

- 1. 介绍
- 2. 配置如何加载规则文件
- 3. 配置规则文件
  - 3.1. 规则名称规范
  - 3.2. 全局配置框架
  - 3.3. Recording rules（记录规则）
    - 3.3.1 格式讲解
    - 3.3.2 示例
  - 3.4. Alerting rules（警报规则）
    - 3.4.1 格式讲解
    - 3.4.2 示例
  - 3.5. 模板化如何使用
    - 3.5.1 变量
    - 3.5.2 配置示例
- 4. 检查规则文件语法
- 5. 发送警报通知

# 1. 介绍

经过前几篇的学习，相信你一定对 Prometheus 有了初步的了解，并且能够进行部署，监控服务器了。如果想要监控更多的应用服务，并实现告警，这还有一段路程要走。别担心，有我在，我会陪着你慢慢的掌握 Prometheus，并能使用它为你所有，实现真正有价值的东西。

上一篇文章我们学习了 PromQL，相信你对它已经不再陌生了，而要很好的理由它，我们还需要学习 Prometheus 中的 Rules（规则）。

Prometheus 规则是一种逻辑表达式，可用于定义有关监控数据的逻辑关系和约束条件。这些规则可以用于告警条件、聚合和转换等。

普罗米修斯支持两种类型的规则，可以对其进行配置，然后定期进行评估：

**评估**就是获取规则中的值进行逻辑运算，获取一个布尔值结果，表示被监控的指标是否达到了已经设定的一个阈值，或者是否达到了某个条件。

- **Alerting** 告警规则：在满足某些条件时触发警报，例如 CPU 使用率超过 90%。
- **Recording** 记录规则：使用 PromQL 表达式进行聚合和转换，将结果记录下来。例如计算平均响应时间。可以作为性能指标的跟踪，以便找到规律优化服务。

通过使用这些规则，您可以轻松地监控和管理您的应用程序和基础设施，并及时发现并解决任何问题。

要在 Prometheus 中使用规则，请创建一个包含所需规则语句的文件，并让 Prometheus 通过 配置文件中的 `rule_files` 字段指定这个文件，这样 Prometheus 就可以加载该文件。规则文件使用 YAML 格式书写。

# 2. 配置如何加载规则文件

需要在 Prometheus 的配置文件中的 `rule_files` 字段下添加配置，`rule_files` 字段的值是一个包含多个规则文件路径的列表，规则文件路径支持通配符。

示例：

```
prometheus.yml
```

...

rule\_files:

- "prometheus.rules.yml" # 指定具体文件
- "rules/\*.yml" # 指定 rules 目录下的所有以 .yml 结尾的文件

示例:

在 Prometheus 部署的服务器上创建如下目录

```
mkdir /usr/local/prometheus/rules
```

接着, 在创建的目录中, 创建 **recording.yml** 文件, 后面我们会用此文件作为众多规则文件中的一个。

```
touch /usr/local/prometheus/rules/recording.yml
```

最后在 Prometheus 的配置文件中添加如下配置:

rule\_files:

- "/usr/local/prometheus/rules/\*.yml"

重新加载配置文件

```
curl -X POST localhost:9090/-/reload
```

检查配置是否生效

The screenshot shows the Prometheus web interface. The top navigation bar includes the Prometheus logo, 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. The 'Status' menu is open, showing options: 'Runtime & Build Information', 'TSDB Status', 'Command-Line Flags', 'Configuration' (highlighted with a red box), 'Rules', 'Targets', and 'Service Discovery'. The main content area is titled 'Configuration' and shows the configuration file path: `rule_files:`  
`- /usr/local/prometheus/rules/*.yml` (highlighted with a red box). The footer of the page reads 'CSDN @shark\_西瓜甜'.

