

Prometheus(普罗米修斯)监控系统

学习目标

- ☐ 能够安装prometheus服务器
- ☐ 能够通过安装node_exporter监控远程linux
- ☐ 能够通过安装mysqld_exporter监控远程mysql数据库
- ☐ 能够安装grafana
- ☐ 能够在grafana添加prometheus数据源
- ☐ 能够在grafana添加监控cpu负载的图形
- ☐ 能够在grafana图形显示mysql监控数据
- ☐ 能够通过grafana+onealert实现报警

任务背景

某某某公司是一家电商网站，由于公司的业务快速发展，公司要求对现有机器进行业务监控，责成运维部门来实施这个项目。

任务要求

- 1) 部署监控服务器，实现7x24实时监控
- 2) 针对公司的业务及研发部门设计监控系统，对监控项和触发器拿出合理意见
- 3) 做好问题预警机制，对可能出现的问题要及时告警并形成严格的处理机制
- 4) 做好监控告警系统，要求可以实现告警分级

一级报警 电话通知

二级报警 微信通知

三级报警 邮件通知

5) 处理好公司服务器异地集中监控问题，K8S内部使用的监控系统就是普罗米修斯

任务分析

为什么要监控？

答: 实时收集数据，通过报警及时发现问题，及时处理。数据为优化也可以提供依据。

监控四要素：

- 监控对象 [主机状态 服务 资源 页面，url]
- 用什么监控 [zabbix-server zabbix-agent] => 普罗米修斯监控
- 什么时间监控 [7x24 5x8]
- 报警给谁 [管理员]

项目选型：

- **mrtg** (Multi Router Traffic Grapher)通过**snmp**协议得到设备的流量信息，并以包含PNG格式的图形的HTML文档方式显示给用户。
- **cacti** (仙人掌) 用php语言实现的一个软件，它的主要功能是用snmp服务获取数据，然后用rrdtool储存和更新数据。官网地址: <https://www.cacti.net/>
- **ntop** 官网地址: <https://www.ntop.org/>
- **nagios** 能够跨平台,插件多,报警功能强大。官网地址: <https://www.nagios.org/>
- **centreon** 底层使用的就是nagios。是一个nagios整合版软件。官网地址:<https://www.centreon.com/>
- **ganglia** 设计用于测量数以千计的节点,资源消耗非常小。官网地址:<http://ganglia.info/>
- **open-falcon** 小米发布的运维监控软件，高效率，高可用。时间较短，用户基数小。官网地址: <http://open-falcon.org/>

- **zabbix** 跨平台，画图，多条件告警，多种API接口。使用基数特别大。官网地址: <https://www.zabbix.com/>
- **prometheus** 基于时间序列的数值数据的容器监控解决方案。官网地址: <https://prometheus.io/>

综合分析：Prometheus比较适合公司的监控需求

一、普罗米修斯概述

Prometheus(由go语言(golang)开发)是一套开源的监控&报警&时间序列数据库的组合。适合监控docker容器。因为kubernetes(俗称k8s)的流行带动了prometheus的发展。

<https://prometheus.io/docs/introduction/overview/>

二、时间序列数据

1、什么是序列数据

时间序列数据(TimeSeries Data)：按照时间顺序记录系统、设备状态变化的数据被称为时序数据。

应用的场景很多, 如：

- 无人驾驶车辆运行中要记录的经度，纬度，速度，方向，旁边物体的距离等等。每时每刻都要将数据记录下来做分析。
- 某一个地区的各车辆的行驶轨迹数据
- 传统证券行业实时交易数据
- 实时运维监控数据等

2、时间序列数据特点

- 性能好

关系型数据库对于大规模数据的处理性能糟糕。NOSQL可以比较好的处理大规模数据，让依然比不上时间序列数据库。

- 存储成本低

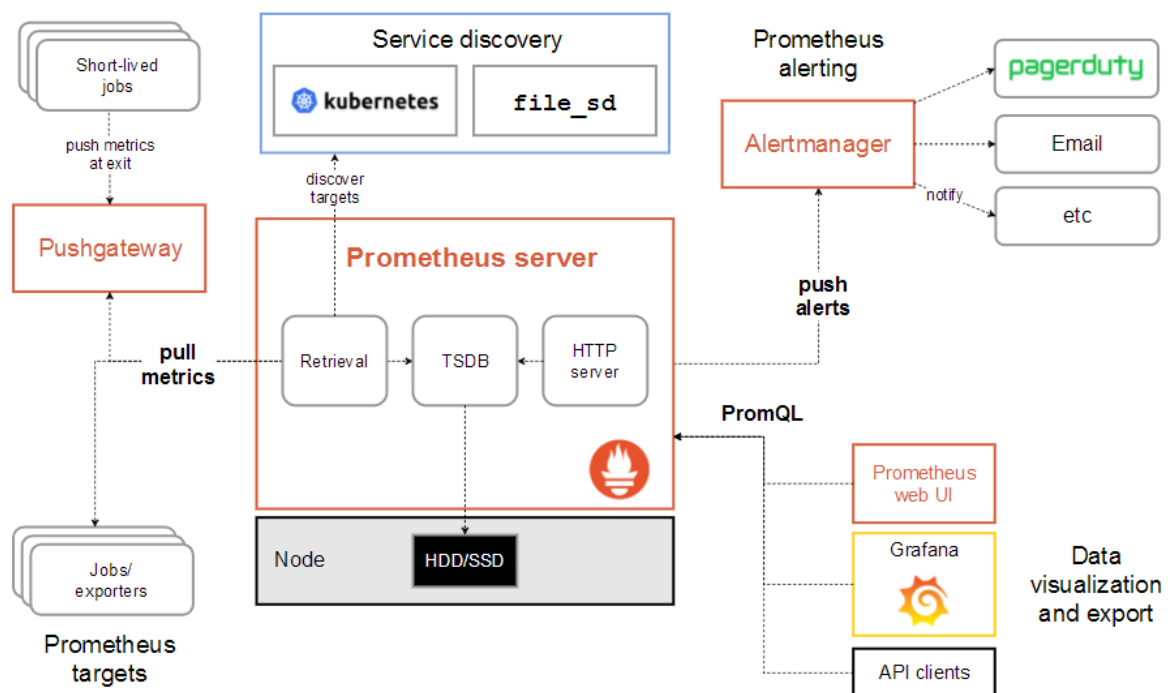
高效的压缩算法，节省存储空间，有效降低IO

Prometheus有着非常高效的时间序列数据存储方法，每个采样数据仅仅占用3.5byte左右空间，上百万条时间序列，30秒间隔，保留60天，大概花了200多G（来自官方数据）

3、Prometheus的主要特征

- 多维度数据模型
- 灵活的查询语言
- 不依赖分布式存储，单个服务器节点是自主的
- 以HTTP方式，通过pull模型拉去时间序列数据
- 也可以通过中间网关支持push模型
- 通过服务发现或者静态配置，来发现目标服务对象
- 支持多种多样的图表和界面展示

4、普罗米修斯原理架构图



三、实验环境准备

grafana服务器

10.1.1.15

Prometheus服务器

10.1.1.13

被监控服务器

10.1.1.14

1. 静态ip(要求能上外网)
2. 主机名

各自配置好主机名

```
# hostnamectl set-hostname --static server.cluster.com
```

三台都互相绑定IP与主机名

```
# vim /etc/hosts
```

```
10.1.1.13 server.cluster.com
```

```
10.1.1.14 agent1.cluster.com
```

```
10.1.1.15 grafana.cluster.com
```

3. **时间同步**(时间同步一定要确认一下)
4. 关闭防火墙,selinux

```
# systemctl stop firewalld
```

```
# systemctl disable firewalld
```

```
# iptables -F
```

1、安装prometheus

从 <https://prometheus.io/download/> 下载相应版本，安装到服务器上

官网提供的是二进制版，解压就能用，不需要编译

```
[root@server ~]# tar xf prometheus-2.5.0.linux-  
amd64.tar.gz -C /usr/local/  
[root@server ~]# mv /usr/local/prometheus-2.5.0.linux-  
amd64/ /usr/local/prometheus
```

