18 Micrometer 实现微服务监控

更新时间：2019-06-28 11:48:57

> 横眉冷对千夫指，俯首甘为孺子牛。
>
> ——鲁迅

不同于单体架构的应用，微服务架构由于服务数量众多，出故障的概率更大，这个在前两篇文章中已经和读者分享过了。这种时候不能单纯依靠“人肉”运维，否则当服务数量越来越多时成本将变得不可控。一个好的解决方案是我们需要对服务进行监控，监控服务运行的数据。当有异常情况出现时，服务能够自动报警，方便运维工程师去处理。

Spring Cloud 中对于服务监控这一个话题也是在不断地变化中。早期的版本（ Greenwich 版之前）服务监控主要使用 Hystrix Dashboard 仪表盘，集群数据监控使用 Turbine，这一技术组合在最新的 Greenwich 版中被建议使用 Micrometer 来替换掉。相对于前者， Micrometer 的使用确实要方便很多，而且容易结合配套工具 Prometheus 以及 Grafana 一起使用，具备自动报警功能，数据展示也更加多样化，方便运维工程师去查看，因此本专栏将不再向读者介绍古老的服务监控的用法，主要向读者介绍 Micrometer 的用法。考虑到很多读者也是第一次使用 Micrometer ，因此本文将分为两部分，首先向读者介绍在 Spring Boot 中 Micrometer 要如何使用，然后再向读者介绍微服务中 Micrometer 的用法。

Micrometer 简介

Micrometer 为数据测量仪表提供了一个简单的外观，它几乎适用于大部分目前最流行的监控系统，允许开发者检测基于 JVM 的应用程序代码， Micrometer 有点类似于 SLF4J ，只不过是针对测量数据的。Micrometer 主要有如下三方面的功能：

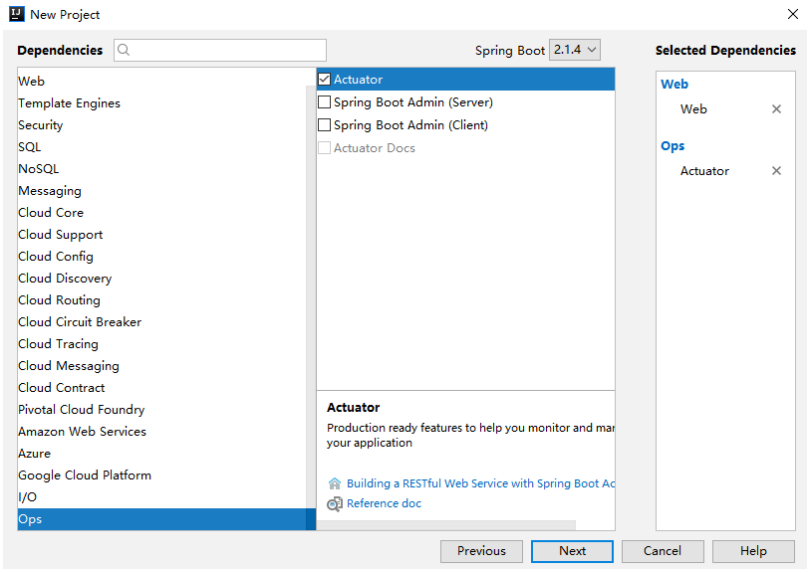
1. Micrometer 提供了度量指标类，例如 timers、gauges 以及 counters等；
2. 一揽子开箱即用的解决方案，例如缓存、类加载器、垃圾收集、处理器利用率以及线程池等；
3. 从 Spring Boot 2.0 开始， 在 Spring Boot Actuator 中使用了 Micrometer 。在早期的 Spring Boot 版本中，也支持通过附加依赖的方式来使用 Micrometer。

Micrometer 支持流行的监控系统，作为一个门面，Micrometer 允许开发者检测代码，并决定是否监控系统。Micrometer 支持 AppOptics、Azure Monitor、Netflix Atlas、CloudWatch、 Datadog、Dynatrace、Elastic、Ganglia、 Graphite、Humio、Influx/Telegraf、 JMX、KairosDB、New Relic、Prometheus、SignalFx、Google Stackdriver、StatsD 以及 Wavefront。

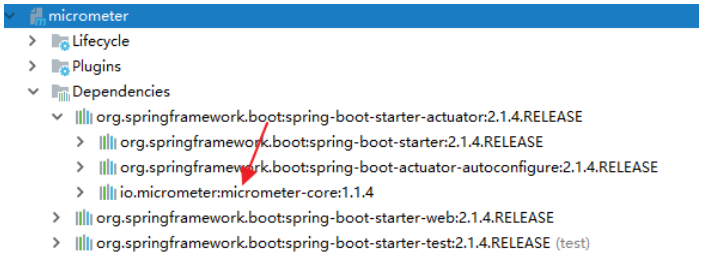
Micrometer 基本用法

首先来看看 Micrometer 在 Spring Boot 项目中的用法：

创建一个 Spring Boot 项目，加入 Web 和 Actuator 依赖，如下图：



项目创建成功后，从依赖关系中我们可以看到，新版的 Actuator 中就是使用了 Micrometer 来实现数据监控，如下图：

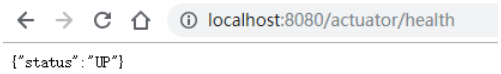


默认情况下，出于安全层面的考虑，只有 health 和 info 端点暴露了。其他端点虽然开启了但是未暴露，通过在 application.yaml 中添加如下配置，我们可以暴露所有的端点：

```
management:
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    metrics:
      enabled: true
```

