GPE监控+ Docker+Docker Compose

# 1前言

搭建一个多维度监控微服务的可视化平台，包括Docker容器监控、MySQL监控、Redis监控和微服务JVM监控等，并且在必要的情况下可以发送预警邮件。

主要用到的组件有Prometheus、Grafana、alertmanager、node\_exporter、mysql\_exporter、redis\_exporter、cadvisor。各自作用如下所示：

1、Prometheus：获取、存储监控数据，供第三方查询；

2、Grafana：提供Web页面，从Prometheus获取监控数据可视化展示；

3、alertmanager：定义预警规则，发送预警信息；

4、node\_exporter：收集微服务端点监控数据；

5、mysql\_exporter：收集MySQL数据库监控数据；

6、redis\_exporter：收集Redis监控数据；

7、cadvisor：收集Docker容器监控数据。

# 2 准备工作

传统方式部署这么多软件比较麻烦，所以还是建议使用Docker来安装。

本次涉及到MySQL和Redis的监控，如果你的linux机器上还没有安装MySQL和Redis，可以使用docker compose创建并运行这两个容器：

version: '3'

services:

mysql:

image: mysql:5.7.24

container\_name: mysql

environment:

MYSQL\_ROOT\_PASSWORD: 123456

ports:

- 3306:3306

volumes:

- / /mysql/data:/var/lib/mysql #挂载 MySQL数据

redis:

image: redis:4.0.14

container\_name: redis

command: redis-server /usr/local/etc/redis/redis.conf --appendonly yes

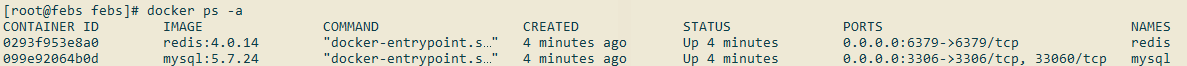
volumes:

- /wm/redis/data:/data #挂载 Redis数据

- /wm/redis/conf/redis.conf:/usr/local/etc/redis/redis.conf #挂载 Redis配置

ports:

- 6379:6379



# 3 接入Prometheus

微服务接入Pormetheus只需要在common模块引入下面这个依赖即可：

<dependency>

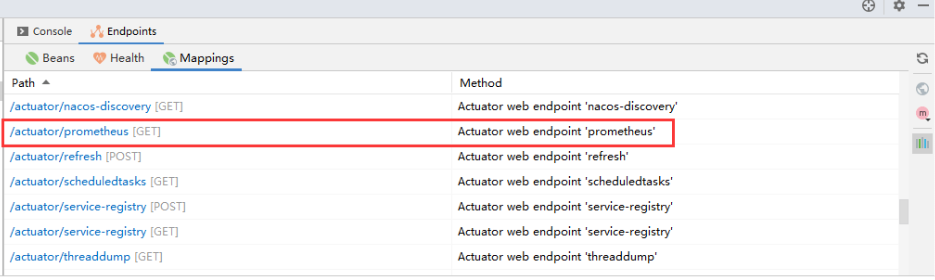
<groupId>io.micrometer</groupId>

<artifactId>micrometer-registry-prometheus</artifactId>

</dependency>

Prometheus端点的暴露依赖于spring-boot-starter-actuator，因为之前在 common模块里引入了spring-boot-admin-client依赖，该依赖包含spring-boot-starter-actuator，所以无需再额外引入了。

启动相关的微服务，然后通过IDEA查看是否已经包含/actuator/prometheus端点：



接着看服务器，因为接着需要创建较多的服务，所以需要先预定下个服务的端口。

|  |  |
| --- | --- |
| **服务** | **端口** |
| Prometheus | 8403 |
| Grafana | 8404 |
| Node\_exporter | 8405 |
| AlertManager | 8406 |
| Cadvisor | 8407 |
| Redis\_expoter | 8408 |
| Mysql\_expoter | 8409 |

promethues.yml配置文件内容如下所示：

global:

scrape\_interval: 15s

evaluation\_interval: 15s

scrape\_configs:

- job\_name: 'prometheus'

static\_configs:

- targets: ['192.168.33.10:8403']

- job\_name: 'node\_exporter'

static\_configs:

- targets: ['192.168.33.10:8405']

labels:

instance: 'node\_exporter'

- job\_name: 'redis\_exporter'

static\_configs:

- targets: ['192.168.33.10:8408']

labels:

instance: 'redis\_exporter'

- job\_name: 'mysql\_exporter'

static\_configs:

- targets: ['192.168.33.10:8409']

labels:

instance: 'mysql\_exporter'

- job\_name: 'cadvisor'

static\_configs:

