使用 Prometheus 对网站进行监控和告警

懒人某天闲得慌，想起来要把我的博客（https://ilostmyname.com/ ）监控一下，有问题自动发个告警邮件。

监控方面的工具挺多的，我搜了一圈，发现 Prometheus 挺热门，就用它了。

实际用起来才发现，这玩意挺复杂，不是那种下载安装就能 run 的。我费了大半天劲，才把它跑起来。

主要复杂的地方在于 Prometheus 是插件式的，要管理它的众多插件。

Prometheus 本身并不进行监控，它是一个时序数据库，带 web 管理系统，支持按多种条件，对监控事件进行时序查询。

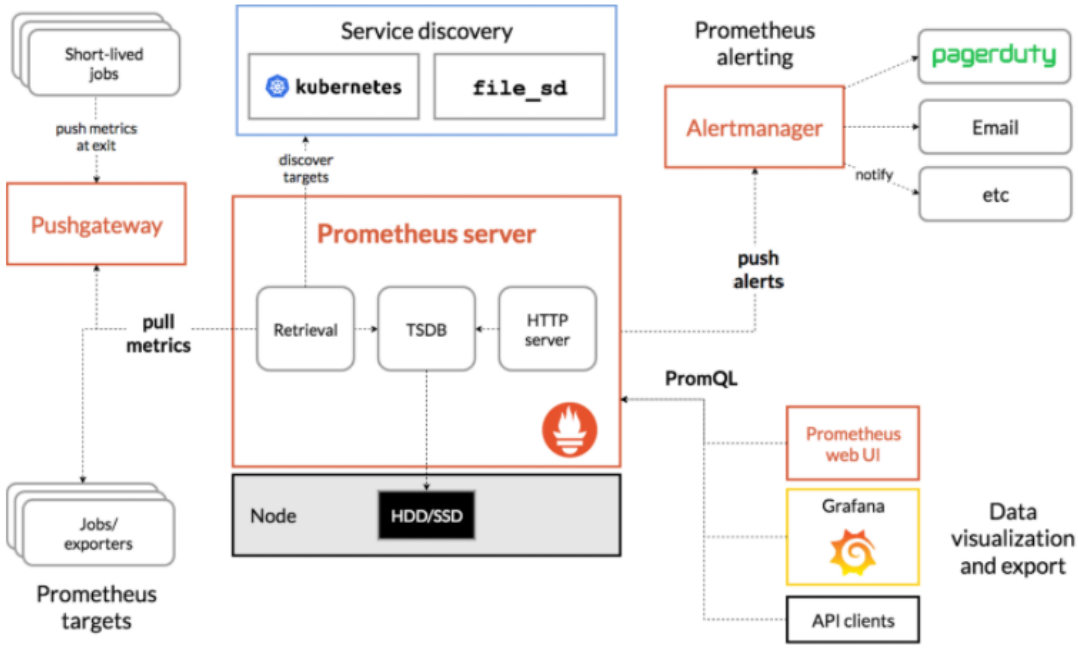
具体监控任务，交给插件进行，Prometheus 体系里把监控插件叫做 exporter。比如，监控本机的系统资源，cpu, mem, disk, bandwidth 等等，由一个叫 node_exporter 的插件完成。

如果监控外部资源，比如网站的 url，那就使用一个叫黑盒（blackbox_exporter）的插件。黑盒的意思是不需要在被监控服务器上部署这个插件，而是从外部暴露的接口进行监控。

