链滴

基于 Prometheus 和 Grafana 的监控平台 - 运维告警

作者：jianzh5

原文链接：https://ld246.com/article/1574661714703

来源网站：链滴

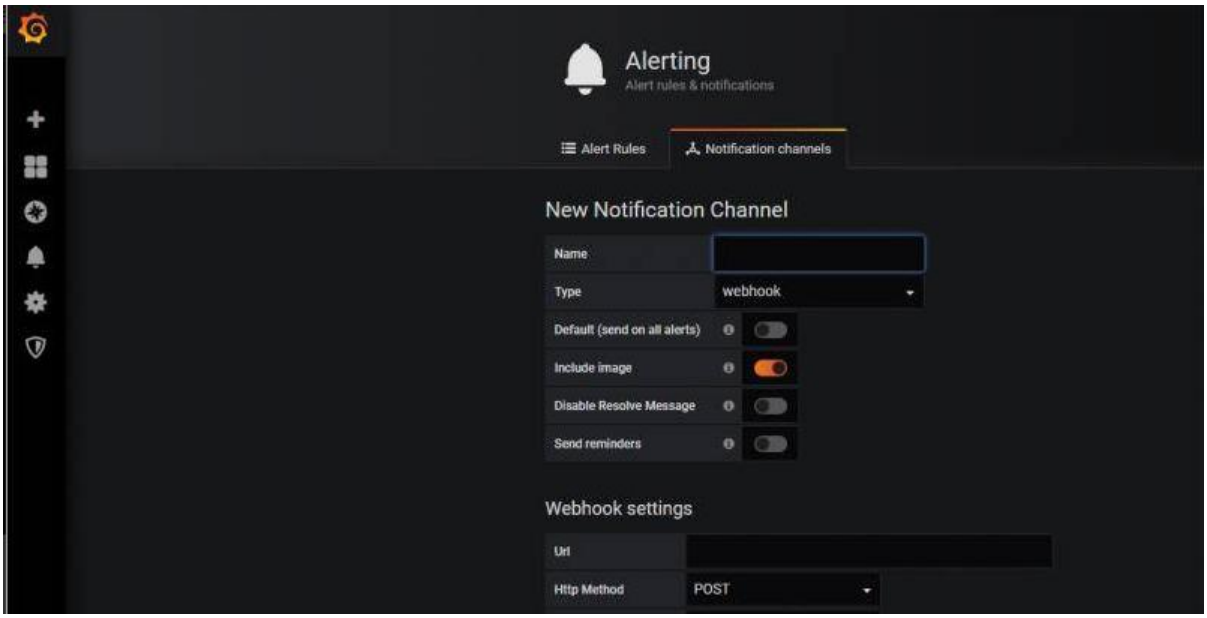
通过前面几篇文章我们搭建好了监控环境并且监控了服务器、数据库、应用，运维人员可以实时了解前被监控对象的运行情况，但是他们不可能时时坐在电脑边上盯着DashBoard，这就需要一个告警功，当服务器或应用指标异常时发送告警，通过邮件或者短信的形式告诉运维人员及时处理。

今天我们就来聊聊 基于Prometheus和Grafana的监控平台的异常告警功能。

告警方式

Grafana

新版本的Grafana已经提供了告警配置，直接在dashboard监控panel中设置告警即可，但是我用过发现其实并不灵活，不支持变量，而且好多下载的图表无法使用告警，所以我们不选择使用Grafana警，而使用Alertmanager。



Alertmanager

相比于Grafana的图形化界面，Alertmanager需要依靠配置文件实现，配置稍显繁琐，但是胜在功能大灵活。接下来我们就一步一步实现告警通知。

告警类型

Alertmanager告警主要使用以下两种：

- 邮件接收器 email_config
- Webhook接收器 webhook_config，会用post形式向配置的url地址发送如下格式的参数。

```
{
  "version": "2",
  "status": "<resolved|firing>",
  "alerts": [{
        "labels":  < object > ,
        "annotations":  < object > ,
        "startsAt": "<rfc3339>",
        "endsAt": "<rfc3339>"
        }]
  }
```

这次主要使用邮件的方式进行告警。

实现步骤

- 下载

从GitHub上下载最新版本的Alertmanager,将其上传解压到服务器上。

```
tar -zxvf  alertmanager-0.19.0.linux-amd64.tar.gz
```

- 配置Alertmanager

```
vi alertmanager.yml
global:
  resolve_timeout: 5m
  smtp_smarthost: 'mail.163.com:25' #邮箱发送端口
  smtp_from: 'xxx@163.com'
  smtp_auth_username: 'xxx@163.com' #邮箱账号
  smtp_auth_password: 'xxxxxx' #邮箱密码
  smtp_require_tls: false
route:
  group_by: ['alertname']
  group_wait: 10s  # 最初即第一次等待多久时间发送一组警报的通知
  group_interval: 10s # 在发送新警报前的等待时间
  repeat_interval: 1h # 发送重复警报的周期 对于email配置中，此项不可以设置过低，否则将会由邮件发送太多频繁，被smtp服务器拒绝
  receiver: 'email'
receivers:
  - name: 'email'
    email_configs:
    - to: 'xxx@xxx.com'
```