- targets: ['192.168.33.10:8407']

labels:

instance: 'cadvisor'

- job\_name: 'wm-server-system-actuator'

metrics\_path: '/actuator/prometheus'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.73.42:8201']

labels:

instance: 'wm-server-system'

- job\_name: 'wm-server-auth-actuator'

metrics\_path: '/actuator/prometheus'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.73.42:8101']

labels:

instance: wm-auth'

- job\_name: 'wm-server-test-actuator'

metrics\_path: '/actuator/prometheus'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.73.42:8202']

labels:

instance: wm-server-test'

rule\_files:

- 'memory\_over.yml'

- 'server\_down.yml'

alerting:

alertmanagers:

- static\_configs:

- targets: ["192.168.33.10:8406"]

配置的具体含义如下图：





值得注意的是，每个服务的targets都是一个数组，言外之意就是可以收集多个服务器下的exporter提供的监控数据。

接着创建上面提到的两个监控规则memory\_over.yml和server\_down.yml：

# 创建memory\_over.yml

vim /wm/prometheus/memory\_over.yml

内容如下所示:

groups:

- name: memory\_over

rules:

- alert: NodeMemoryUsage

expr: (node\_memory\_MemTotal\_bytes - (node\_memory\_MemFree\_bytes+node\_memory\_Buffers\_bytes+node\_memory\_Cached\_bytes )) / node\_memory\_MemTotal\_bytes \* 100 > 80

for: 20s

labels:

user: user

annotations:

summary: "{{$labels.instance}}: High Memory usage detected"

description: "{{$labels.instance}}: Memory usage is above 80% (current value is:{{ $value }})"

含义是，当某个节点的内存使用率大于80%，并且持续时间大于20秒后，触发监控预警。

接着创建server\_down.yml：

vim /wm/prometheus/server\_down.yml

**内容如下所示：**

groups:

- name: server\_down

rules:

- alert: InstanceDown

expr: up == 0

for: 20s

labels:

user: wanggq

annotations:

summary: "Instance {{ $labels.instance }} down"

description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 20 s."

含义是，当某个节点宕机（up==0表示宕机，1表示正常运行）超过20秒后，则触发监控。

接着创建alertmanager相关挂载目录和配置：

mkdir -p /wm/alertmanager

vim /wm/alertmanager/alertmanager.yml

alertmanager.yml内容如下所示：

global:

smtp\_smarthost: 'smtp.126.com:465'

smtp\_from: 'user@126.com'

smtp\_auth\_username: user

smtp\_auth\_password: '123456'

smtp\_require\_tls: false

#templates:

# - '/alertmanager/template/\*.tmpl'

route:

group\_by: ['alertname']

group\_wait: 10s

group\_interval: 5m

repeat\_interval: 5m

receiver: 'default-receiver'

receivers:

- name: 'default-receiver'

email\_configs:

- to: xxx@qq.com'

send\_resolved: true

配置文件的含义如下：



其中，smtp\_auth\_password并不是邮件的密码，而是授权码，以126为例，授权码可以在下面这个页面定义：



创建好后，接着在目录下创建一个docker-compose.yml文件，用于构建上面提到的那些服务：

# 创建docker-compose.yml

vim /wm/docker-compose.yml

内容如下所示：

version: '3'

services:

prometheus:

image: prom/prometheus

container\_name: prometheus

volumes:

- /wm/prometheus/:/etc/prometheus/

ports:

- 8403:9090

restart: on-failure

node\_exporter:

image: prom/node-exporter

container\_name: node\_exporter

ports:

- 8405:9100

redis\_exporter:

image: oliver006/redis\_exporter

container\_name: redis\_exporter

command:

- '--redis.addr=redis://192.168.33.10:6379'

ports:

- 8408:9121

restart: on-failure

mysql\_exporter:

image: prom/mysqld-exporter

container\_name: mysql\_exporter

environment:

- DATA\_SOURCE\_NAME=root:123456@(192.168.33.10:3306)/

ports:

- 8409:9104

cadvisor:

image: google/cadvisor

container\_name: cadvisor

volumes:

- /:/rootfs:ro

- /var/run:/var/run:rw

- /sys:/sys:ro

- /var/lib/docker/:/var/lib/docker:ro

ports:

- 8407:8080

alertmanager:

image: prom/alertmanager

container\_name: alertmanager

volumes:

- /wm/alertmanager/alertmanager.yml:/etc/alertmanager/alertmanager.yml

ports:

- 8406:9093

grafana:

