**Grafana ，Alertmanager与钉钉对接文档**

|  |  |  |
| --- | --- | --- |
| 作者 | 版本 | 最后修订时间 |
| 张国栋 | 1.0 | 2020/11/11 |

# Grafana

1. 安装Grafana环境

1.1 安装Grafana

系统: Ubuntu Server 18.04 使用deb包安装

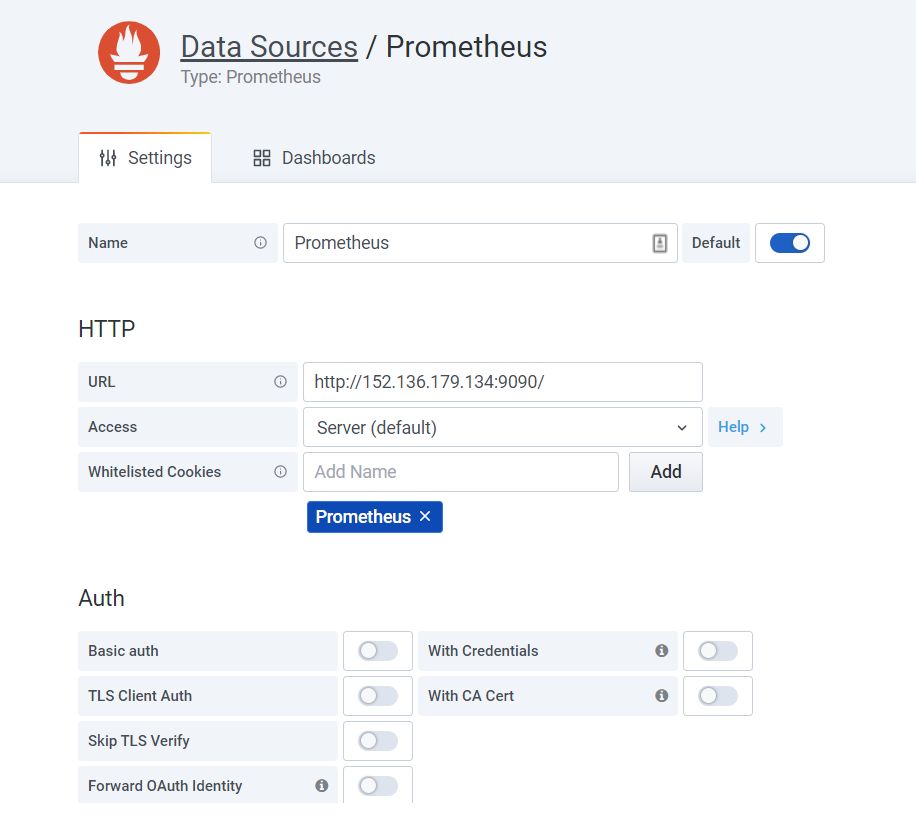
1. wget https://dl.grafana.com/oss/release/grafana\_7.3.1\_amd64.deb
2. sudo dpkg -i grafana\_7.3.1\_amd64.deb

启动Grfana-server

1. systemctl start grafana-server
2. systemctl status grafana-server

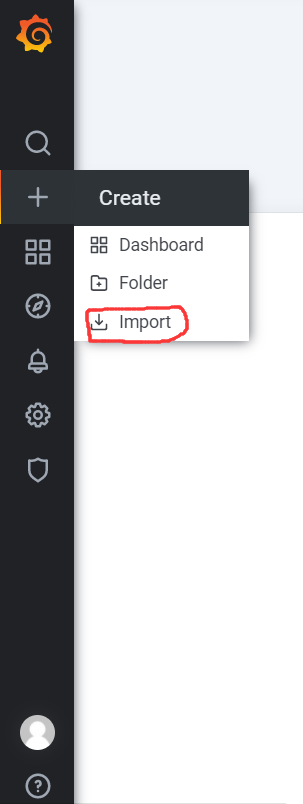
granafa默认端口为3000，可以在浏览器中输入<http://ip:3000/>

 granafa首次登录账户名和密码admin/admin，可以修改

 配置数据源Data sources->Add data source -> Prometheus，输入prometheus数据源的信息，主要是输入name和url

