

Prometheus 算是一个全能型选手，原生支持容器监控，当然监控传统应用也不是吃干饭的，所以就是容器和非容器他都支持，所有的监控系统都具备这个流程，数据采集→数据处理→数据存储→数据展示→告警，本文就是针对 Prometheus 展开的，所以先看看 Prometheus 概述

Prometheus 概述

先来看一下 Prometheus 是个啥

Prometheus 是什么

中文名普罗米修斯，最初在 SoundCloud 上构建的监控系统，自 2012 年成为社区开源项目，用户非常活跃的开发人员和用户社区，2016 年加入 CNCF，成为继 kubernetes 之后的第二个托管项目，[官方网站](#)

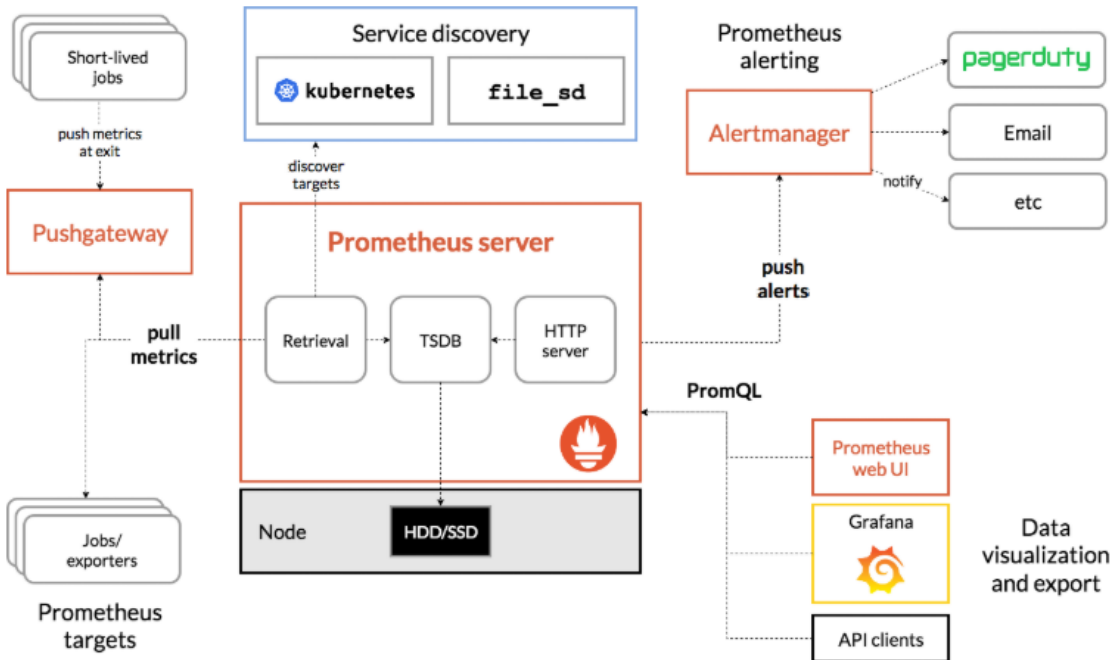
Prometheus 特点

官方扒过来的

- 多维数据模型：由度量名称和键值对标识的时间序列数据
- PromSQL: 一种灵活的查询语言，可以利用多维数据完成复杂的查询
- 不依赖分布式存储，单个服务器节点可直接工作
- 基于 HTTP 的 pull 方式采集时间序列数据
- 推送时间序列数据通过 PushGateway 组件支持
- 通过服务发现或静态配置发现目标
- 多种图形模式及仪表盘支持 (grafana)

Prometheus 组成与架构

来看一张图，官方扒到的



名称	说明
Prometheus Server	收集指标和存储时间序列数据，并提供查询接口
Push Gateway	短期存储指标数据，主要用于临时性任务
Exporters	采集已有的三方服务监控指标并暴露 metrics
Alertmanager	告警

Web UI	简单的 WEB 控制台
--------	-------------

集成了数据的采集，处理，存储，展示，告警一系列流程都已经具备了

数据模型

Prometheus 将所有数据存储为时间序列，具有相同度量名称以及标签属于同个指标，也就是说 Prometheus 从数据源拿到数据之后都会存到内置的 TSDB 中，这里存储的就是时间序列数据，它存储的数据会有一个度量名称，譬如你现在监控一个 nginx，首先你要给他起个名字，这个名称也就是度量名，还会有 N 个标签，你可以理解名称为表名，标签为字段，所以，每个时间序列都由度量标准名称和一组键值对 (也称为标签) 唯一标识。

时间序列的格式是这样的，

```
<metricname> {<labelname>=<labelvalue>,...}
```

metric name 指的就是度量标准名称，label name 也就是标签名，这个标签可以有多个，例子

```
jvm_memory_max_bytes{area="heap",id="Tenured Gen",}
```

这个度量名称为 jvm_memory_max_bytes，后面是两个标签，和他们各对应的值，当然你还可以继续指定标签，你指定的标签越多查询的维度就越多。

指标类型

看表格吧

类型名称	说明
Counter	递增计数器，适合收集接口请求次数
Gauge	可以任意变化的数值，适用 CPU 使用率
Histogram	对一段时间内数据进行采集，并对有所数值求和于统计数量
Summary	与 Histogram 类型类似

任务和实例

实例指的就是你可以抓取的目标target，这个会在 Prometheus 配置文件中体现，任务是具有相同目标的实例集合，你可以理解为是一个组(比如，订单服务多台实例机器，可以放入一个任务里，分多个实例target抓取)，一会写配置文件的时候会详细解析，下面开始安装 Prometheus。

Prometheus 部署

我们借助docker来安装，新建目录docker-monitor，在里面创建文件docker-compose.yml，内容如下：

```
1 version: "3"
2 services:
3   prometheus:
4     image: prom/prometheus:v2.4.3
5     container_name: 'prometheus'
6     volumes:
7       - ./prometheus:/etc/prometheus/ #映射prometheus的配置文件
8       - /etc/localtime:/etc/localtime:ro #同步容器与宿主机的时间，这个非常重要，如果时间不一致，会导致prometheus抓不到数据
9     ports:
10      - '9090:9090'
```