直接使用默认配置文件启动

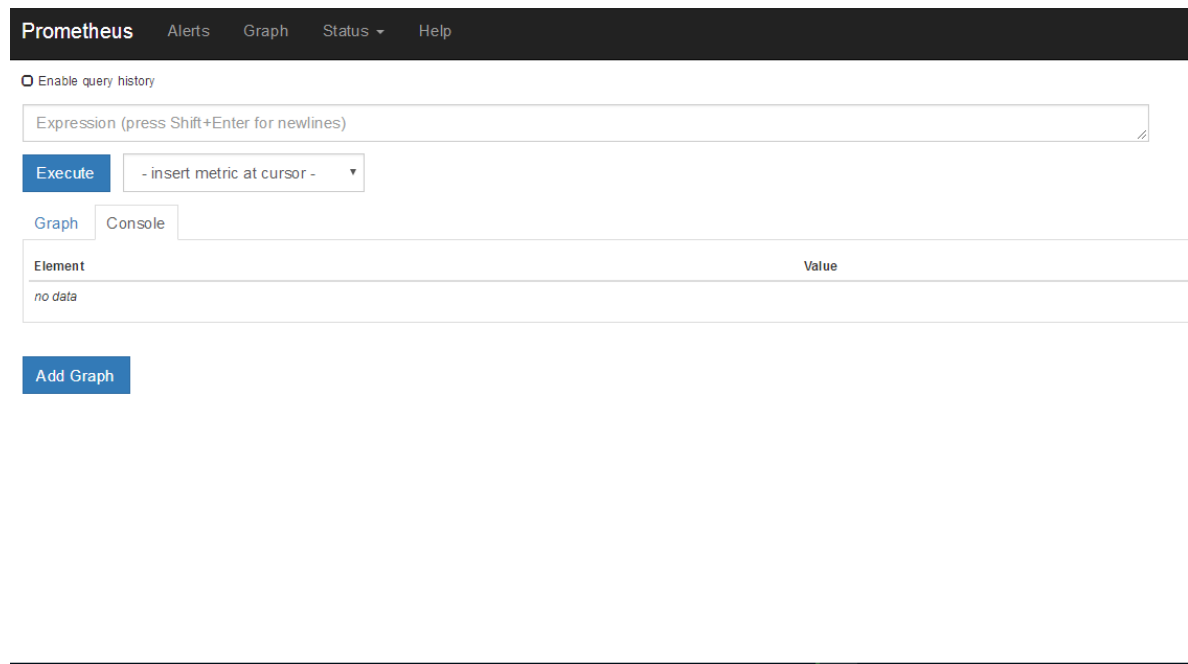
```
[root@server ~]# /usr/local/prometheus/prometheus --  
config.file="/usr/local/prometheus/prometheus.yml" &
```

确认端口(9090)

```
[root@server ~]# lsof -i:9090
```

2、prometheus界面

通过浏览器访问<http://服务器IP:9090>就可以访问到prometheus的主界面



默认只监控了本机一台，点Status --》点Targets --》可以看到只监控了本机

Prometheus Alerts Graph **Status** Help

Targets

All Unhealthy

prometheus (1/1 up) show less

- Runtime & Build Information
- Command-Line Flags
- Configuration
- Rules
- Targets
- Service Discovery

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	7.571s ago	38.99ms	

1, 点 Status

2, 点 Targets

这里可以看到默认监控了本机，也就是监控服务器

3、主机数据展示

通过<http://服务器IP:9090/metrics>可以查看到监控的数据

```
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 0.000682652
go_gc_duration_seconds{quantile="0.25"} 0.001064976
go_gc_duration_seconds{quantile="0.5"} 0.001743837
go_gc_duration_seconds{quantile="0.75"} 0.002586473
go_gc_duration_seconds{quantile="1"} 0.01326839
go_gc_duration_seconds_sum 0.0419986
go_gc_duration_seconds_count 16
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 36
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.11.1"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.4427e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.19677512e+08
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.46774e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 753737
# HELP go_memstats_gc_cpu_fraction The fraction 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 6.119982162383281e-05
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 2.387968e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 1.4427e+07
# 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 1.4427e+07
```

在web主界面可以通过关键字查询监控项



4、监控远程Linux主机

① 在远程linux主机(被监控端agent1)上安装node_exporter组件

下载地址: <https://prometheus.io/download/>

```
[root@agent1 ~]# tar xf node_exporter-0.16.0.linux-  
amd64.tar.gz -C /usr/local/  
[root@agent1 ~]# mv /usr/local/node_exporter-0.16.0.linux-  
amd64/ /usr/local/node_exporter
```

里面就一个启动命令node_exporter,可以直接使用此命令启动

```
[root@agent1 ~]# ls /usr/local/node_exporter/  
LICENSE  node_exporter  NOTICE  
[root@agent1 ~]# nohup  
/usr/local/node_exporter/node_exporter &
```

确认端口(9100)

```
[root@agent1 ~]# lsof -i:9100
```

扩展: **nohup**命令: 如果把启动node_exporter的终端给关闭,那么进程也会随之关闭。nohup命令会帮你解决这个问题。

② 通过浏览器访问<http://被监控端IP:9100/metrics>就可以查看到node_exporter在被监控端收集的监控信息


```
# HELP go_gc_duration_seconds A summary of the GC invocation durations.
# 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 6
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.9.6"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 1.400808e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.400808e+06
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.443392e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 980
# HELP go_memstats_gc_cpu_fraction The fraction 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 of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 169984
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 1.400808e+06
# 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 96304
```

③ 回到prometheus服务器的配置文件里添加被监控机器的配置段

在主配置文件最后加上下面三行

```
[root@server ~]# vim /usr/local/prometheus/prometheus.yml
- job_name: 'agent1' # 取一个job名称来代表被监控的机器
```

static_configs:

```
- targets: ['10.1.1.14:9100'] # 这里改成被监控机器的IP, 后面端口接9100
```

改完配置文件后, 重启服务

```
[root@server ~]# pkill prometheus
```

```
[root@server ~]# lsof -i:9090 # 确认端口没有进程占用
```

```
[root@server ~]# /usr/local/prometheus/prometheus --config.file="/usr/local/prometheus/prometheus.yml" &
```

```
[root@server ~]# lsof -i:9090 # 确认端口被占用, 说明重启成功
```

④ 回到web管理界面 --》点Status --》点Targets --》可以看到多了一台监控目标

Prometheus Alerts Graph Status Help

Targets

All Unhealthy

agent1 (1/1 up) show less

这里看到agent1已经成功被监控

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.1.1.14:9100/metrics	UP	instance="10.1.1.14:9100" job="agent1"	5.328s ago	30.33ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	12.112s ago	31.31ms	

练习: 加上本机prometheus的监控

答: 在本机安装node_exporter , 也使用上面的方式监控起来。

5、 监控远程MySQL

① 在被管理机agent1上安装mysqld_exporter组件

下载地址: <https://prometheus.io/download/>

安装mysqld_exporter组件

```
[root@agent1 ~]# tar xf mysqld_exporter-0.11.0.linux-amd64.tar.gz -C /usr/local/
[root@agent1 ~]# mv /usr/local/mysqld_exporter-0.11.0.linux-amd64/ /usr/local/mysqld_exporter
[root@agent1 ~]# ls /usr/local/mysqld_exporter/
LICENSE  mysqld_exporter  NOTICE
```

安装mariadb数据库, 并授权

```
[root@agent1 ~]# yum install mariadb* -y
[root@agent1 ~]# systemctl restart mariadb
[root@agent1 ~]# systemctl enable mariadb
[root@agent1 ~]# mysql
```

MariaDB [(none)]> grant select,replication client,process ON *.* to 'mysql_monitor'@'localhost' identified by '123';

(注意: 授权ip为localhost, 因为不是prometheus服务器来直接找mariadb获取数据, 而是prometheus服务器找mysql_exporter,mysql_exporter再找mariadb。所以这个localhost是指的mysql_exporter的IP)

```
MariaDB [(none)]> flush privileges;
```

```
MariaDB [(none)]> quit
```

创建一个mariadb配置文件，写上连接的用户名与密码(和上面的授权的用户名和密码要对应)

```
[root@agent1 ~]# vim /usr/local/mysql_exporter/.my.cnf
[client]
user=mysql_monitor
password=123
```

启动mysql_exporter

```
[root@agent1 ~]# nohup
/usr/local/mysql_exporter/mysql_exporter --config.my-
cnf=/usr/local/mysql_exporter/.my.cnf &
```

确认端口(9104)

```
[root@agent1 ~]# lsof -i:9104
```

② 回到prometheus服务器的配置文件里添加被监控的mariadb的配置段

在主配置文件最后再加上下面三行

```
[root@server ~]# vim /usr/local/prometheus/prometheus.yml
- job_name: 'agent1_mariadb' # 取一个job
名称来代表被监控的mariadb
  static_configs:
  - targets: ['10.1.1.14:9104'] # 这里改成
被监控机器的IP，后面端口接9104
```

改完配置文件后,重启服务

```
[root@server ~]# pkill prometheus
[root@server ~]# lsof -i:9090
[root@server ~]# /usr/local/prometheus/prometheus --
config.file="/usr/local/prometheus/prometheus.yml" &
[root@server ~]# lsof -i:9090
```

③ 回到web管理界面 --》点Status --》点Targets --》可以看到监控mariadb了

Targets

All Unhealthy

agent1 (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.1.1.14:9100/metrics	UP	instance="10.1.1.14:9100" job="agent1"	3.814s ago	61.69ms	