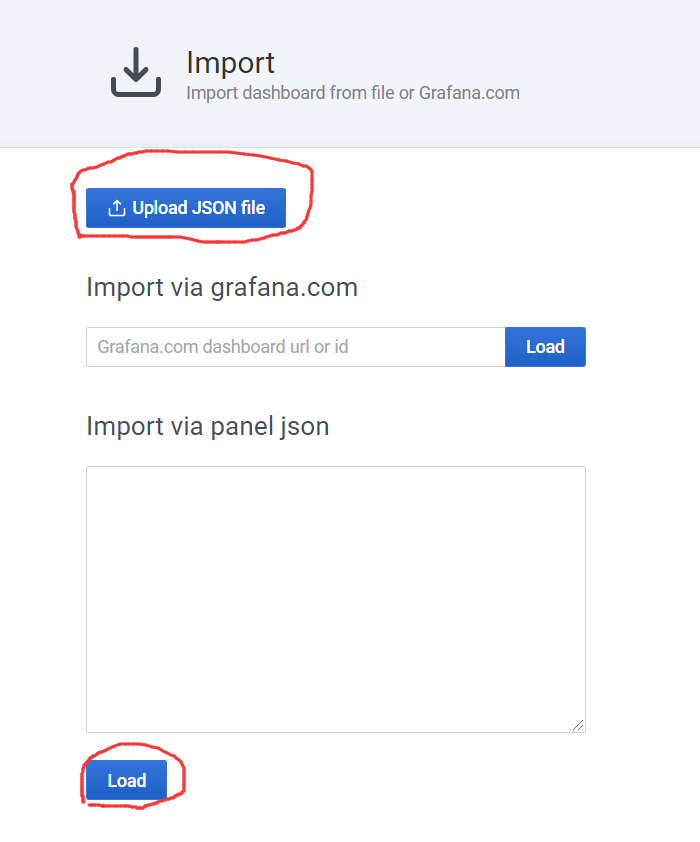
1.2 导入Grafana Dashboard

用admin账号登录，在左侧菜单栏点击”+”按钮，选择Import

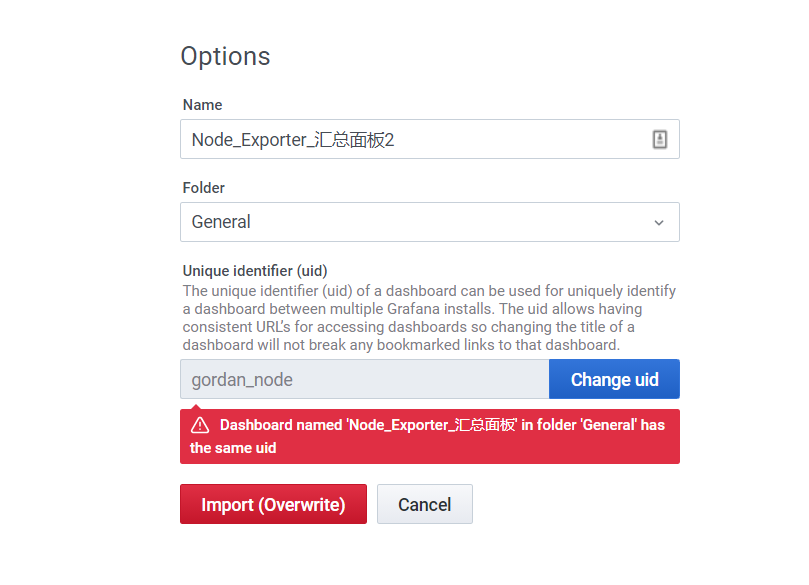


选择Upload JSON file，选择dashboard导出的json文件

或者点击Load，将json文件中的内容粘贴到下面的文本框中：



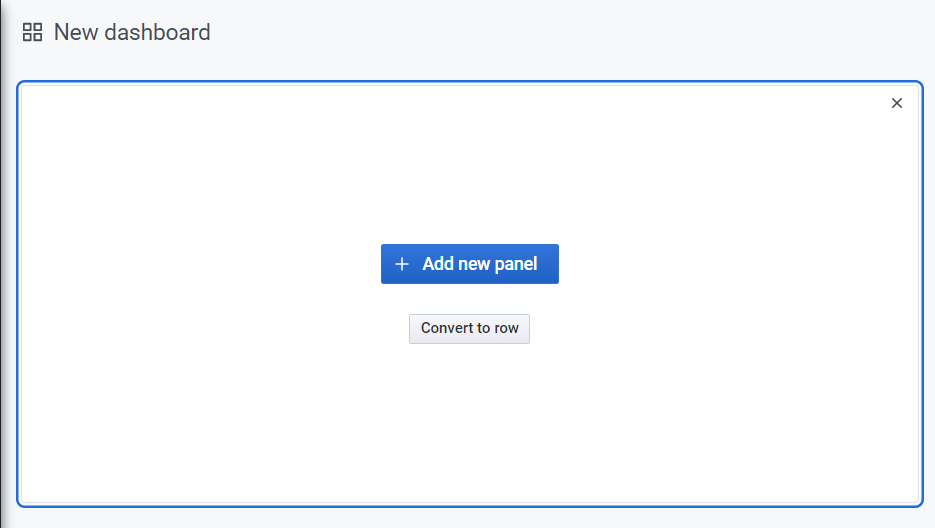
点击Import，即可看到面板：



1. Grafana Dashboard

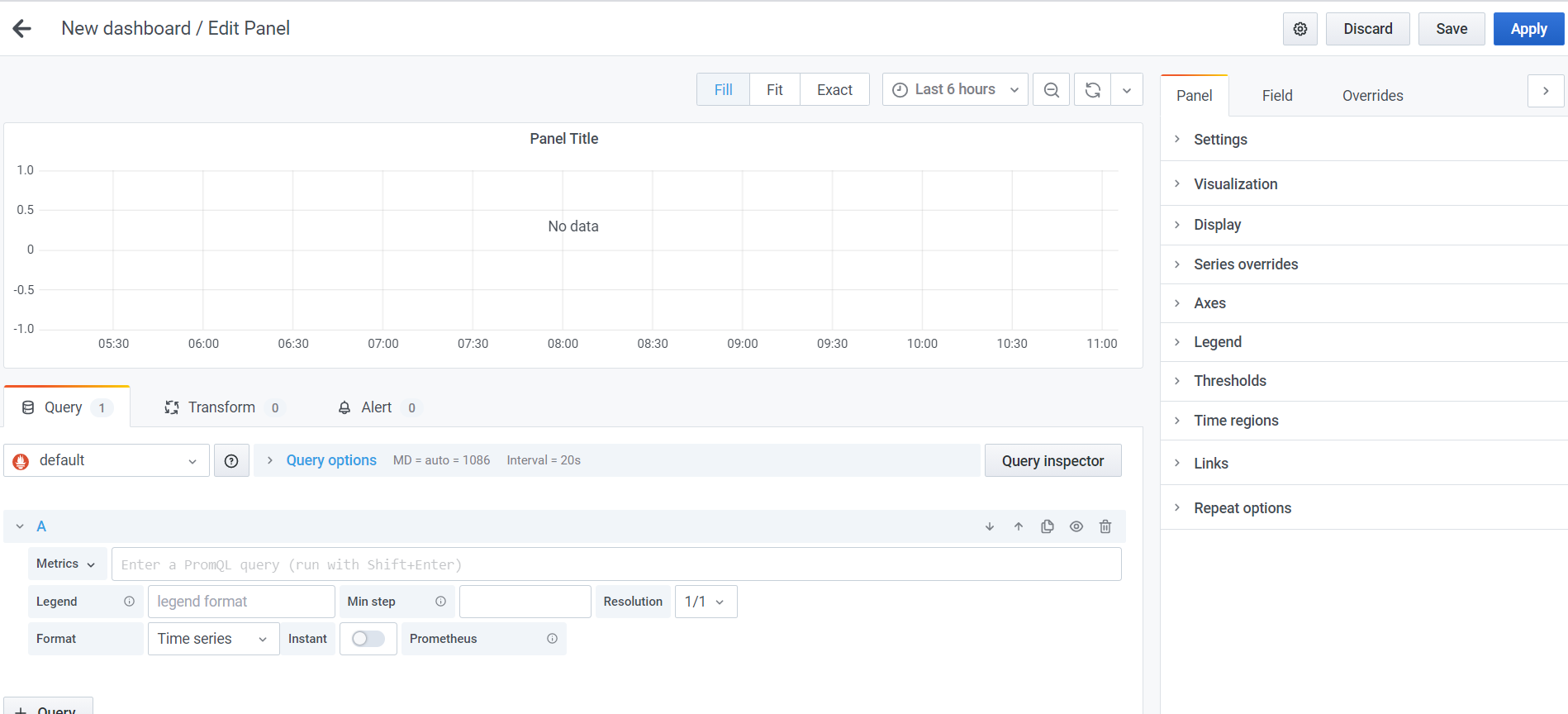