## 3. 配置规则文件

记录和警报规则存在于规则组中。组中的规则会以固定的时间间隔按顺序运行，评估时间相同。

### 3.1. 规则名称规范

记录规则的名称必须是有效的度量值名称。警报规则的名称必须是有效的标签值。

记录规则名称需要符合正则表达式: `[a-zA-Z_:][a-zA-Z0-9_]*`

警报规则名称需要符合正则表达式: `[a-zA-Z_][a-zA-Z0-9_]*` 以双下划线 ( `__` ) 开头的标签名称保留供内部使用

### 3.2. 全局配置框架

```
groups:
- name: example
  [ interval: <duration> | default = global.evaluation_interval ]

  [ limit: <int> | default = 0 ]

rules:
  [- <rule> ...]
```

- `name` 一个规则组的名称，在当前文件中需唯一。
- `interval` 每次查询规则的间隔时间，比如设置为 5s, 1m 等。默认是 prometheus 配置文件中全局配置 `evaluation_interval` 的值。
- `limit` 限制警报规则和记录规则可以产生的系列警报的数量。0 没有限制。一般按默认。
- `rules` 值是一个列表，列表中每个元素是这个规则组中的一条规则。

### 3.3. Recording rules (记录规则)

#### 3.3.1 格式讲解

记录规则允许您预先计算经常需要的或计算成本高昂的表达式，并将其结果保存为一组新的时间序列。

```
groups:
- name: example
  interval: 1m
  rules:
  - record: code:prometheus_http_requests_total:sum
    expr: sum by(code) (prometheus_http_requests_total)
    labels:
      [ <labelname>: <labelvalue> ]
```

- `record` 输出的时间序列的名称。必须是有效的度量值名称。
- `expr` 要计算的PromQL表达式。每个评估周期都在当前时间进行评估，结果记录为一组新的时间序列，度量名称由 `record` 给出。
- `labels` 要为每个警报添加或覆盖的标签。标签值可以使用 go 的模板变量。

#### 3.3.2 示例

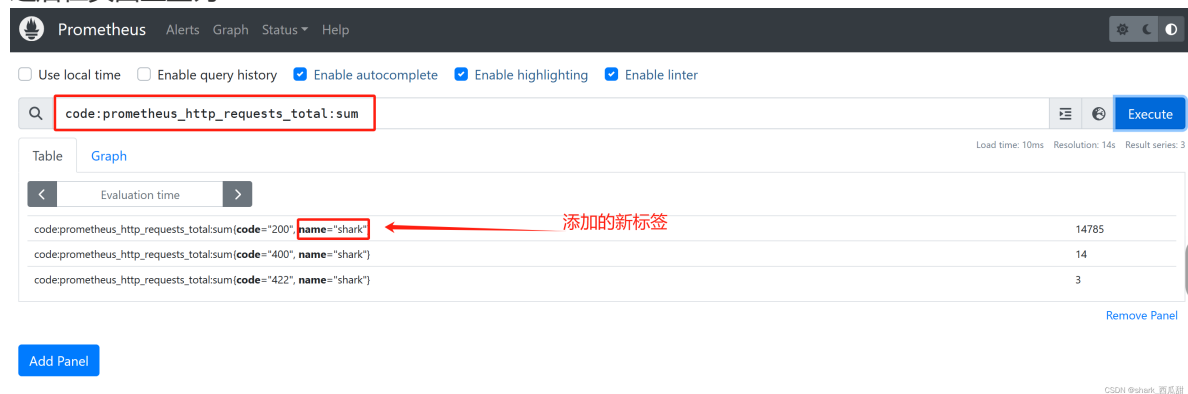
将如下配置添加到 `/usr/local/prometheus/rules/recording.yml` 中。

```
groups:
- name: sum-code-http-prom
  interval: 1s # 为了尽快看到实验效果这里设置 1 秒，实际生产需根据具体情况设置
  rules:
- record: code:prometheus_http_requests_total:sum
  expr: sum by(code) (prometheus_http_requests_total)
  labels:
    name: shark
```

重新加载配置文件

```
curl -X POST localhost:9090/-/reload
```

之后在页面上查询



The screenshot shows the Prometheus web interface. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below it, there are checkboxes for 'Use local time', 'Enable query history', 'Enable autocomplete', 'Enable highlighting', and 'Enable linter'. A search bar contains the query 'code:prometheus\_http\_requests\_total:sum'. Below the search bar, there are tabs for 'Table' and 'Graph'. The 'Table' tab is selected, showing a table with 3 rows. The first row has a value of 14785, the second 14, and the third 3. A red arrow points to the 'name' label in the first row, with the text '添加的新标签' (Added new label) next to it. At the bottom, there are buttons for 'Add Panel' and 'Remove Panel'.

code:prometheus_http_requests_total:sum	name="shark"	
code:prometheus_http_requests_total:sum(code="200", name="shark")		14785
code:prometheus_http_requests_total:sum(code="400", name="shark")		14
code:prometheus_http_requests_total:sum(code="422", name="shark")		3