这里的 `*` 表示启用所有端点，由于 `*` 在 yaml 中有特殊含义，因此大家一定要注意给 `*` 加上双引号。

配置完成后，启动 Spring Boot 项目，即可看到各个数据端口均已打开，例如访问 `health` 端点（默认前缀为 `/actuator`），效果如下：



各数据端口及含义如下表：

端点	端点描述	是否开启
auditevents	展示当前应用程序的审计事件信息	Yes
beans	展示所有SpringBeans信息	Yes
conditions	展示一个自动配置类的使用报告，该报告展示所有自动配置类及它们被使用或未被使用的原因	Yes
configprops	展示所有@ConfigurationProperties的列表	Yes
env	展示系统运行环境信息	Yes
flyway	展示数据库迁移路径	Yes
health	展示应用程序的健康信息	Yes
httptrace	展示trace信息（默认为最新的100条HTTP请求）	Yes
info	展示应用的定制信息，这些定制信息以info开头	Yes
loggers	展示并修改应用的日志配置	Yes
liquibase	展示任何Liquibase数据库迁移路径	Yes
metrics	展示应用程序度量信息	Yes
mappings	展示所有@RequestMapping路径的集合列表	Yes
scheduledtasks	展示应用的所有定时任务	Yes
shutdown	远程关闭应用接口	No
sessions	展示并操作SpringSession会话	Yes
threaddump	展示线程活动的快照	Yes
heapdump	返回一个GZip压缩的hprof堆转储文件	Yes
jolokia	展示通过HTTP暴露的JMXbeans	Yes
logfile	返回日志文件内容	Yes
prometheus	展示一个可以被Prometheus服务器抓取的metrics数据	Yes

但是这样的数据查看不够直观，不易分析出问题，结合 Prometheus 可以更好的将数据展示出来，那么什么是 Prometheus 呢？

Prometheus

Prometheus 是一个最初在 SoundCloud 上构建的开源系统监视和警报工具包 。自2012年成立以来，许多公司和组织都采用了 Prometheus ，该项目拥有一个非常活跃的开发人员和用户社区。它现在是一个独立的开源项目，可以独立于任何公司进行维护。

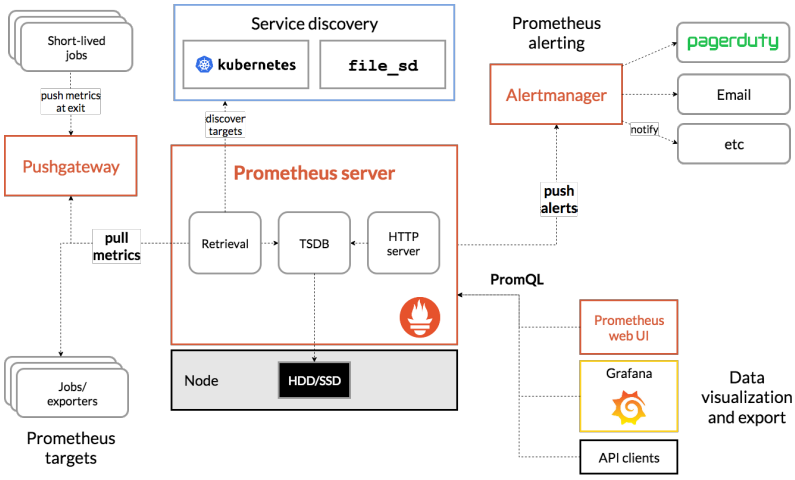
特征

Prometheus 主要有如下特点：

- 具有由度量名称和 key/value 标识的时间序列数据的多维数据模型
- PromQL，一种灵活的查询语言
- 不依赖分布式存储，单个服务器节点是自治的
- 使用 HTTP 协议，自动拉取数据
- 通过服务发现或静态配置发现目标
- 多种图形和仪表盘支持，数据展示友好
- 可以非常方便地实现扩展

架构

通过下面一张图大家可以看到 Prometheus 的一个大致工作原理：



由这张图中，我们可以大致分析出 Prometheus 的工作过程：

1. 首先 Prometheus Server 定期从 targets 或者服务注册中心拉取数据；
2. exporters 负责向 Prometheus Server 做数据汇总。不同的数据汇总由不同的 exporters 实现，例如监控主机有 node-exporters，MySQL 有 MySQL Server exporter；
3. 由于 Prometheus 采用数据拉取的模式，实际生产环境可能由于各个服务不在一个子网或者防火墙的原因，导致 Prometheus 无法直接拉取各个 target 数据，此时可以通过 Pushgateway 来推送 metrics 到 Prometheus Server；
4. Alertmanager 则可以通过提前配置好的邮件地址，对收到的警告信息发出报警；
5. Grafana 则可以通过 PromQL 查询监控数据，进行更丰富的展示。

具体使用

说了这么多，那么在我们的 Spring Boot 项目中，到底要如何使用 Prometheus 呢？很简单，整体上来说，可以分为两个步骤：

1. 在Spring Boot 项目中引入相关依赖；
2. 安装 Prometheus 软件并配置。

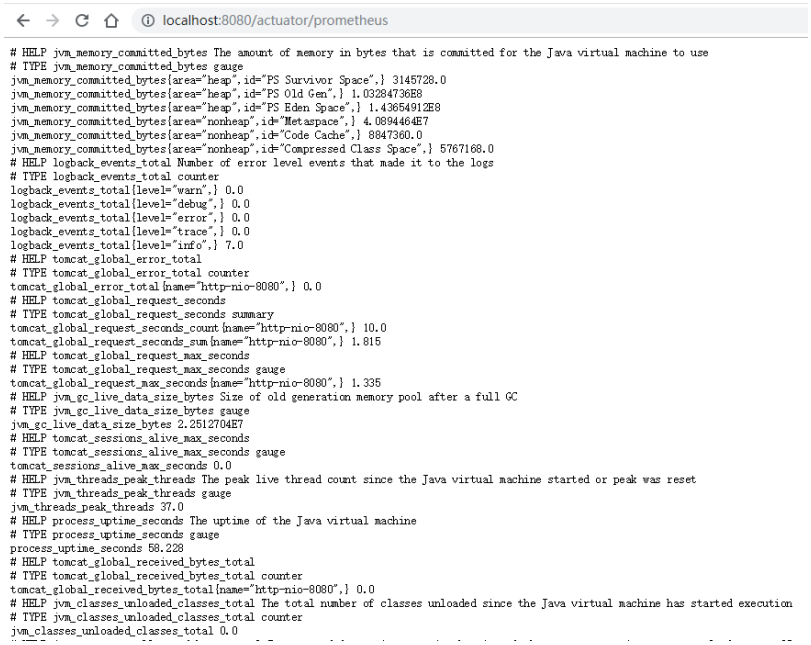
下面分别来看，首先在 Spring Boot 项目中添加如下依赖：

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

添加成功后，修改配置，开启 Prometheus ，如下:

```yaml
management:
  metrics:
    export:
      prometheus:
        enabled: true
  endpoints:
    web:
      exposure:
        include: "*"
  endpoint:
    prometheus:
      enabled: true
    metrics:
      enabled: true
```