监控web应用性能指标

在docker-monitor目录下新增prometheus目录，在里面创建prometheus配置文件prometheus.yml，内容如下：

```

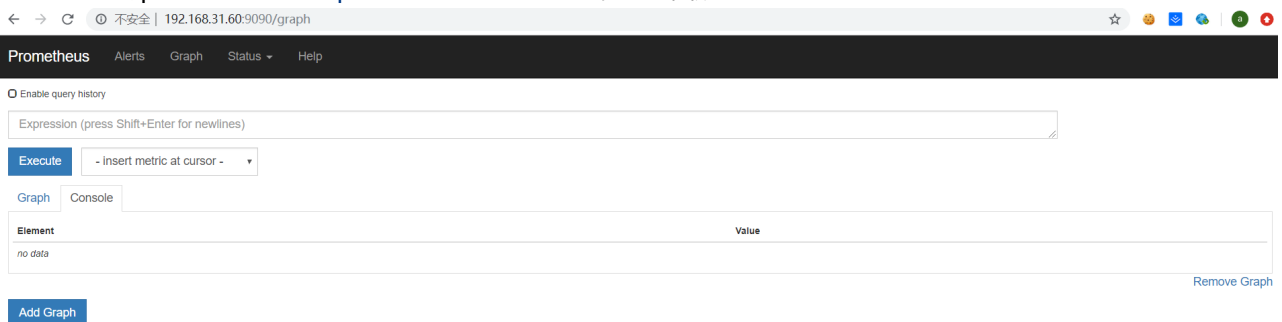
1 global: #全局配置
2   scrape_interval: 15s #全局定时任务抓取性能数据间隔
3
4 scrape_configs: #抓取性能数据任务配置
5   - job_name: 'tulingmall-order' #抓取订单服务性能指标数据任务，一个job下可以配置多个抓取的targets，比如订单服务多个实例机器
6     scrape_interval: 10s #每10s抓取一次
7     metrics_path: '/actuator/prometheus' #抓取的数据url
8     static_configs:
9       - targets: ['192.168.31.60:8844'] #抓取的服务器地址
10      labels:
11        application: 'tulingmall-order-label' #抓取任务标签
12
13   - job_name: 'prometheus' #抓取prometheus自身性能指标数据任务
14     scrape_interval: 5s
15     static_configs:
16       - targets: ['localhost:9090']

```

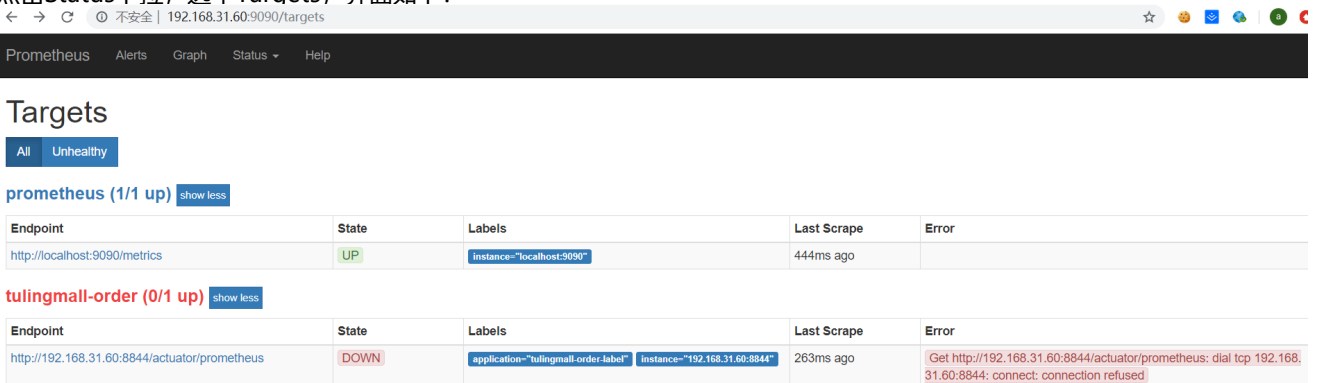
在docker-monitor目录下执行如下命令启动prometheus

```
1 docker-compose up -d
```

在浏览器访问prometheus: <http://192.168.31.60:9090>，如下图所示：



点击Status下拉，选中Targets，界面如下：



这里显示了在prometheus里配置的两个抓取任务，不过tulingmall-order任务是失败的，state是down，接下来我们需要配置下tulingmall-order服务才能让prometheus抓取数据。首先需要在tulingmall-order服务下增加pom依赖，如下：

```

1 <!-- 开启springboot的应用监控 -->
2 <dependency>
3   <groupId>org.springframework.boot</groupId>
4   <artifactId>spring-boot-starter-actuator</artifactId>
5 </dependency>
6 <!-- 增加prometheus整合 -->
7 <dependency>
8   <groupId>io.micrometer</groupId>
9   <artifactId>micrometer-registry-prometheus</artifactId>

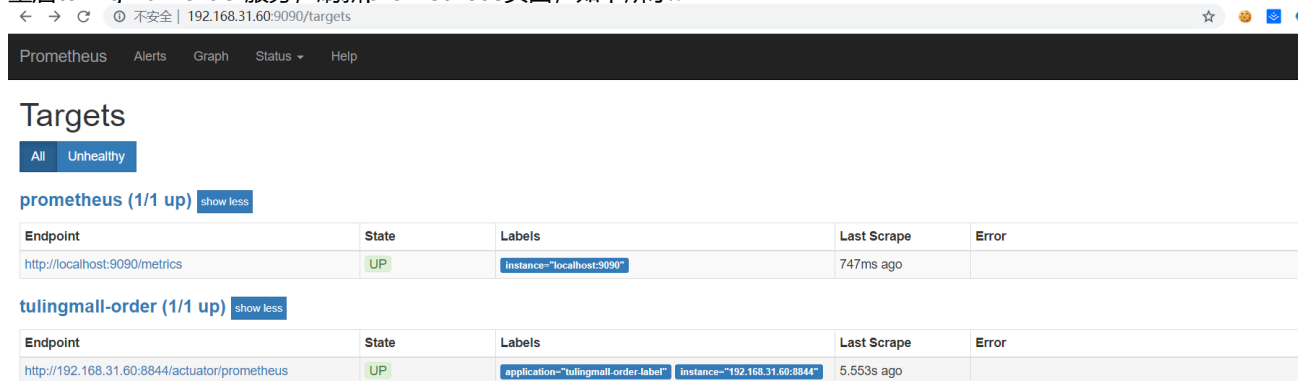
```

```
10 </dependency>
```

还需要再tulingmall-order服务的配置文件里增加开启springboot admin监控的配置，如下：

```
1 management: #开启SpringBoot Admin的监控
2   endpoints:
3     prometheus:
4       enable: true
5   web:
6     exposure:
7       include: '*'
8   endpoint:
9     health:
10      show-details: always
```

