

今日学习目标:

- ☐ 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

普罗米修斯

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。适合监控容器平台。因为kubernetes(俗称k8s)的流行带动了prometheus的发展。

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

数据库分类:

- 关系型 mysql,oracle,sql server,sybase,db2,access等
- 非关系型(nosql)
 - key-value memcache redis etcd
 - 文档型 mongodb elasticsearch
 - 列式 hbase
 - 时序 prometheus
 - 图形数据库

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

时间序列数据库的主要优点:

- 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

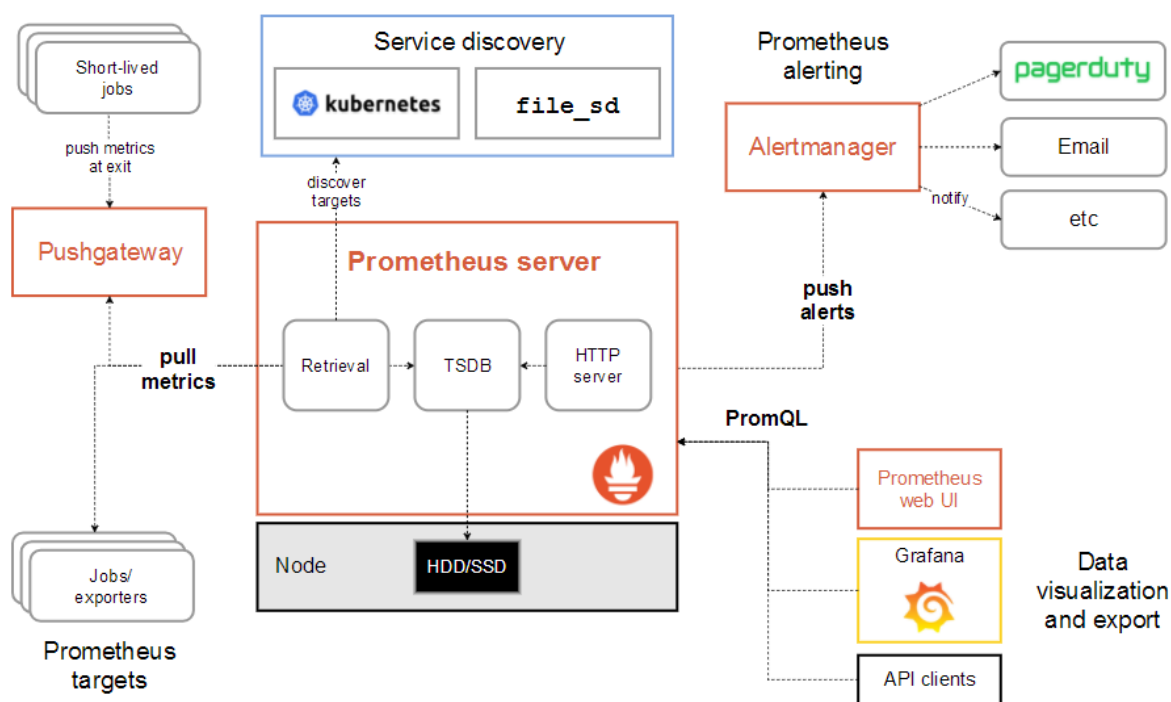
高效的压缩算法，节省存储空间，有效降低IO

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）

Prometheus的主要特征有：

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

普罗米修斯原理架构图



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)

```
# systemctl restart ntpd
```

```
# systemctl enable ntpd
```

4. 关闭防火墙,selinux

```
# systemctl stop firewalld
```

```
# systemctl disable firewalld
```

```
# iptables -F
```

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

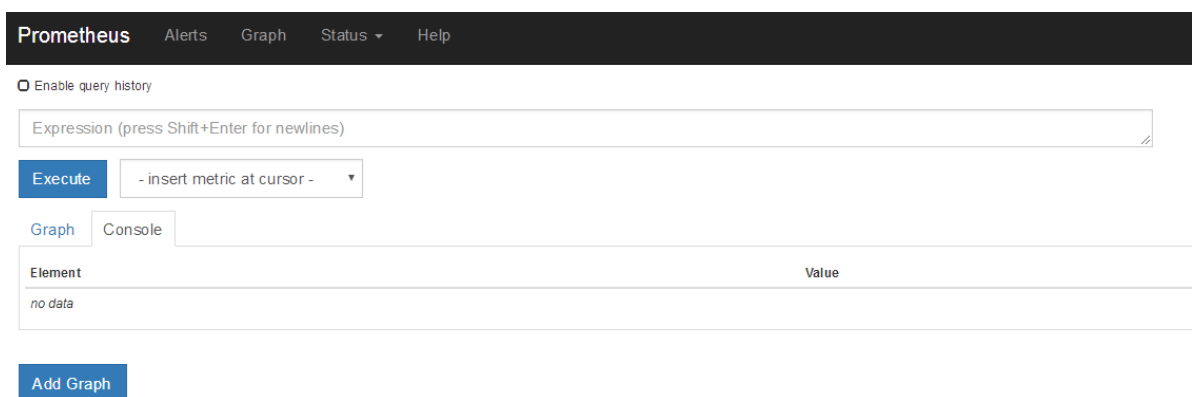
```
[root@server ~]# /usr/local/prometheus/prometheus --  
config.file="/usr/local/prometheus/prometheus.yml" &
```

确认端口(9090)

