搭建多维度监控体系

建立日期：2020年8月3日

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 版本 | 变更原因 | 变更内容简述 | 编制/修订者 | 适用范围 | 发布日期 |
| V1.0 | 建立 |  | 陈文华 | 开发、测试 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 搭建Prometheus

Master节点内存不是很足，在node2节点（192.168.84.47）搭建多维度监控服务。

prometheus 官网 <https://prometheus.io/>

### 引入依赖

在common模块引入以下依赖

<dependency>

<groupId>io.micrometer</groupId>

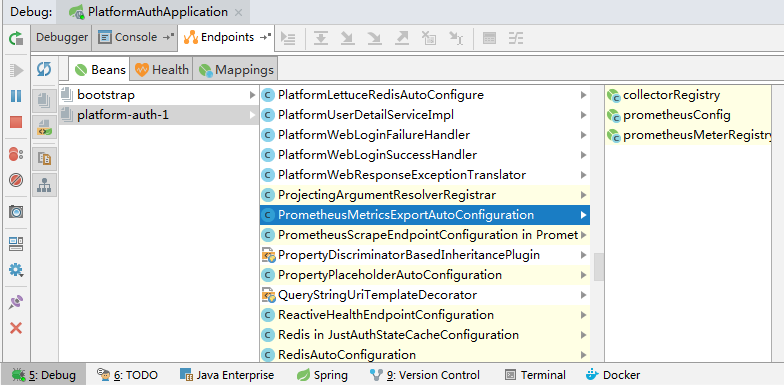
<artifactId>micrometer-registry-prometheus</artifactId>

</dependency>

Spring Boot使用Micrometer，一个应用metrics组件，将actuator metrics整合到外部监控系统中。一旦你增加上述的依赖，Spring Boot会自动配置一个PrometheusMeterRegistry和CollectorRegistry来收集和输出格式化的metrics数据，使得Prometheus服务器可以爬取。

所有应用的metrics数据是根据一个叫/prometheus的endpoint来设置是否可用。Prometheus服务器可以周期性的爬取这个endpoint来获取metrics数据。

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，所以无需再引入了。



### 2、服务列表

下表是接下来需要创建的服务，由于服务比较多，所以先设置下各个服务的端口号，方便管理。

|  |  |  |
| --- | --- | --- |
| 服务 | 说明 | 端口 |
| prometheus | 收集、分析监控的数据，给服务提供查询接口 | 8403 |
| grafana | 从Prometheus获取监控数据并进行可视化展示 | 8404 |
| node\_exporter | 收集微服务监控数据 | 8405 |
| alertmanager | 配置预警规则，发送告警信息 | 8406 |
| cadvisor | 收集Docker监控数据 | 8407 |
| redis\_exporter | 收集Redis监控数据 | 8408 |
| mysql\_exporter | 收集Mysql监控数据 | 8409 |

### 3、创建Prometheus挂载目录和配置文件

创建Prometheus挂载目录

mkdir -p /platform/prometheus

创建Prometheus配置文件

vim /platform/prometheus/prometheus.yml

内容如下

global:

scrape\_interval: 15s

evaluation\_interval: 15s

scrape\_configs:

- job\_name: 'prometheus'

static\_configs:

- targets: ['192.168.84.47:8403']

- job\_name: 'node\_exporter'

static\_configs:

- targets: ['192.168.84.47:8405']

labels:

instance: 'node\_exporter'

- job\_name: 'redis\_exporter'

static\_configs:

- targets: ['192.168.84.47:8408']

labels:

instance: 'redis\_exporter'

- job\_name: 'mysql\_exporter'

static\_configs:

- targets: ['192.168.84.47:8409']

labels:

instance: 'mysql\_exporter'

- job\_name: 'cadvisor'

static\_configs:

- targets: ['192.168.84.47:8407']

labels:

instance: 'cadvisor'

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

metrics\_path: '/actuator/prometheus'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.108.151:8201']

labels:

instance: 'platform-server-system'

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

metrics\_path: '/actuator/prometheus'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.108.151:8101']

labels:

instance: 'platform-auth'

rule\_files:

- 'memory\_over.yml'

- 'server\_down.yml'

alerting:

alertmanagers:

- static\_configs:

- targets: ["192.168.84.47:8406"]

targets指定服务器的ip地址和端口，labels:instance指定当前服务的名称，auth，system和gateway服务配置为我本机192.168.108.151的地址和端口，如果在84.48上部署了服务，则修改为服务器地址。

### 4、创建监控规则

创建memory\_over.yml

vim /platform/prometheus/memory\_over.yml

内容如下

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: crystal

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。

创建server\_down.yml

vim /platform/prometheus/server\_down.yml

内容如下

groups:

- name: server\_down

rules:

- alert: InstanceDown

expr: up == 0

for: 20s

labels:

user: crystal

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。

### 5、创建alertmanager挂载目录和配置文件

创建alertmanager挂载目录

mkdir -p /platform/alertmanager

创建alertmanager配置文件

vim /platform/alertmanager/alertmanager.yml

内容如下所示

global:

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

smtp\_from: '764471698@qq.com'

smtp\_auth\_username: '764471698'

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: 'chenwenhua666@aliyun.com'

send\_resolved: true

smtp\_auth\_password填写qq邮箱授权码即可，receivers指定接收者。

### 6、构建服务

创建docker-compose.yml用于构建以上服务

vim /platform/docker-compose.yml

内容如下

version: '3'

services:

prometheus:

image: prom/prometheus

container\_name: prometheus

volumes:

- /platform/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://192.168.84.47:6379'

- '--redis.password=123456'

ports:

- 8408:9121

restart: on-failure

mysql\_exporter:

image: prom/mysqld-exporter

container\_name: mysql\_exporter

environment:

- DATA\_SOURCE\_NAME=root:123456@(192.168.84.47:3306)/

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:

- /platform/alertmanager/alertmanager.yml:/etc/alertmanager/alertmanager.yml

ports:

- 8406:9093

grafana:

image: grafana/grafana

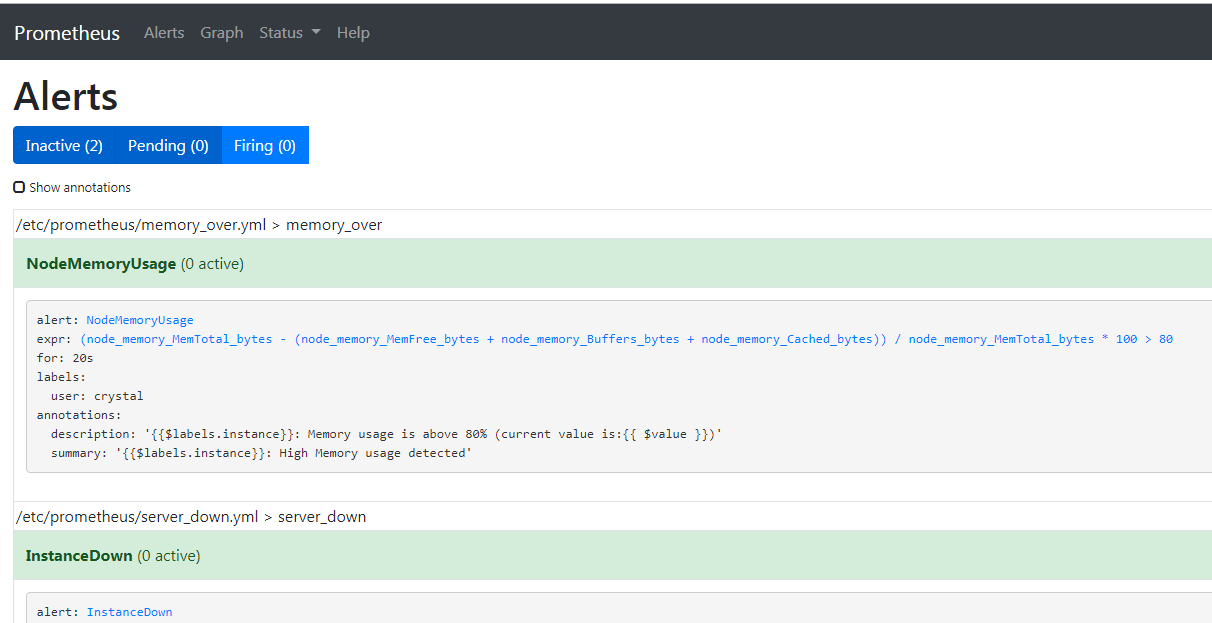
container\_name: grafana

ports:

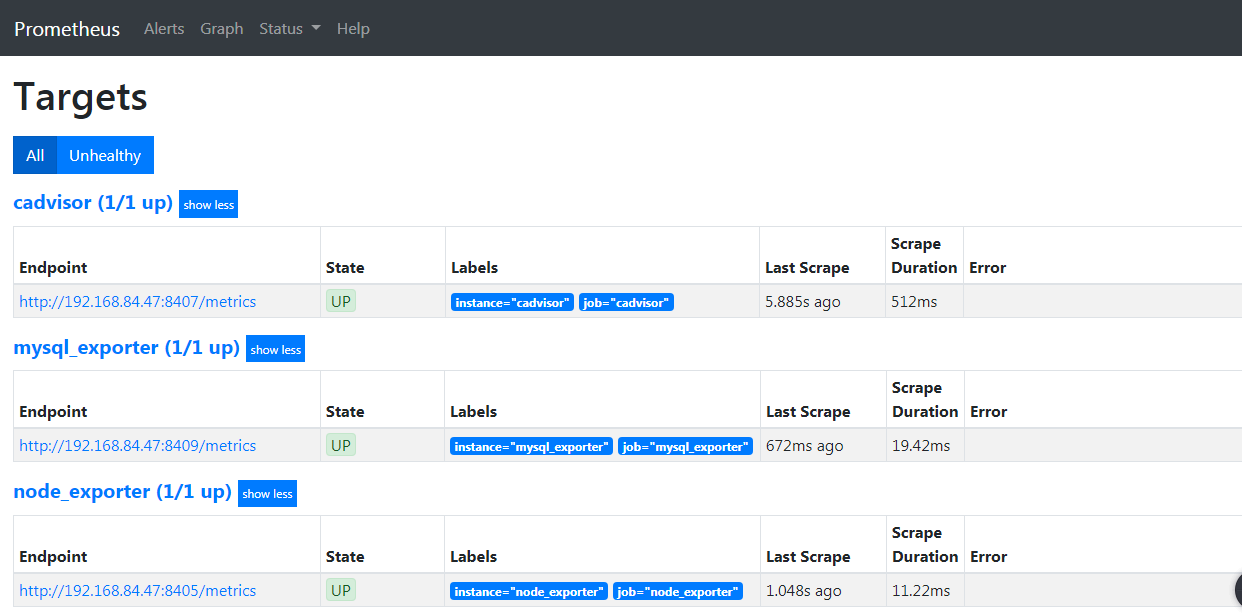
- 8404:3000

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。执行docker-compose up -d启动容器。

访问 <http://192.168.84.47:8403/alerts> 查看预警规则



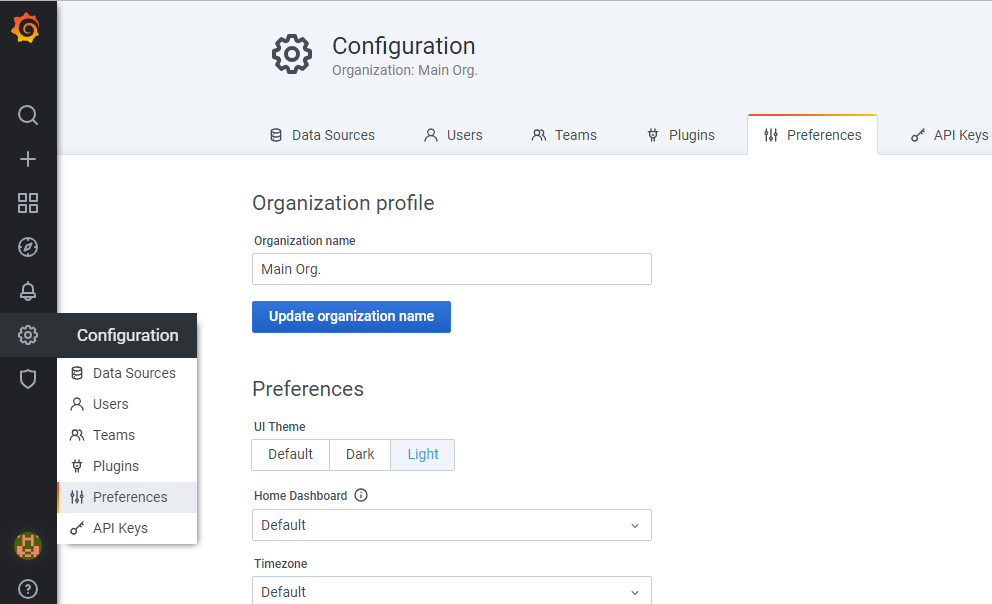
访问 <http://192.168.84.47:8403/targets> 查看监控状态