配置完成后，重启 Spring Boot 项目，此时就可以访问 prometheus 端点了，如下：



从这里可以看到系统的各项运行数据，数据已经汇总了，但是还没有可视化，因此，接下来我们需要下载 Prometheus 并安装。

Prometheus 下载地址

下载完成后，将下载文件解压，在解压目录中可以看到 prometheus.yml 配置文件，在 prometheus.yml 文件中配置要查看的数据接口，如下：

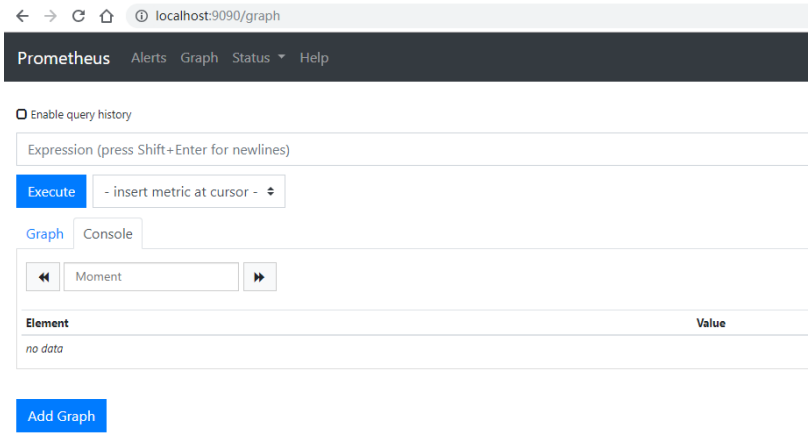
```yaml
scrape_configs:
  # The job name is added as a label `job=<
  - job_name: 'prometheus'
    scrape_interval: 5s
    metrics_path: '/actuator/prometheus'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['localhost:8080']
```

要配置的地方就三处：

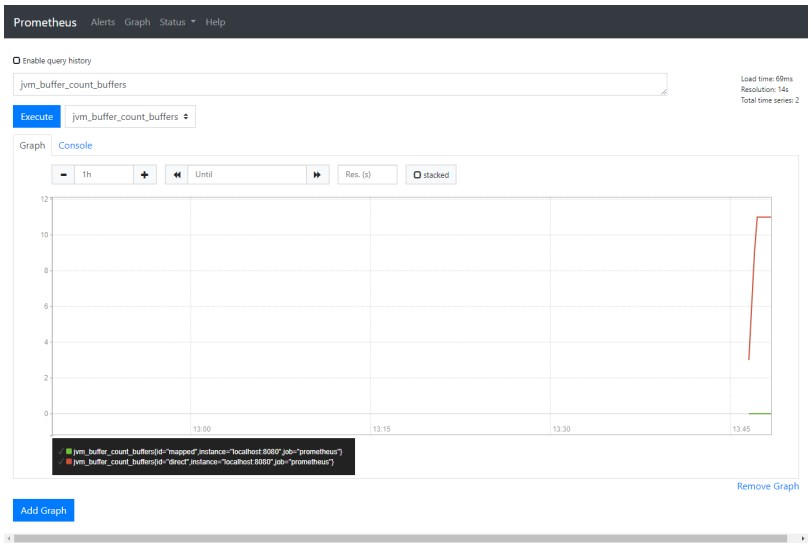
1. scrape_interval 表示每隔 5 秒抓取一次数据；
2. metrics_path 表示数据路径；
3. targets 中配置的则是服务地址。

配置完成后，Windows 环境下，直接双击 prometheus.exe 启动 Prometheus ，如果是 Linux 环境，则执行 `./prometheus --config.file=prometheus.yml` 命令启动 Prometheus 。

启动成功后，在浏览器中输入 `http://localhost:9090` ，看到如下界面，说明服务已经搭建成功了：



上面的选项卡分别是警告、图表展示、状态以及帮助，默认看到的就是图表，在下拉框中选择要查看的参数，点击 Execute 按钮，即可看到相关数据：



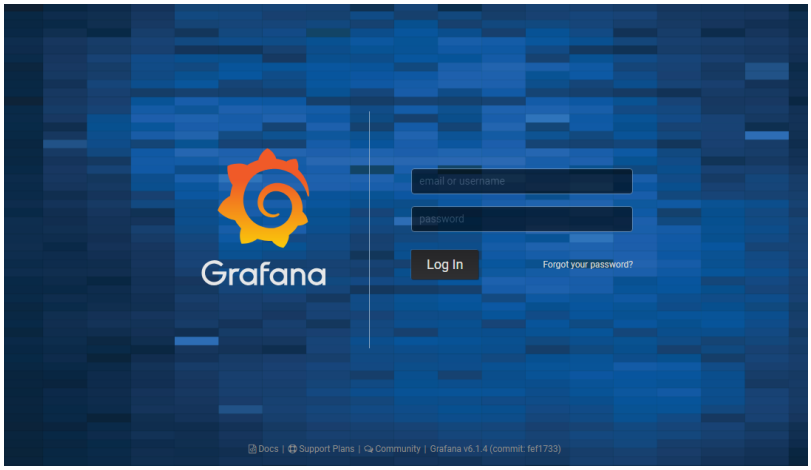
Grafana

仅仅做到上面这一步还不够，前面和大家说过 Prometheus 的工作原理，在 Prometheus 工作流程的最后一步我们提到， Prometheus 中的数据可以通过 Grafana 图表更好地展现出来，这里我们就再结合 Grafana 来看下使用。