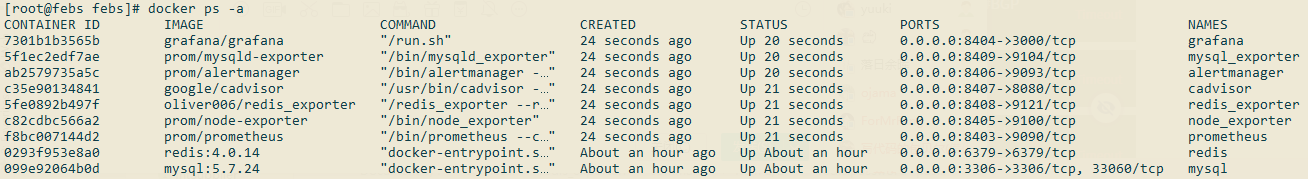
image: grafana/grafana

container\_name: grafana

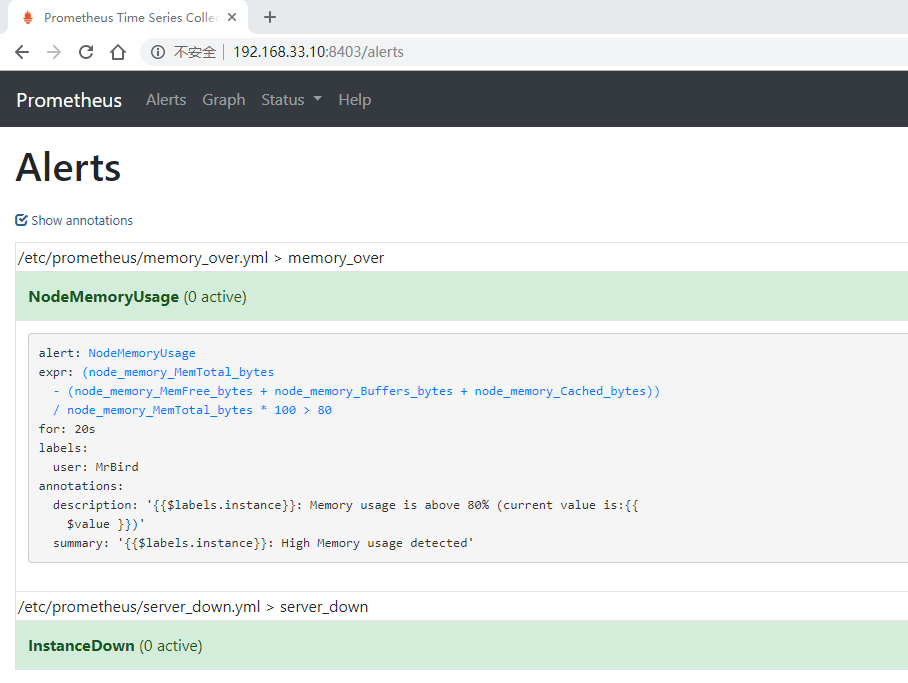
ports:

- 8404:3000

其中，在创建redis\_exporter和mysql\_exporter的时候需要指定它们的IP、端口和密码等配置。然后在 指定目录下使用docker-compose up -d启动这些容器：