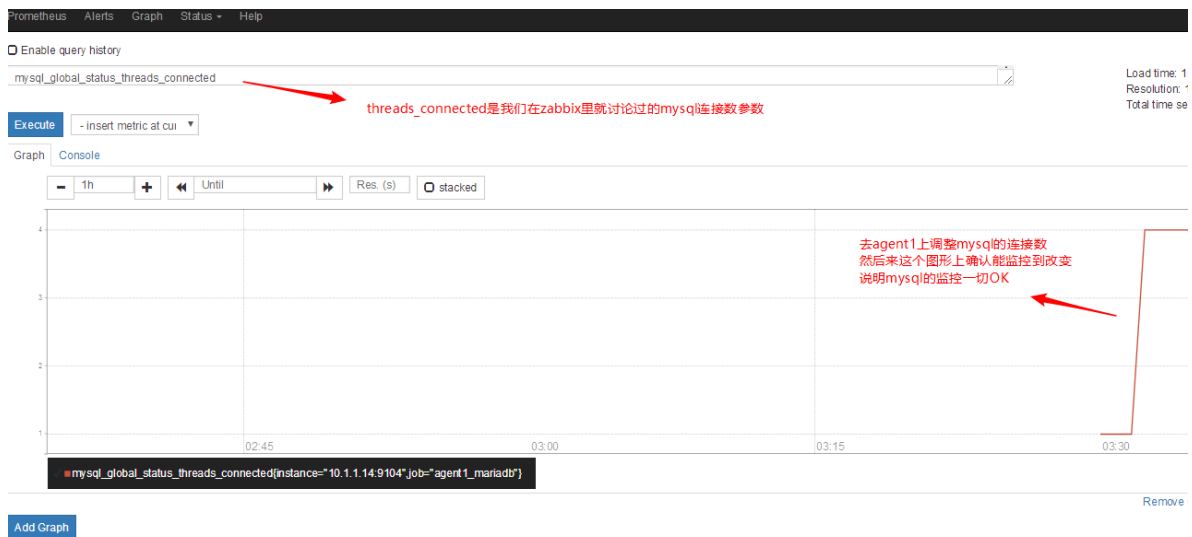
agent1_mariadb (1/1 up) show less

这里就可以看到这个mariadb的监控也开启了

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://10.1.1.14:9104/metrics	UP	instance="10.1.1.14:9104" job="agent1_mariadb"	12.551s ago	45.55ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	10.601s ago	24.63ms	



四、Grafana可视化图形工具

1、什么是Grafana

Grafana是一个开源的度量分析和可视化工具，可以通过将采集的数据分析，查询，然后进行可视化的展示,并能实现报警。



网址: <https://grafana.com/>

2、使用Grafana连接Prometheus

① 在grafana服务器上安装grafana

下载地址:<https://grafana.com/grafana/download>

我这里选择的rpm包，下载后直接rpm -ivh安装就OK

```
[root@grafana ~]# rpm -ivh /root/Desktop/grafana-5.3.4-1.x86_64.rpm
```

启动服务

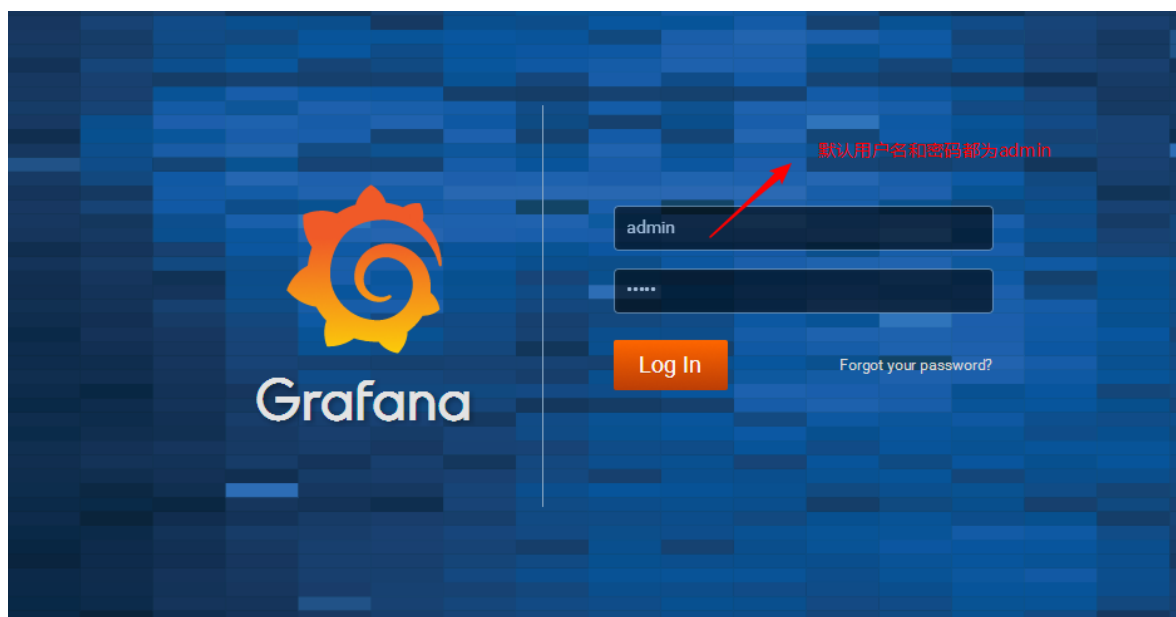
```
[root@grafana ~]# systemctl start grafana-server
```

```
[root@grafana ~]# systemctl enable grafana-server
```

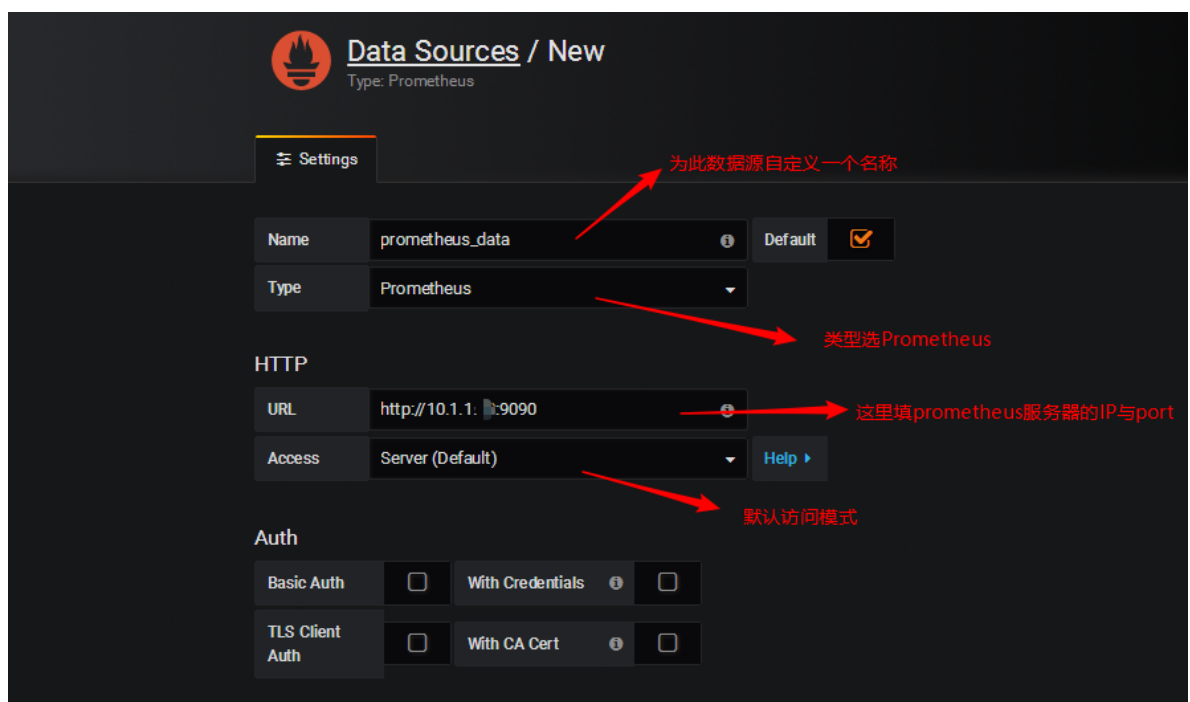
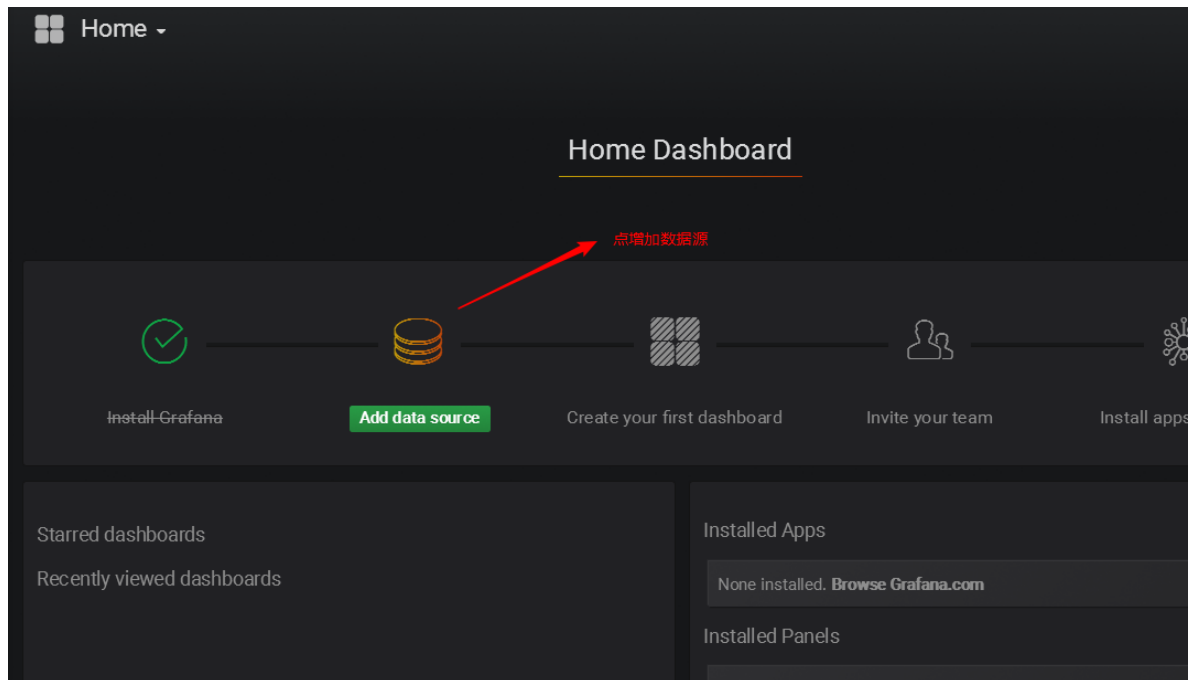
确认端口(3000)

