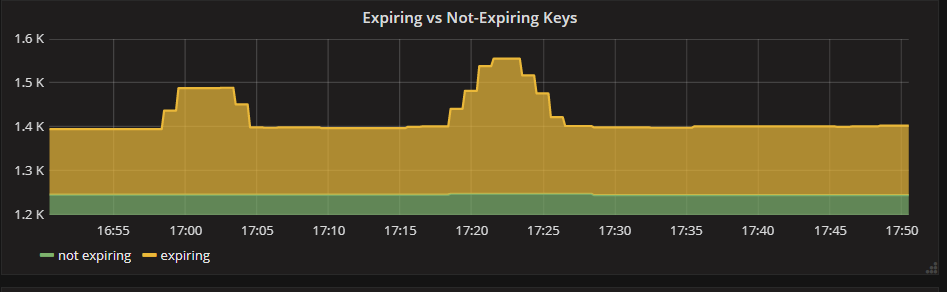
**每个DB key的数量**

查看每个DB key的数量



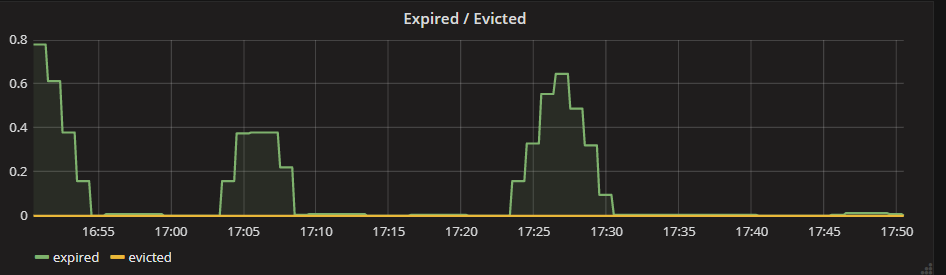
**Expiring/Not-Expiring key的数量**

查看有过期和无过期 key的数量



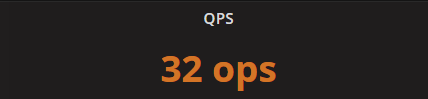
**每秒Expired的数量**

查看每秒过期key的数量



### MySQL监控

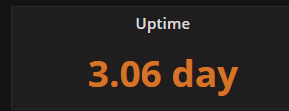
**每秒查询数**



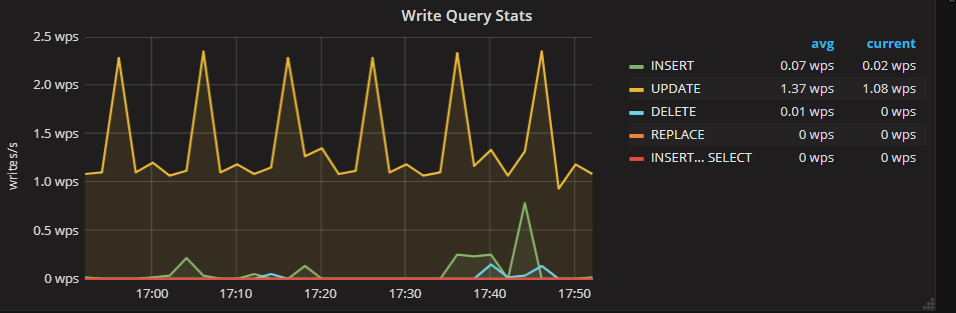
**客户端连接数**



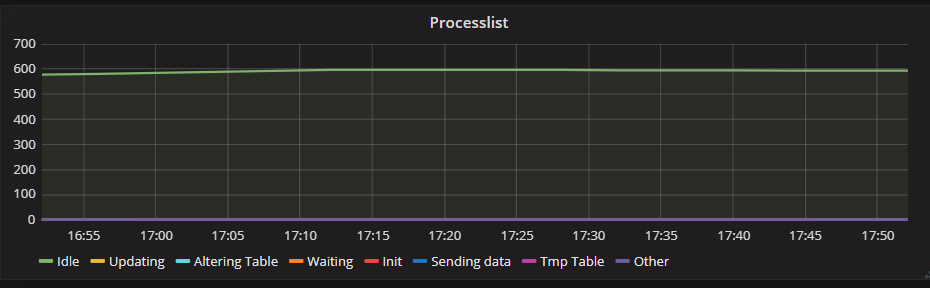