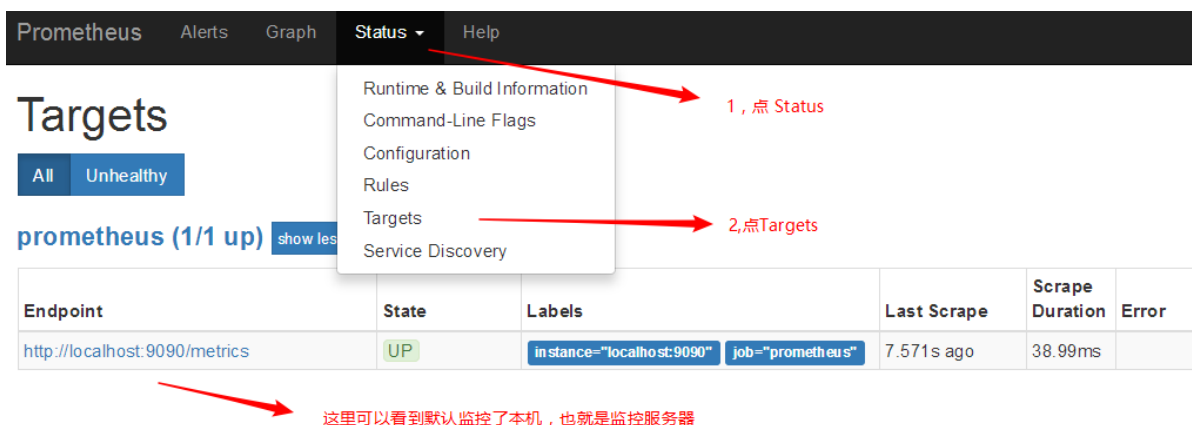
```
[root@server ~]# lsof -i:9090
```

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



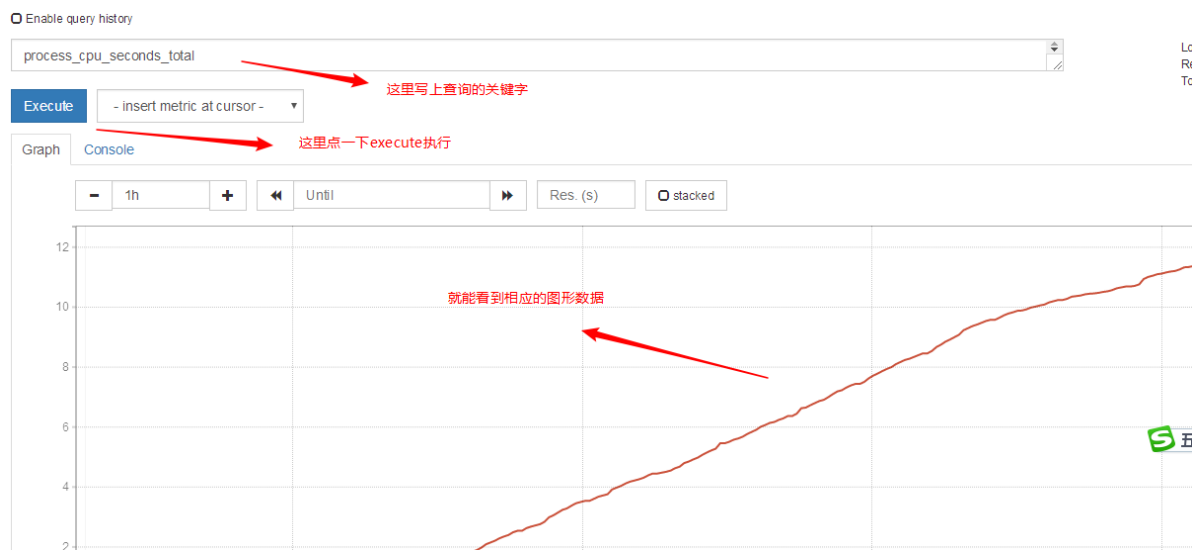
默认只监控了本机一台，点Status --》点Targets --》可以看到只监控了本机



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



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 98304
```

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

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。

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-server -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/mysqld_exporter/.my.cnf
[client]
user=mysql_monitor
password=123
```

```
启动mysqld_exporter
[root@agent1 ~]# nohup /usr/local/mysqld_exporter/mysqld_exporter --config.my-
cnf=/usr/local/mysqld_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	



grafana

使用grafana连接prometheus

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。

The leading **open source** software for time series analytics

开源的时序数据分析平台

 **Grafana**

No matter where your data is, or what kind of database it lives in, you can bring it together with Grafana. Beautifully.

目前支持52种数据源，其中prometheus就是其中支持的一种

As of right now, there are **52 data sources**, **42 panels**, **17 apps** and **1529 dashboards** available.

网址: <https://grafana.com/>

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

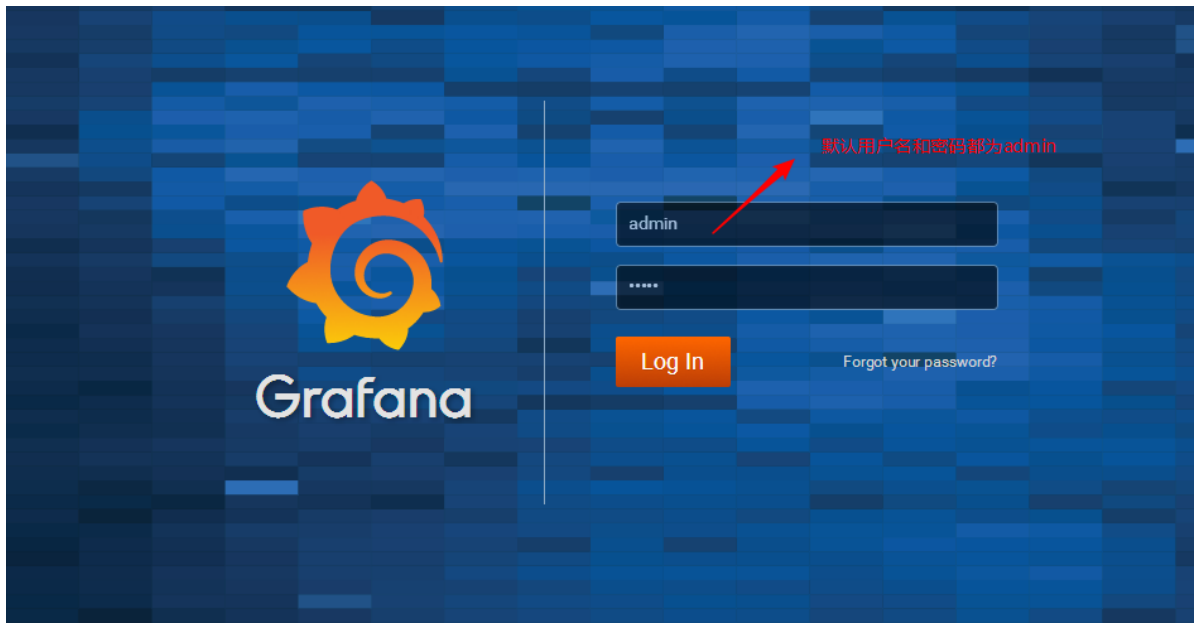
```
[root@grafana ~]# systemctl start grafana-server
```

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

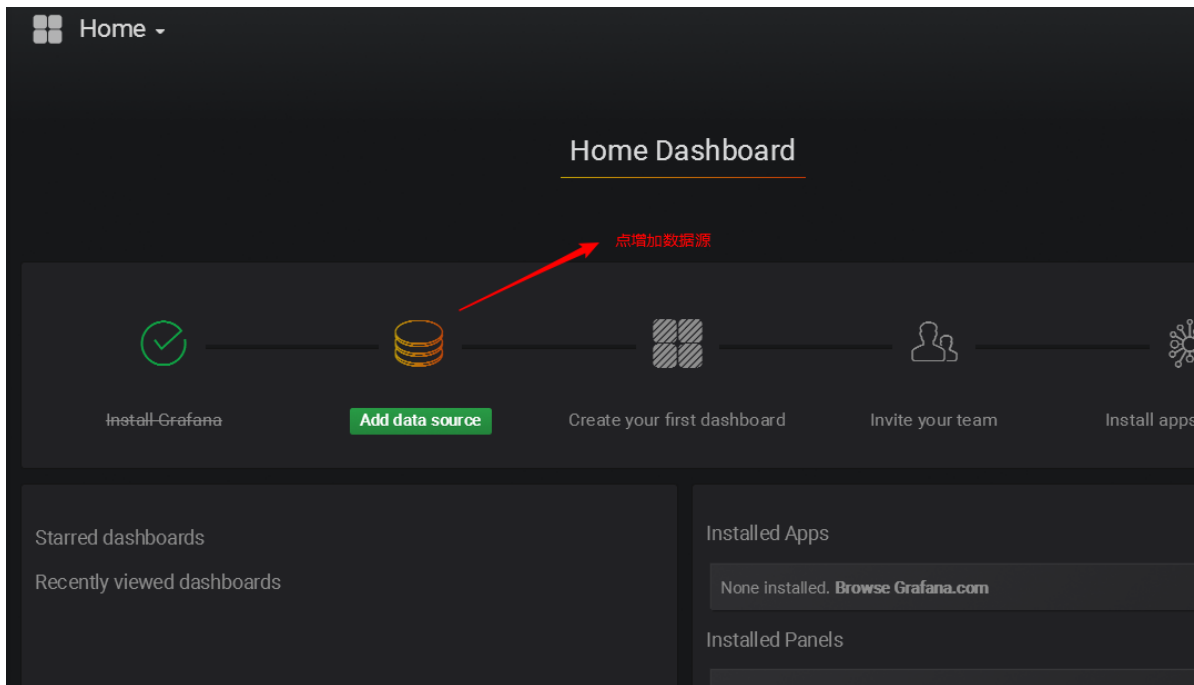
确认端口(3000)