修改完成后可以使用 ./amtool check-config alertmanager.yml校验文件是否正确。

```
[root@localhost alertmanager]# ./amtool check-config alertmanager.yml
Checking 'alertmanager.yml'  SUCCESS
Found:
 - global config
 - route
 - 1 inhibit rules
 - 2 receivers
 - 0 templates
```

校验正确后启动alertmanager。nohup ./alertmanager &。（第一次启动可以不使用nohup静默启，方便后面查看日志）

我们只定义了一个路由，那就意味着所有由Prometheus产生的告警在发送到Alertmanager之后都会过名为email的receiver接收。实际上，对于不同级别的告警，会有不同的处理方式，因此在route中我们还可以定义更多的子Route。具体配置规则大家可以去百度进一步了解。

● 配置Prometheus

在Prometheus安装目录下建立rules文件夹，放置所有的告警规则文件。

```
alerting:
   alertmanagers:
   - static_configs:
       - targets: ['192.168.249.131:9093']

rule_files:
   - rules/*.yml
```

在rules文件夹下建立告警规则文件service_down.yml,当服务器下线时发送邮件。

```
groups:
 - name: ServiceStatus
   rules:
    - alert: ServiceStatusAlert
      expr: up == 0
      for: 2m
      labels:
        team: node
      annotations:
        summary: "Instance {{ $labels.instance }} has bean down"
        description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 2
minutes."
        value: "{{ $value }}"
```

配置详解

alert：告警规则的名称。

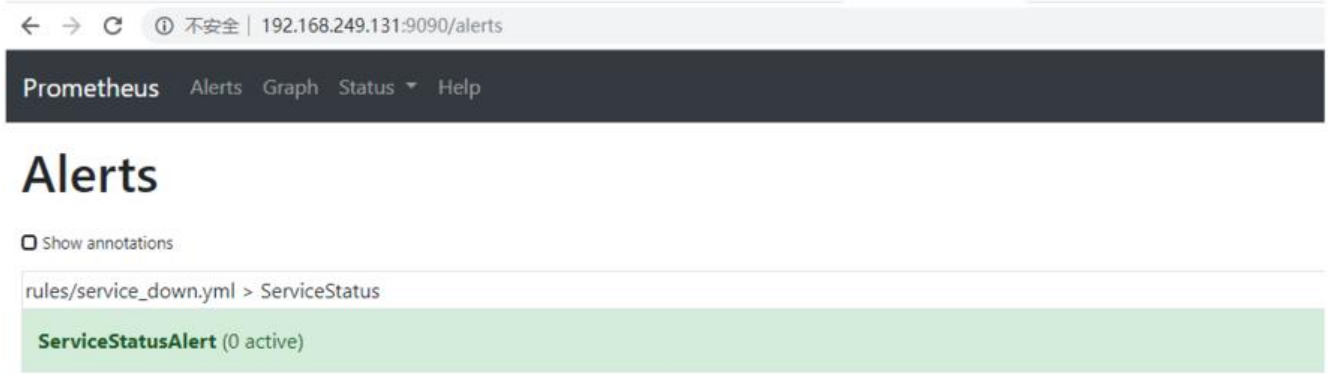
expr：基于PromQL表达式告警触发条件，用于计算是否有时间序列满足该条件。

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。在等待期间新生告警的状态为PENDING，等待期后为FIRING。