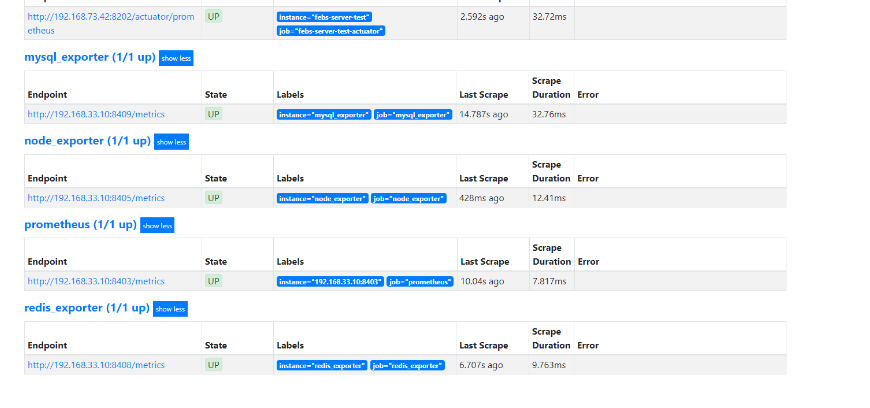


容器运行成功后，在浏览器里访问：<http://192.168.33.10:8403/alerts>：



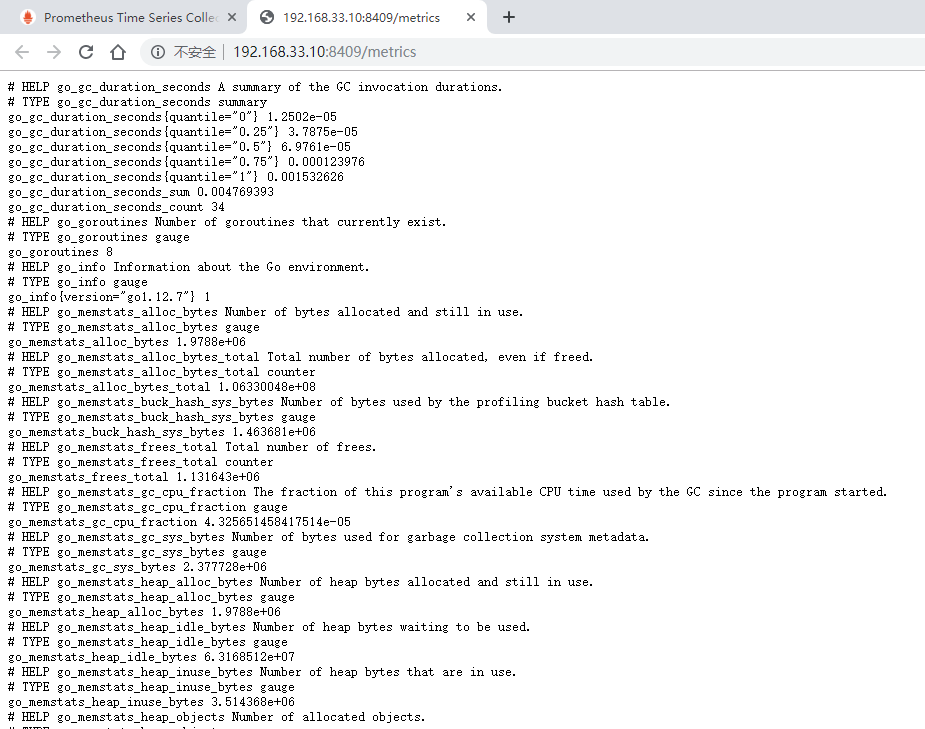
可以看的刚刚定义的两个告警规则已经成功加载。

访问<http://192.168.33.10:8403/targets>，观察在Prometheus配置文件里定义的各个job的状态：



可以看的都是监控的UP状态。

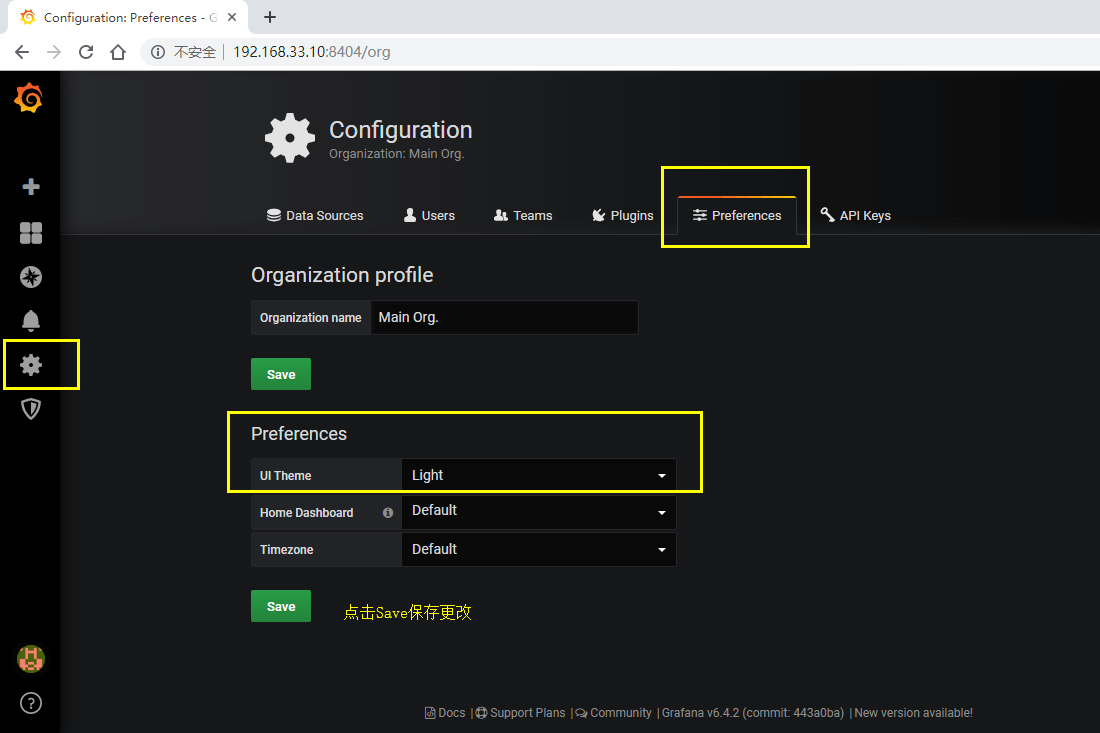
还可以点击上面这个页面的各个xxx\_exporter的链接，如果页面显示出了收集的数据，则说明各个exporter已经成功采集到了数据，以mysql\_exporter为例子，访问<http://192.168.33.10:8409/metrics>：