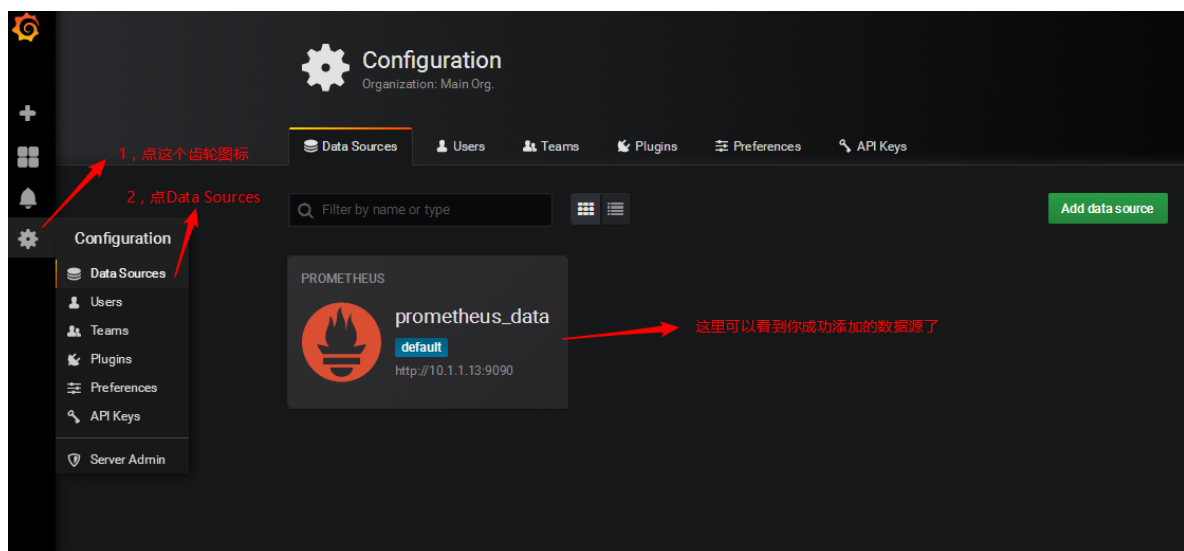
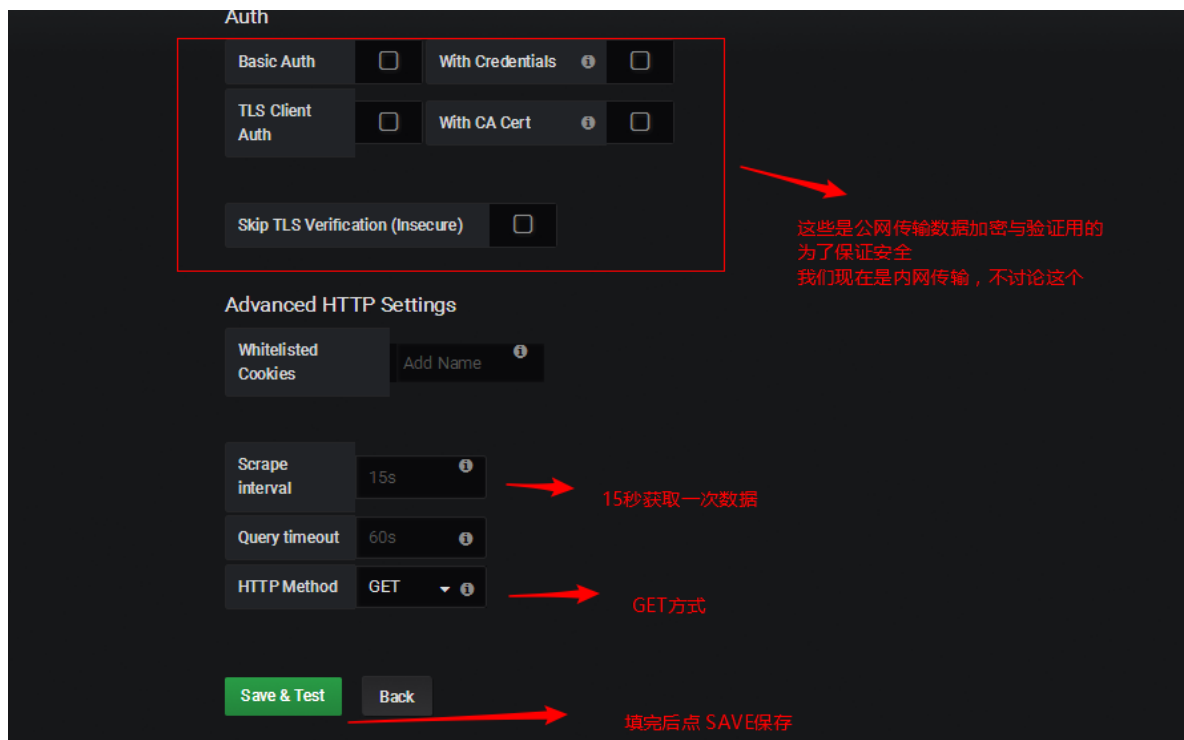
```
[root@grafana ~]# lsof -i:3000
```