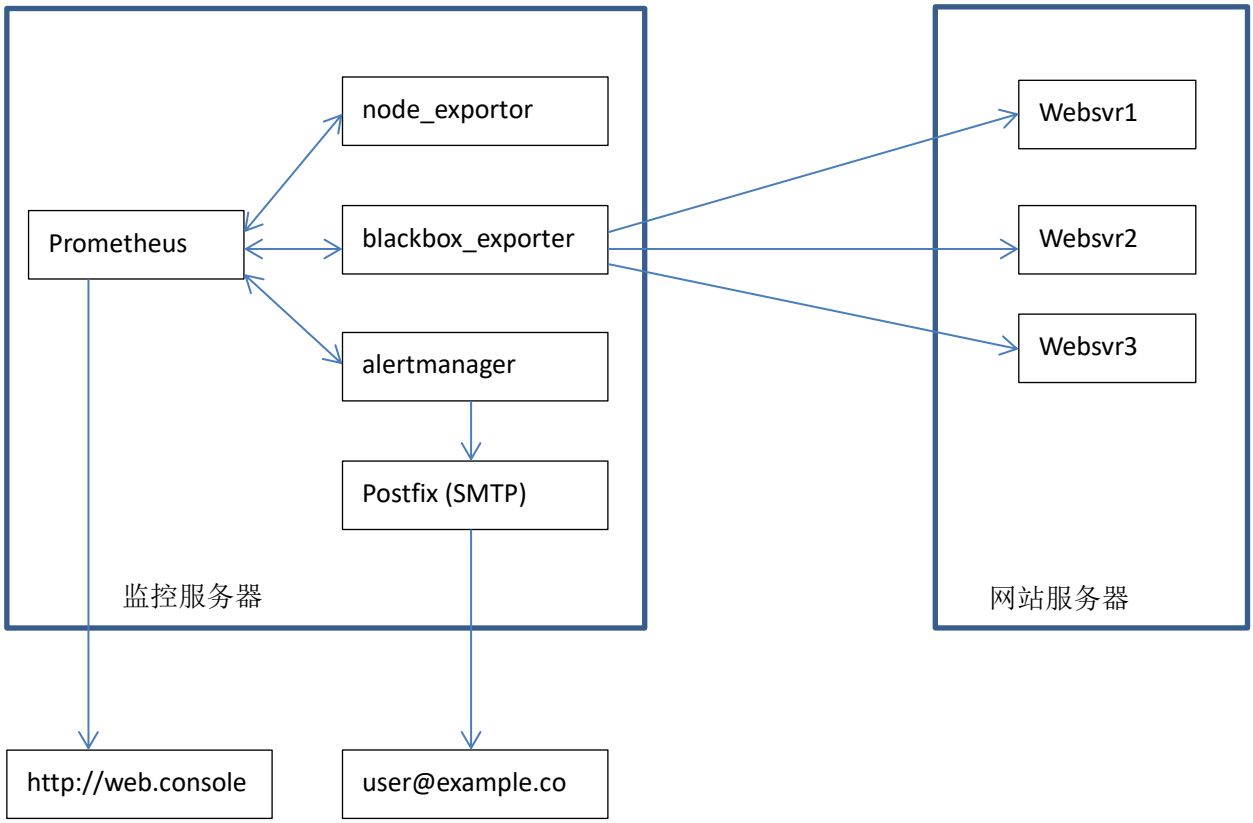
同样，告警也是由一个叫做 alertmanager 的插件完成。这个插件管理具体的告警方式，比如邮件告警、短信告警的发送和接收渠道。但是，触发告警的 rules，还是在 Prometheus 本身的配置文件里设置。

Prometheus 体系的结构图如下图，引用自官网：

https://prometheus.io/docs/introduction/overview/


Short-lived jobs
push metrics at exit
Pushgateway
Service discovery
kubernetes
file_sd
discover targets
Prometheus server
Retrieval
TSDB
HTTP server
pull metrics
Node
HDD/SSD
Jobs/ exporters
Prometheus targets
Prometheus alerting
Alertmanager
pagerduty
Email
etc
notify
push alerts
PromQL
Prometheus web UI
Grafana
API clients
Data visualization and export

我部署的方式如下图：


node_exportor
Prometheus
blackbox_exporter
alertmanager
Postfix (SMTP)
监控服务器
Websvr1
Websvr2
Websvr3
网站服务器
http://web.console
user@example.co

在一台服务器上部署了 Prometheus 程序，以及上面提到的 3 个插件（两个监控、一个告警）。其中 node_exporter 插件在这里用途不大，它用来监控安装了 Prometheus 程序的监控机自身。