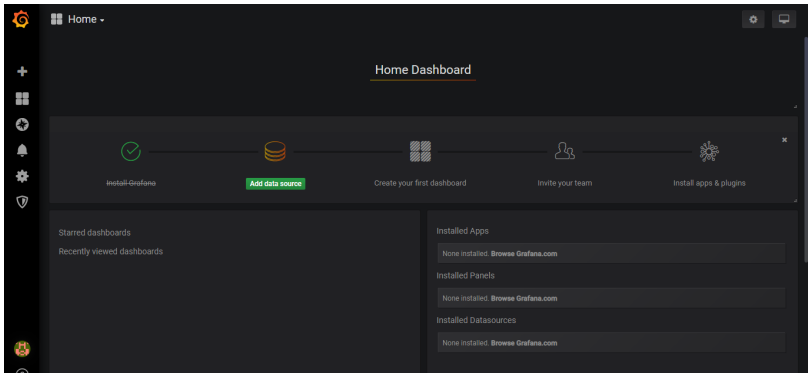
Grafana 是一个开源的指标量监测和数据可视化工具，常见的使用场景是展示基础设施的时序数据并且对应用程序运行进行分析。 Grafana 的 Dashboard 可以用一个非常炫酷的方式将监控数据展示出来，Grafana 可以展示多种不同的数据源， Prometheus 只是其中一种。

Grafana下载地址

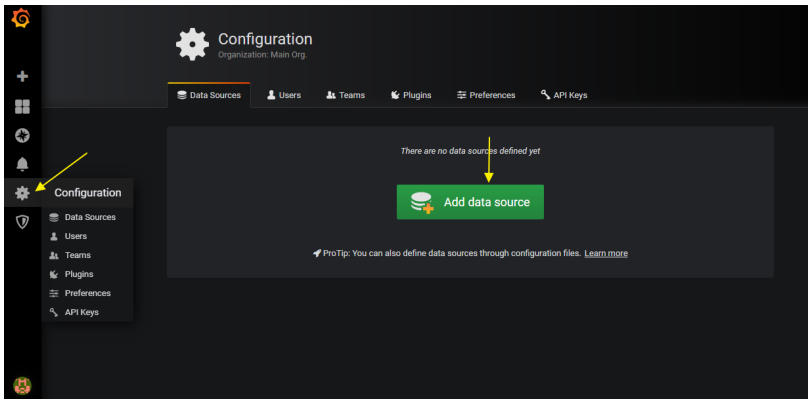
下载成功之后，对下载文件进行解压。解压后，双击 bin 目录下的 grafana-server.exe 文件启动 Grafana ，启动之后，在浏览器中输入 `http://localhost:3000` 即可看到非常炫酷的登录界面，默认的用户名密码都是 `admin` ，如下：