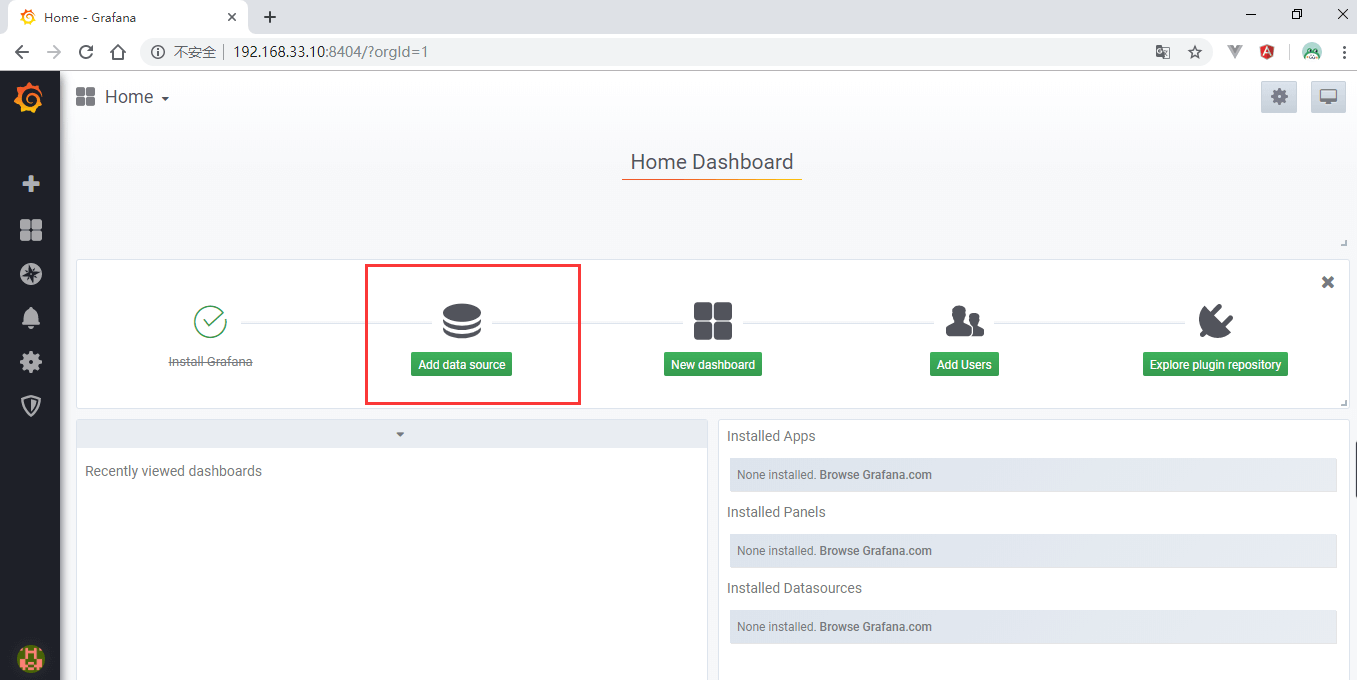


访问<http://192.168.33.10:8406/#/status> 查看状态

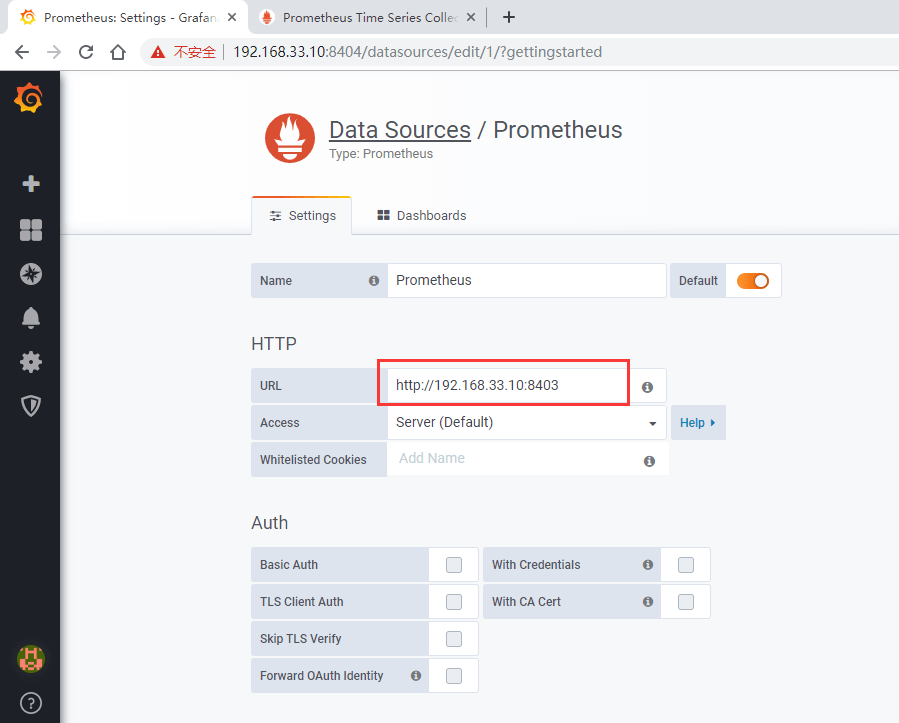
# 4 Grafana的使用

使用浏览器访问[http://192.168.33.10:8404](http://192.168.33.10:8404/)，用户名密码为admin/admin，首次登录需要修改密码。进入主页后，我们修改下主题，让接下来的演示更为清晰：