监控机是 Ubuntu 18.04 系统，64 位。网站服务器上部署了 3 个网站，它们是普通的 HTTP webserver。

这里用到的是 blackbox_exporter 插件和 alertmanager 插件，一个用来监控博客和网站，一个用来告警。

接下来描述如何配置它们。

首先，安装上述所有程序。

到这个页面下载所有程序：https://prometheus.io/download/

请下载 Prometheus 和各个插件的二进制程序，下载最新的和你的平台对应的版本。

下载完后解压，然后进入到程序目录，直接运行即可。

我的是 Linux 版。作者也挺懒，居然都没做成系统服务，直接在前台运行。

我也更乐的省事，直接一个 Shell 脚本，用 nohup 的方式，在后台启动了所有服务。

Shell 脚本如下：

```
#!/bin/bash

# node monitor for local linux's cpu,mem,disk etc
cd /root/node_exporter-1.0.1.linux-amd64
nohup ./node_exporter &

# http monitor for remote host
cd /root/blackbox_exporter-0.17.0.linux-amd64
nohup ./blackbox_exporter &

#alert manager
cd /root/alertmanager-0.21.0.linux-amd64
nohup ./alertmanager --cluster.listen-address="" --config.file=alertmanager.yml &

# start prometheus itself
cd /root/prometheus-2.21.0-rc.0.linux-amd64
nohup ./prometheus --config.file=prometheus.yml &
```

注意 alertmanager 启动时，要加上--cluster.listen-address=""这个选项，表示不启用集群。

执行上述 Shell 脚本，就启动了所有 4 个服务。

上述 4 个服务打开的端口如下：

启动后运行 netstat –ntlp 可以看到进程打开的所有端口。

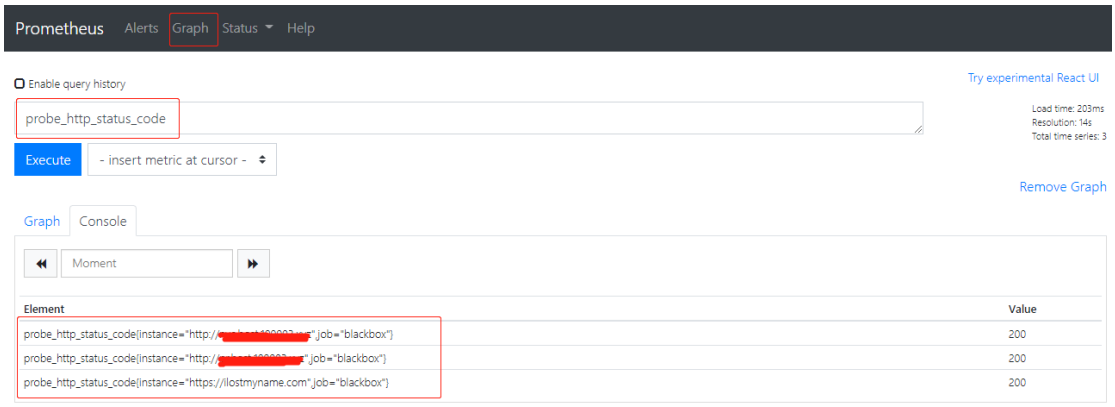
如果程序运行有问题，请进入对应的程序目录，tail –f nohup.out 查看进程的输出日志。

使用如下 url 访问 Prometheus 服务的 web 管理页面：

http://sample.com:9090/graph

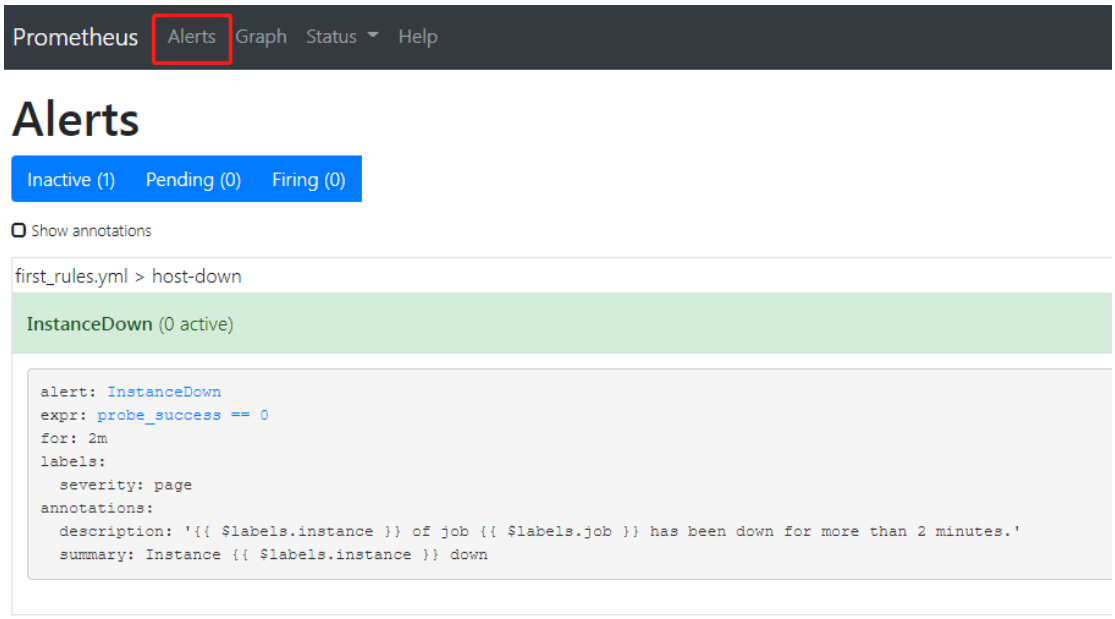
sample.com 是监控机的域名，Prometheus 的 web server 运行在 9090 端口。