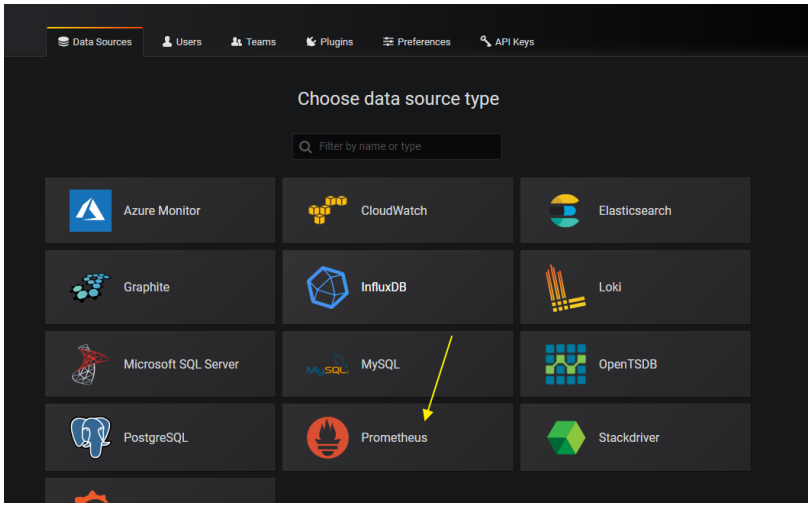
使用默认，密码进行登录，首次登录需要修改密码，修改之后就算登录成功了，如下图：



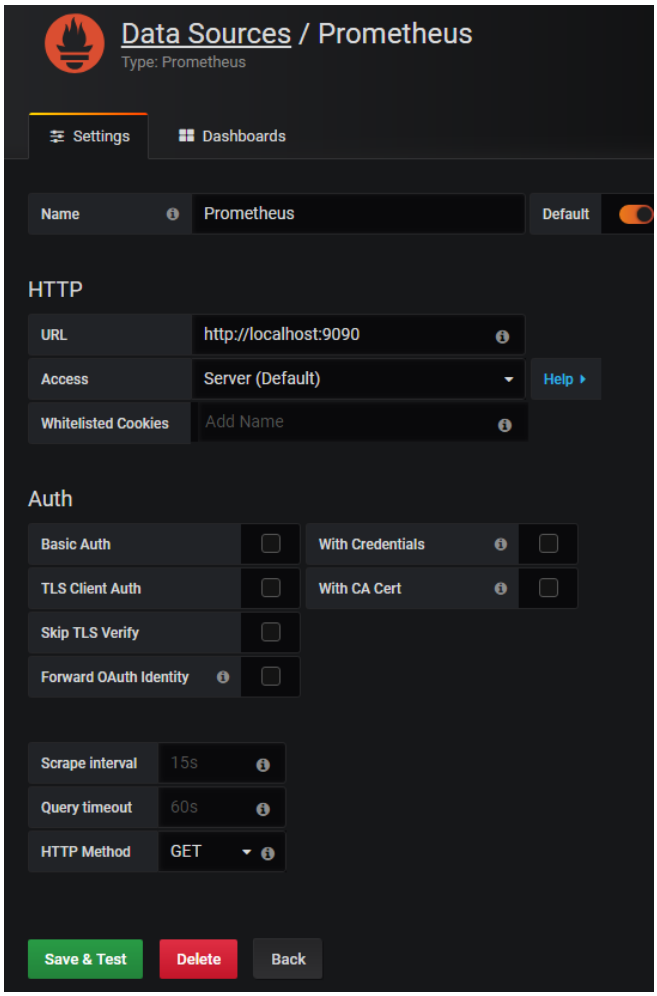
接下来点击左边的设置配置数据源：



在配置数据源时，选择 Prometheus ， 如下：



然后对 Prometheus 进行配置：

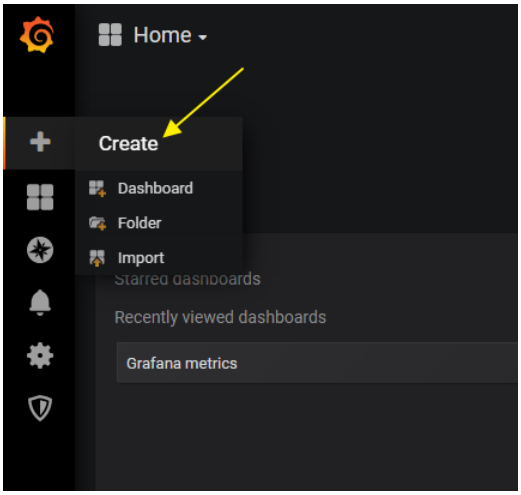


可以看到，这里只需要配置一下基本信息即可：

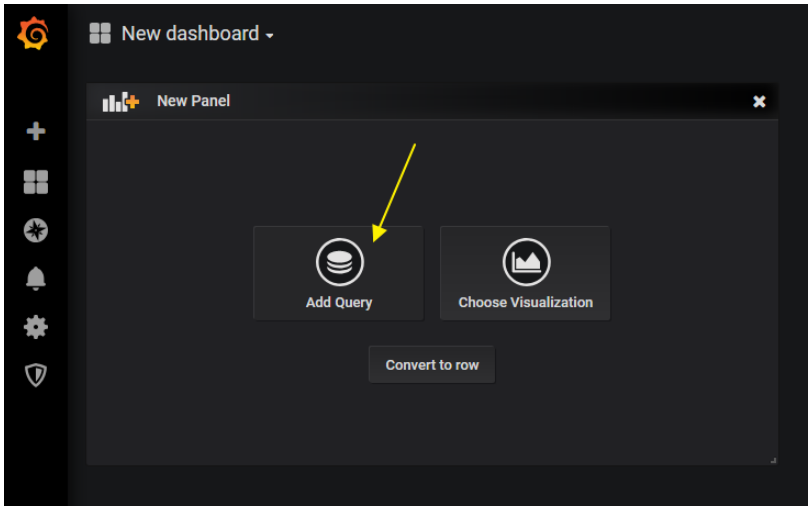
1. name 表示数据源名字，这个自己随意取一个；
2. HTTP 中的 URL 表示数据的地址，配置 Prometheus 的数据地址即可；
3. 配置完成后，点击最下面的 `Save & Test` 按钮。