## 配置Grafana

### 1、登陆

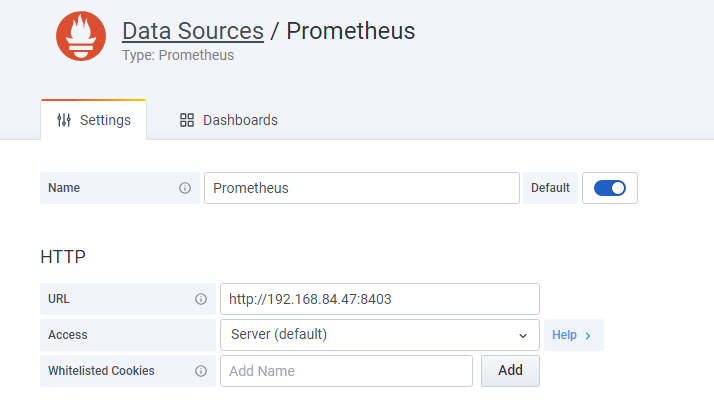
浏览器访问http://192.168.84.47:8404，用户名密码为admin/admin，首次登录需要修改密码。默认UI为暗黑风格，可修改为明亮风格



### 2、添加数据源

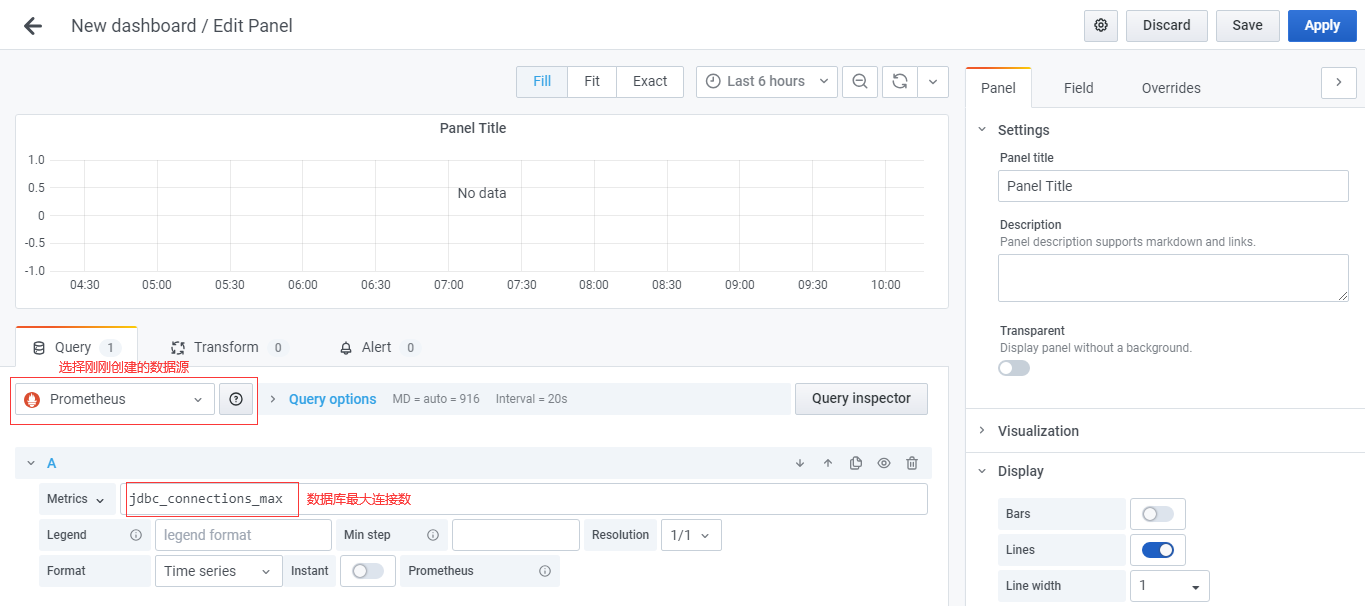
主页选择Add data source，Time series databases选择Prometheus，