如下是 Graph 功能，这里可以对 metrics 进行查询，比如我查询 probe_http_status_code 这条 metrics，在 web 页面上展示出对 3 个网站进行监控的结果。



点上述 Console 旁边的 Graph 按钮，可以按照时序关系，画出监控结果的历史报表图。

如下是 Alerts 功能，查看配置的告警方式。



上述表示系统里配置了一条处于非激活状态的告警，文本内容是告警规则（rules）。

那么，这 4 个服务是如何配置的呢？下面一一道来。

（1） 配置 Prometheus 服务

Prometheus 有 2 份配置文件，一份是 Prometheus 自身，一份是告警的 rules。

Prometheus 自身的配置文件是 prometheus.yml，位于程序同一目录。

我的内容如下：

```
# my global config
global:
  scrape_interval:     60s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 60s # Evaluate rules every 15 seconds. The default is every 1 minute.

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
        - 127.0.0.1:9093   #这里重要，指定 alertmanager 的地址端口

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - "first_rules.yml"   # 加载第一个告警规则文件
  # - "second_rules.yml"
```

```
# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'   # 这里是 prometheus 自身的状态监控

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['localhost:9090']


  - job_name: node   # 这里配置 node_exporter 的监控方式
    static_configs:
    - targets: ['localhost:9100']


  - job_name: 'blackbox'   # 这里配置 blackbox_exporter 的监控方式
    metrics_path: /probe
    params:
      module: [http_2xx]   # Look for a HTTP 200 response.
    static_configs:
      - targets:
        - https://ilostmyname.com       # 监控我的博客
        - http://sample.com     # 监控第 2 个网站
        - http://sample2.com    # 监控第 3 个网站
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: instance
      - target_label: __address__
        replacement: 127.0.0.1:9115    # The blackbox exporter's real hostname:port.
```

告警规则文件 first_rules.yml，也位于 Prometheus 程序的同一目录。

我的规则如下：

```
groups:
- name: host-down
  rules:
```

```
  # Alert for any instance that is unreachable for >2 minutes.
  - alert: InstanceDown     # 定义告警的 item 名字，可以基于名字进行分组
    expr: probe_success == 0     # blackbox_exporter 的 metrics 里有这条，表示探测失败
    for: 2m    # 持续失败 2 分钟
    labels:
      severity: page
    annotations:    # 定义告警的描述
      summary: "Instance {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 2
minutes."
```

配置完后，运行如下命令工具，检查下格式：

```
# ./promtool check config prometheus.yml
Checking prometheus.yml
  SUCCESS: 1 rule files found

Checking first_rules.yml
  SUCCESS: 1 rules found
```

配置文件相关说明，请见官方文档：

https://prometheus.io/docs/prometheus/latest/configuration/configuration/

（2） 配置 node_exporter 插件

node_exporter 对监控机自身进行监控，不需要特别配置，它甚至都没有配置文件。

（3） 配置 blackbox_exporter 插件

blackbox_exporter 对外部网站服务器进行监控，也不需要特别配置。它有一个默认配置文件 blackbox.yml，我没有去动它。

（4） 配置 alertmanager 插件

alertmanager 定义告警的方式，比如邮件、短信告警。它的配置文件是 alertmanager.yml，位于程序同一目录。

我的内容如下：

```
global:
  smtp_smarthost: 'localhost:25'     # 需要打开本机的 postfix 服务，默认方式开启即可
  smtp_from: 'info@sample.com'     # 发送邮件的地址
  smtp_require_tls: false     # SMTP 通讯的 TLS 设为 false
```

```
route:
  group_by: ['alertname']  # 按告警 item 的名字进行分组
  group_wait: 10s
  group_interval: 10s
  repeat_interval: 1h
  receiver: team-X-mails  # 设置告警方式

receivers:  # 定义告警方式
- name: 'team-X-mails'
  email_configs:
  - to: 'user@xxx.org'    # 设置接收人邮箱


inhibit_rules:  # 这里的规则表示 critical 的规则覆盖 warning 的，避免重复告警
- source_match:
      severity: 'critical'
  target_match:
      severity: 'warning'
  equal: ['alertname']
```