接下来我们就可以通过 Dashboard 来查看监控数据了，如下：

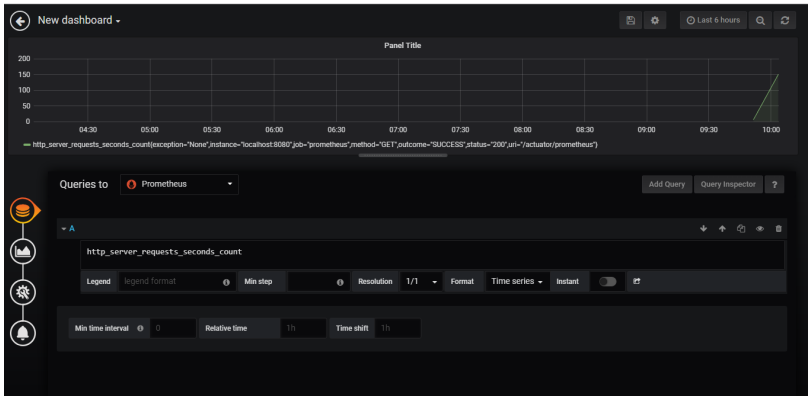
首先点击左边栏添加 Dashboard ，如下：



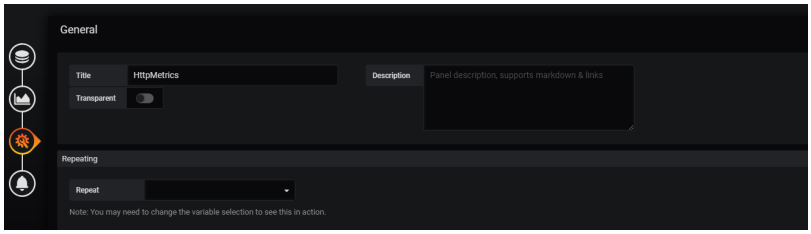
在新的页面中点击 Add Query 按钮，如下：



在新打开的页面中，主要做如下配置：

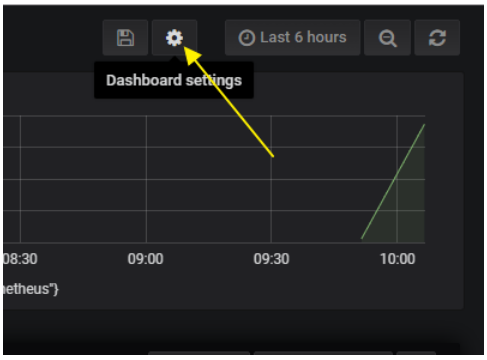


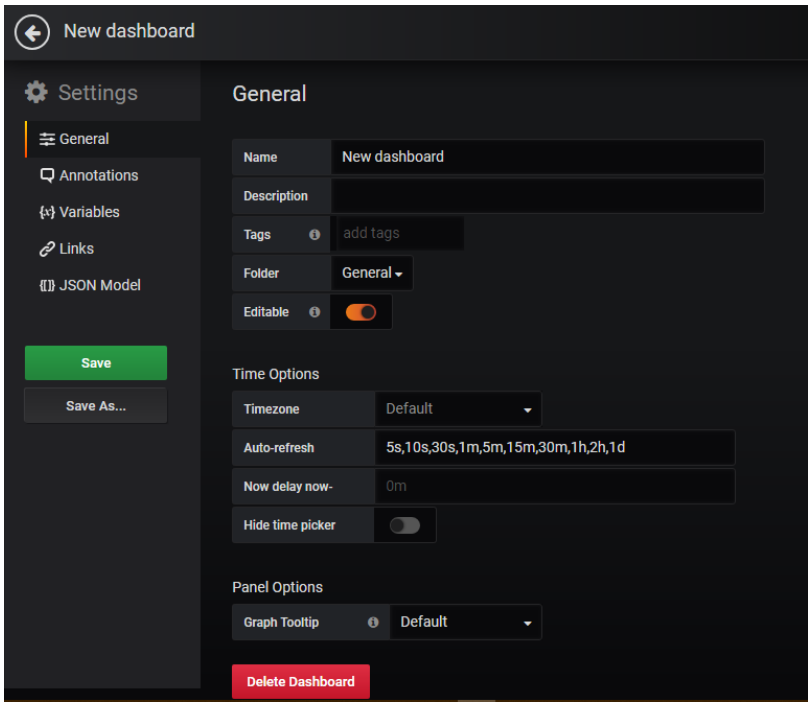
刚进来，只需要添加两个配置，一个是选择 Queries to 为我们一开始配置的数据源，然后输入要查询的数据，这个数据要是不清楚怎么写的话，可以参考一开始 Prometheus 中的写法。这两个配置完成后，光标移动到其它地方，相关数据就在图表中显示出来了，如上图。当然开发者也可以点击左边的 Setting 按钮，进行关于 Panel 的更详细的配置，例如要修改 Panel 的名字，如下：



这里可以配置 Panel 的 Title 属性，Panel 的详细描述以及图表背景是否透明等属性。

当然开发者也可以点击当前面板右上角的设置图标，来进行详细配置，如下图：



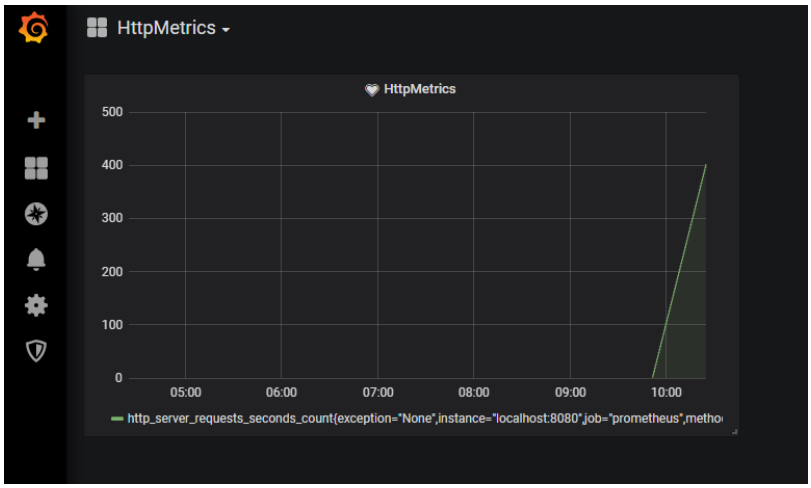


这里配置的参数含义分别如下：

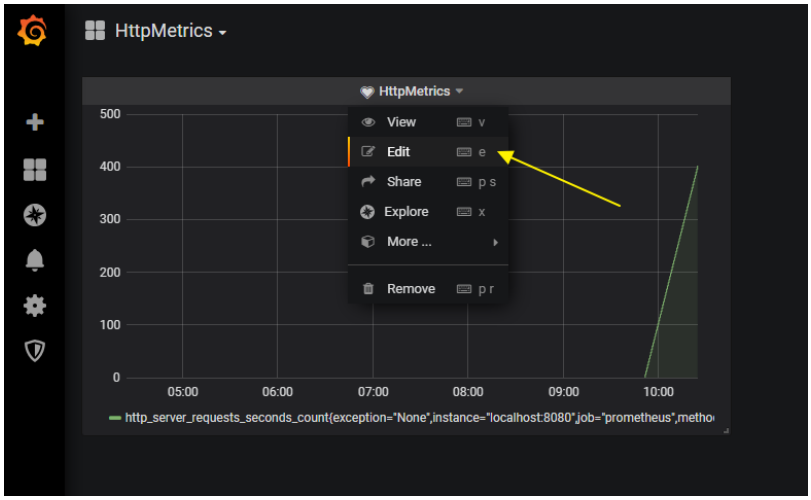
1. Name 表示面板的名称
2. Description 表示对面板的描述
3. Tags 表示可以给面板设置一个标签
4. Folder 表示这个面板归属的文件夹，开发者可以创建多个文件夹，对面板分类管理
5. Editable 表示面板是否可编辑的
6. Timezone 表示时区
7. Auto-refresh 表示面板数据自动刷新时间间隔
8. Now delay now- 表示延迟时间
9. Hide time picker 表示是否隐藏时间控件

配置完成后，就可以点击左边的 Save 按钮，将面板保存下来。

保存完成后，效果如下：



点击 Panel 的 title 属性，可以对面板继续进行编辑，如下图：



好了，这是单个面板的创建，如果要创建多个面板，重复上面的步骤即可，不再赘述。

警报

让运维工程师一直盯着 Panel 查看服务是否出错，显然不太现实，我们可以开启服务出错警报功能来解决这一难题。

我们在 Prometheus 中可以定义 AlertRule ，Prometheus 会定时对 AlertRule 进行计算，看看是否满足，如果满足条件就会向 Alertmanager 发送告警信息，因此这里的流程实际上是分为两个步骤，接下来分别向大家介绍。