填写 <http://192.168.84.47:8403> 点击Save&Test



### 3、添加Dashboard

主页选择Dashboards -> Add new panel

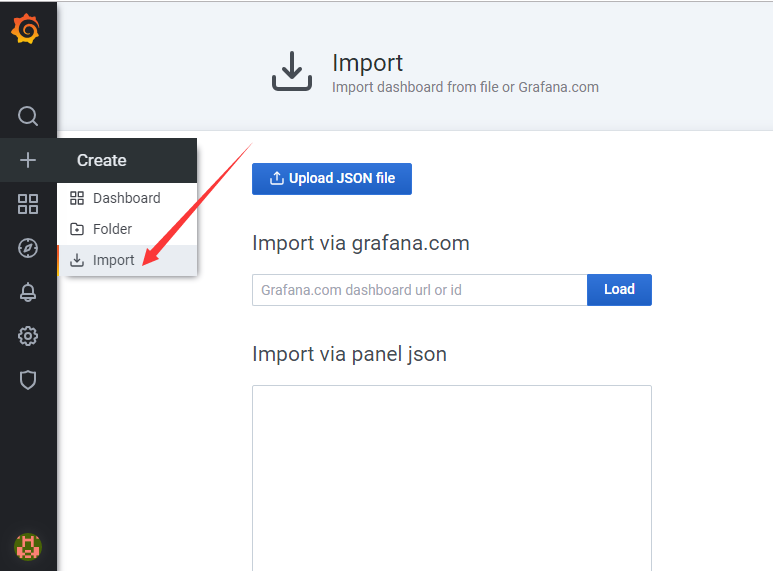


### 4、创建监控面板

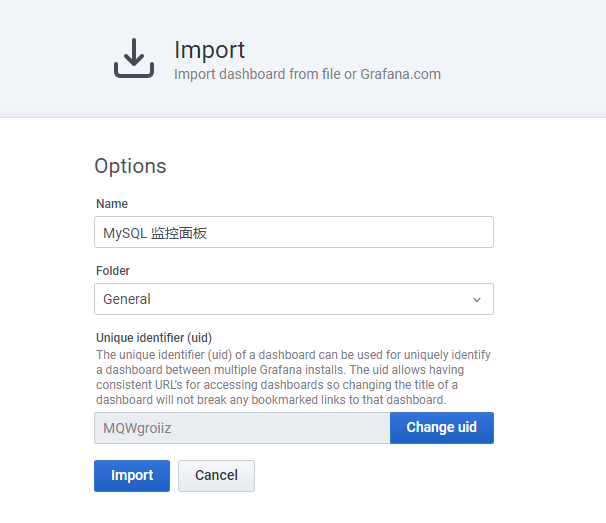
以上的方式创建监控面板比较麻烦，可以去官网下载模板

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

导入模板



选择导入JSON文件

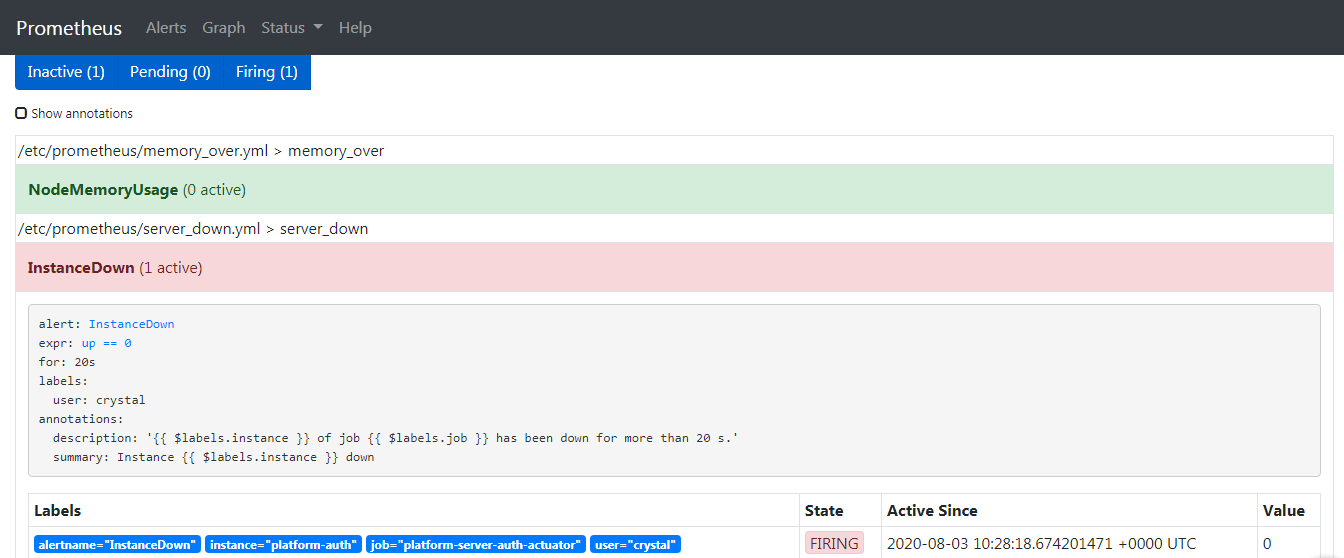