2.1 Grafana自定义Dashboard

点击左侧菜单”+”按钮，Create->Dashboard，即可新建Dashboard，见下图：



2.2 绘制Panel

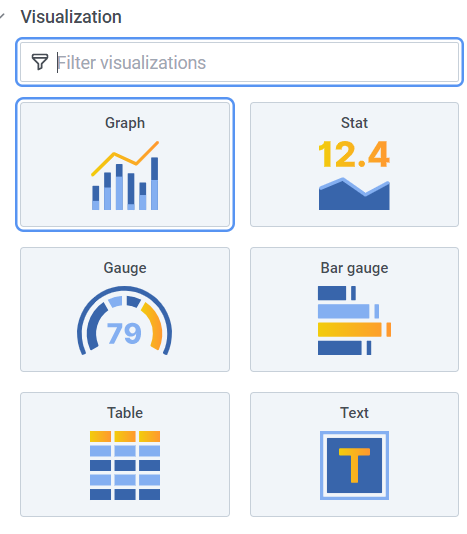
点击Add new panel，进入面板编辑页面：



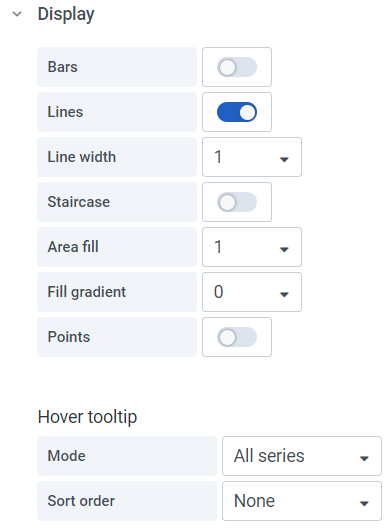
右侧的菜单为面板的选项:

Settings为Panel的标题，描述设置；

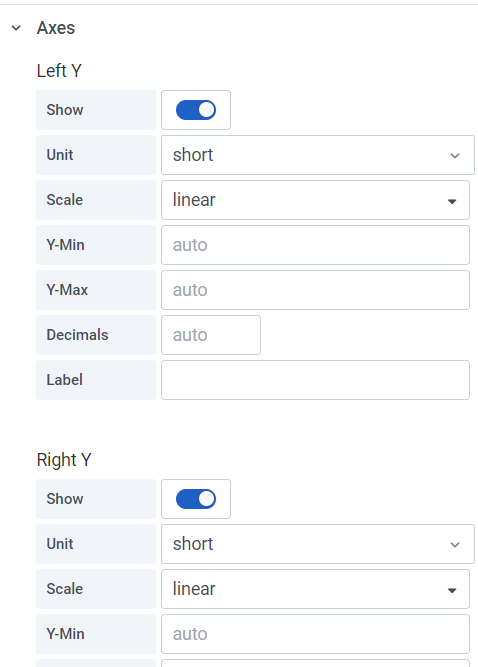
Viusalization可以选择图表的类型，比如线形图，条形图，柱形图，表格，仪表盘等；