AlertRule

AlertRule 是我们在 Prometheus 中定义的告警规则，我这里通过一个服务宕机的例子来给大家演示 AlertRule 的配置方式。

首先我们需要额外定义一个 yaml 配置文件，用来描述所有的 AlertRule，我这里定义了名为 `rules.yml` 的文件，内容如下：

```yaml
groups:
- name: server-dwon
  rules:
  - alert: InstanceDown
    expr: up==0
    for: 5s
    labels:
      severity: page
    annotations:
      summary: "Instance {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 seconds."
```

各项配置的含义分别如下：

1. name: 表示配置的名称
2. alert：告警规则的名称
3. expr：基于 PromQL 表达式告警触发条件，用于计算是否有时间序列满足该条件，关于 PromQL 读者可以参考初识PromQL
4. for：评估等待时间，可选参数。用于表示只有当触发条件持续一段时间后才发送告警。在等待期间新产生告警的状态为 pending
5. labels：自定义标签，允许用户指定要附加到告警上的一组附加标签
6. annotations：用于指定一组附加信息，比如用于描述告警详细信息的文字等，annotations的内容在告警产生时会一同作为参数发送到Alertmanager

这个 rules.yml 文件可以定义在任意位置，但是建议大家放在 Prometheus 的安装目录中，这样方便查找，rules.yml 配置完成后，接下来我们需要在 Prometheus 的配置文件中加载这个配置，如下：

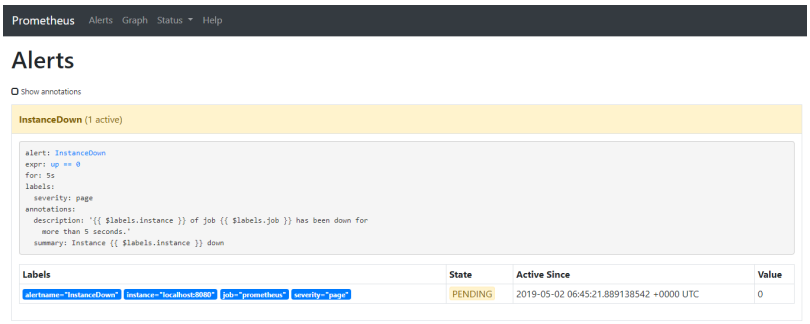
prometheus.yml

```yaml
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Defa
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is e
  # scrape_timeout is set to the global default (10s).
# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evalu
rule_files:
  - C:\devtools\prometheus-2.9.1.windows-amd64\rules.yml
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
```

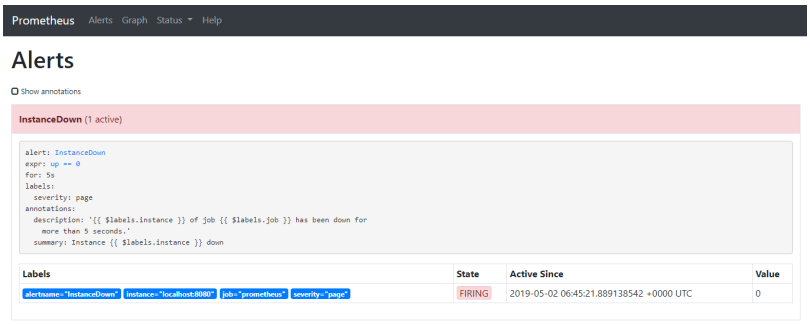
配置完成后，就可以启动 Prometheus 了（如果已经启动了，则需要重启），启动之后，点击 Prometheus 标题栏的 Alerts，就可以看到我们定义的 AlertRule 了，如下：

能看到这个页面，表示配置成功，0 表示这个告警规则未被触发。

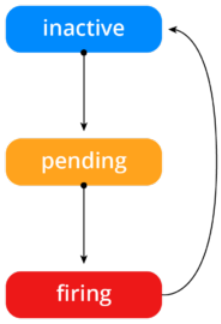
此时，我们尝试关闭 Spring Boot 项目，关闭之后，最多 5 秒，我们刷新 Prometheus 的 Alerts 页面，如下：



表示规则已经被触发，但是 State 的值是 PENDING，再停留一会我们再次刷新该页面，结果如下：



此时 State 的值变为 FIRING ，前后三个不同的 State 值，说明了 Prometheus 在这里的工作过程，如下：



- inactive：警报未被触发，一切安好。
- Pending：这个警报必须被触发，但是，警报可以被分组、压抑/抑制或者静默/静音。一旦所有的验证都通过了，告警就转到 Firing。
- Firing：警报发送到 Notification Pipeline ，它将联系警报的所有接收者，警报解除后，又回到 inactive 状态。