配置完后，运行如下命令检查下格式：

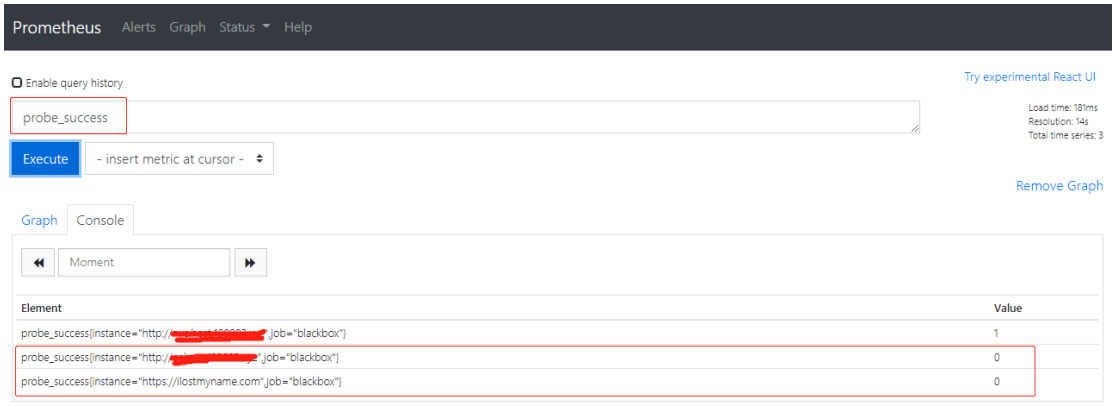
```
# ./amtool check-config alertmanager.yml
Checking 'alertmanager.yml'  SUCCESS
Found:
 - global config
 - route
 - 1 inhibit rules
 - 1 receivers
 - 0 templates
```