```
[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。



 **Data Sources / New**
Type: Prometheus

Settings

Name

prometheus_data

?

Default

☒

Type

Prometheus

▼

HTTP

URL

http://10.1.1.1:9090

?

Access

Server (Default)

▼

Help ▶

Auth

Basic Auth

☐

With Credentials

?

☐

TLS Client Auth

☐

With CA Cert

?

☐

Auth

Basic Auth

☐

With Credentials

?

☐

TLS Client Auth

☐

With CA Cert

?

☐

Skip TLS Verification (Insecure)

☐

Advanced HTTP Settings

Whitelisted Cookies

Add Name

?

Scrape interval

15s

?

Query timeout

60s

?

HTTP Method

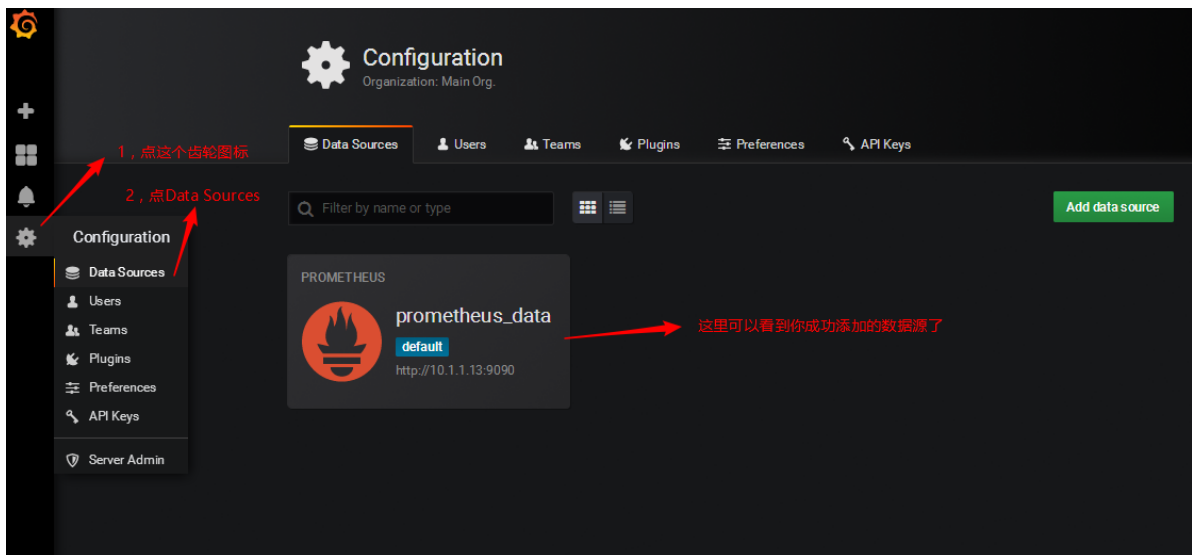
GET

▼

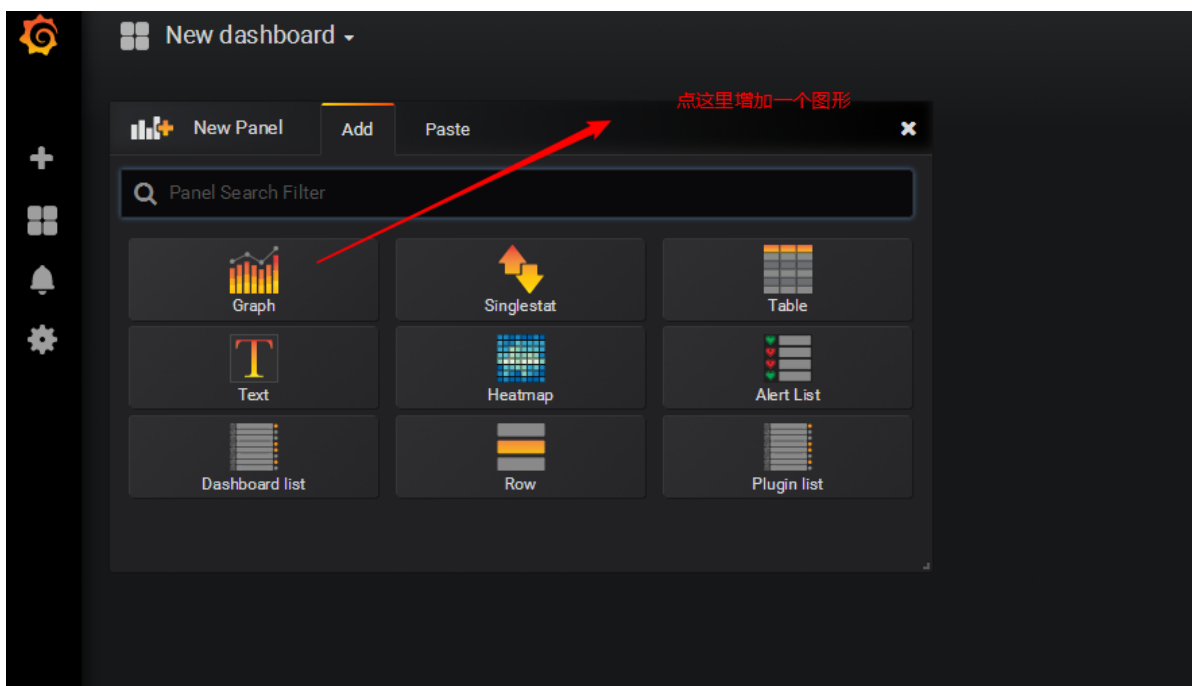
?