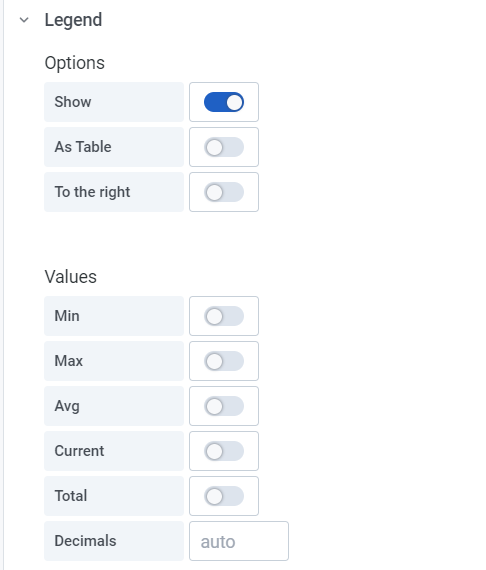
Display可以设置数据的格式，单位，鼠标悬浮时显示的东西等；



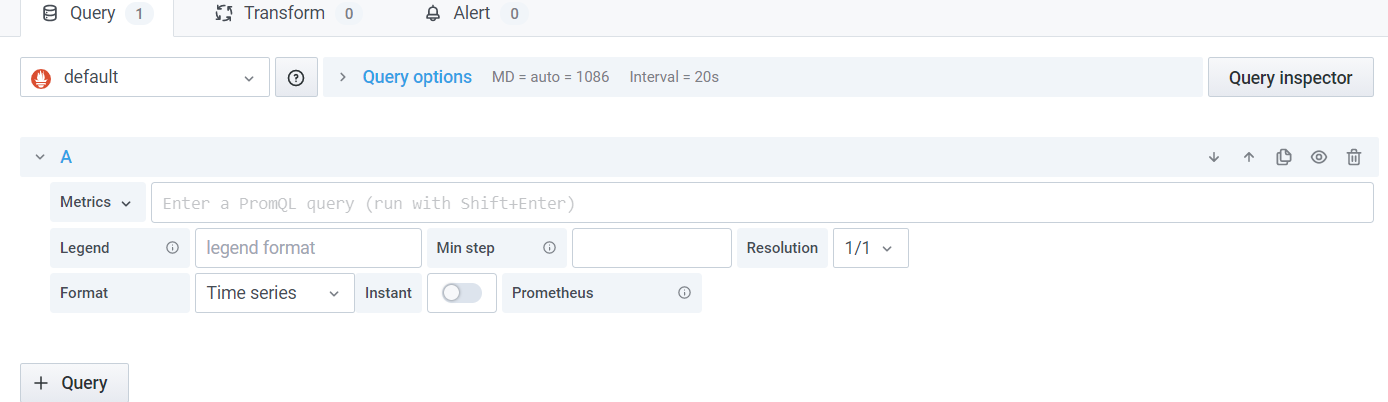
Axes可以设置x，y坐标轴的名称，单位，精度等；



Legend可以设置图例，以及图中数据的最大值/最小值，平均值，当前值等；



下方的菜单用来写查询表达式：



Metrics中写Prometheus的promQL表达式，Legend设置图例名称，可以使用模板的形式，如{{instance}}表示图例为label instance的值，模板只可以使用Metrics中有的label

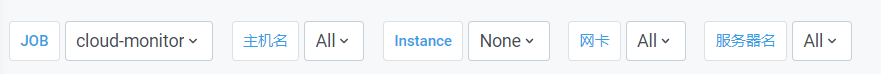
Format设置可以用Time Series，或者Table的形式展示；

Instant开关表示是否采用实时数据

1. Grafana模板变量

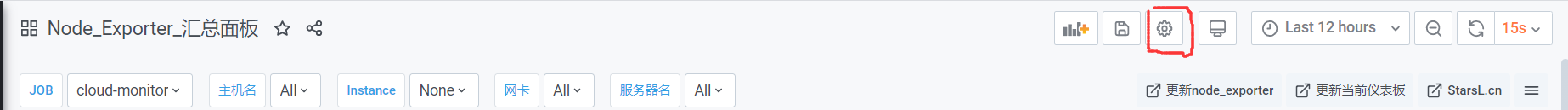
模板变量可以嵌入到Grafana的promQL中，代替掉Label的Value，用以聚合数据，替换掉硬编码：