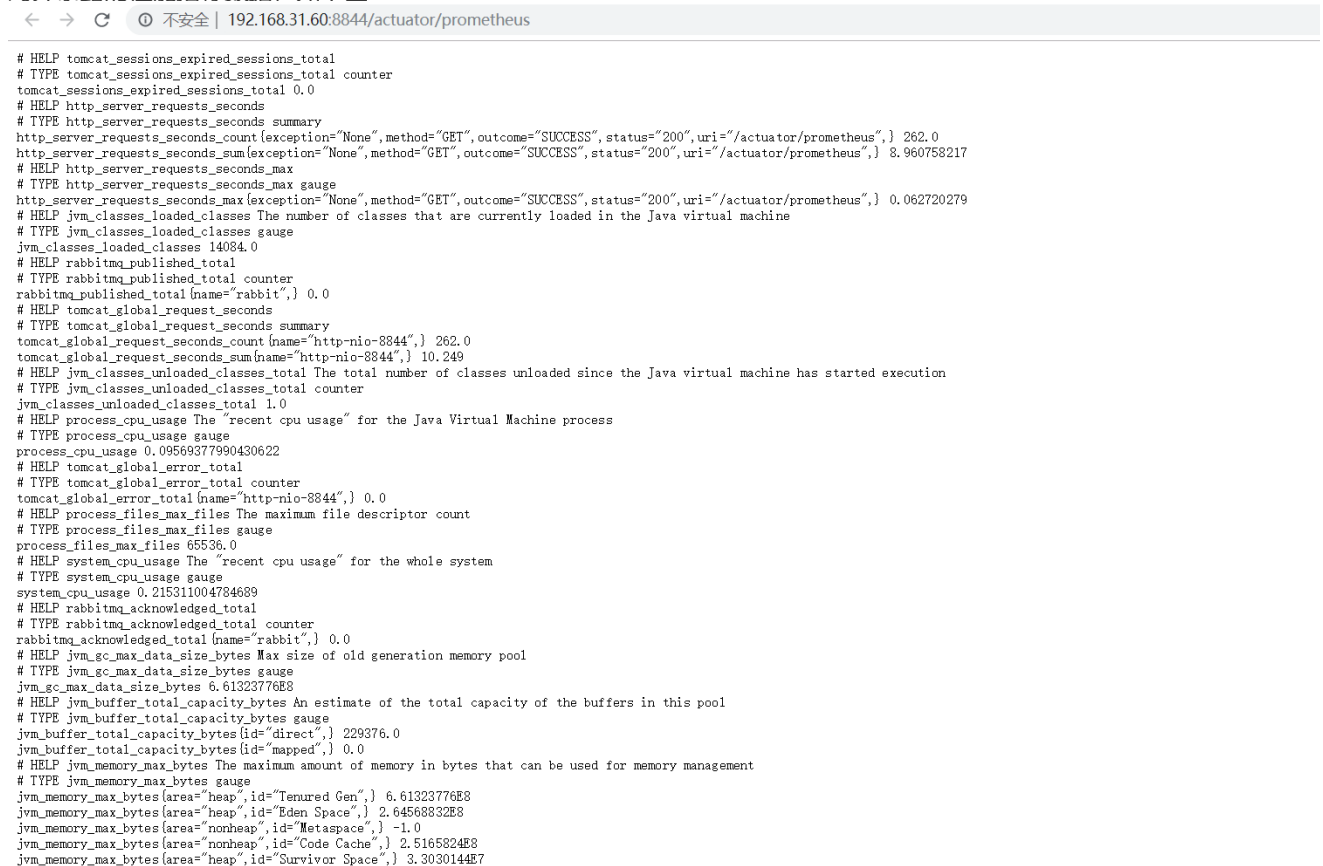
重启tulingmall-order服务，刷新prometheus页面，如下所示：



The screenshot shows the Prometheus web interface. The 'Targets' tab is selected, and the 'Unhealthy' filter is active. Two target groups are listed:

- prometheus (1/1 up)**: Shows a single target at `http://localhost:9090/metrics` with a state of 'UP' and a last scrape time of 747ms ago.
- tulingmall-order (1/1 up)**: Shows a single target at `http://192.168.31.60:8844/actuator/prometheus` with a state of 'UP' and a last scrape time of 5.553s ago. This target is highlighted with a blue background.

点击tulingmall-order下面的prometheus链接：<http://192.168.31.60:8844/actuator/prometheus>，会打开order服务对外暴露的性能指标数据，如下图：



The screenshot shows the Prometheus metrics page for the tulingmall-order service. The page displays a list of metrics, including JVM and system metrics. The metrics are organized into sections, with the first section showing JVM metrics.

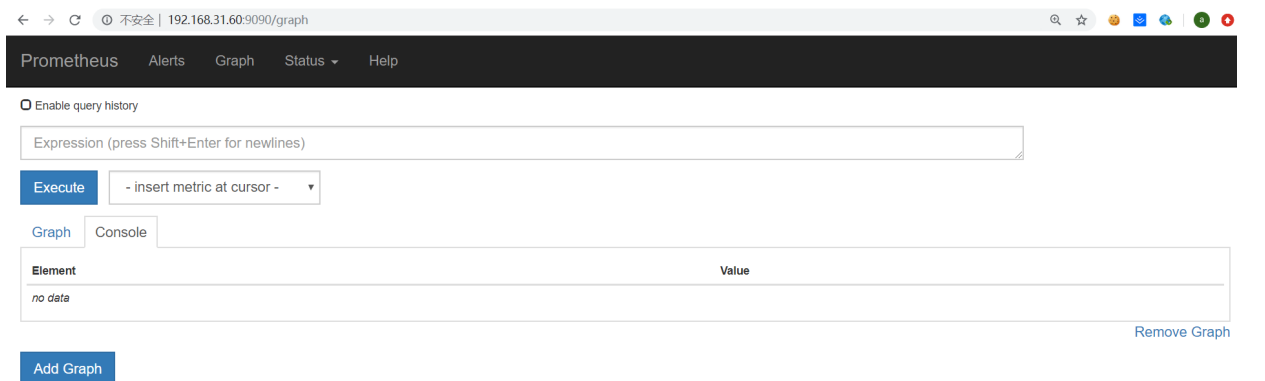
Key metrics visible include:

- `tomcat_sessions_expired_sessions_total`: 0.0
- `http_server_requests_seconds`: 0.0
- `http_server_requests_seconds_max`: 0.062720279
- `jvm_classes_loaded_classes`: 14084.0
- `tomcat_global_request_seconds`: 0.0
- `tomcat_global_request_seconds_max`: 0.0
- `jvm_classes_unloaded_classes_total`: 1.0
- `process_cpu_usage`: 0.09569377990430622
- `tomcat_global_error_total`: 0.0
- `process_files_max_files`: 65536.0
- `system_cpu_usage`: 0.215311004784689
- `rabbitmq_acknowledged_total`: 0.0
- `jvm_gc_max_data_size_bytes`: 6.61323776E8
- `jvm_buffer_total_capacity_bytes`: 2.29376E9
- `jvm_memory_max_bytes`: 2.5165824E8

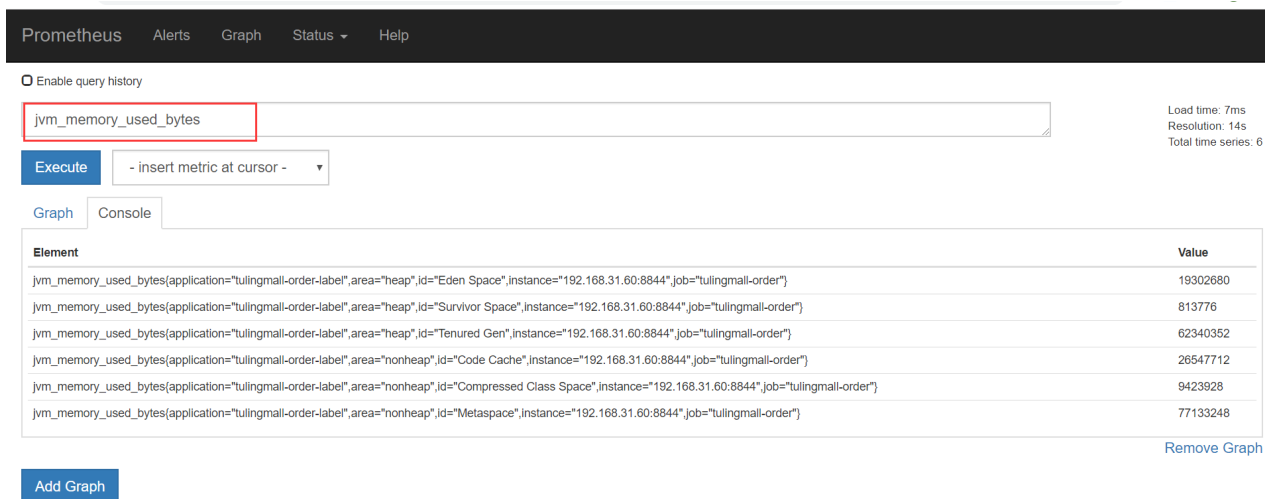
拿其中一个指标举例：`jvm_threads_states_threads{state="runnable"} 13.0`，这代表jvm_threads_states_threads这个