## 三、测试预警

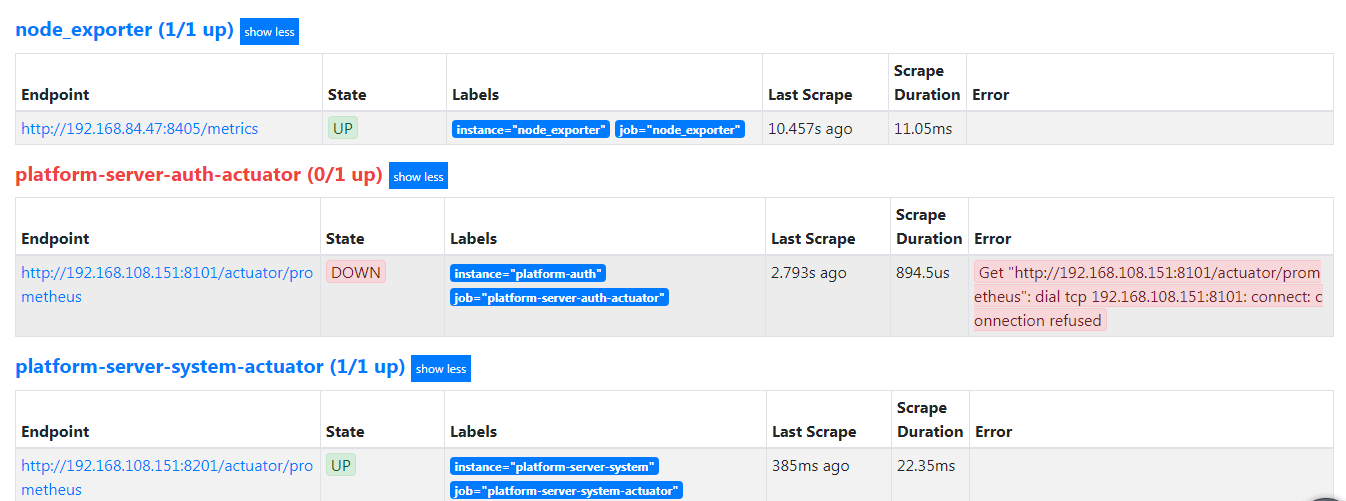
1. 关闭auth服务

访问<http://192.168.84.47:8403/alerts>

查看Status

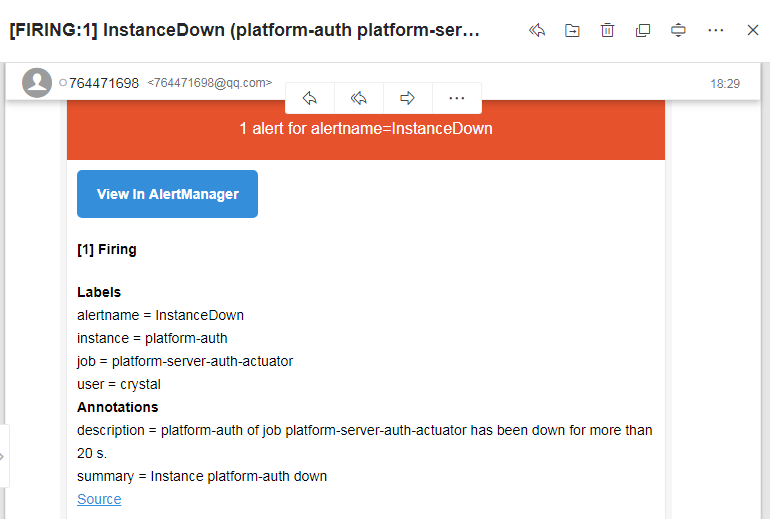


访问<http://192.168.84.47:8403/targets> 可以看到状态栏为DOWN

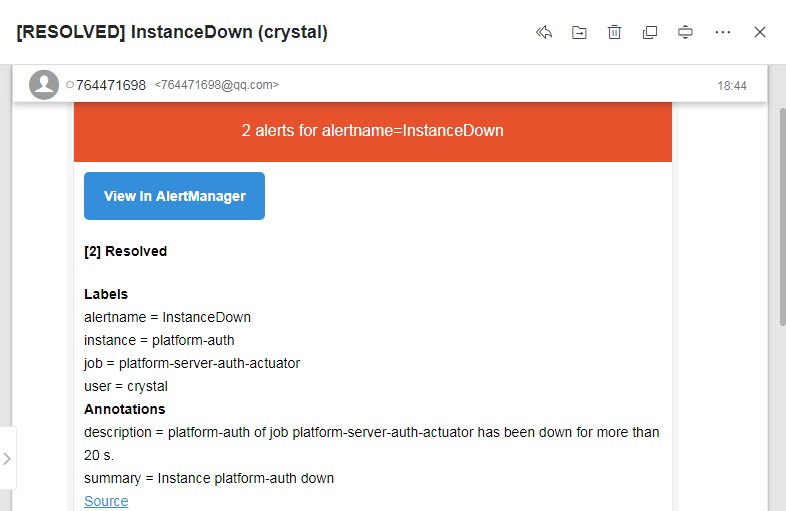


访问[http://192.168.84.47:8406/#/alerts](http://192.168.84.47:8406/" \l "/alerts) 可以看到alert groups