alertmanager 相关的配置和说明请见官方文档：

https://prometheus.io/docs/alerting/latest/overview/

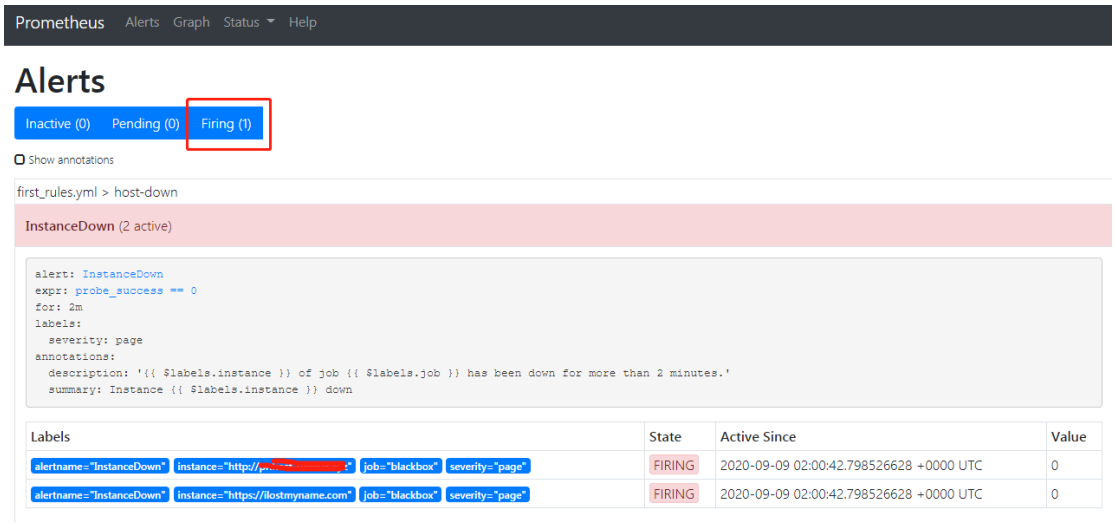
接下来，测试监控和告警。

关掉其中 2 个网站，查看监控页面，如下图：

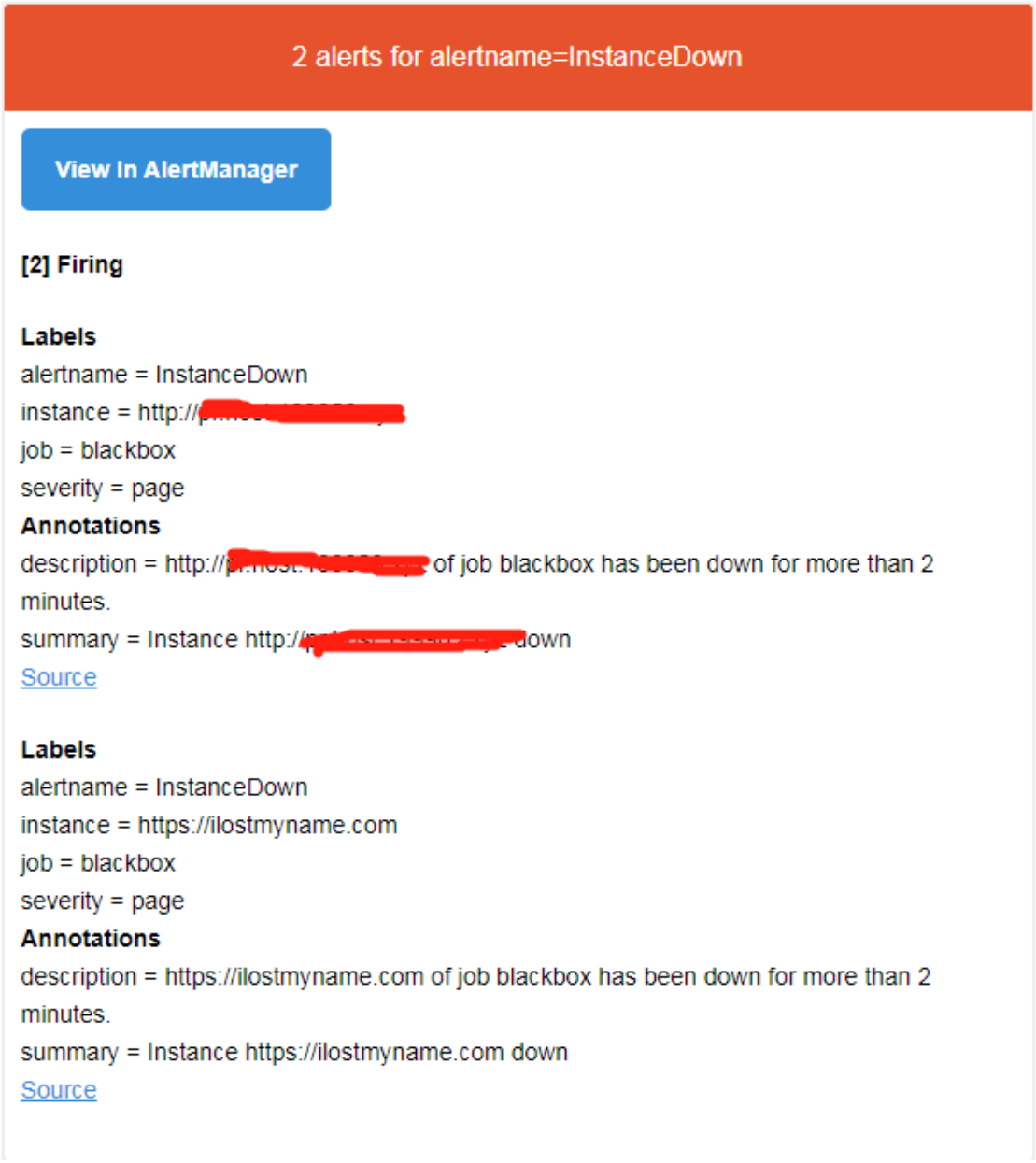


我们查询 probe_success 这条 metrics，被关掉的网站的结果是 0.

再回到告警页面，如下图：



看到一条告警已经被触发了。检查我的邮箱，收到的告警邮件内容如下：



邮件内容告知，由 blackbox 监控的网站实例，有 2 个挂掉了，宕机时间超过 2 分钟。

至此，Prometheus 配置完成，工作正常。

如果你的配置有问题，可以咨询懒人，我有空看一下。联系方式：

https://ilostmyname.com/about