度量指标，其中state等于runnable的数据有13条

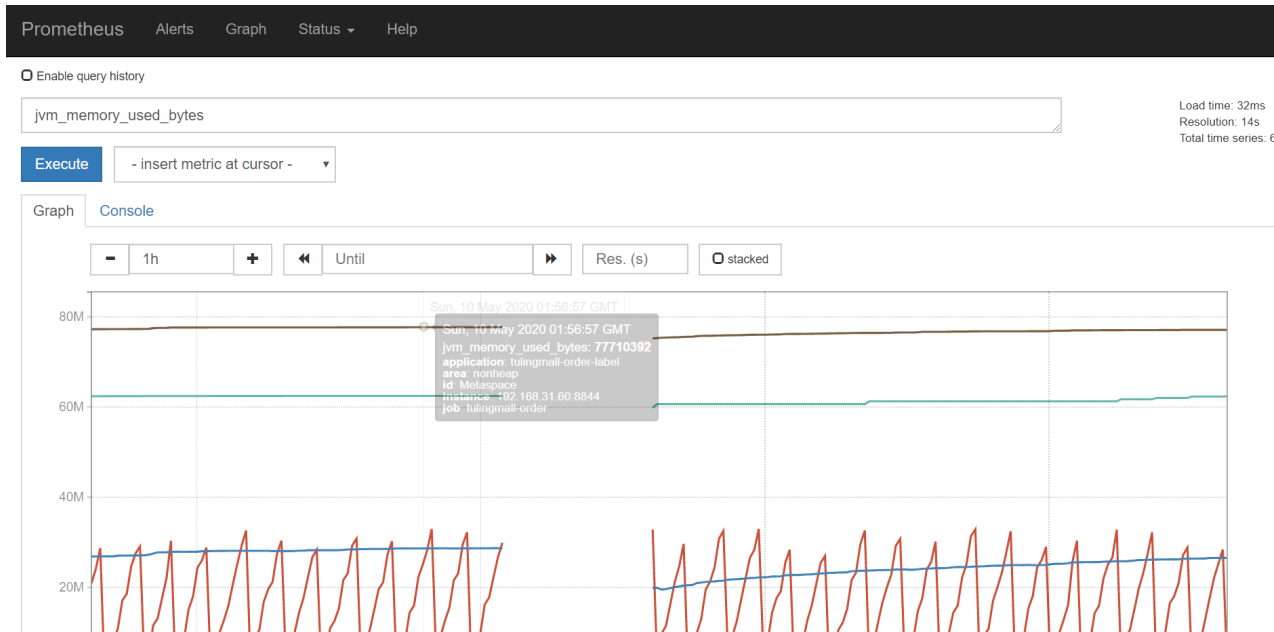
我们点prometheus页面的Graph链接，进入指标查询页面可以查询相关指标，如下：



将度量指标输入查询框，点击Execute按钮，如下：



点击Execute按钮下的Graph链接可以查看指标对应的图标，如下：



以上是prometheus自带的指标查询界面，但是太简陋，一般我们都是使用grafana图形展示工具配合prometheus一起使用

Grafana 部署

先用docker来安装下grafana，在上面的docker-compose.yml文件里加入grafana的安装配置，如下所示：

```
1 version: "3"
2 services:
3   prometheus:
4     image: prom/prometheus:v2.4.3
5     container_name: 'prometheus'
```

```

6 volumes:
7 - ./prometheus:/etc/prometheus/ #映射prometheus的配置文件
8 - /etc/localtime:/etc/localtime:ro #同步容器与宿主机的时间，这个非常重要，如果时间不一致，会导致prometheus
  抓不到数据
9 ports:
10 - '9090:9090'
11 grafana:
12 image: grafana/grafana:5.2.4
13 container_name: 'grafana'
14 ports:
15 - '3000:3000'
16 volumes:
17 - ./grafana/config/grafana.ini:/etc/grafana/grafana.ini #grafana报警邮件配置
18 - ./grafana/provisioning:/etc/grafana/provisioning/ #配置grafana的prometheus数据源
19 - /etc/localtime:/etc/localtime:ro
20 env_file:
21 - ./grafana/config.monitoring #grafana登录配置
22 depends_on:
23 - prometheus #grafana需要在prometheus之后启动

```

在docker-monitor目录下新增grafana目录，在里面创建文件config.monitoring，内容如下：

```

1 GF_SECURITY_ADMIN_PASSWORD=password #grafana管理界面的登录用户密码，用户名是admin
2 GF_USERS_ALLOW_SIGN_UP=false #grafana管理界面是否允许注册，默认不允许

```

在grafana目录下创建目录provisioning，在里面创建datasources目录，在datasources目录里新建文件datasource.yml，内容如下：

```

1 # config file version
2 apiVersion: 1
3
4 deleteDatasources: #如果之前存在name为Prometheus，orgId为1的数据源先删除
5 - name: Prometheus
6   orgId: 1
7
8 datasources: #配置Prometheus的数据源
9 - name: Prometheus
10   type: prometheus
11   access: proxy
12   orgId: 1
13   url: http://prometheus:9090 #在相同的docker compose下，可以直接用prometheus服务名直接访问
14   basicAuth: false
15   isDefault: true
16   version: 1
17   editable: true