1. 查看邮箱是否收到了邮件



1. 重新启动auth服务，等待一会，收到已解决resolved邮件。



## 四、企业微信预警

1、登录<https://work.weixin.qq.com/> 注册企业微信，在应用管理界面创建应用。

2、修改alertmanager.yml 内容如下

**route:  
 group\_by:** [**'alertname'**]  
 **group\_wait:** 10s  
 **group\_interval:** 5m  
 **repeat\_interval:** 5m  
 **receiver: 'wechat'  
  
receivers:** - **name: 'wechat'  
 wechat\_configs:** - **corp\_id: 'xxx'  
 to\_party: '1'  
 agent\_id: '1000002'  
 api\_secret: 'xxx'**

corp\_id为企业ID，在“我的企业”-> “企业信息”最下面

to\_party为部门（组）号，默认企业微信号的部门名（组名）的值为1

agent\_id和api\_secret为创建的应用的AgentId和Secret



3、prometheus.yml 的rule\_files添加rules.yml

内容如下：

groups:

- name: node\_exporter

rules:

- alert: server\_status

expr: up{job=" node\_exporter "} == 0

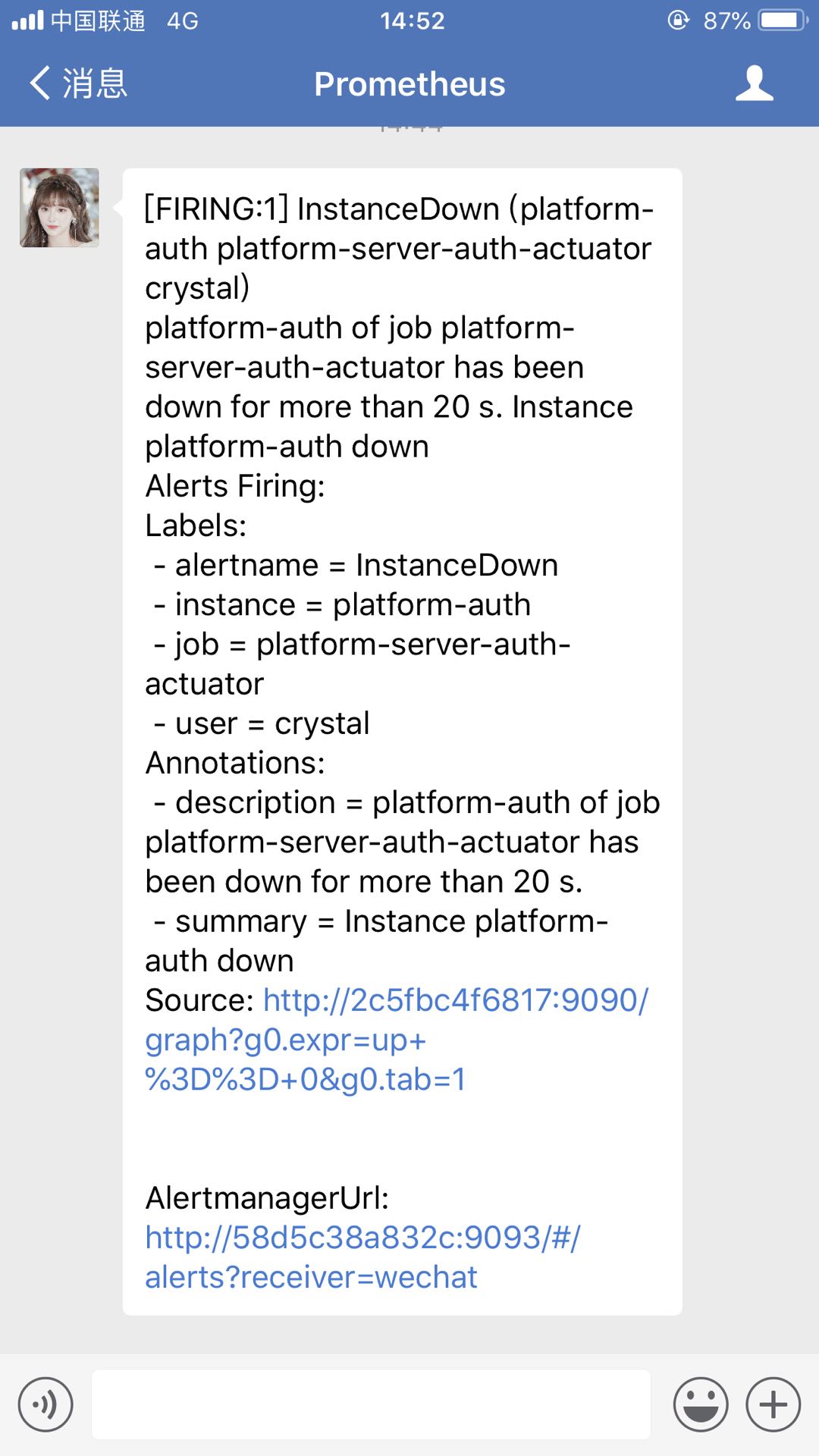
for: 15s

annotations:

summary: "服务 {{ $labels.instance }} 宕机"