② 通过浏览器访问 **http:// grafana服务器IP:3000**就到了登录界面,使用默认的admin用户,admin密码就可以登陆了

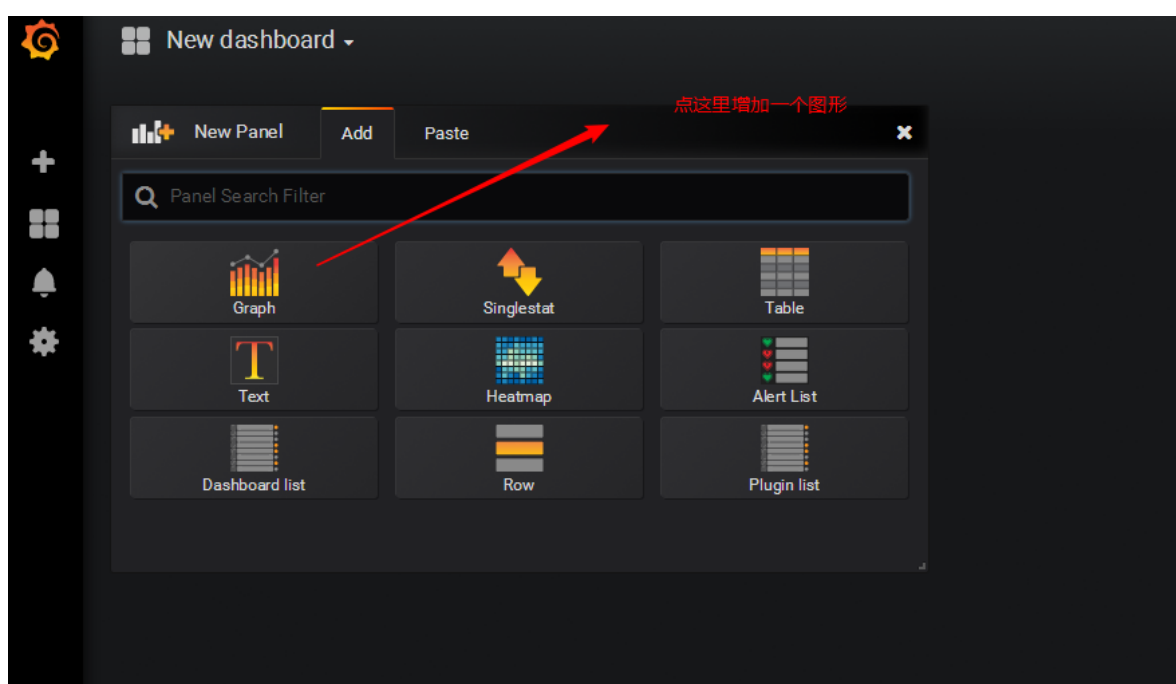


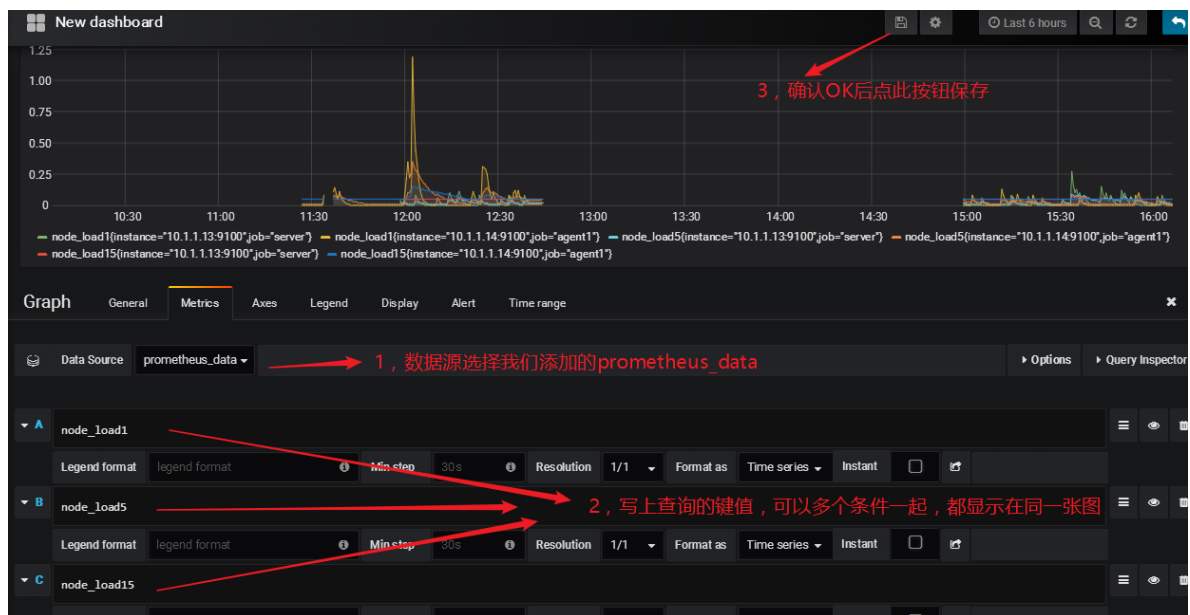
③ 下面我们把prometheus服务器收集的数据做为一个数据源添加到grafana,让grafana可以得到prometheus的数据。



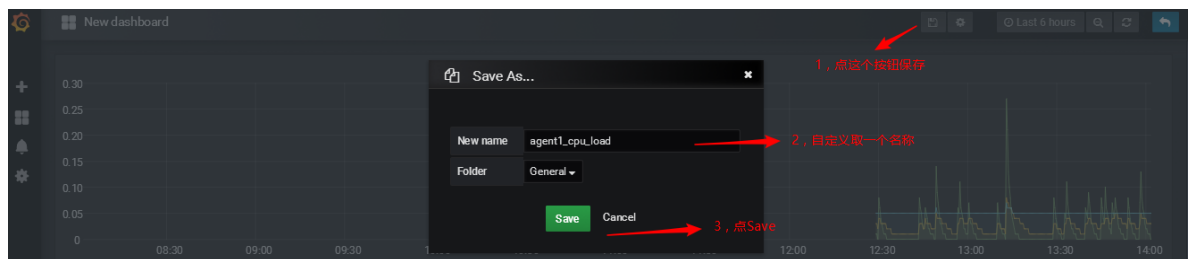


④ 然后为添加好的数据源做图形显示

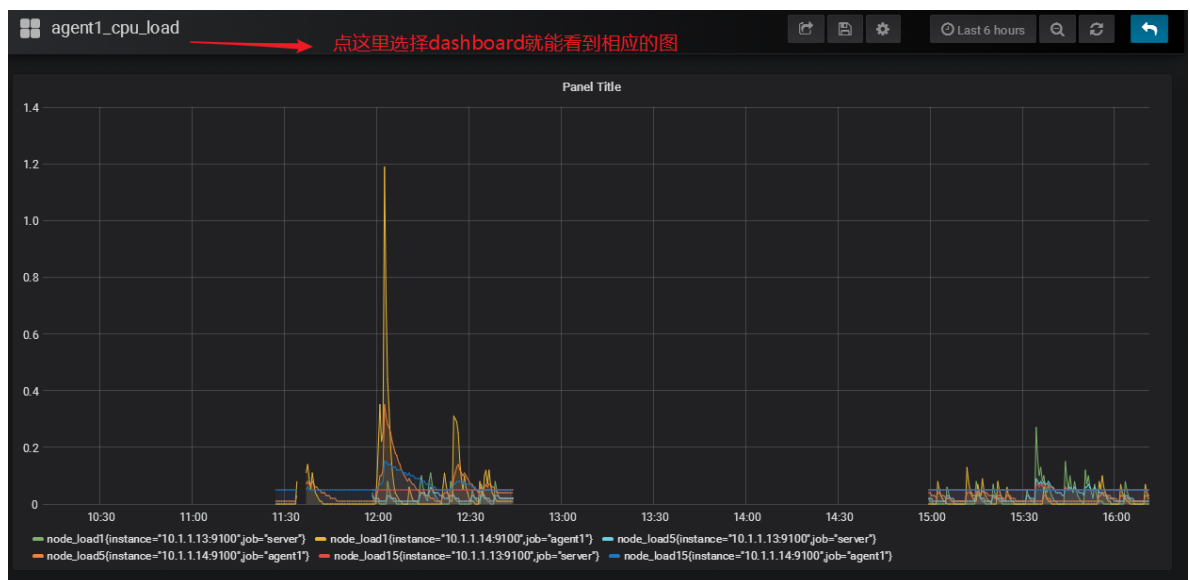




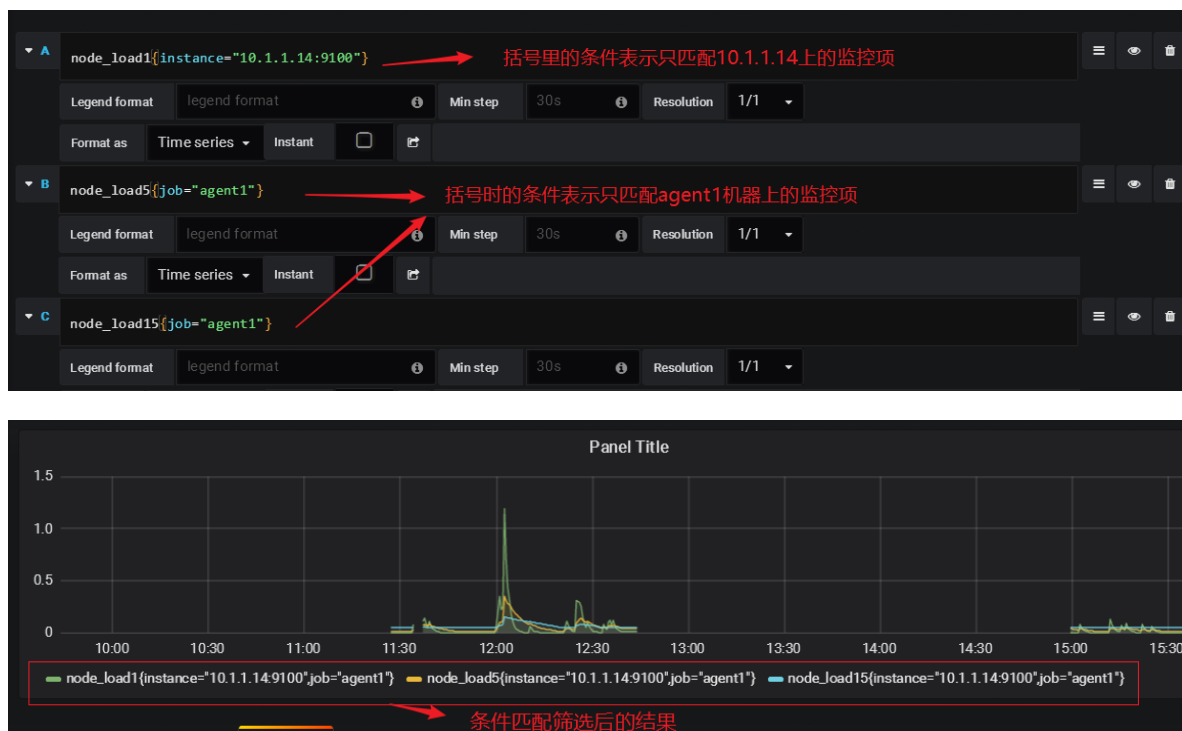
⑤ 保存



⑥ 最后在dashboard可以查看到



⑦ 匹配条件显示



3、Grafana图形显示MySQL监控数据

① 在grafana上修改配置文件,并下载安装mysql监控的dashboard (包含相关json文件 , 这些json文件可以看作是开发人员开发的一个监控模板)