```

在grafana目录下创建目录config，在里面创建文件grafana.ini，内容如下：

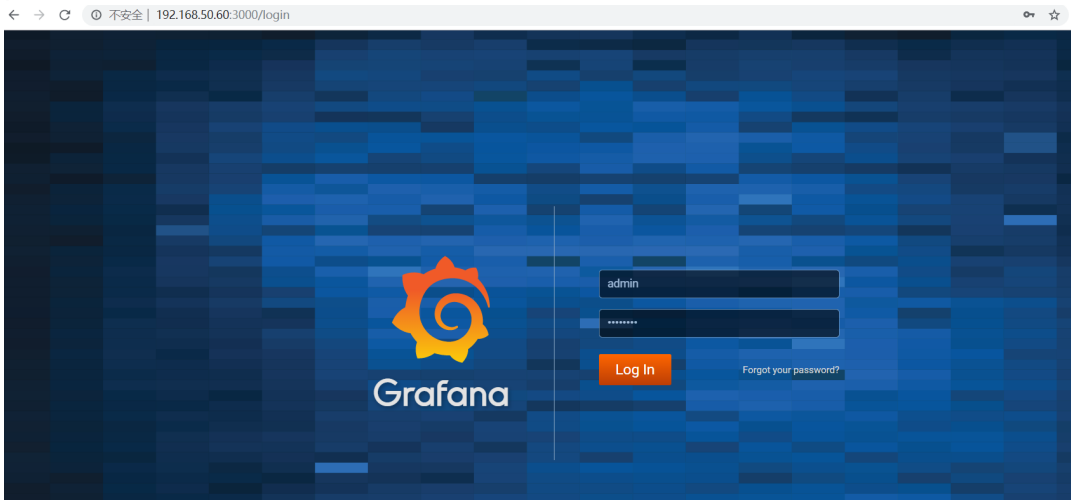
```

1 ##### SMTP / Emailing #####
2 # 配置邮件服务器
3 [smtp]
4 enabled = true
5 # 发件服务器
6 host = smtp.qq.com:465
7 # smtp账号
8 user = 3376224996@qq.com
9 # smtp 授权码，授权码获取请参看课上视频演示
10 password = test123
11 # 发信邮箱

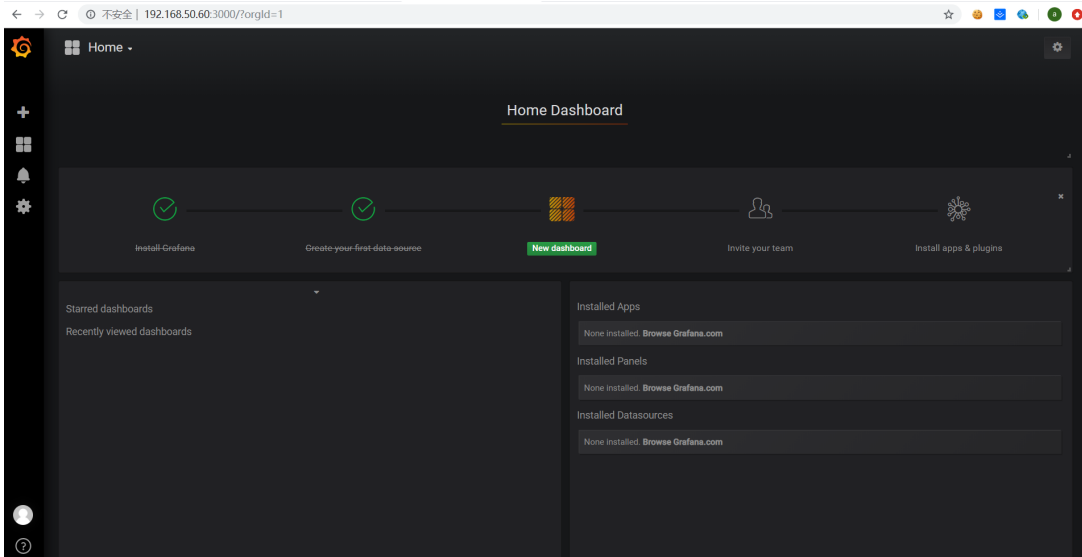
```

```
12 from_address = 3376224996@qq.com
13 # 发信人
14 from_name = zhuge
```

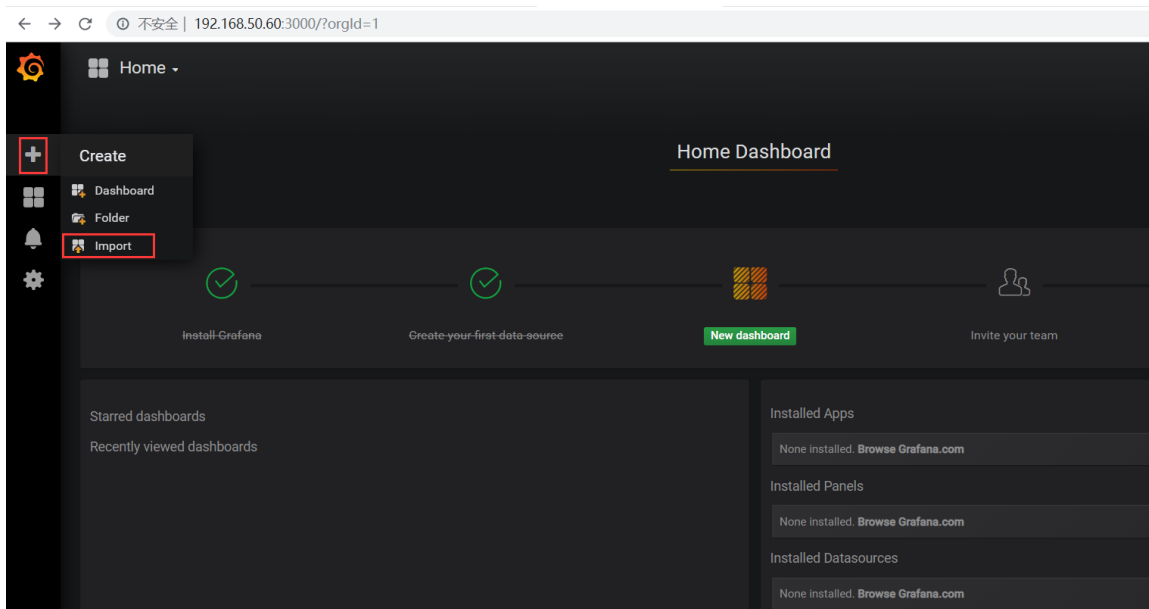
用docker compose启动grafana, 访问grafana页面: <http://192.168.31.60:3000>, 用户名为admin, 密码为password, 如下:



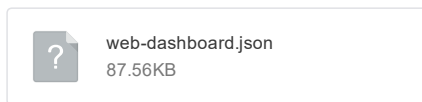
登录进去首页如下:



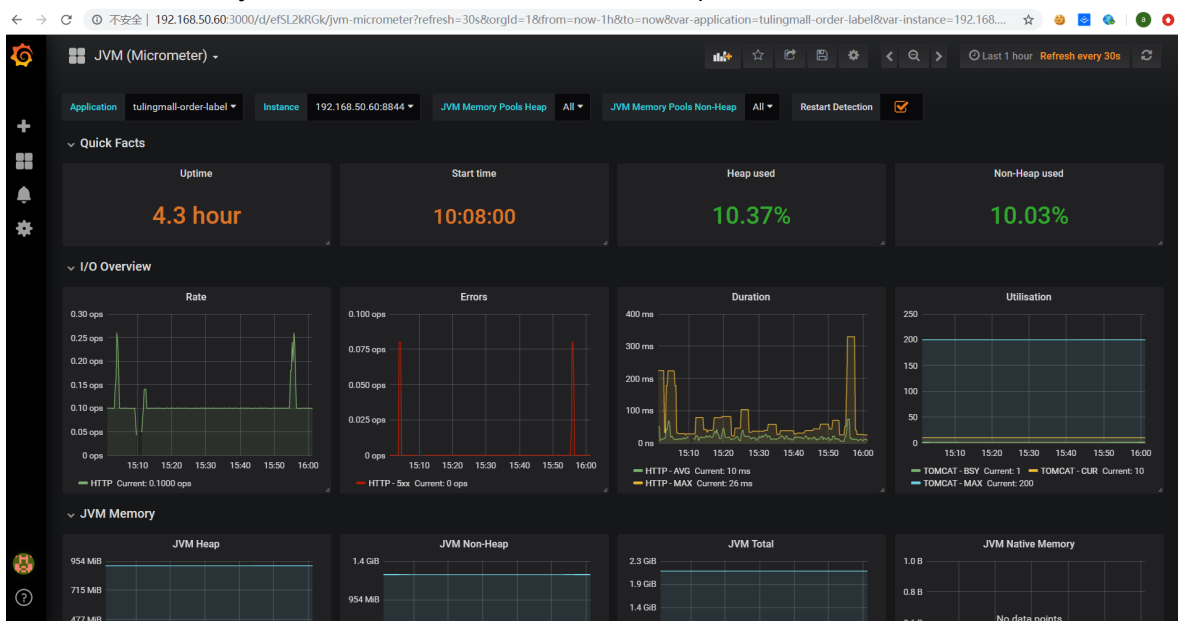
点击左边的加号并import一个我们事先准备好的可视化指标文件web-dashboard.json(都是些运维的指标, 网上可以找现成的)



web-dashboard.json内容参考附件:

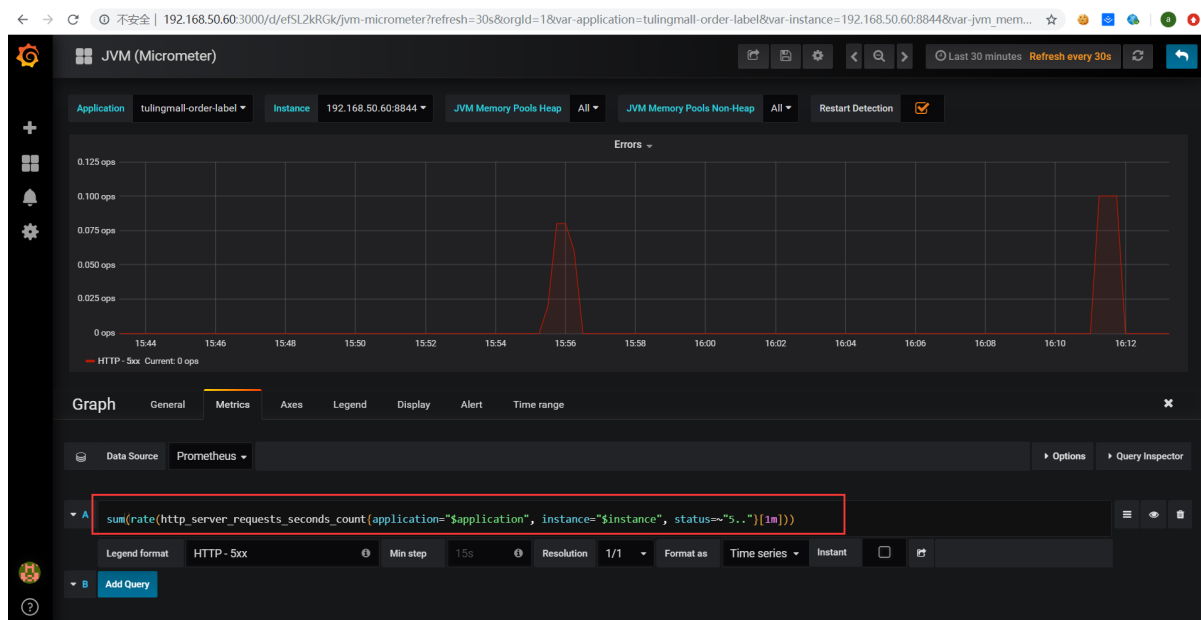


导入web-dashboard.json后在页面上选择Prometheus，点击import按钮之后页面显示如下(有可能没有任何数据):