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。

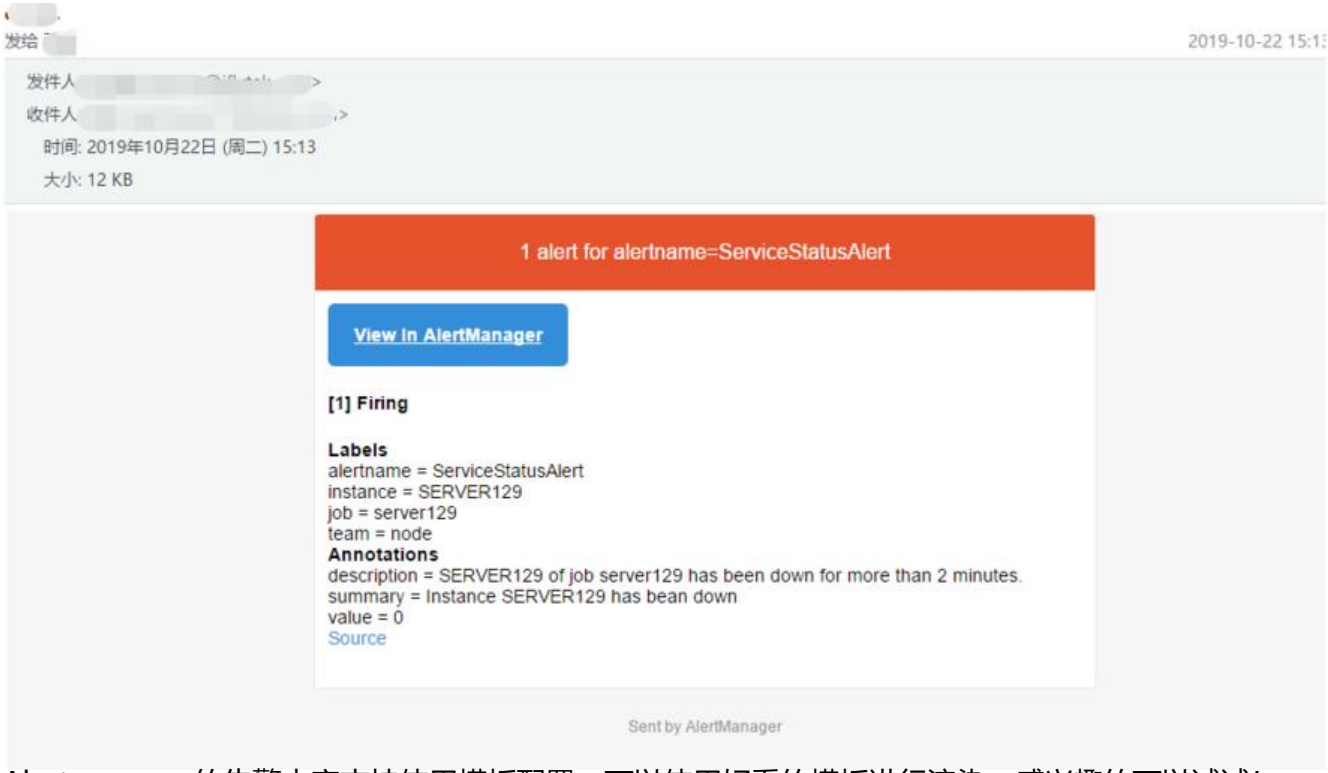
annotations：用于指定一组附加信息，比如用于描述告警详细信息的文字等，annotations的内容告警产生时会一同作为参数发送到Alertmanager。

配置完成后重启Prometheus，访问Prometheus查看告警配置。



- 测试

关闭node_exporter,过2分钟就可以收到告警邮件啦，截图如下：



Alertmanager的告警内容支持使用模板配置，可以使用好看的模板进行渲染，感兴趣的可以试试！

The More

node exporter的一些计算语句

- CPU使用率(单位为percent)

```
(avg by (instance) (irate(node_cpu_seconds_total{mode="idle"}[5m])) * 100)
```

- 内存已使用(单位为bytes)

```
node_memory_MemTotal_bytes - node_memory_MemFree_bytes - node_memory_Cached_bytes - node_memory_Buffers_bytes - node_memory_Slab_bytes
```

- 内存使用量(单位为bytes/sec)

```
node_memory_MemTotal_bytes - node_memory_MemFree_bytes - node_memory_Cached_by
```

es - node_memory_Buffers_bytes - node_memory_Slab_bytes

- 内存使用率(单位为percent)

((node_memory_MemTotal_bytes - node_memory_MemFree_bytes - node_memory_Cached_bytes - node_memory_Buffers_bytes - node_memory_Slab_bytes)/node_memory_MemTotal_bytes) * 100

- server1的内存使用率(单位为percent)

((node_memory_MemTotal_bytes{instance="server1"} - node_memory_MemAvailable_bytes{instance="server1"})/node_memory_MemTotal_bytes{instance="server1"}) * 100

- server2的磁盘使用率(单位为percent)

((node_filesystem_size_bytes{fstype=~"xfs|ext4",instance="server2"} - node_filesystem_free_bytes{fstype=~"xfs|ext4",instance="server2"}) / node_filesystem_size_bytes{fstype=~"xfs|ext4",instance="server2"}) * 100

- uptime时间(单位为seconds)

time() - node_boot_time

- server1的uptime时间(单位为seconds)

time() - node_boot_time_seconds{instance="server1"}

- 网络流出量(单位为bytes/sec)

irate(node_network_transmit_bytes_total{device!~"lo|bond[0-9]|cbr[0-9]|veth.*"}[5m]) > 0

- server1的网络流出量(单位为bytes/sec)

irate(node_network_transmit_bytes_total{instance="server1", device!~"lo|bond[0-9]|cbr[0-9]|veth.*"}[5m]) > 0

- 网络流入量(单位为bytes/sec)

irate(node_network_receive_bytes_total{device!~"lo|bond[0-9]|cbr[0-9]|veth.*"}[5m]) > 0

- server1的网络流入量(单位为bytes/sec)

irate(node_network_receive_bytes_total{instance="server1", device!~"lo|bond[0-9]|cbr[0-9]|veh.*"}[5m]) > 0

- 磁盘读取速度(单位为bytes/sec)

irate(node_disk_read_bytes_total{device=~"sd.*"}[5m])