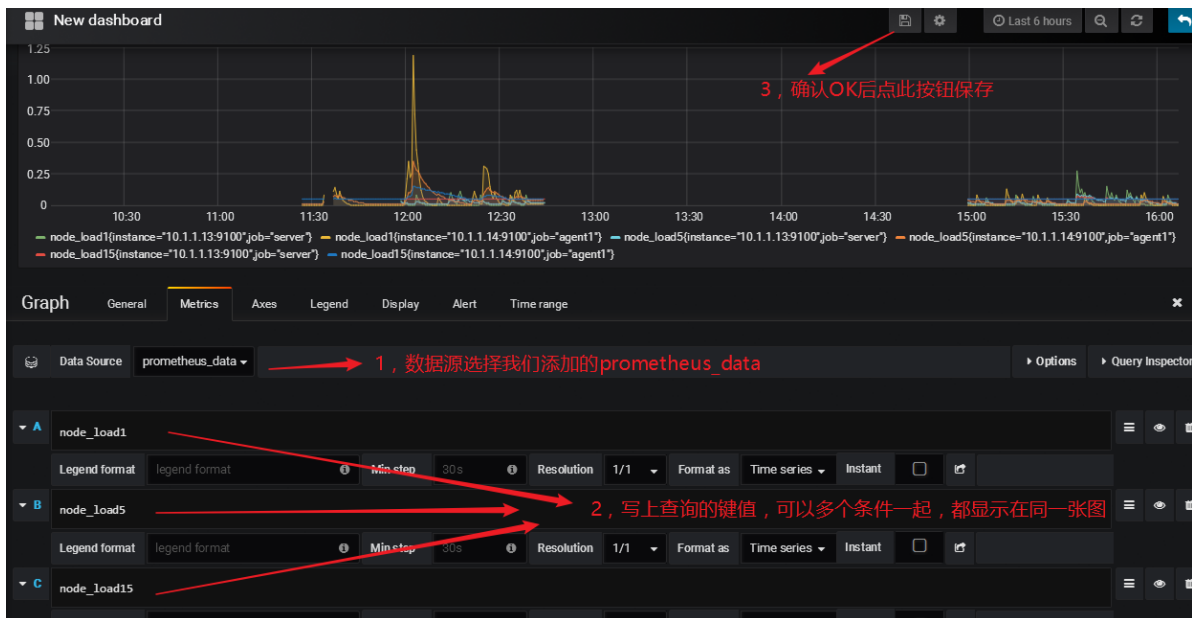
Save & Test

Back

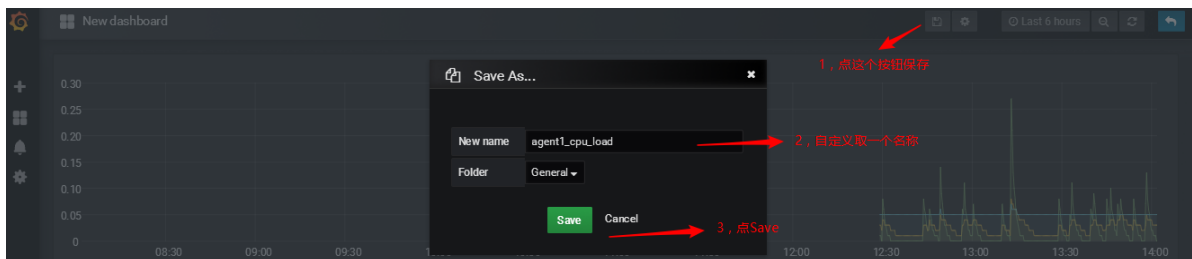


4, 然后为添加好的数据源做图形显示

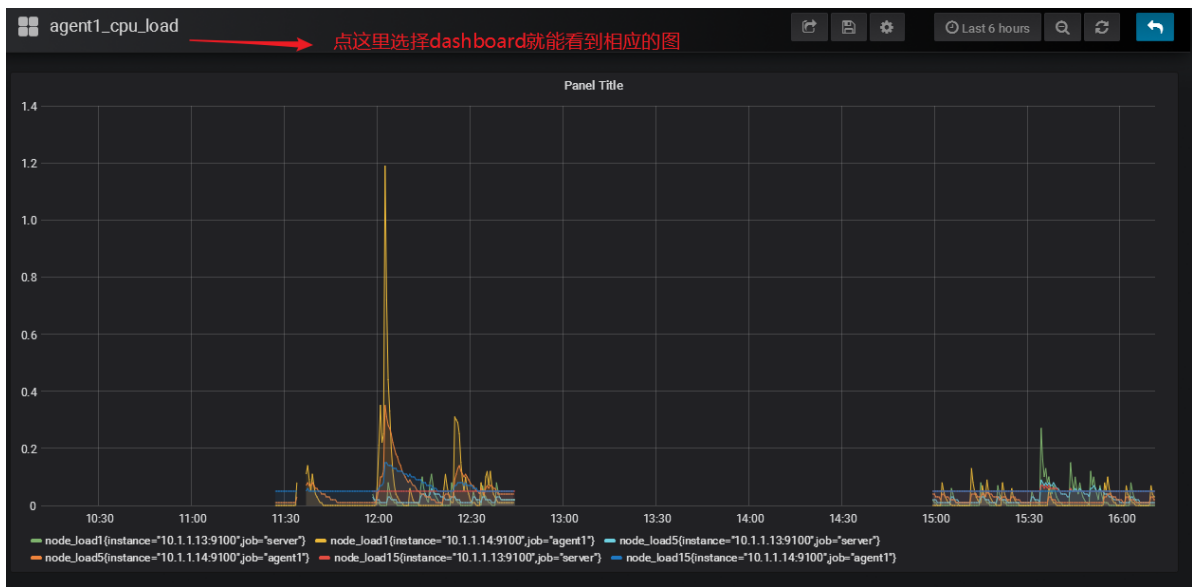




5, 保存



6, 最后在dashboard可以查看到



匹配条件显示



grafana图形显示mysql监控数据

根据上面的思路，我们可以将 `mysql_global_status_threads_connected` 这个metrics加到 dashboard实现对mysql数据库的当前连接数的监控。

但是mysql需要监控的状态非常的多(`mysql> show status` 得到的状态信息几乎都可以监控)，一个个的手动添加太累了。有没有类似zabbix里的模板那种概念呢？

答案是有的,需要开发人员开发出相应的json格式的模板,然后导入进去就可以了。那么问题来了,谁开发？

有这么几种途径：

- 如果公司有这方面的专业开发支持, 就可以实现定制化的监控, 运维工程师配合就好
- 当然运维工程师也可以学习并实现这方面的开发
- 寻找别人开发好的开源项目

grafana-dashboards就是这样的开源项目

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

1, 下载grafana-dashboards开源项目

下载方法:

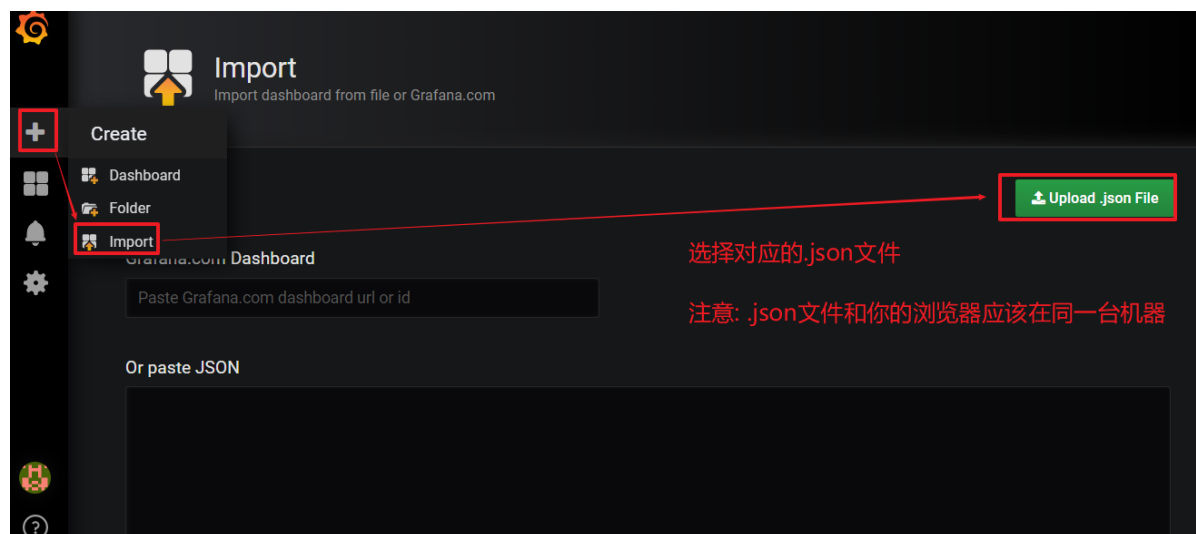
```
# git clone https://github.com/percona/grafana-dashboards.git
```