**启动时间**



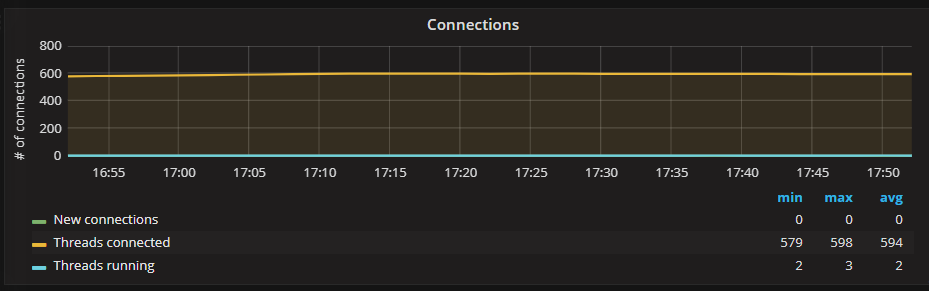
**每秒钟insert/update/delete数**



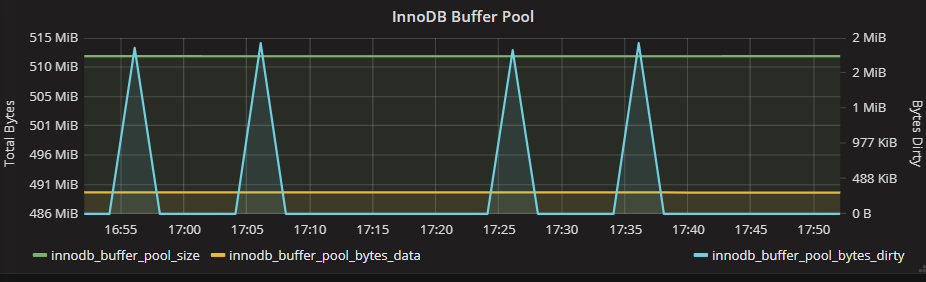
**Proceslist信息**



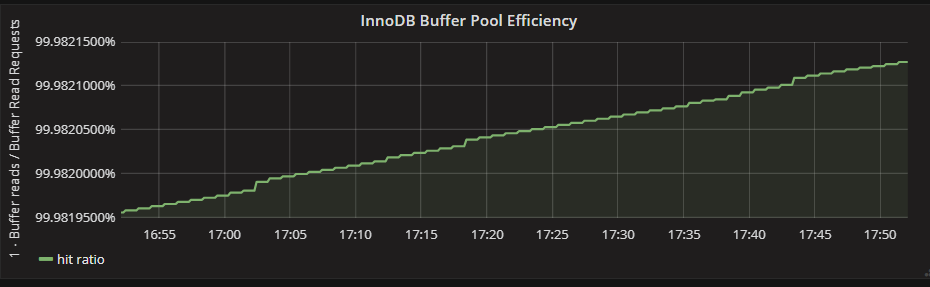
**Connection信息**



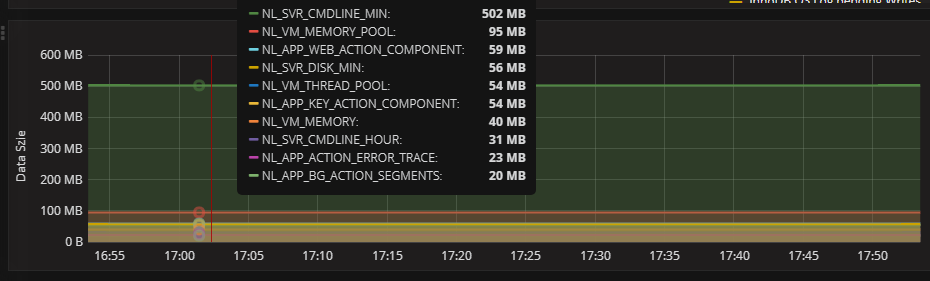
**InnoDB Buffer Pool使用信息**



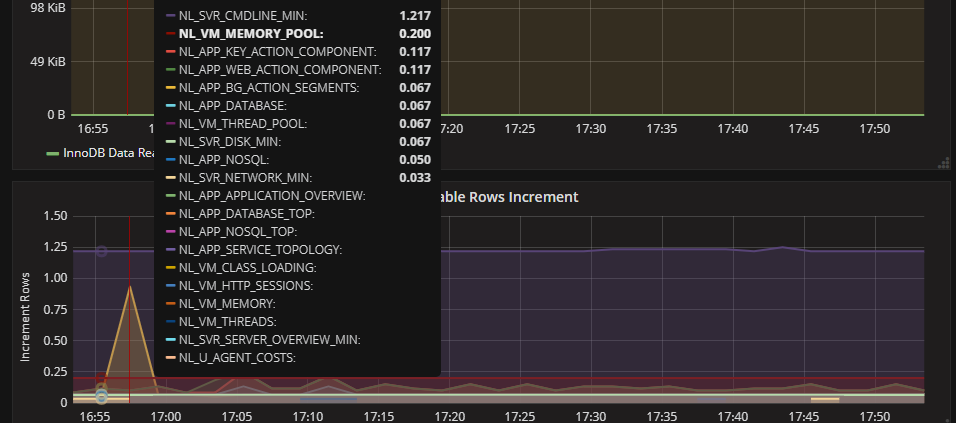