## 3.4. Alerting rules (警报规则)

### 3.4.1 格式讲解

警报规则文件的格式几乎和记录规则文件格式一样，只是 **rule** (规则)配置中部分字段不同。

```
groups:
- name: example
  rules:
- alert: <string>
  expr: <string>
  [ for: <duration> | default = 0s ]
  [ keep_firing_for: <duration> | default = 0s ]
  labels:
    [ <labelname>: <tmpl_string> ]
  annotations:
    [ <labelname>: <tmpl_string> ]
```

- **alert** 警报的名称。必须符合有效标签名称的规则。
- **expr** 要计算的PromQL表达式。
- **fro** 当 **expr** 内书写的表达式条件满足后，警报会立刻变成 **pending** 状态。  
持续超过 **for** 指定的时间后，警报会被视为触发，并转为 **firing** 状态, 处于 **fring** 状态时才会把告警信息发送 **Alertmanager**。  
**pending** 和 **firing** 也会在 Prometheus Web 页面中看到。
- **keep\_firing\_for** 触发警报的条件清除后，警报仍然保持触发状态多长时间。

- `annotations` `annotations` 子句指定一组信息标签，这些标签可用于存储更长的附加信息，如警报描述或runbook链接。注解值可以模板化。

## 3.4.2 示例

编辑告警规则文件 `/usr/local/prometheus/rules/alerting.yml` 并添加如下内容

```
groups:
- name: example
  rules:
  - alert: ServerDown
    expr: up == 0
    for: 10s
    labels:
      severity: page
    annotations:
      summary: Server is down
```

重新加载配置文件

```
curl -X POST localhost:9090/-/reload
```

检查是否生效

The screenshot shows the Prometheus web interface. The 'Rules' tab is selected, and the 'example' rule is highlighted. The rule details are as follows:

Rule	State	Error	Last Evaluation	Evaluation Time
<b>example</b> alert: ServerDown expr: up == 0 for: 10s labels: severity: page annotations: summary: Server is down	OK		4.772s ago	0.372ms

尝试关闭你已经启动的某一 `node_exporter`，关闭后不断的刷新 Prometheus 的管理页面中的 **Alerts**

```
systemctl stop node-exporter
```

在 Prometheus 的管理页面查看规则状态的变化。

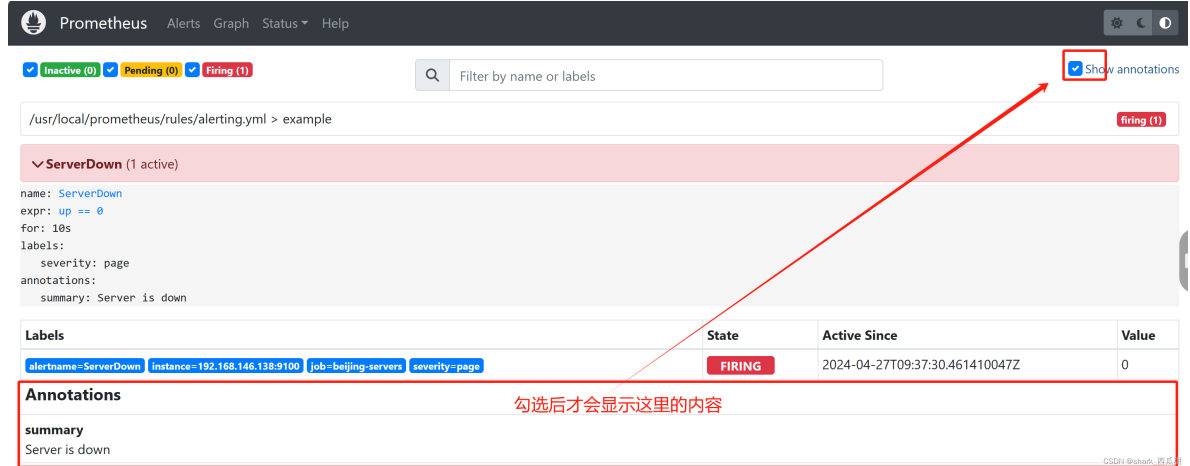
触发告警之前

The screenshot shows the Prometheus web interface. The 'Alerts' tab is selected. The page shows a list of alerts, with the 'ServerDown' alert highlighted. The alert is currently 'Inactive'.