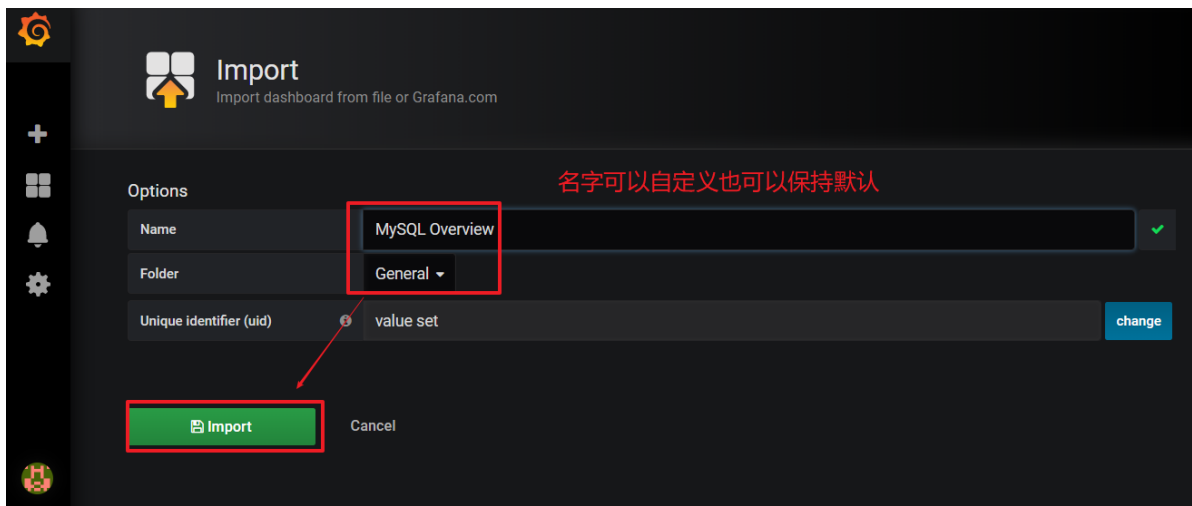
学习完git与github相关课程后就明白了

因为github下载网速非常慢, 我这里已经下载好了分享给大家



2, 在grafana图形界面导入相关json文件





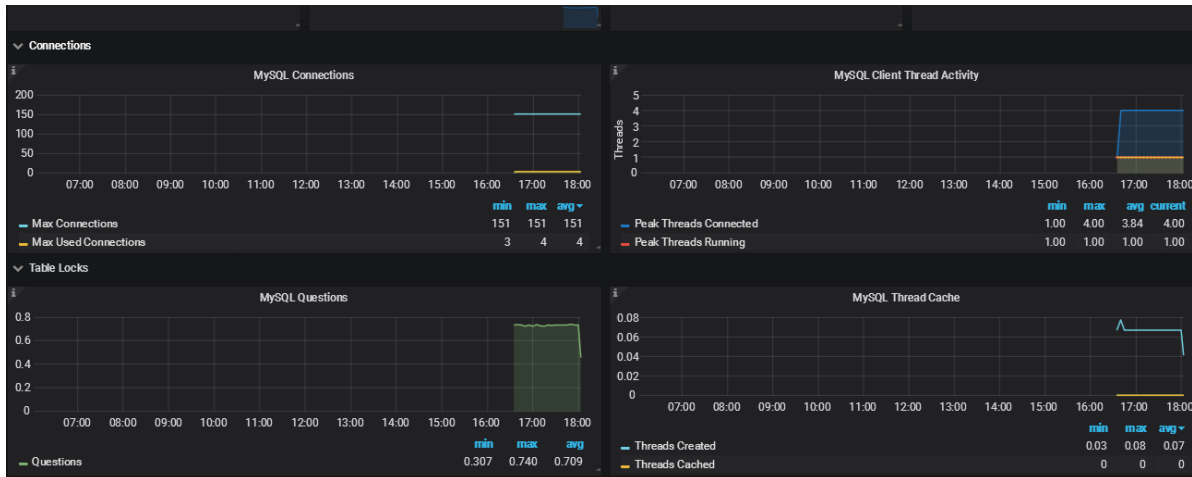
3,点import导入后,报prometheus数据源找不到,因为这些json文件里默认要找的就是叫Prometheus的数据源,但我们前面建立的数据源却是叫prometheus_data

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)



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。

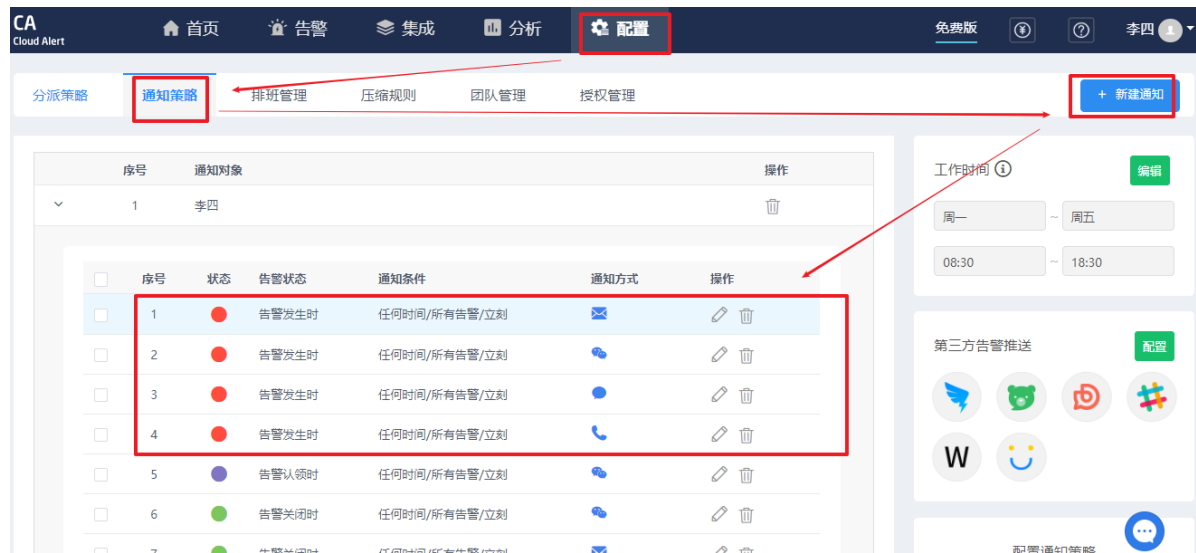
1,先在onealert里添加grafana应用(申请onealert账号在zabbix已经讲过)

The screenshot shows the OneAlert (OneAlert) interface. The top navigation bar includes 'CA Cloud Alert', '首页' (Home), '告警' (Alerts), '集成' (Integrations), and '分析' (Analysis). The '集成' tab is selected, and the '监控工具' (Monitoring Tools) section is highlighted. A red box is drawn around the '增加grafana应用' (Add Grafana Application) button. Below this, a grid of monitoring tools is displayed, including Zabbix, Prometheus, Nagios, Open-Falcon, Amazon, 阿里云 (Alibaba Cloud), solarwinds, 睿象云 (Ruixiang Cloud), 监控宝 (Jianguanbao), Grafana, vmware, Site24x7, Cloud REST API, and Cloud Email. A red box is drawn around the Grafana icon. Below the grid, a table lists the installed applications:

序号	状态	应用名称	类型	appkey	自动关闭	分派策略	操作
1	●	zabbix报警	zabbix	2842d6d7-f7...	30 分钟	点击查看	

Below the table, the configuration page for the 'granfana告警' (Grafana Alert) application is shown. The '应用名称' (Application Name) is 'granfana告警' and the 'AppKey' is '7378c70d-01d6-2b16-aef8-26c93154d432'. The '自动关闭时间' (Auto-close time) is '30 分钟'. A red box is drawn around the 'AppKey' field. The '关联分派' (Associated distribution) is '查看' (View) and the '维护状态' (Maintenance status) is '备注: 无' (Remarks: None). A red box is drawn around the '自定义应用名称, 并产生AppKey' (Customize application name, and generate AppKey) text.