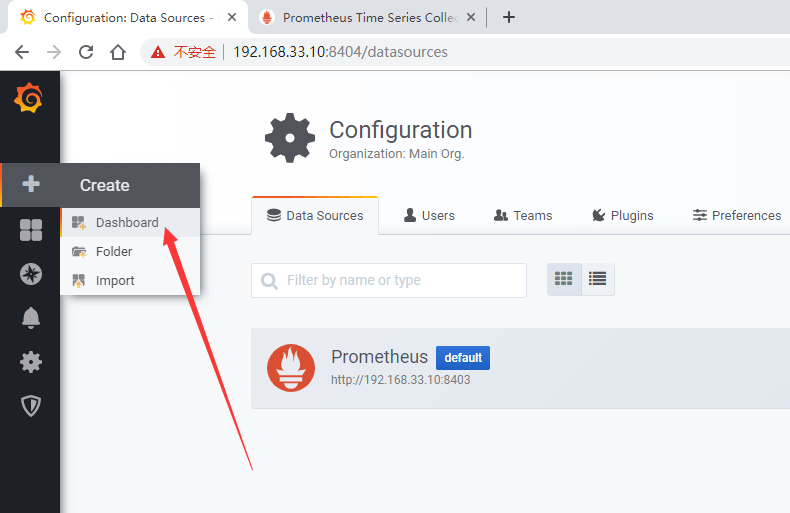


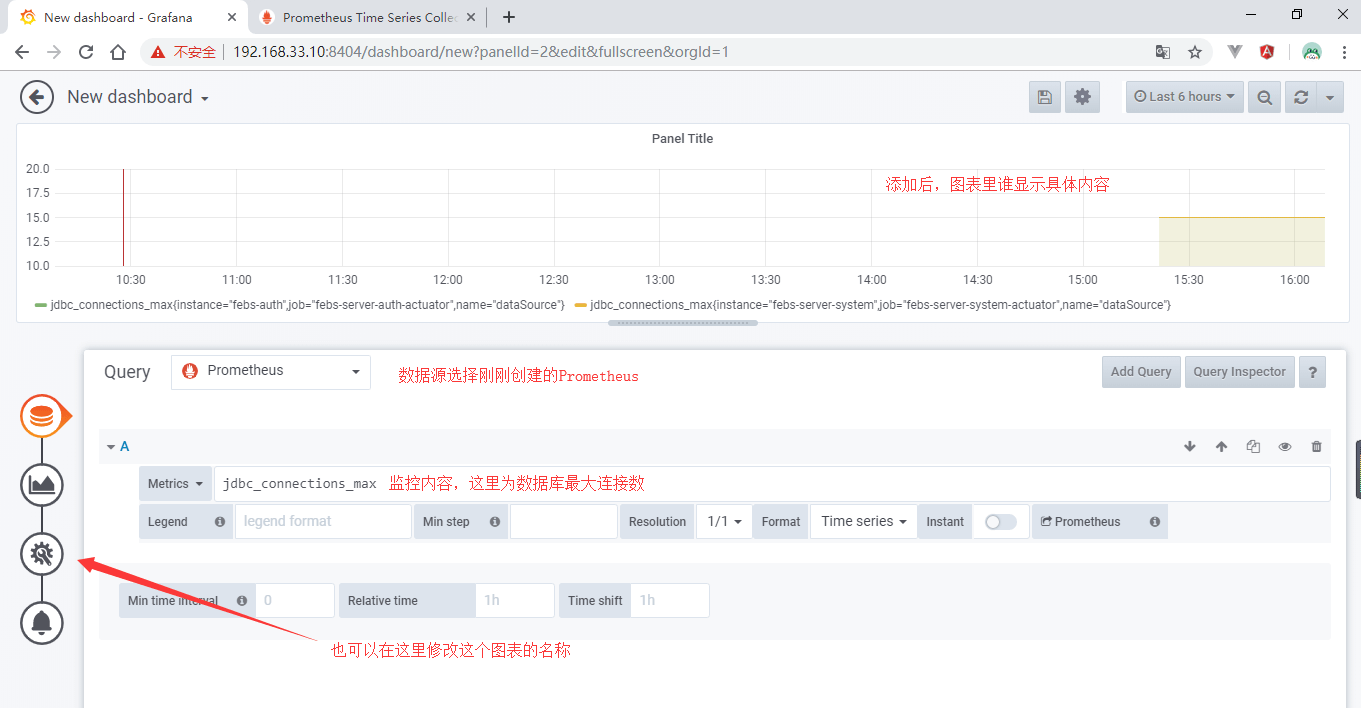


选择Prometheus，

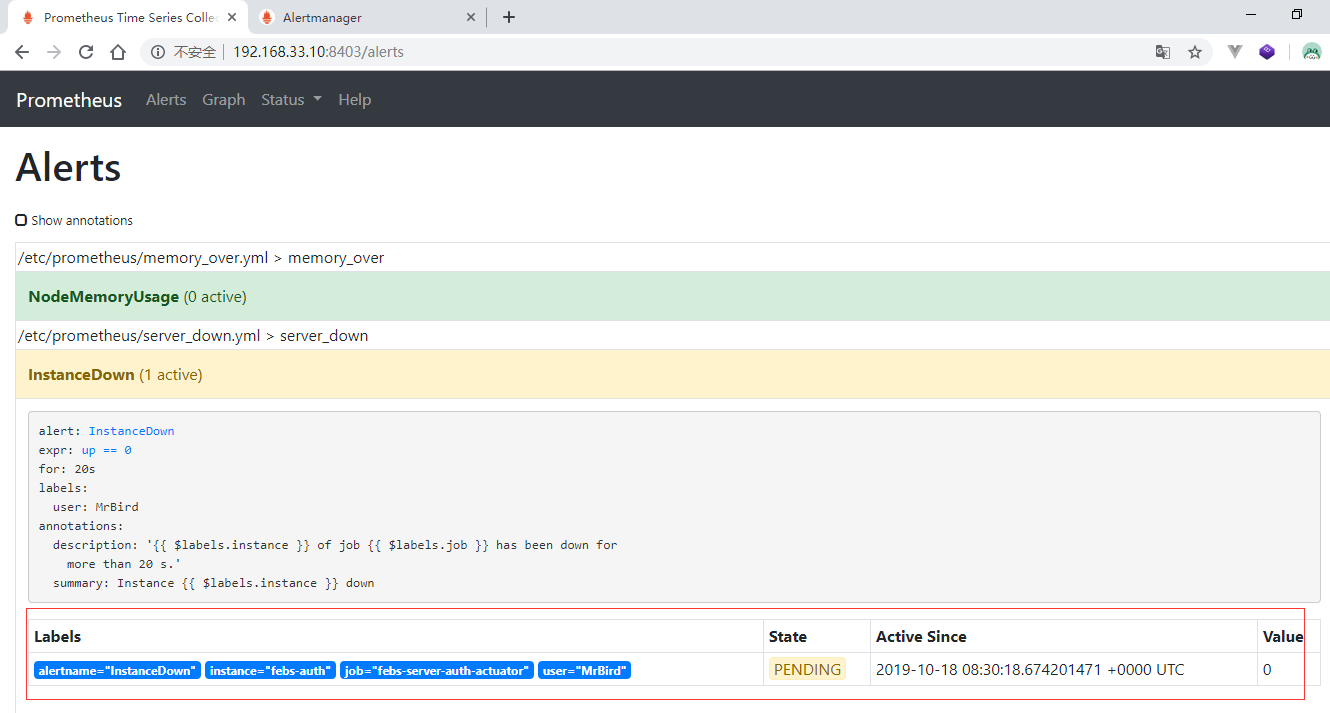