触发告警之后,会先进入 **pending** 状态。



在经过 **for** 指定的时间之后,此警报才会视为 触发, 转为 **firing** 状态, 处于 **fring** 状态时才会把告警信息发送 **Alertmanager**。



## 3.5. 模板化如何使用

### 3.5.1 变量

`labels` 和 `annotations` 字段的配置可以使用 go 语言中的模板进行模板化。

- `$labels` 变量包含警报实例的标签键/值对。
- 可以通过 `$externalLabels` 变量访问配置的外部标签。
- `$value` 变量保存警报实例 `expr` 的计算结果值, 注意不是返回条件表达式的 布尔值。  
例如: `up == 0` 中, `$value` 的值是 `up` 的计算结果: `0`, 官方称为警报实例的评估值。

### 3.5.2 配置示例

编辑 `/usr/local/prometheus/rules/diskislow.yml` 文件, 添加如下内容:

```
groups:
- name: example
  rules:

# 对任何超过5分钟无法访问的实例发出警报。
- alert: InstanceDown
  expr: up == 0
  for: 5m
  labels:
    severity: page
  annotations:
    summary: "实例: {{ $labels.instance }} 已停机"
    description: "xx 项目作业 Job: { {{ $labels.job }} } 的 { {{ $labels.instance }} } 已停机超过5分钟。当前值: {{ $value }}。"

# 针对请求延迟中值>1s的任何实例发出警报
- alert: RootSysfileAvailabilityLow
```

```
expr: node_filesystem_avail_bytes{mountpoint="/" } /
node_filesystem_size_bytes{mountpoint="/" } * 100 > 85
for: 1m
annotations:
  summary: "实例 {{ $labels.instance }} 根分区可用率低"
  description: "{{ $labels.instance }} 的根分区可用率低于 85 %(当前值: {{ $value
}}s)"
```

## 查看警报内容

Prometheus

Alerts

Graph

Status ▾

Help

Inactive (2)

Pending (0)

Firing (1)

Filter by name or labels

Show annotations

/usr/local/prometheus/rules/alerting.yml > example

inactive

> ServerDown (0 active)

/usr/local/prometheus/rules/diskislow.yml > example

inactive firing (1)

> InstanceDown (0 active)

▼ RootSysfileAvailabilityLow (1 active)

name: RootSysfileAvailabilityLow

expr: node\_filesystem\_avail\_bytes{mountpoint="/" } / node\_filesystem\_size\_bytes{mountpoint="/" } \* 100 < 85

for: 1m

annotations:

description: {{ \$labels.instance }} 的根分区可用率低于 85 %(当前值: {{ \$value }}s)

summary: 实例 {{ \$labels.instance }} 根分区可用率低

Labels

State

Active Since

Value

alertname=RootSysfileAvailabilityLow device=/dev/mapper/centos-root fstype=xfs instance=192.168.146.138:9100 job=beijing-servers mountpoint=/

FIRING

2024-04-27T12:04:49.429085256Z

81.10660913964585

Annotations

description

192.168.146.138:9100 的根分区可用率低于 85 %(当前值: 81.10660913964585s)

summary

实例 192.168.146.138:9100 根分区可用率低

## 4. 检查规则文件语法

在不启动 Prometheus 的情况下也可以检查规则文件中的语法是否正确。

Prometheus 安装包中的 promtool 工具可以支持规则文件语法的检查。

```
./promtool check rules /usr/local/prometheus/rules/alerting.yml
```

```
[root@prome prometheus]# ./promtool check rules /usr/local/prometheus/rules/alerting.yml
Checking /usr/local/prometheus/rules/alerting.yml
SUCCESS: 1 rules found

[root@prome prometheus]# ./promtool check rules /usr/local/prometheus/rules/*
Checking /usr/local/prometheus/rules/alerting.yml
SUCCESS: 1 rules found

Checking /usr/local/prometheus/rules/diskislow.yml
SUCCESS: 2 rules found

Checking /usr/local/prometheus/rules/recording.yml
SUCCESS: 1 rules found

[root@prome prometheus]#
```

## 5. 发送警报通知

普罗米修斯的警报规则善于发现目前出现的问题，但它们并不是一个成熟的通知解决方案。在简单的警报定义之上，还需要另一层来添加摘要、通知速率限制、静音和警报依赖关系。在普罗米修斯的生态系统中，Alertmanager 扮演着这个角色。下个章节详细介绍 Alertmanager。