Grafana里内置了模板变量，我们在Dashboard的顶部看到的下拉框就是模板变量，见下图：

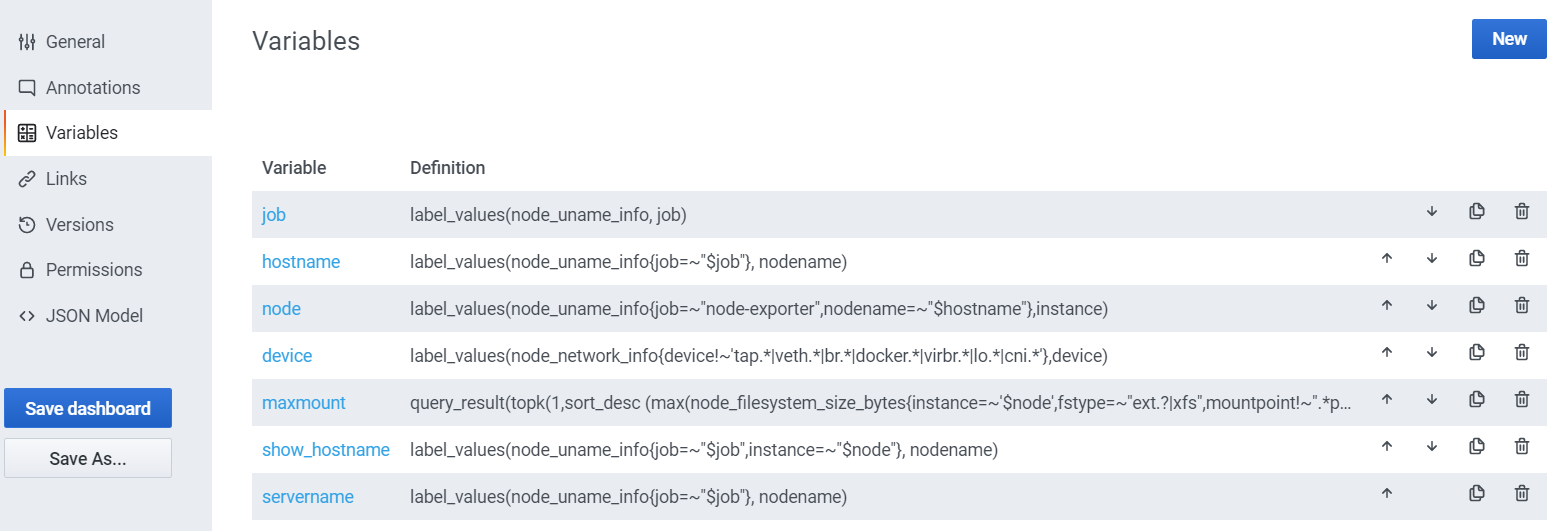


模板变量可以在Dashboard中设置：

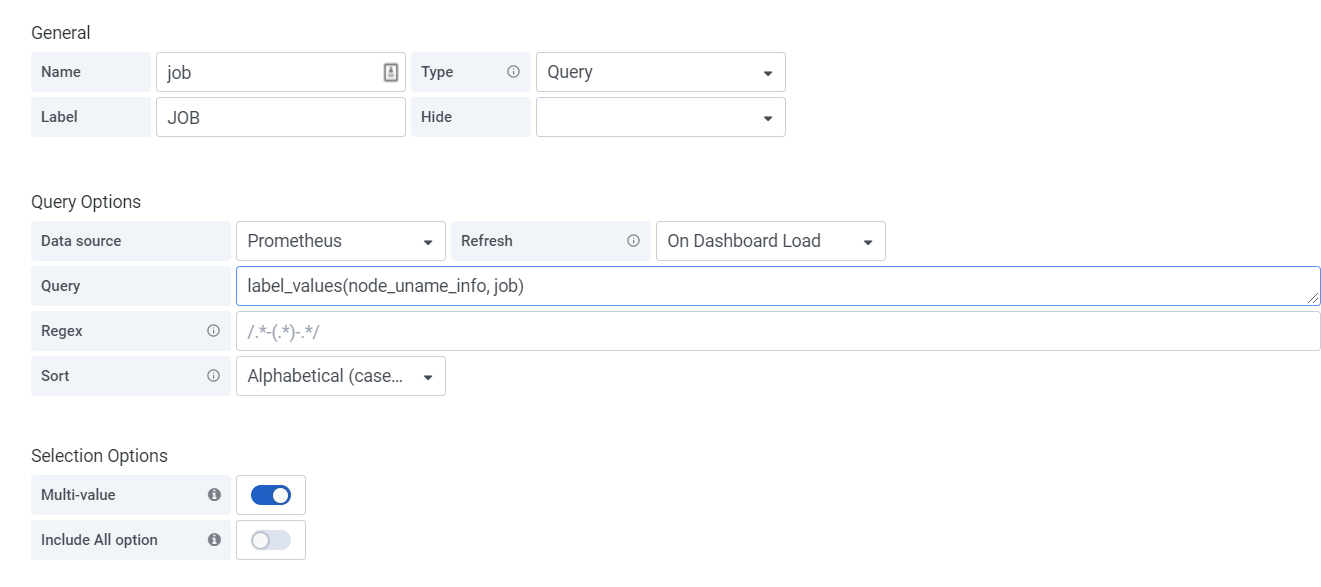
3.1点击Dashboard右上角的齿轮，进入设置：



* 1. 点击Variables，可以看到当前Dashboard所有模板变量：



* 1. 点击New，自定义模板变量：



Name就是变量名，Label是变量显示在Dashboard上的名称，类型选择Query

在下面的Query Options中，Data source选择Prometheus，Refresh选择On Dashboard Load，在Query中填写查询变量，用以获得指标，标签和标签值的列表。Grafana内置的查询变量见下表：

|  |  |
| --- | --- |
| 名称 | 说明 |
| label\_\\_\_names() | 返回标签名称的列表 |
| label\_\\_\_values(label) | 返回每个度量标准中标签的标签值的列表 |
| label\_\\_\_values(metric, label) | 返回指定度量标准中标签的标签值列表 |
| metrics(metric) | 返回与指定指标正则表达式匹配的指标列表 |
| query\_\\_\_result(query) | 返回查询的Prometheus查询结果列表 |

例子：查询出当前所有的job标签的值

Query: label\_values(node\_uname\_info, job)

Regex: /"([^"]+)"/

结果: “cloud-monitor”

结果会显示在下方Preview of values中.

* 1. 在查询中使用模板变量

有2种语法：

* $<varname> Example: rate(http\_requests\_total{job=~"$job"}[5m])
* [[varname]] Example: rate(http\_requests\_total{job=~"[[job]]"}[5m])

为什么有两种方式？ 第一种语法更易于读写，但不允许您在单词中间使用变量。 启用“多值”或“包含所有值”选项后，Grafana会将标签从纯文本转换为与regex兼容的字符串。 这意味着您必须使用=〜而不是=

请注意，使用了模板变量的promQL是不能直接放进prometheus中查询的。

# Alertmanager

Alertmanager处理Prometheus服务器发送的警报，包括静音，禁止以及通过电子邮件，钉钉，企业微信等方式发送通知。

1. AlertManager安装

下载程序并解压

地址: <https://github.com/prometheus/alertmanager>

1. 配置Prometheus与Alertmanager集成

Alertmanager通过命令行参数和配置文件进行配置。要查看所有可用的命令行标志，请运行alertmanager -h.