**InnoDB Buffer Pool命中率**



**TOP 10表大小**



**TOP 10表行数每秒增量**



**从库Slave IO Running状态**

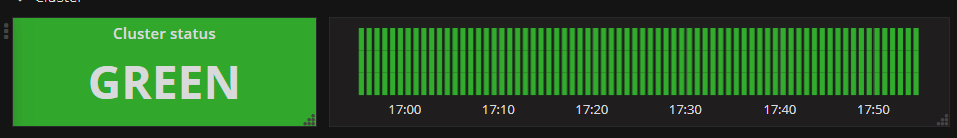
**从库Slave Sql Runing 状态**

**从库Seconds behind Master**

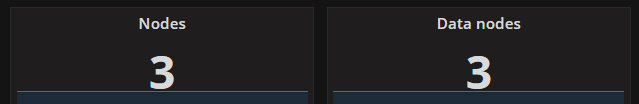
**从库Skip Counter**

### ElasticSearch 监控

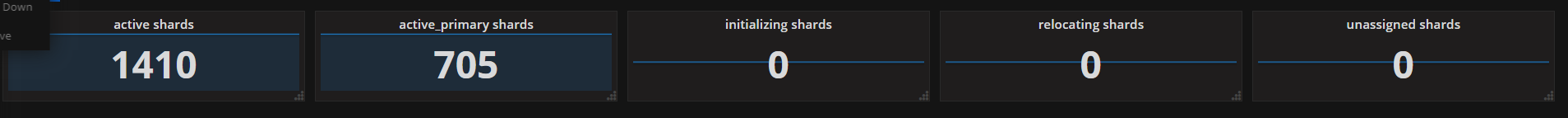
**集群健康状态**



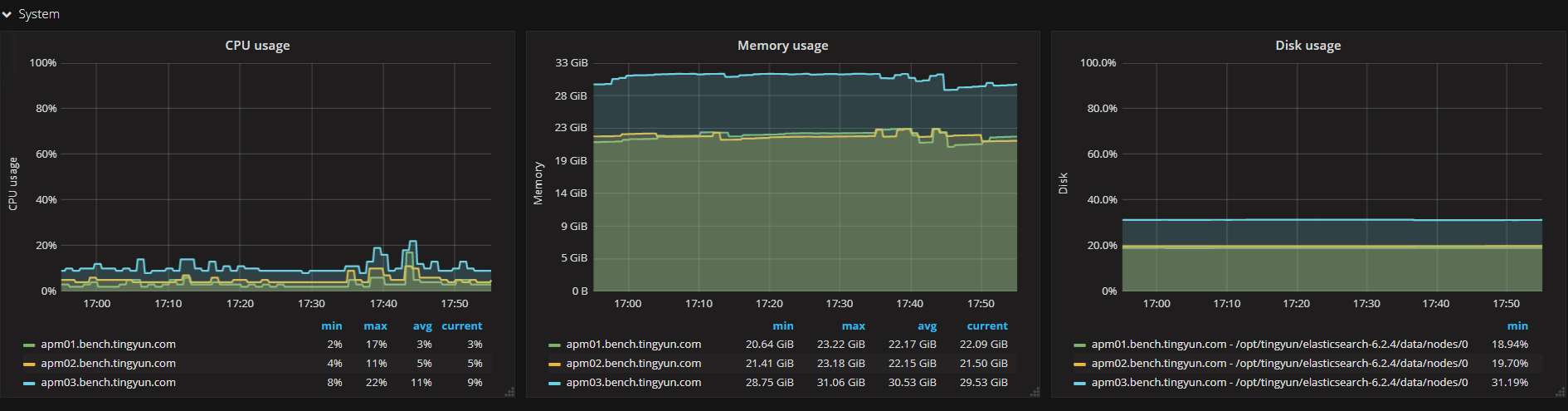
**集群节点数信息**



**shard信息**

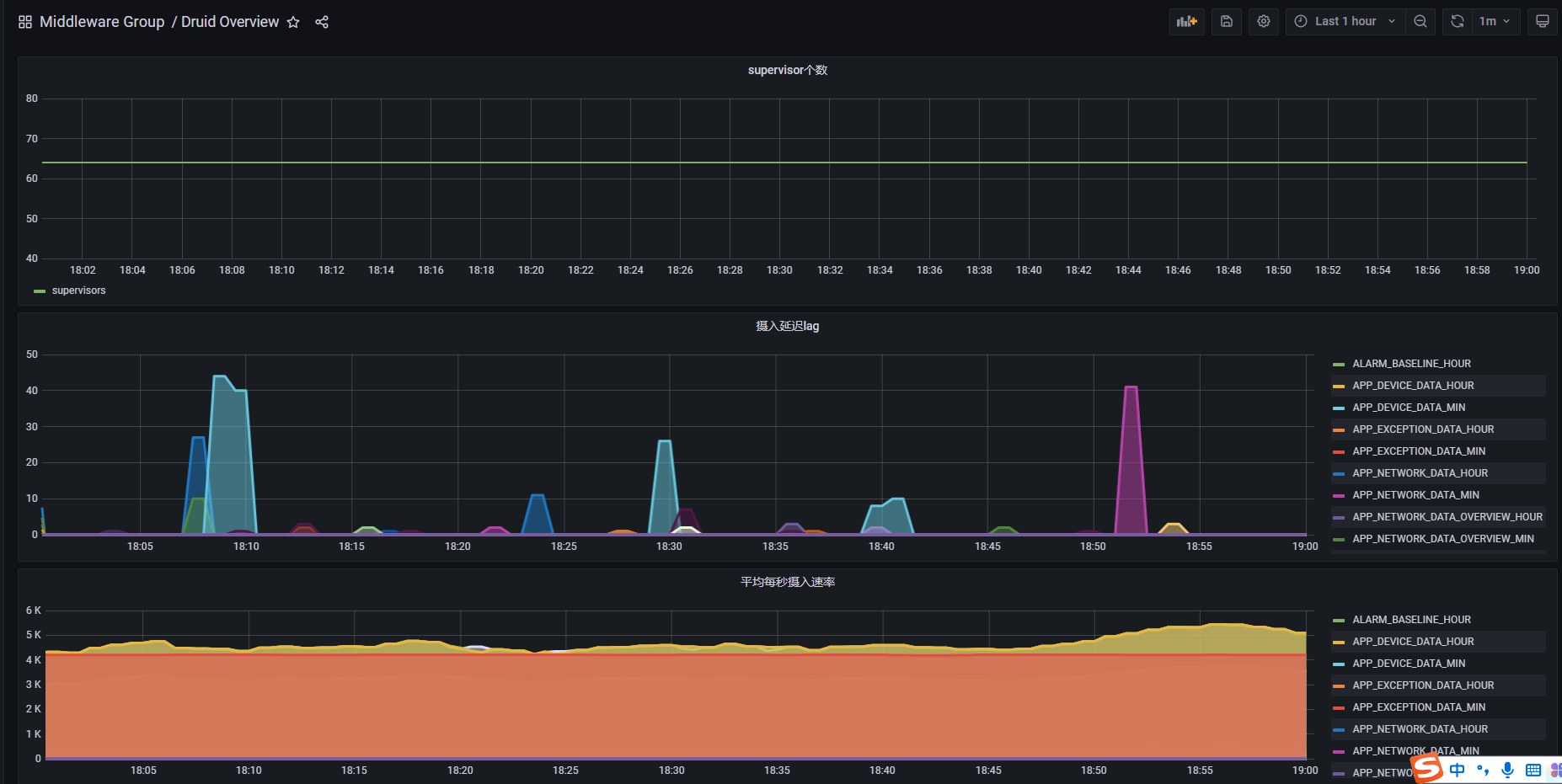