参考网址: <https://github.com/percona/grafana-dashboards>

在grafana配置文件里最后加上以下三行

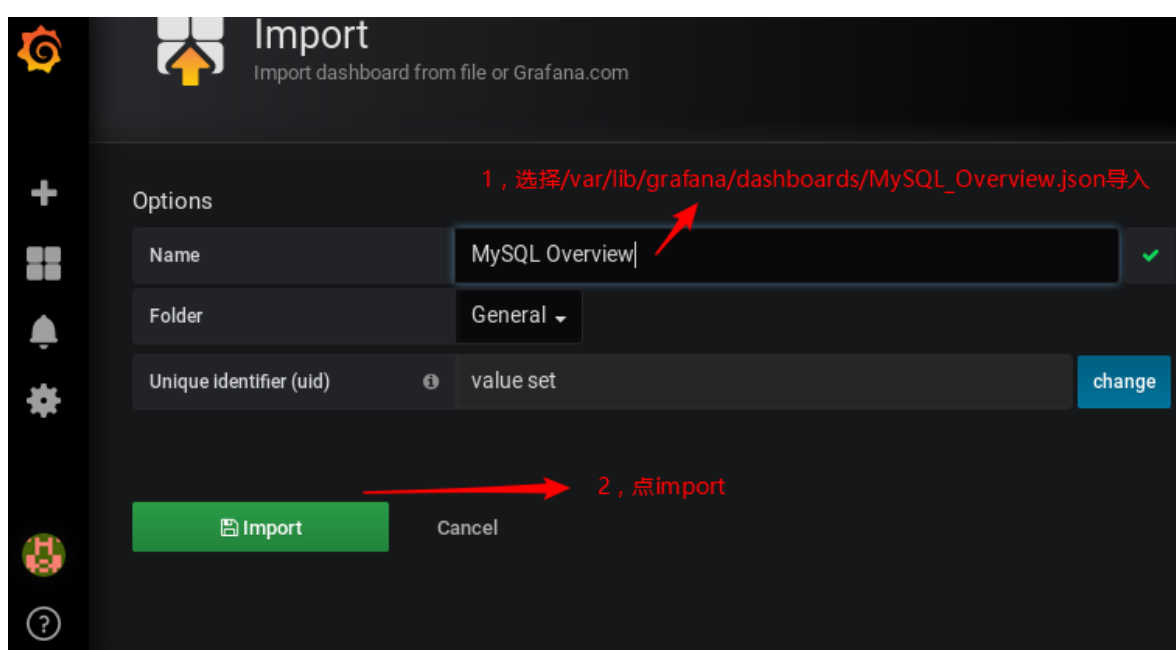
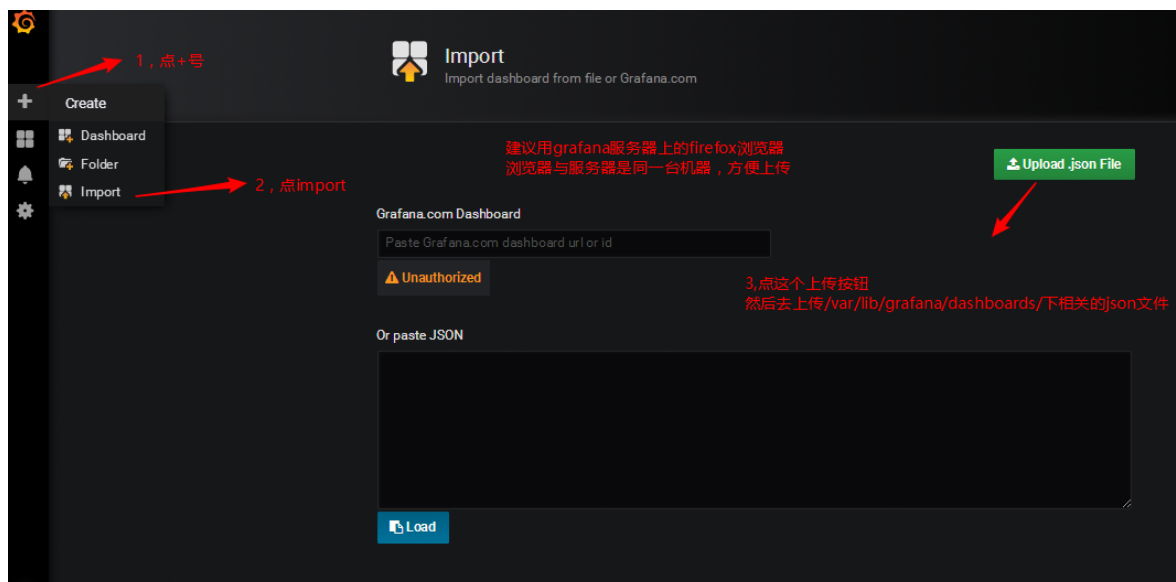
```
[root@grafana ~]# vim /etc/grafana/grafana.ini
[dashboards.json]
enabled = true
path = /var/lib/grafana/dashboards
```

```
[root@grafana ~]# cd /var/lib/grafana/
[root@grafana grafana]# git clone
https://github.com/percona/grafana-dashboards.git
[root@grafana grafana]# cp -r grafana-
dashboards/dashboards/ /var/lib/grafana/
```

重启grafana服务

```
[root@grafana grafana]# systemctl restart grafana-server
```

② 在grafana图形界面导入相关json文件



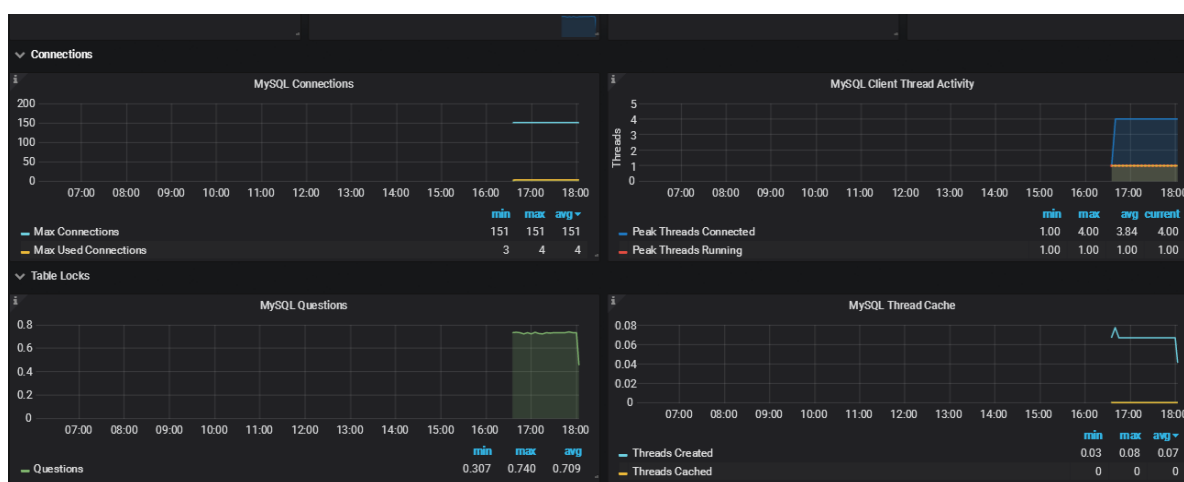
③ 点import导入后,报prometheus数据源找不到,因为这些json文件里默认要找的就是叫Prometheus的数据源,但我们前面建立的数据源却是叫prometheus_data(坑啊)

那么请自行把原来的prometheus_data源改名为**Prometheus**即可(注意:第一个字母P是大写)

然后再回去刷新,就有数据了(如下图所示)



④ 过段时间再看，就会有数据了(如下图所示)



4、Grafana+onealert报警

prometheus报警需要使用alertmanager这个组件，而且报警规则需要手动编写(对运维来说不友好)。所以我这里选用grafana+onealert报警。