这是 AlertRule 的配置，配置完成之后，现在还不能实现自动邮件报警，要实现邮件报警，还需要配置 AlertManager，请继续看下文。

Alertmanager

准备工作

我们得先做一点邮件发送的准备工作，这里我以网易邮箱为例，来给大家演示一下邮件发送的准备工作：

首先我们需要先登录网易邮箱网页版，点击上方的设置按钮：

然后勾选开启 POP3/SMTP 服务：

开启过程需要手机号码验证，按照步骤操作即可，不赘述。开启成功之后，需要自己设置一个客户端登录授权码，将该号码保存好，一会使用。

如此之后，准备工作就算完成了。

Alertmanger

配置 Alertmanager 需要我们首先去下载 Alertmanager，这是一个 GitHub 上的开源项目，我们直接下载二进制安装包即可。

Alertmanager下载地址

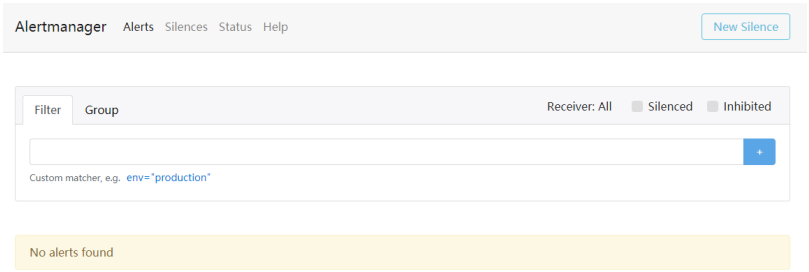
下载之后解压即可。

解压之后，找到 alertmanager.yml 配置文件，在这个文件中添加邮件发送相关的配置，如下：

```yaml
global:
  resolve_timeout: 5s
  smtp_smarthost: 'smtp.163.com:465'
  smtp_from: 'wa[illegible]ng@163.com'
  smtp_auth_username: 'wa[illegible]ng@163.com'
  smtp_auth_password: '[illegible]'
  smtp_require_tls: false
route:
  group_by: ['alertname']
  group_wait: 10s
  group_interval: 10s
  repeat_interval: 1h
  receiver: 'email'
receivers:
- name: 'email'
  email_configs:
  - to: '5[illegible]@qq.com'
inhibit_rules:
  - source_match:
      severity: 'critical'
    target_match:
      severity: 'warning'
    equal: ['alertname', 'dev', 'instance']
```

global 中配置的是发件账户的信息，这里的密码是指准备工作配置的授权码。receivers 配置的收件人的信息，收件人可以配置多个。

配置完成后，双击 alertmanager.exe 启动 Alertmanager ，启动成功之后，浏览器输入 `http://localhost:9093` ，结果如下：

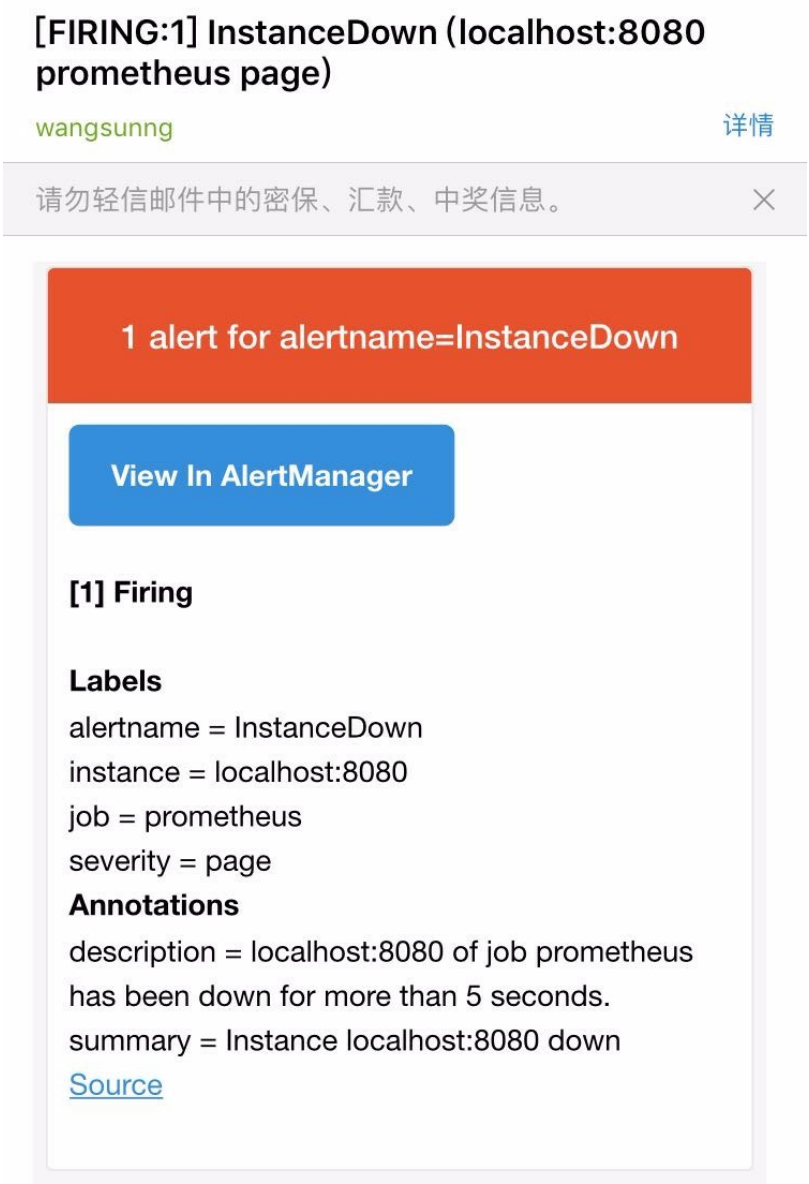


这就表示 Alertmanager 启动成功了。

接下来，将 Prometheus 和 Alertmanager 关联起来即可，修改 Prometheus 的配置文件，如下：

```yaml
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 se
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The c
  # scrape_timeout is set to the global default (10s).
# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets: [localhost:9093]

# Load rules once and periodically evaluate them according to the g
rule_files:
  - C:\devtools\prometheus-2.9.1.windows-amd64\rules.yml
  # - "second_rules.yml"
```

配置完成后，重启 Prometheus。启动成功后，此时我们再次尝试关闭 Spring Boot 项目，关闭后，稍等一会，就会收到一封服务下线告警邮件，如下：



注意，这个邮件可能被判定为垃圾邮件，如果没收到可以去垃圾箱看下有没有。

好了，至此，我们的邮件告警就算全部配置完成了。

关于自定义告警邮件模板等，大家可以参考告警模板详解。

小结

本文主要和大家分享了 Micrometer 监控微服务，顺便说了 Prometheus、Grafana以及 Alertmanger 的用法，有了这些工具的配合，可以有效地提高运维工程师的工作效率。本章至此，也就先告一个段落。

参考资料：

1. prometheus-book
2. 如何以优雅的姿势监控kubernetes

本文作者：纯洁的微笑、江南一点雨

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论