**CPU、内存、磁盘**



### Druid监控

展示当前supervisor个数，以及摄入是否有延迟和摄入速率。若有延迟则需要关注。

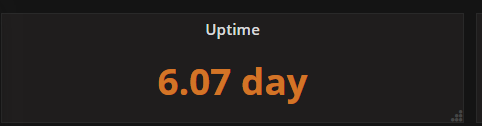


## JVM监控

针对所有java进程，都可进行jvm监控，下边详述下jvm的监控指标

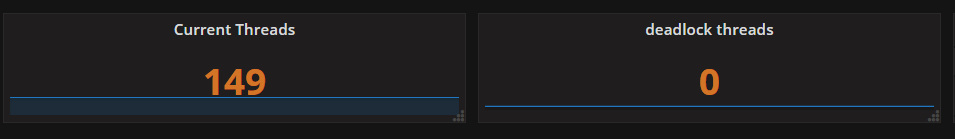
### JVM启动时间

可通过jvm取出改进程的启动时间，用于判断是否有过正常运维外的异常重启



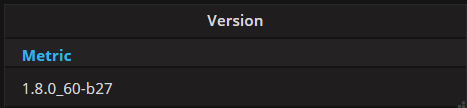
### 当前线程数/deadlock线程数

可查看当前线程数和死锁线程数，用于判断是否存在死锁

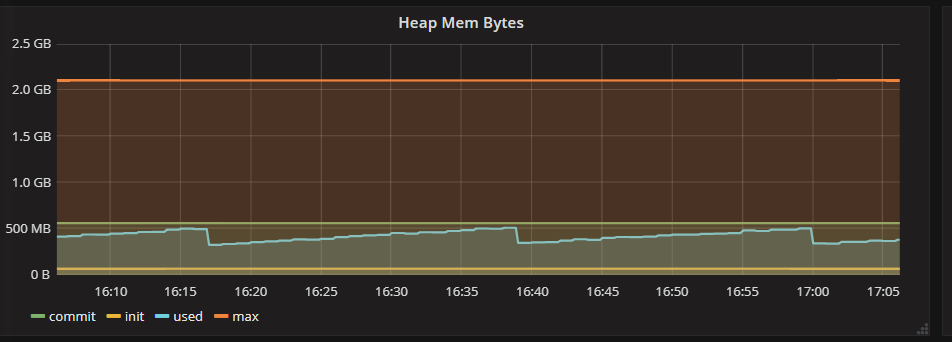