数据源添加好后，如下图所示创建一个新的监控：

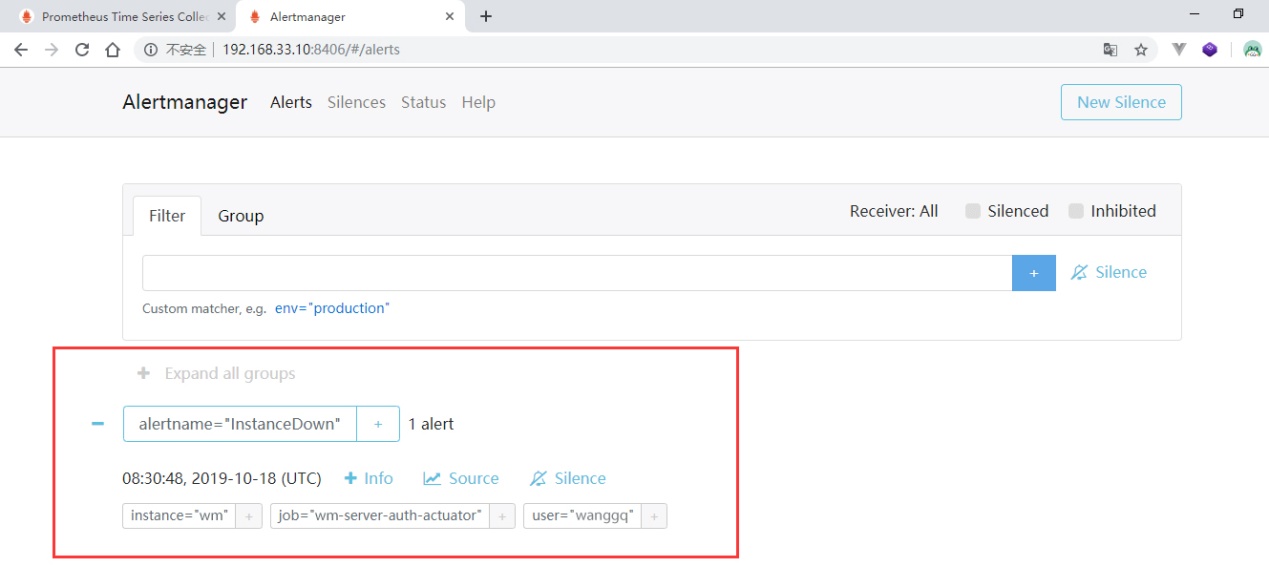




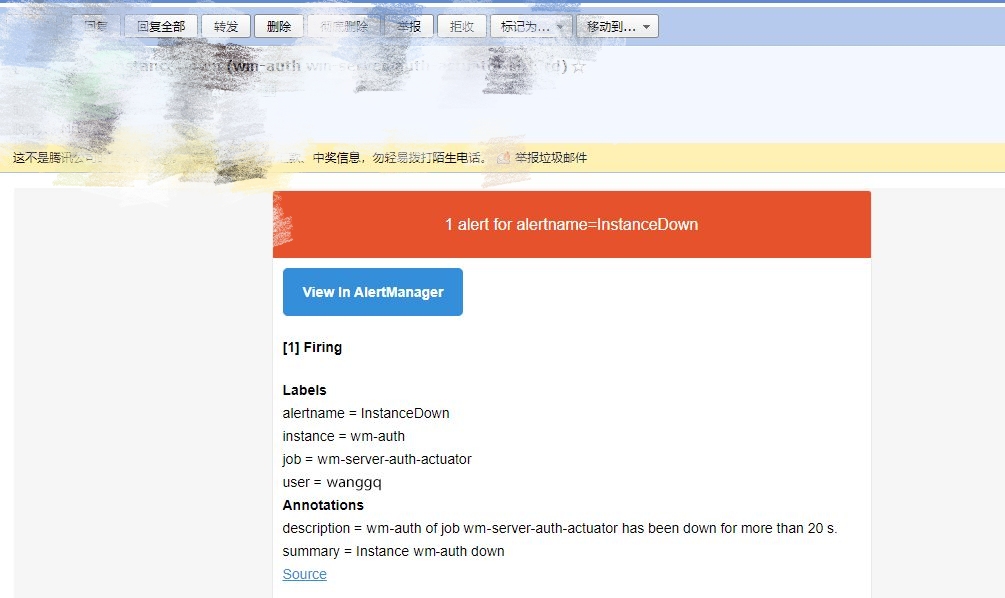
# 5测试警告



该端点已经处于PENDING（等待）状态，20秒后（server\_down.yml里定义的时间）应用还是不可用的话，则认为该服务已经是（宕机）状态，则会触发server\_down预警：



过一会（5秒的扫描时间+20秒的等待确认时间+10秒的分组等待时间）你的邮箱将收到如下所示的邮件：



重新启动wm- auth服务，不久后你将收到resolved（告警已解决）邮件：