2, 配置通知策略



3, 在grafana增加通知通道

 配置

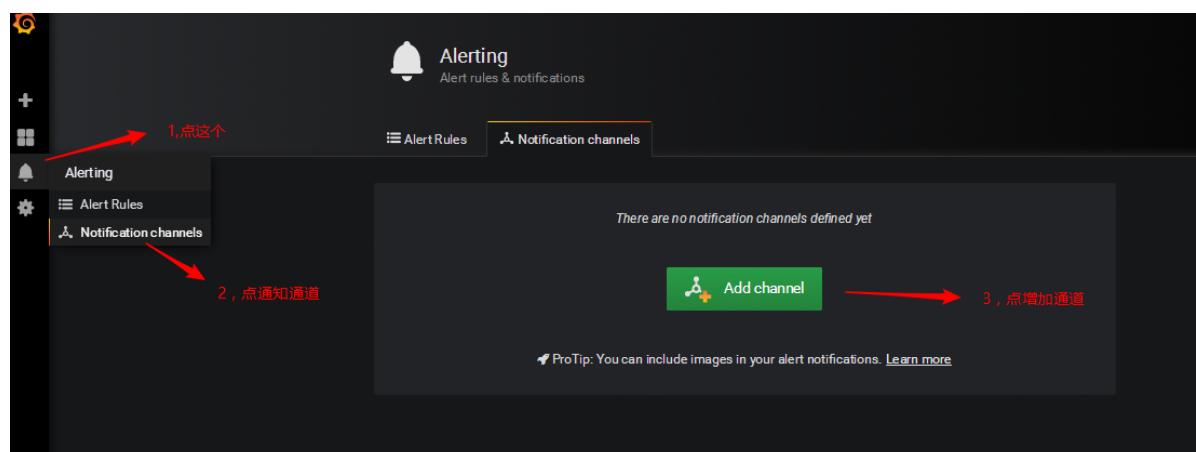
一、在Grafana中配置Webhook URL


- 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，Cloud Alert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.aiops.com/alert/api/event/grafana/v1/7378c70d-01d6-2b16-aef8-26c93154d432/`

- 4、Http Method选择POST；
- 5、Send Test&Save；

参考配置步骤进行配置



 **Alerting**
Alert rules & notifications

Alert Rules | **Notification channels**

New Notification Channel

Name	onealert
Type	webhook
Send on all alerts	<input checked="" type="checkbox"/>
Include image	<input checked="" type="checkbox"/>
Send reminders	<input type="checkbox"/>


Webhook settings

Url	http://api.onealert.com/alert/api/event/grafana/v1/4...
Http Method	POST
Username	
Password	

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

Annotations:

- 名称自定义
- 类型选择webhook
- 这两个勾上
- 这个URL是在onealert那里产生的，复制过来
- 选择POST
- 可以先点一下Send Test测试一下报警媒介是否OK再点Save保存