### JDK版本

可查看JDK版本，判断是否jdk版本不兼容

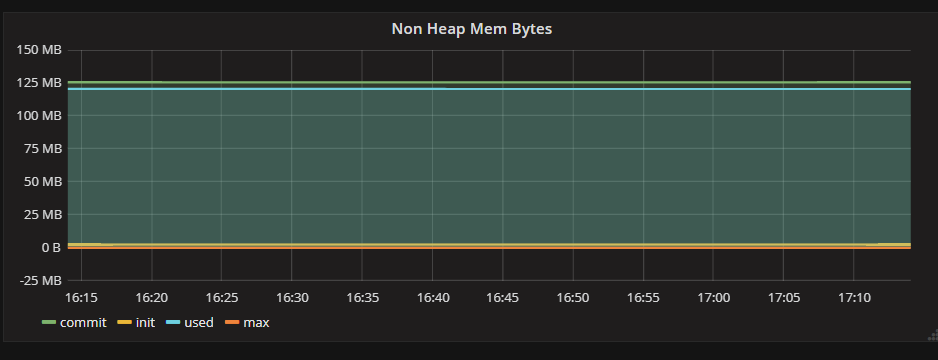


### Heap区内存使用情况

用于查看堆区内存使用情况，用来进行jvm的调优，如commit严重低于max值，可调低jvm的Xmx值，如used和commit超过max的90%，建议增大jvm的Xmx值,如used超过初始的commit值，可适当增大jvm的Xms值。通过长期观察used的使用情况，应用运行稳定后可将Xms和Xmx设置成一直以减小反复申请内存的开销，提升jvm应用的效率。

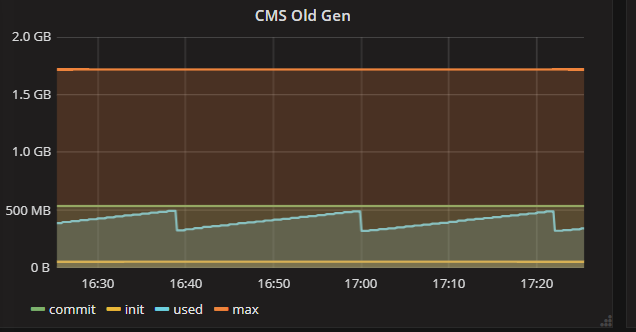
### Non Heap区内存使用情况

查看非堆区内存占用大小，堆区+非堆区内存大小可简单评估出该应用占用的总内存。进一步可评估是否机器内存满足要求。