4、重新启动容器。

停掉auth服务，等待一会，企业微信收到了服务宕机的消息



## 五、附录

alertmanager支持的预警配置，示例：

groups:

- name: example #定义规则组

rules:

- alert: InstanceDown #定义报警名称

expr: up == 0 #Promql语句，触发规则

for: 1m # 一分钟

labels: #标签定义报警的级别和主机

name: instance

severity: Critical

annotations: #注解

summary: " {{ $labels.appname }}" #报警摘要，取报警信息的appname名称

description: " 服务停止运行 " #报警信息

value: "{{ $value }}%" # 当前报警状态值

- name: Host

rules:

- alert: HostMemory Usage

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

for: 1m

labels:

name: Memory

severity: Warning

annotations:

summary: " {{ $labels.appname }} "

description: "宿主机内存使用率超过80%."

value: "{{ $value }}"

- alert: HostCPU Usage

expr: sum(avg without (cpu)(irate(node\_cpu\_seconds\_total{mode!='idle'}[5m]))) by (instance,appname) > 0.65

for: 1m

labels:

name: CPU

severity: Warning

annotations:

summary: " {{ $labels.appname }} "

description: "宿主机CPU使用率超过65%."

value: "{{ $value }}"

- alert: HostLoad

expr: node\_load5 > 4

for: 1m

labels:

name: Load

severity: Warning

annotations:

summary: "{{ $labels.appname }} "

description: " 主机负载5分钟超过4."

value: "{{ $value }}"

- alert: HostFilesystem Usage

expr: 1-(node\_filesystem\_free\_bytes / node\_filesystem\_size\_bytes) > 0.8

for: 1m

labels:

name: Disk

severity: Warning

annotations:

summary: " {{ $labels.appname }} "

description: " 宿主机 [ {{ $labels.mountpoint }} ]分区使用超过80%."

value: "{{ $value }}%"

- alert: HostDiskio

expr: irate(node\_disk\_writes\_completed\_total{job=~"Host"}[1m]) > 10

for: 1m

labels:

name: Diskio

severity: Warning

annotations:

summary: " {{ $labels.appname }} "

description: " 宿主机 [{{ $labels.device }}]磁盘1分钟平均写入IO负载较高."

value: "{{ $value }}iops"

- alert: Network\_receive

expr: irate(node\_network\_receive\_bytes\_total{device!~"lo|bond[0-9]|cbr[0-9]|veth.\*|virbr.\*|ovs-system"}[5m]) / 1048576 > 3

for: 1m

labels:

name: Network\_receive

severity: Warning

annotations:

summary: " {{ $labels.appname }} "

description: " 宿主机 [{{ $labels.device }}] 网卡5分钟平均接收流量超过3Mbps."

value: "{{ $value }}3Mbps"

- alert: Network\_transmit

expr: irate(node\_network\_transmit\_bytes\_total{device!~"lo|bond[0-9]|cbr[0-9]|veth.\*|virbr.\*|ovs-system"}[5m]) / 1048576 > 3

for: 1m

labels:

name: Network\_transmit

severity: Warning

annotations:

summary: " {{ $labels.appname }} "

description: " 宿主机 [{{ $labels.device }}] 网卡5分钟内平均发送流量超过3Mbps."

value: "{{ $value }}3Mbps"

- name: Container

rules:

- alert: ContainerCPU Usage

expr: (sum by(name,instance) (rate(container\_cpu\_usage\_seconds\_total{image!=""}[5m]))\*100) > 60

for: 1m

labels:

name: CPU

severity: Warning

annotations:

summary: "{{ $labels.name }} "

description: " 容器CPU使用超过60%."

value: "{{ $value }}%"

- alert: ContainerMem Usage

# expr: (container\_memory\_usage\_bytes - container\_memory\_cache) / container\_spec\_memory\_limit\_bytes \* 100 > 10

expr: container\_memory\_usage\_bytes{name=~".+"} / 1048576 > 1024

for: 1m

labels:

name: Memory

severity: Warning

annotations:

summary: "{{ $labels.name }} "

description: " 容器内存使用超过1GB."

value: "{{ $value }}G"