注意: 实现报警前把所有机器**时间同步**再检查一遍。

① 先在onealert里添加grafana应用(申请onealert账号在zabbix已经讲过)



名称：	grafana报警	自定义一个名称
应用 Key：	4bcc1d0d-4048-b6ed-38ec-bc2e41b50f11	保存后得到key
自动关闭时间：	30分钟	

② 在Grafana中配置Webhook URL

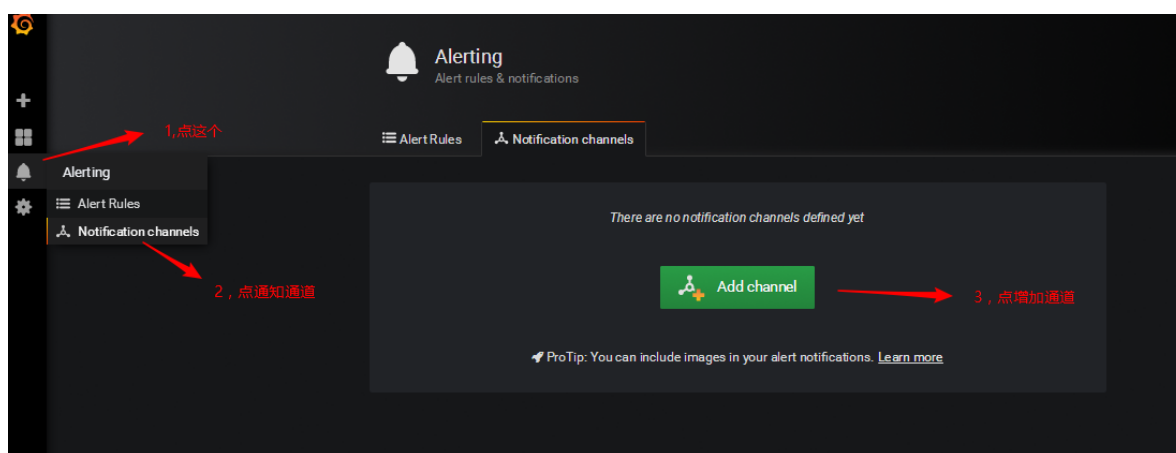
按照提示来配置grafana


1. 在Grafana中创建Notification channel，选择类型为Webhook；
2. 推荐选中Send on all alerts和Include image，OneAlert体验更佳；
3. 将第一步中生成的Webhook URL填入Webhook settings Url；
URL格式：
`http://api.onealert.com/alert/api/event/grafana/v1/4bcc1d0d-4048-b6ed-38ec-bc2e41b50f11/`
4. Http Method选择POST；
5. Send Test&Save；

② 配置通知策略



③ 在grafana增加通知通道



 **Alerting**
Alert rules & notifications

Alert Rules | **Notification channels**

New Notification Channel


Name	onealert	名称自定义
Type	webhook	类型选择webhook
Send on all alerts	<input checked="" type="checkbox"/>	这两个勾上
Include image	<input checked="" type="checkbox"/>	
Send reminders	<input type="checkbox"/>	

Webhook settings

Url	http://api.onealert.com/alert/api/event/grafana/v1/4...	这个URL是在onealert那里产生的，复制过来
Http Method	POST	选择POST
Username		
Password		

可以先点一下Send Test测试一下报警媒介是否OK
再点Save保存

Save **Send Test** **Back**

 **Alerting**
Alert rules & notifications

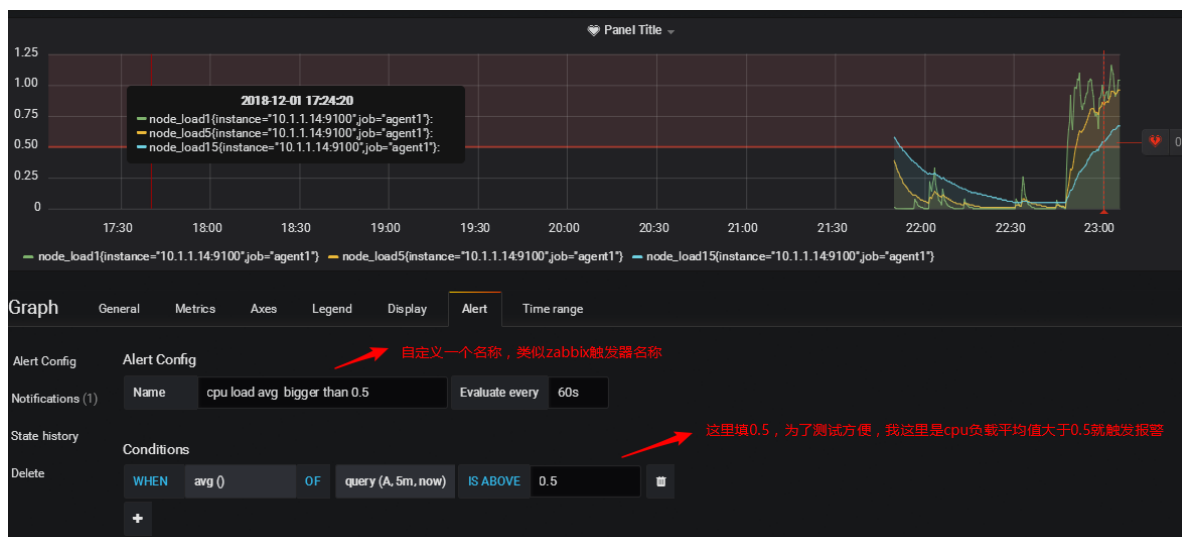
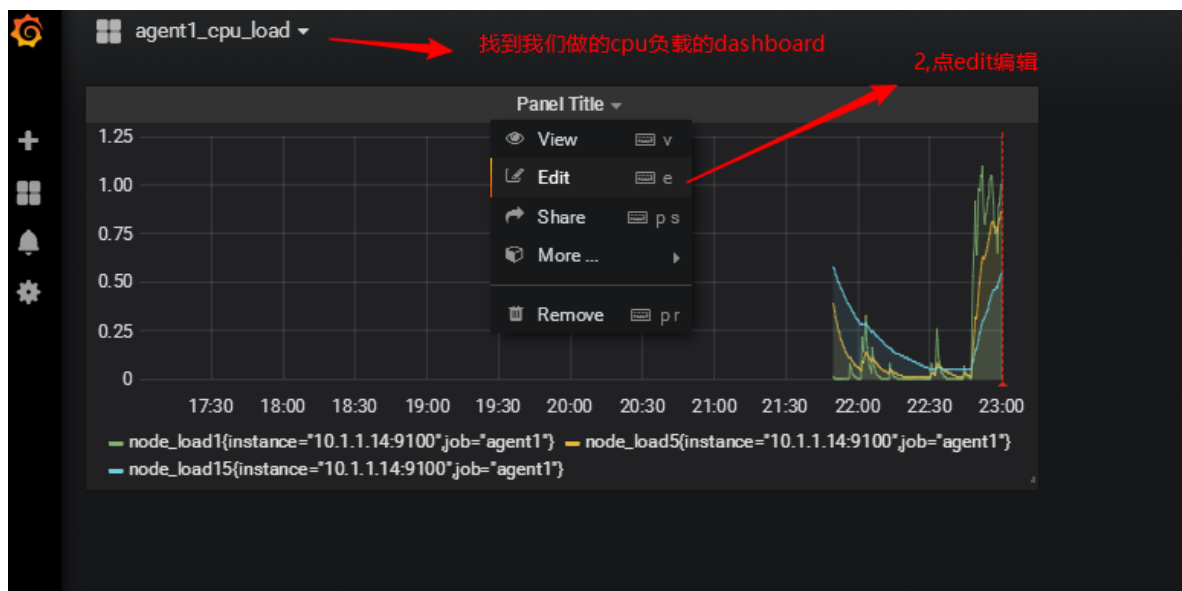
Alert Rules | **Notification channels**

+ New Channel

Name	Type
onealert	webhook default x

创建成功

④ 现在可以去设置一个报警来测试了(这里以我们前面加的cpu负载监控来做测试)