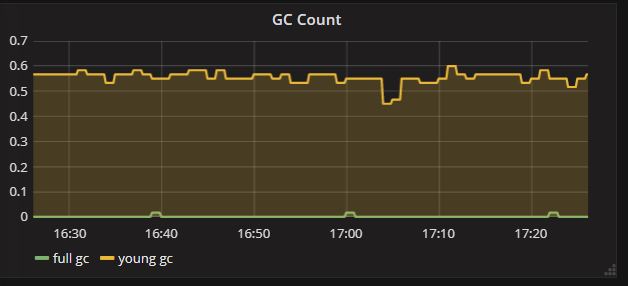
### Eden/Survivor/Old区使用情况

查看Eden、Survivor、Old区的使用情况，一般对backend应用要重点看下Old区的使用，如果长期使用率在90%以上，说明Full GC情况严重，需要增大jvm的Xmx内存或做对应用进行分片处理。



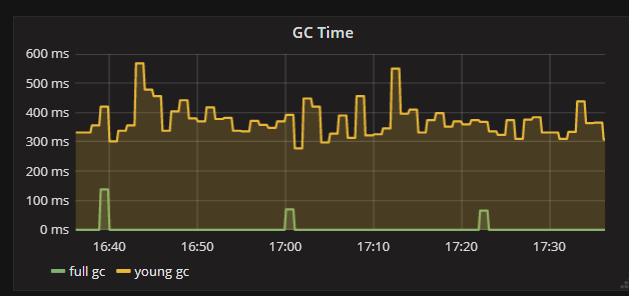
### 平均每秒GC次数

用于查看young gc 和full gc的平均次数，Full GC次数不能太频繁



### 平均每次GC时间

查看yong gc 和full gc的平均执行时间，如果是对响应时间敏感的应用，Full GC的时间不能太长。



### 进程负载

可以通过jvm观察该java应用对cpu的负载使用情况