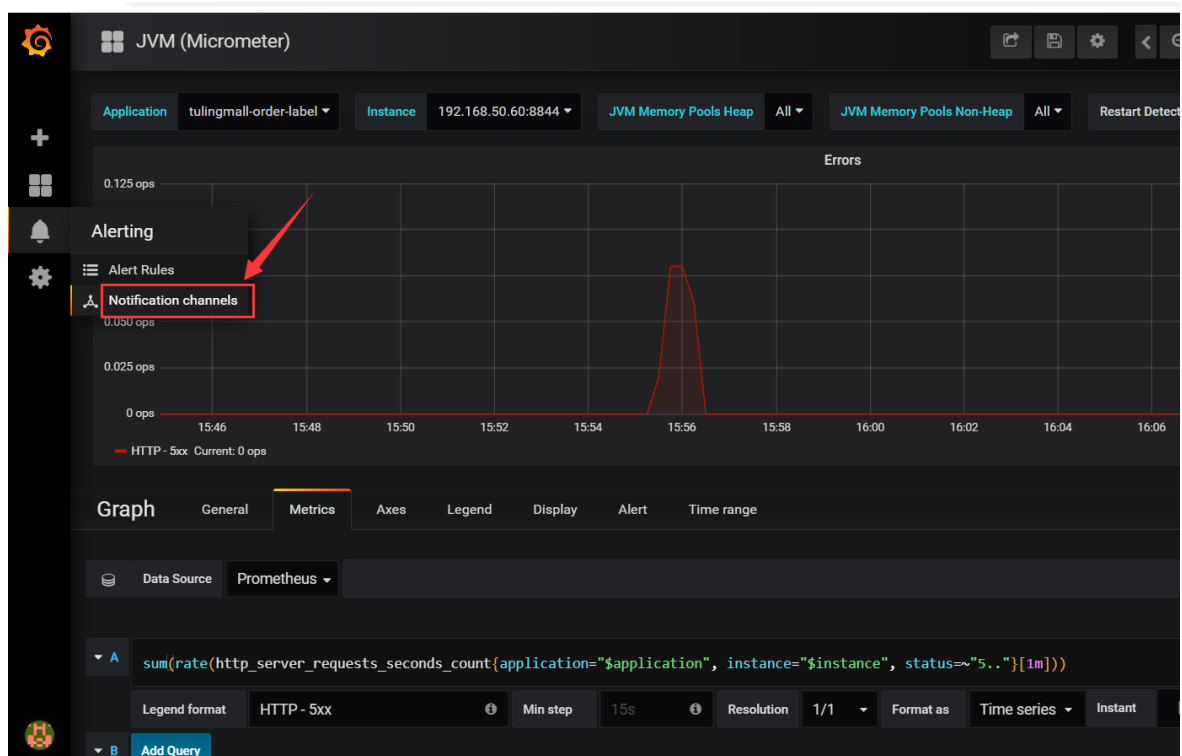


写一个监控指标报警示例，比如系统报错5XX达到一定程度就报警发邮件通知：

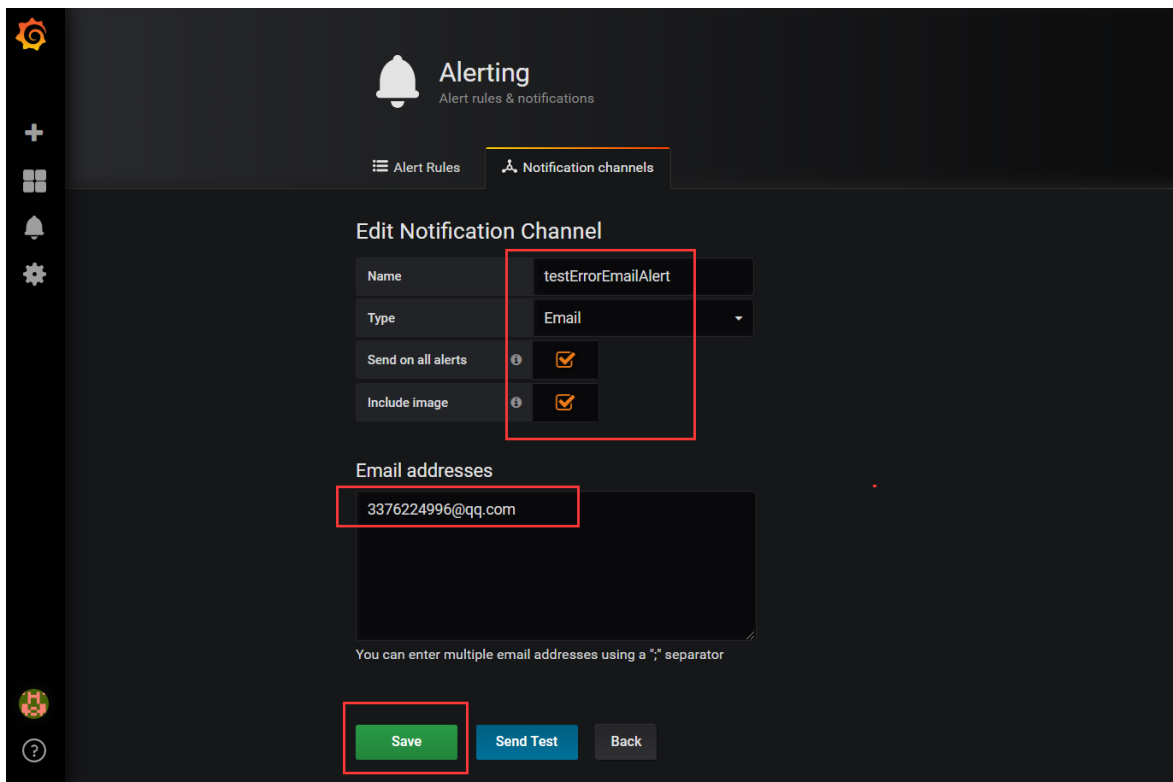
点击Errors面板选择Edit，进入到Errors指标的详细面板，如下：



点击下图所示新增报警渠道：



然后选择邮件报警，也可以选择webhook方式配置一个报警通知的http调用接口，这个可以间接实现所有的通知方式，如下：



The image shows the 'Alerting' section of a dashboard, specifically the 'Edit Notification Channel' page. The page has a dark theme. At the top, there's a header with a bell icon and the word 'Alerting'. Below it, there are tabs for 'Alert Rules' and 'Notification channels'. The 'Notification channels' tab is active. The main content area is titled 'Edit Notification Channel'. It contains a form with the following fields: 'Name' (testErrorEmailAlert), 'Type' (Email), 'Send on all alerts' (checked), and 'Include image' (checked). Below these fields is a section for 'Email addresses' with a text input field containing '3376224996@qq.com'. At the bottom, there are three buttons: 'Save' (green), 'Send Test' (blue), and 'Back' (grey). Red boxes highlight the 'Name' and 'Type' fields, the 'Email addresses' input field, and the 'Save' button.

Alerting
Alert rules & notifications

Alert Rules Notification channels

Edit Notification Channel

Name	testErrorEmailAlert
Type	Email
Send on all alerts	<input checked="" type="checkbox"/>
Include image	<input checked="" type="checkbox"/>