 **Alerting**
Alert rules & notifications

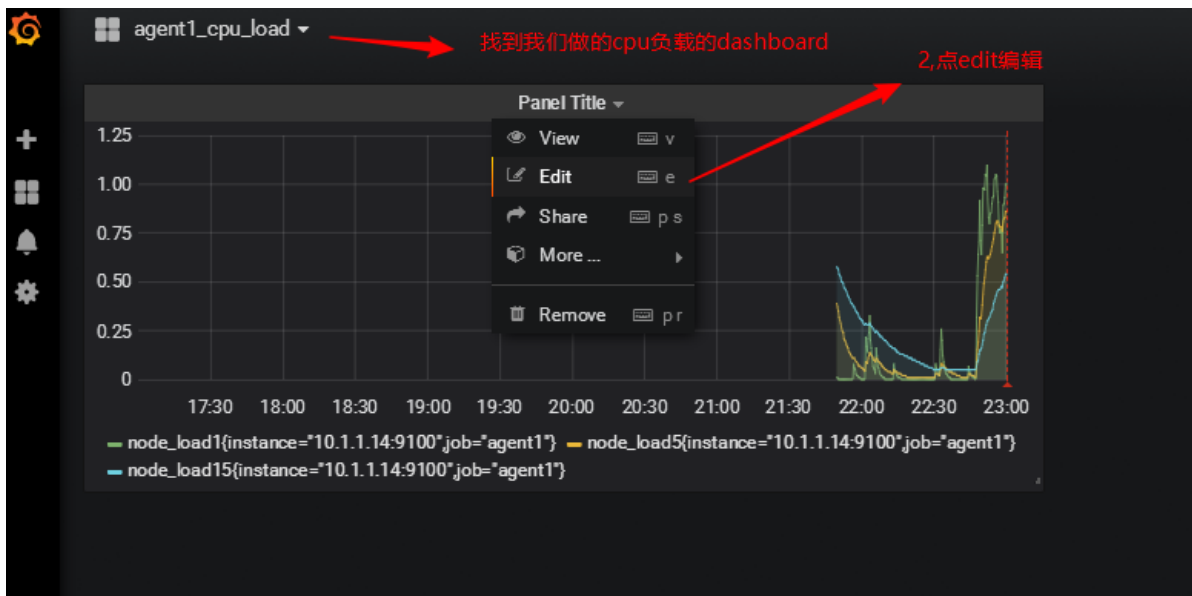
Alert Rules | **Notification channels**

+ New Channel

Name	Type
onealert	webhook

Annotation: 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

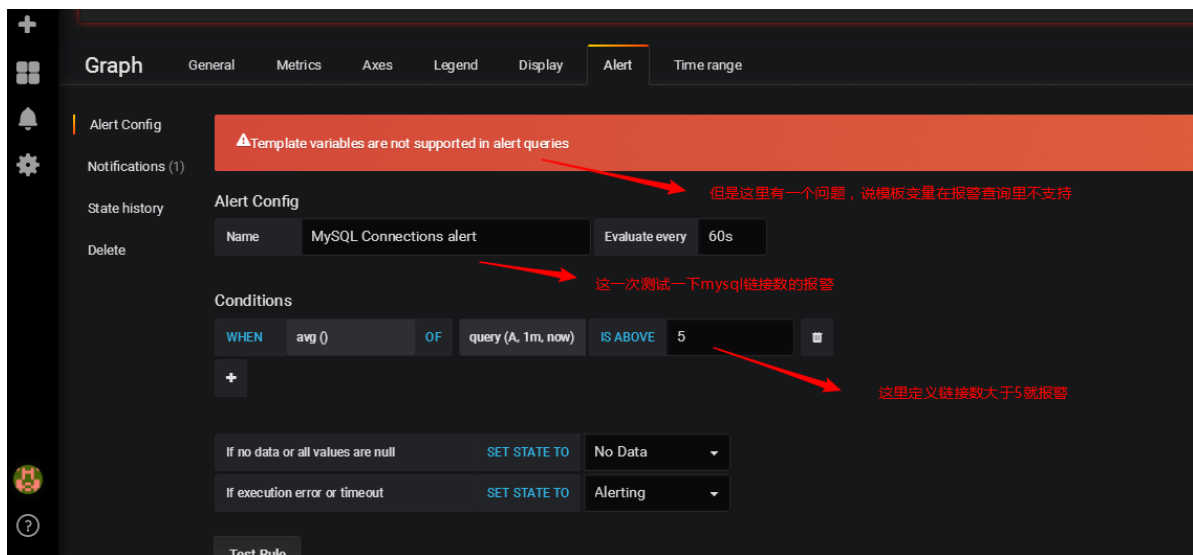
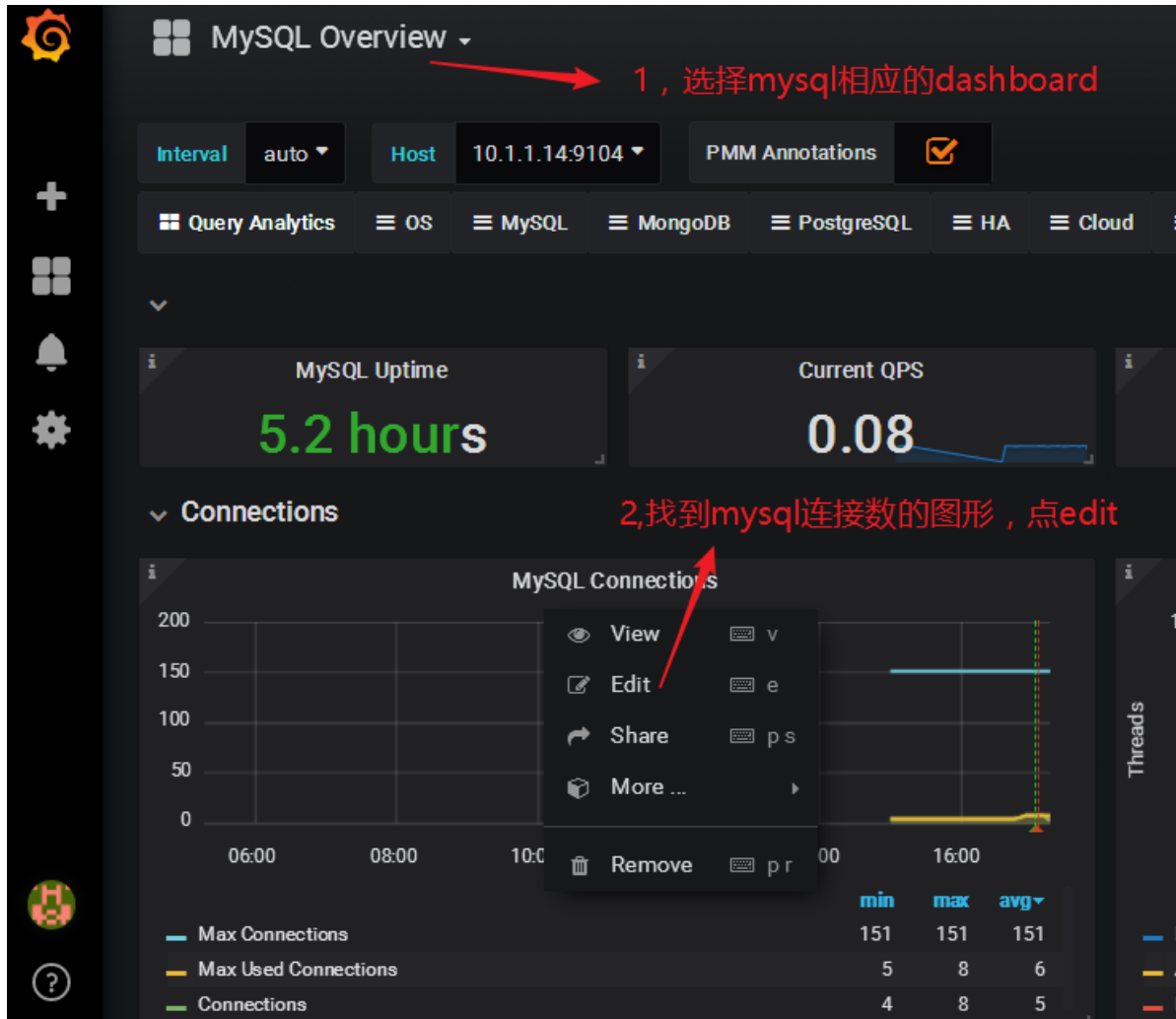
如果agent1上的cpu负载还没有到0.5, 你可以试试0.1,或者运行一些程序把agent1负载调大。最终能测试报警成功。



最终的邮件报警效果



测试mysql连接数报警



将原来的\$host变量替换成被监控的mysql机器的IP与端口

将\$interval改成60s时间间隔

Query Editor (Prometheus)

Query A: `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

Query C: `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

Query B: `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

MySQL Overview

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

Graph: Max Connections (151), Max Used Connections (1)

Alert Config

Notifications (1)

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

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

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。

地址为: <https://grafana.com/grafana/dashboards>

Grafana Labs

Docs Community Events GrafanaCon Blog Log In

Grafana Grafana Cloud Grafana Enterprise **Dashboards** Plugins Get Grafana

1, 官网的Dashboards下有很多官方和社区开发的dashboard可下载使用

Dashboards

Official & community built dashboards

2, 左边可搜索查询

这些都是可供下载的dashboard,一般为json格式,可导入后直接使用

Filter by:

Data Source: All

Panel Type: All

Category: All

Collector: All

1 Node Exporter 0.16 0.17 for Prometheus 监控展示看板 by StarsLn

使用 Node Exporter v0.16 0.17, 精简优化重要指标展示。包含: CPU 内存 磁盘 IO 网络...

Downloads: 3620

1 Node Exporter 0.17 (Host Metrics) by fchiorascu

Dynamic Dashboard covering metrics for: 1 VM, x VMs or all VMs. @Florian-Romel CHIORASCU

Downloads: 732

1. Kubernetes Deployment Statefulset Daemonset metrics by prat0318

Monitors Kubernetes deployments in cluster using Prometheus. Shows overall cluster CPU / Memory...

Downloads: 7054

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)