Alertmanager可用在运行时重新加载其配置。如果新配置格式不正确，则更改将不会应用。通过SIGHUP发送到进程或将HTTP POST请求发送到/-/ reload短短来触发配置重新加载。

要指定要加载的配置文件，请使用--config.file标志

1. ./alertmanager --config.file=alertmanager.yml

该文件以YAML格式写入，由以下所述的方案定义。方括号表示参数是可选的。 对于非列表参数，该值设置为指定的默认值。

通用占位符定义如下：

1. <duration>：与正则表达式[0-9] +（ms | [smhdwy]）匹配的持续时间
2. <labelname>：与正则表达式[a-zA-Z \_] [a-zA-Z0-9 \_] \*匹配的字符串
3. <labelvalue>：一串unicode字符
4. <filepath>：当前工作目录中的有效路径
5. <boolean>：可以采用true或false值的布尔值
6. <string>：常规字符串
7. <secret>：作为机密的常规字符串，例如密码
8. <tmpl\_string>：使用前已模板扩展的字符串
9. <tmpl\_secret>：一个字符串，在使用前经过模板扩展，这是一个秘密
10. <int>：一个整数值

由于alertmanager.yml的配置项众多，这里只介绍已经部署的alertmanager的配置项，其他配置项请参考[prometheus官网](https://prometheus.io/docs/alerting/latest/configuration/)

全局配置指定在所有其他配置上下文中有效的参数。 它们还用作其他配置部分的默认设置。

路由块定义了路由树中的节点及其子节点。 如果未设置，则其可选配置参数将从其父节点继承。

每个警报都会在已配置的顶级路由处进入路由树，该路由树必须与所有警报匹配（即没有任何已配置的匹配器）。 然后遍历子节点。 如果将continue设置为false，它将在第一个匹配的子项之后停止。 如果在匹配的节点上true为true，则警报将继续与后续的同级进行匹配。 如果警报与节点的任何子节点都不匹配（不存在匹配的子节点或不存在子节点），则根据当前节点的配置参数来处理警报。

当存在与另一组匹配器匹配的警报（源）时，禁止规则会使与一组匹配器匹配的警报（目标）静音。

接收器是一个或多个通知集成的命名配置。一般通过webhook来实现自定义接收器（见下文:Alertmanager Webhook DingTalk）

在云上部署的alertmanager配置文件如下：

1. global:
2. resolve\_timeout: 5m
3. route:
4. receiver: webhook
5. group\_wait: 30s
6. group\_interval: 5m
7. repeat\_interval: 12h
8. group\_by: [alertname]
9. routes:
10. - receiver: webhook
11. group\_wait: 10s
12. match:
13. team: node
14. receivers:
15. - name: webhook
16. webhook\_configs:
17. - url: http://localhost:8060/dingtalk/ops\_dingding/send
18. send\_resolved: true

<global>：

resolve\_timeout: ResolveTimeout是Alertmanager使用的默认值，经过此时间后，如果尚未更新，则可以将警报声明为已解决。这对Prometheus的警报没有影响；

<route>:

receiver: 将传入警报分组的标签。这里的webhook与下面receivers标签中的webhook相对应；

group\_wait: 最初等待为一组警报发送通知的时间。 允许等待禁止警报到达或为同一组收集更多初始警报。 （通常〜0秒到几分钟。）默认为30s；

group\_interval : 发送有关新警报的通知之前要等待的时间，该通知已添加到已为其发送了初始通知的一组警报中。 （通常〜5m或更多。）默认为5m；

repeat\_interval : 如果已成功发送警报以再次发送通知，则要等待多长时间。 （通常约3小时或更长时间）。默认为4h；

group\_by : ＃要按所有可能的标签进行聚合，请使用特殊值'...'作为唯一的标签名称，例如这里是根据警报名称alertname进行分组；

routes : 配置子路由

team: 警报必须满足的一组相等匹配器才能匹配节点。这里team:node表示所有带有team = frontend标签的警报均与此子路由匹配；

receivers : 定义接收器。可配置多个不同的接收器；

webhook\_config : 配置webhook；

url: 向其发送HTTP POST请求的端点；

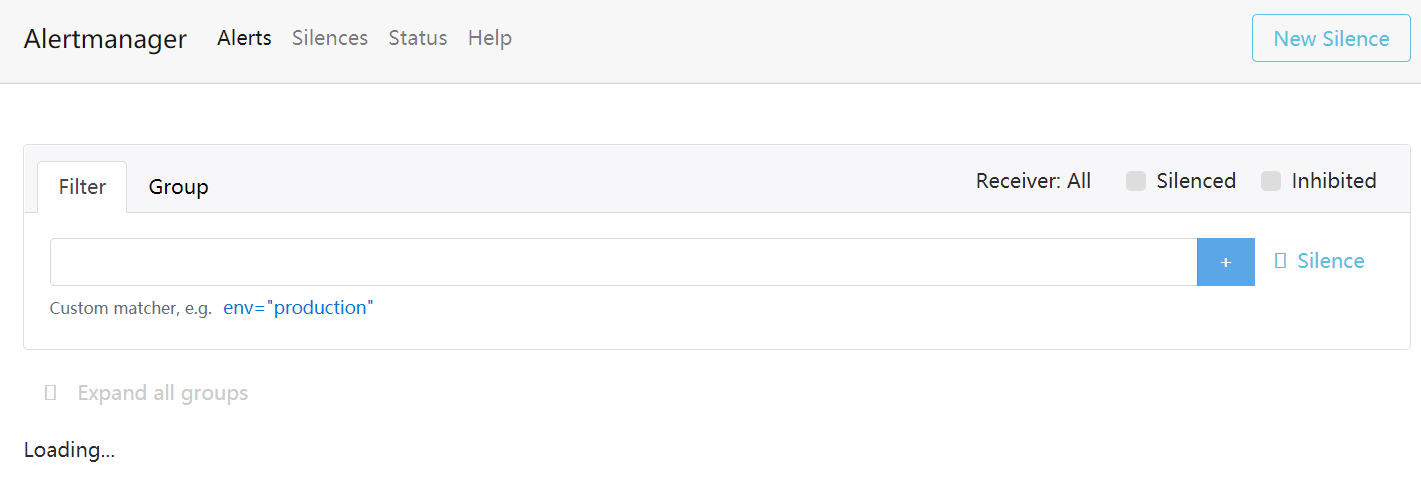
send\_resolved: 警报解除时是否通知。默认为true。

Prometheus配置

在prometheus.yml中添加以下内容:

1. # Alertmanager configuration
2. alerting:
3. alertmanagers:
4. - static\_configs:
5. - targets:
6. - localhost:9093

Alertmanager默认端口是9093，启动后会出现一个webui，如果想访问记得把9093端口开放。



1. 在Prometheus中创建预警规则

3.1 定义报警规则

警报规则允许您基于Prometheus表达式语言表达式定义警报条件，并将有关触发警报的通知发送到外部服务。 每当警报表达式在给定时间点生成一个或多个向量元素时，警报将计为这些元素的标签集的活动状态。

预警规则同样使用yml文件，先在prometheus.yml中添加以下内容：

1. rule\_files:
2. - "rules/\*rules.yml"

这样将解析rules目录下所有命名以rules结尾的规则文件

预警规则文件示例：

1. groups:
2. - name: example
3. rules:
4. - alert: HighErrorRate
5. expr: job:request\_latency\_seconds:mean5m{job="myjob"} **>** 0.5
6. for: 10m
7. labels:
8. severity: page
9. annotations:
10. summary: High request latency

name指定警报组的名称

rules指定全部的规则

alert指定每条警报的名称

expr 指定预警表达式，使用promQL

for 指定触发警报前的等待时间。设定为10m表示Prometheus将在每次评估期间检查警报是否继续处于活动状态10分钟，然后再触发警报。处于活动状态但尚未触发的元素处于暂挂状态。

label指定要附加到警报的一组附加标签。任何现有的冲突标签都将被覆盖。标签值可以是模板化的

severity指定警报的等级，可以自定义值;

annotations 指定一组信息标签，可用于存储更长的附加信息，值可以是模板化的。

* 1. 模板

可以使用控制台模板模板化标签和注释值。$labels变量保存警报实例的标签键/值对,$value保存警报实例的评估值。

1. # 要插入触发元素的标签值：
2. {{ $labels.**<labelname>** }}
3. # 要插入触发元素的数值表达式值：
4. {{ $value }}

例子：

1. groups:
2. - name: example
3. rules:
5. # 对于任何无法访问**>** 5分钟的实例的警报。
6. - alert: InstanceDown
7. expr: up == 0
8. for: 5m
9. labels:
10. severity: page
11. annotations:
12. summary: "Instance {{ $labels.instance }} down"
13. description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 5 minutes."
15. # 对中值请求延迟**>** 1s的任何实例发出警报。
16. - alert: APIHighRequestLatency
17. expr: api\_http\_request\_latencies\_second{quantile="0.5"} **>** 1
18. for: 10m
19. annotations:
20. summary: "High request latency on {{ $labels.instance }}"
21. description: "{{ $labels.instance }} has a median request latency above 1s (current value: {{ $value }}s)"
    1. 在运行时检查警报

要手动检查哪些警报处于活动状态(待处理或触发)，请导航至Prometheus实例的”警报”选项卡。这将显示每个定义的警报当前处于活动状态的确切标签集。

对于待处理和触发警报，Promethues还存储ALERTS{alertname="<alert name>",alertstate ="pending|firing",<additional alert labels>}形式的合成时间序列。只要警报处于指示的活动(挂起或触发)状态，样本值就会设置为1，并且当不再是这种情况时，系列会标记为过时。

# Alertmananger Webhook DingTalk(与钉钉对接)

1. 安装webhook并配置

地址：<https://github.com/timonwong/prometheus-webhook-dingtalk>

alertmanager配置webhook：alertmanager.yml

1. receivers:
2. - name: webhook
3. webhook\_configs:
4. - url: http://localhost:8060/dingtalk/ops\_dingding/send
5. send\_resolved: true

webhook-dingtalk使用：

usage: prometheus-webhook-dingtalk [<flags>]

Flags:

-h, --help Show context-sensitive help (also try --help-long and --help-man).

--web.listen-address=:8060

The address to listen on for web interface.

--web.enable-ui Enable Web UI mounted on /ui path

--web.enable-lifecycle Enable reload via HTTP request.

--config.file=config.yml Path to the configuration file.

--log.level=info Only log messages with the given severity or above. One of: [debug, info, warn, error]

--log.format=logfmt Output format of log messages. One of: [logfmt, json]

--version Show application version.