Email addresses

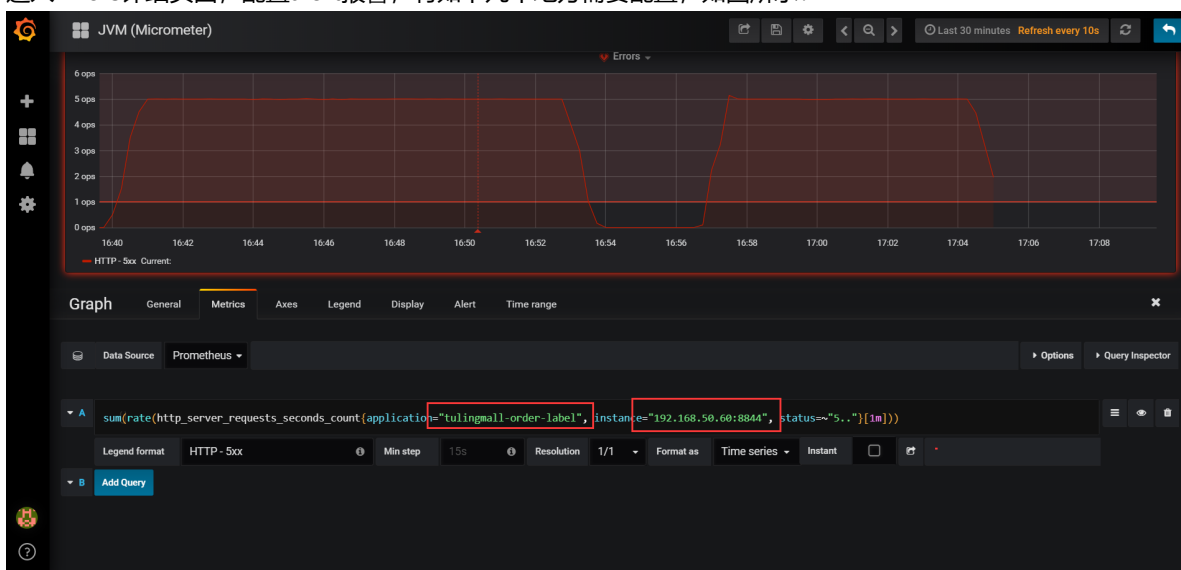
3376224996@qq.com

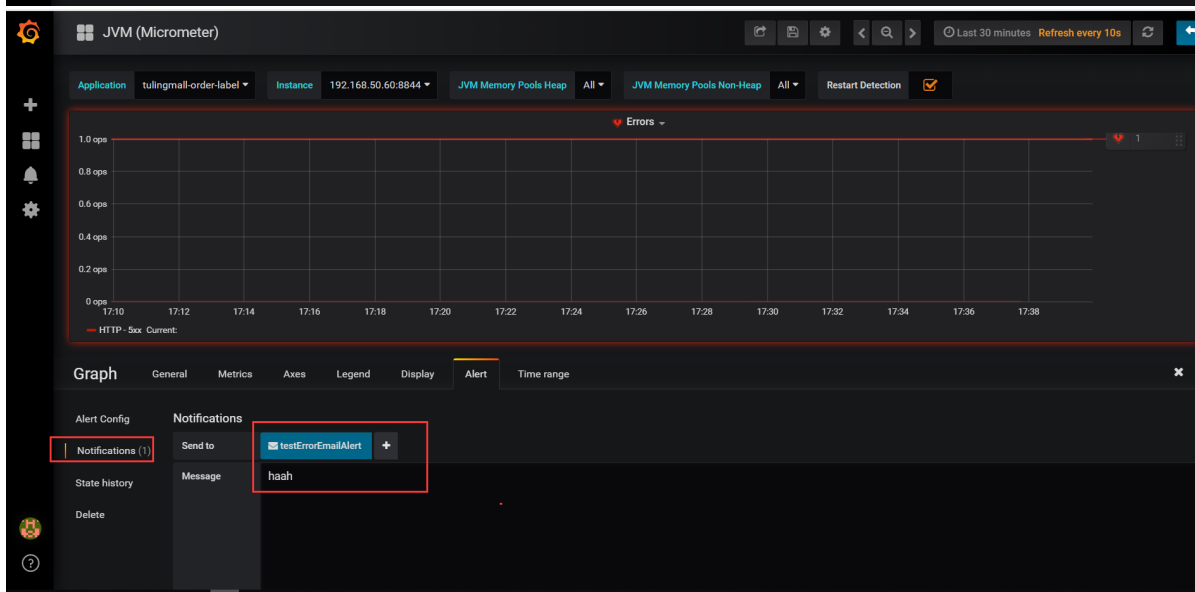
You can enter multiple email addresses using a "," separator

Save Send Test Back

最后点击save按钮保存

进入Errors详细页面，配置alert报警，有如下几个地方需要配置，如图所示：





报警邮件如下所示：



监控Mysql性能指标

1、下载mysql客户端的exporter镜像

```
1 docker pull prom/mysql-exporter
```

2、启动监控的数据库连接，容器创建的时候需要指定

```
1 docker run -d -p 9104:9104 -e DATA_SOURCE_NAME="root:password@(mysql服务器ip:3306)/databaseName" prom/mysql-exporter
```

3、在prometheus.yml文件末尾追加如下配置：

```
1 - job_name: 'mysql'
2   scrape_interval: 5s
3   static_configs:
4     - targets: ['192.168.50.60:9104']
5   labels:
6     instance: mysql
```

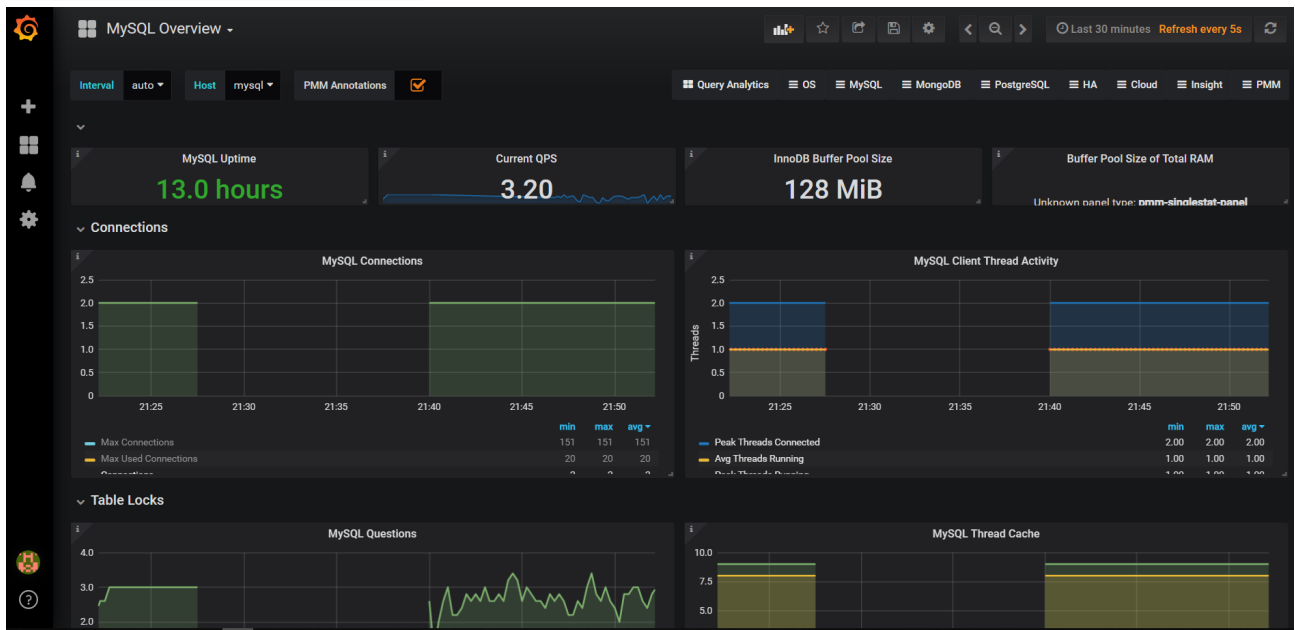
4、重新启动Prometheus镜像，查看Prometheus是否启动完成，访问：http://服务器ip:9090

```
1 docker-compose up --force-recreate -d
```

5、导入Prometheus模板，添加mysql-dashboard.json格式模板，更多模板下载地址：

<https://github.com/percona/grafana-dashboards.git>

mysql-dashboard.json
167.48KB



监控Redis性能指标

1、下载redis客户端的exporter镜像

```
1 docker pull oliver006/redis_exporter
```

2、启动监控的数据库连接，容器创建的时候需要指定

```
1 docker run -d -p 9121:9121 oliver006/redis_exporter --redis.addr redis://redis连接IP:6379
```

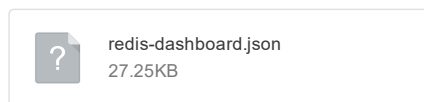
3、在prometheus.yml文件末尾追加如下配置：

```
1 - job_name: 'redis'
2   scrape_interval: 5s
3   static_configs:
4     - targets: ['192.168.50.60:9121']
5   labels:
6     instance: redis
```

4、重新启动Prometheus镜像，查看Prometheus是否启动完成，访问：http://服务器ip:9090

```
1 docker-compose up --force-recreate -d
```

5、导入Prometheus模板，添加redis-dashboard.json格式模板



文档：03-自动化监控系统Prometheus&Grafana实战

链接：[http://note.youdao.com/noteshare?](http://note.youdao.com/noteshare?id=e01ad636d78aa11dcf012d3181a20bc6&sub=BB445FBB954F4BB39BCEF496F95258C6)

[id=e01ad636d78aa11dcf012d3181a20bc6&sub=BB445FBB954F4BB39BCEF496F95258C6](http://note.youdao.com/noteshare?id=e01ad636d78aa11dcf012d3181a20bc6&sub=BB445FBB954F4BB39BCEF496F95258C6)