⑤ 保存后就可以测试了

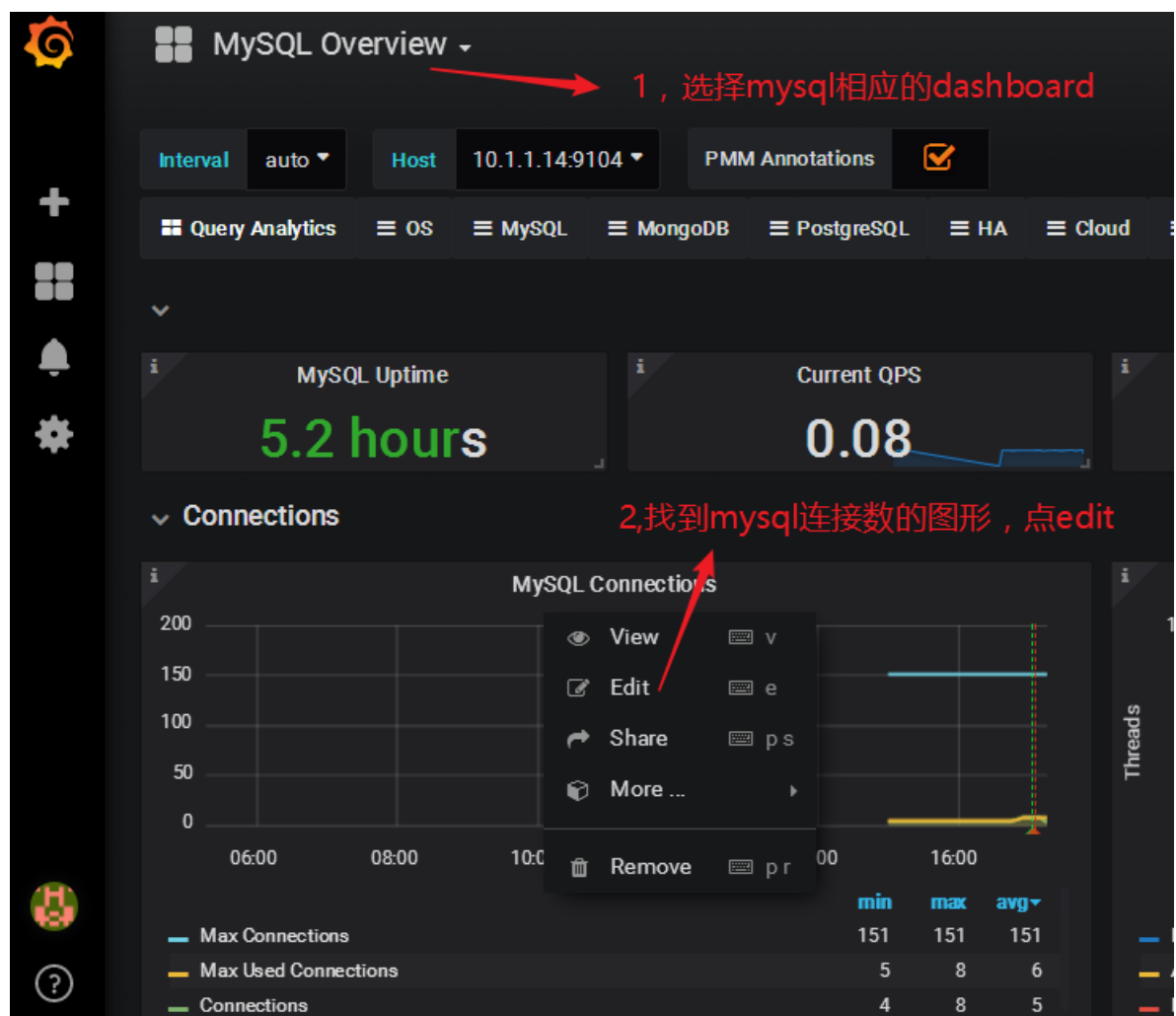
如果agent1上的cpu负载还没有到0.5, 你可以试试0.1, 或者运行一些程序把agent1负载调大。最终能测试报警成功。



最终的邮件报警效果：



测试mysql链接数报警



Graph General Metrics Axes Legend Display **Alert** Time range

Alert Config

Notifications (1)

State history

Delete

Alert Config

Name MySQL Connections alert Evaluate every 60s

Conditions

WHEN avg () OF query (A, 1m, now) IS ABOVE 5

If no data or all values are null SET STATE TO No Data

If execution error or timeout SET STATE TO Alerting

Test Rule

Template variables are not supported in alert queries

但是这里有一个问题，说模板变量在报警查询里不支持

这一次测试一下mysql连接数的报警

这里定义连接数大于5就报警

Data Source Prometheus Options Query Insp

将原来的\$host变量替换成被监控的mysql机器的IP与端口

status_threads_connected{instance="10.1.1.14:9104"}[60s] or mysql_global_status_threads_connected{instance="10.1.1.14:9104"}

Legend format Connections Min step 60s Resolution 1/1

Format as Time series Instant

mysql_global_status_max_used_connections{instance="10.1.1.14:9104"}

Legend format Max Used Connections Min step 60s Resolution 1/1

Format as Time series Instant

mysql_global_variables_max_connections{instance="10.1.1.14:9104"}

Legend format Max Connections Min step 60s Resolution 1/1

Format as Time series Instant

将\$interval改成60s时间间隔

MySQL Overview

1月23, 2019 16:56:52 to 1月24, 2019 16:56:52

100 50 0

18:00 20:00 22:00 00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00

Max Connections 151 151

Max Used Connections 1 8

Graph General Metrics Axes Legend Display **Alert** Time range

Alert Config

Notifications

Send to onealert

Message mariadb连接数超过5，测试。.....

State history

Delete

选择onealert发送通道

选择通知

MySQL Overview

State history

Delete

Conditions

WHEN avg () OF query (A, 1m, now) IS ABOVE 5

+

If no data or all values are null SET STATE TO No Data

If execution error or timeout SET STATE TO Alerting

Test Rule

用test Rule得到测试的当前值

```
firing: true
state: "alerting"
conditionEvals: "true = true"
timeMs: "7.917ms"
▼matches: Array[1]
  ▼0: Object
    metric: "Connections"
    value: 9
  ▼logs: Array[2]
    ▼0: Object
      message: "Condition[0]: Query Result"
      ►data: Array[1]
    ▼1: Object
      message: "Condition[0]: Eval: true, Metric: Connections, Value: 9.000"
      data: null
```

超过了触发条件，就为报警状态了

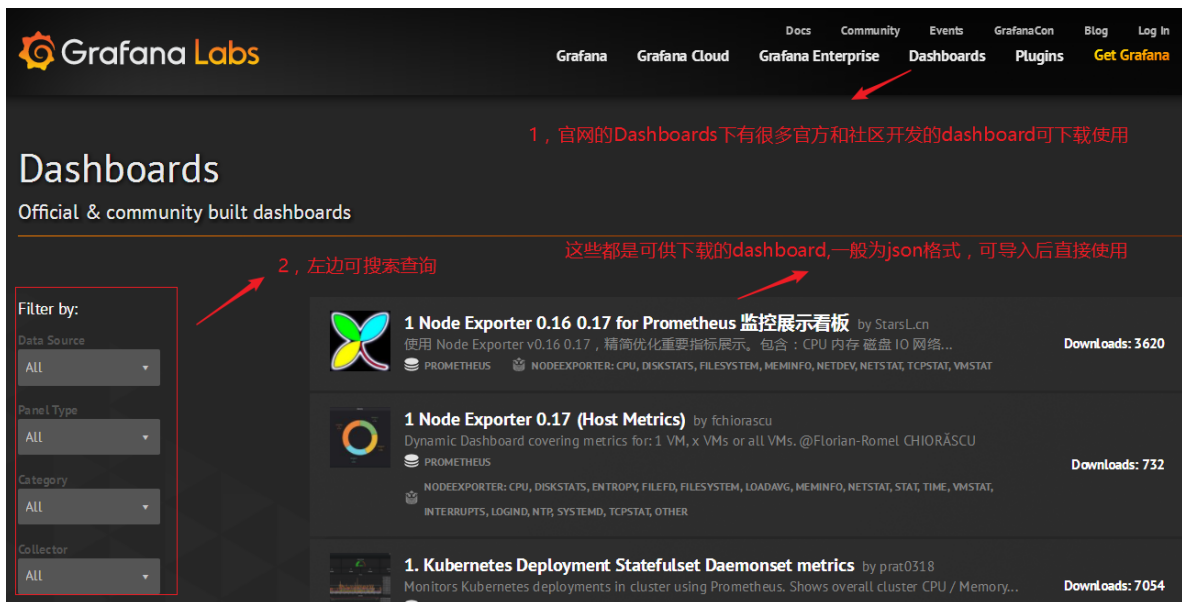
得到的值

5、总结报警不成功的可能原因

- 各服务器之间时间不同步，这样时序数据会出问题，也会造成报警出问题
- 必须写通知内容，留空内容是不会发报警的
- 修改完报警配置后，记得要点右上角的保存
- 保存配置后，需要由OK状态变为alerting状态才会报警(也就是说，你配置保存后，就已经是alerting状态是不会报警的)
- grafana与onealert通信有问题

6、课外扩展

prometheus目前还在发展中，很多相应的监控都需要开发。但在官网的dashboard库中,也有一些官方和社区开发人员开发的dashboard可以直接拿来用。



示例:



有兴趣的同学可以下载几个尝试一下(不一定版本兼容,如果不兼容,可多试几个不同版本)