1. 启用钉钉机器人

请参考官方文档：[自定义机器人](https://ding-doc.dingtalk.com/doc#/serverapi2/krgddi)。

添加机器人后获取机器人的hook（机器人好像只能在钉钉群里面添加）,在后面部署会用到。

机器人hook：<https://oapi.dingtalk.com/robot/send?access_token=xxxxxx>

获取到hook后，在webhook-dingtalk中配置。在目录下的config.yaml中，添加如下内容：

1. targets:
2. webhook1:
3. url: https://oapi.dingtalk.com/robot/send?access\_token=https://oapi.dingtalk.com/robot/send?access\_token=627c34ca398613ff446e643ff2c6581a4039dfdd8de94886d34da
4. 配置报警模板

在contrib/templates/legacy目录下，新建ding\_alert.tmpl文件：

1. {{ define "\_\_text\_alert\_list" }}{{ range .  }}
2. ==========云报警发生==========
3. 告警程序：hpcc\_alter
4. 告警级别：{{ .Labels.severity }}
5. 告警类型：{{ .Labels.alertname }}
6. 主机IP: {{ .Labels.addr }}
7. 主机名: {{ .Labels.addr }}
8. 告警主题: {{ .Annotations.summary }}
9. 告警描述: {{ .Annotations.description }}
10. 触发时间: {{ dateInZone "2006.01.02 15:04:05" (.StartsAt) "Asia/Shanghai" }}
11. \*\*图表:\*\* [查看图表]({{ .GeneratorURL }})
12. ------------------------
13. {{ end }}{{ end }}

16. {{ define "\_\_text\_resolve\_list" }}{{ range .  }}
17. ==========云报警恢复==========
18. 恢复程序：hpcc\_alter
19. 恢复类型：{{ .Labels.alertname }}
20. 主机IP: {{ .Labels.addr }}
21. 主机名: {{ .Labels.addr }}
22. 恢复描述: {{ .Annotations.description }}
23. 警报值: {{ .Annotations.value }} {{ .Labels.unit }}

告警模板基于Go模板系统，请参考：

<https://www.jianshu.com/p/01268c1ab972>

<https://golang.org/pkg/text/template/>

接下来在命令行参数中指定告警模板的位置:

1. --template.file=/etc/alertmanager/prometheus-webhook-dingtalk/contrib/templates/legacy/ding\_alert.tmpl

同时也在命令行参数中指定钉钉机器人hook：

--ding.profile="ops\_dingding=https://oapi.dingtalk.com/robot/send?access\_token=627c34ca398613ff446e643ff2c6581a4039dfdd8de94886d34da568f824b174"

以system service形式启动：

vim /etc/systemd/system/prometheus-dingtalk.service

1. [Unit]
2. Description=prometheus-webhook-dingtalk
3. After=network-online.target
5. [Service]
6. Restart=on-failure
7. ExecStart=/etc/alertmanager/prometheus-webhook-dingtalk/prometheus-webhook-dingtalk \
8. --template.file=/etc/alertmanager/prometheus-webhook-dingtalk/contrib/templates/legacy/ding\_alert.tmpl \
9. --web..listen-address=:8060 \
10. --web.enable-ui \
11. --web.enable-lifecycle \
12. --ding.profile="ops\_dingding=https://oapi.dingtalk.com/robot/send?access\_token=627c34ca398613ff446e643ff2c6581a4039dfdd8de94886d34da568f824b174"

15. [Install]
16. WantedBy=multi-user.target

systemctl start prometheus-dingtalk

systemctl enable prometheus-dingtalk

先启动webhook-dingtalk，再启动alertmanager

这样，就会达到如下的效果





1. 实现每日汇总，主机上线/失联功能

思路：跑定时任务python脚本

每日汇总通过prometheus的httpapi /api/v1/query 和/api/v1/query\_range，获取数据，并通过钉钉机器人发送；

主机上线/失联通过up表达式；

主机上线：在每天0点时执行一脚本getUpEarly.py 将up==1即上线的机器的查询结果写入到json中；然后每5分钟执行一次脚本detectNewHost.py，再次使用up==1查询上线的机器，将数据放入map中，取出json中的数据到map，比较2map的差集，如果新查询的map中不属于json中的数据，说明有主机上线，这时发送信息，并更新json。

主机失联：利用范围查询，查询今天(当前时间-24小时~当前时间)和前一天(当前时间-48小时~当前时间-24小时)的up==1的数据，比较2次数据的差集，如果前一天中有的数据，在今天的数据中没有，那说明有主机失联，这时发送信息。

具体请见python目录下的代码。

crontab任务:

1. # Edit this file to introduce tasks to be run by cron.
2. #
3. # Each task to run has to be defined through a single line
4. # indicating with different fields when the task will be run
5. # and what command to run for the task
6. #
7. # To define the time you can provide concrete values for
8. # minute (m), hour (h), day of month (dom), month (mon),
9. # and day of week (dow) or use '\*' in these fields (for 'any').#
10. # Notice that tasks will be started based on the cron's system
11. # daemon's notion of time and timezones.
12. #
13. # Output of the crontab jobs (including errors) is sent through
14. # email to the user the crontab file belongs to (unless redirected).
15. #
16. # For example, you can run a backup of all your user accounts
17. # at 5 a.m every week with:
18. # 0 5 \* \* 1 tar -zcf /var/backups/home.tgz /home/
19. #
20. # For more information see the manual pages of crontab(5) and cron(8)
21. #
22. # m h  dom mon dow   command
23. \*/2 \* \* \* \* python3 /home/zhgd01/py/tellTime.py **>>** /home/zhgd01/py/telltime.log 2**>**&1
24. 1 0 \* \* \* python3 /home/zhgd01/py/getUpEarly.py **>>** /home/zhgd01/py/getUpEarly.log 2**>**&1
25. 0 8,18 \* \* \* python3 /home/zhgd01/py/daliyNewLost.py **>>** /home/zhgd01/py/daliyNewLost.log 2**>**&1
26. 0 8,18 \* \* \* python3 /home/zhgd01/py/dailySummary.py **>>** /home/zhgd01/py/dailySummary.log 2**>**&1
27. 5 \* \* \* \* python3 /home/zhgd01/py/detectNewHost.py **>>** /home/zhgd01/py/detectNewHost.log 2**>**&1