

prometheus基础

prometheus的数据类型

exporter介绍

pushgateway

Prometheus安装与基本使用

prometheus安装步骤

prometheus主配置文件

服务发现

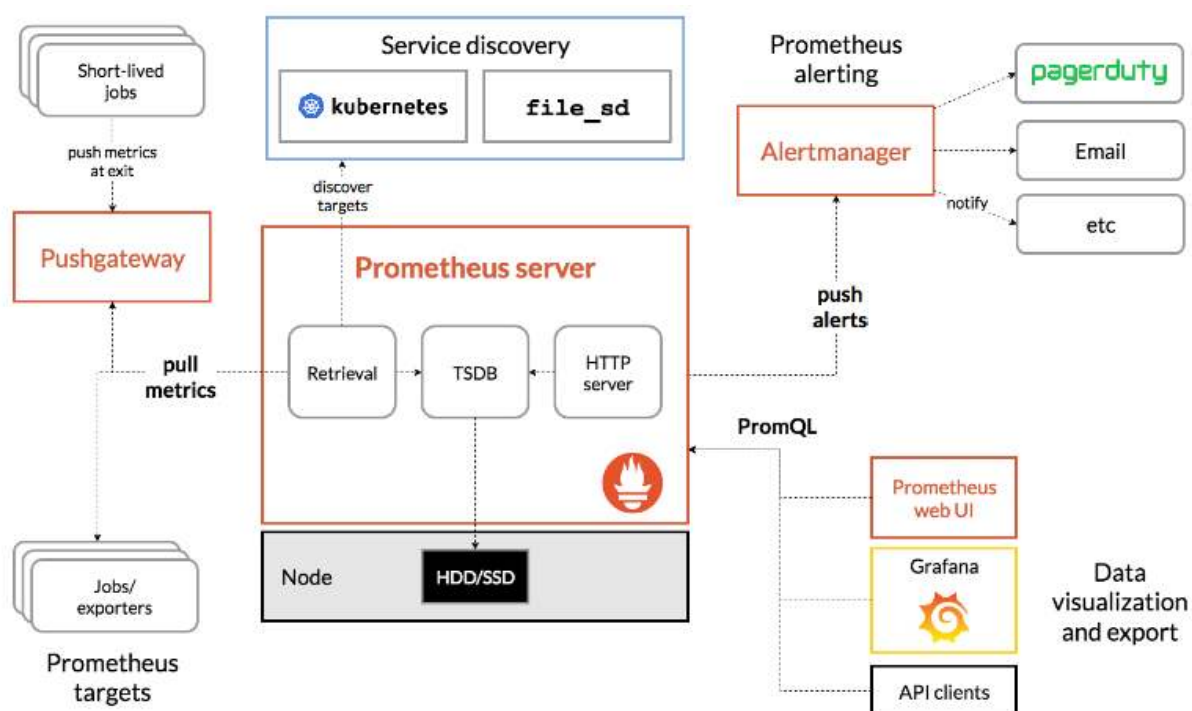
prometheus采集客户端

prometheus常用函数

Grafana

添加数据源

prometheus基础



Grafana: 绘图工具，prometheus自身已有绘制图形的能力，但grafana绘制的图形更精确，更美观。

Pagerduty: 收费的报警系统

Prometheus的优势:

1. 监控数据的精度高，理论上可精确到1~5秒；
2. 集群部署的速度快；

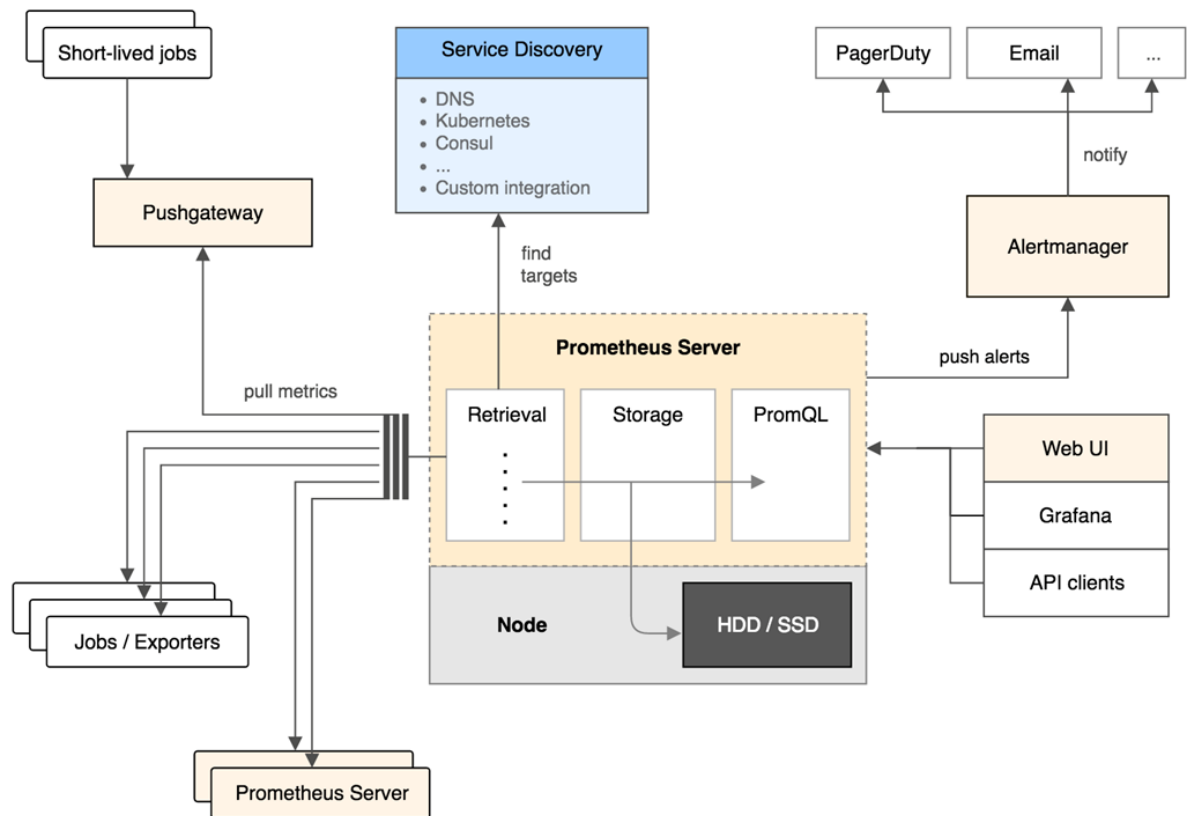
3. 周边插件很丰富，大多数都不需自己开发；
4. 本身基于数据计算模型，有大量实用的函数，可以实现很复杂规则的业务逻辑监控；
5. 可以嵌入很多开源工具的内部进行监控，数据更准时，更可信；
6. 本身开源，更新速度更快，bug修复快，支持多种语言实现其自身和插件的二次开发；
7. 与grafana结合后展示的图形很美观。

监控项目：

1. 业务监控：用户访问QPS、DAU日活、访问状态、业务接口（登陆、注册、聊天、上传、留言、短信、搜索）、产品转化率、用户投诉等；
2. 系统监控：主要是跟操作系统相关的基本监控项，CPU、内存、硬盘、I/O、TCP连接、流量等；
3. 网络监控：对网络状态的监控，丢包率、延迟等；
4. 日志监控：监控中的重头戏（Splunk, ELK），往往单独设计和搭建，全部种类的日志都有采集；
5. 程序监控：一般需要和开发人员配合，程序中嵌入各种接口直接获取数据或者特质的日志格式。

未来监控体系：

1. 完整自愈式监控体系：监控和报警在自愈系统下各种层级的问题都会被自动化、持续集成、人工智能、灾备、系统缓冲等技术自行修复（docker, kubernetes）。
2. 真实链路式监控：监控和报警的准确性、真实性发展到最终的一个理想化模型。



Prometheus监控的体质特性：

1. 基于time series时间序列模型，时间序列是一系列有序的数据，通常是等时间间隔的采样数据；
2. prometheus的本地T-S（Time Series）数据库以每两个小时为间隔来划分block（块）来存储，每个块中又分为多个chunk文件，chunk文件是用来存放采集过来的数据的T-S数据、metadata和索引（index）。
3. index文件是对metrics（prometheus中一次k/v采集数据叫做一个metric）和labels（标签）进行索引，然后存储在chunk中，chunk是作为存储的基本单位，index和metadata是作为其子集。
4. 基于K/V的数据模型，例如：`{disk_size: 80 }`，其数据格式简单，速度快，易于维护；
5. 采样的数据的查询完全基于数学运算而不是其他的表达式，并提供专有的查询输入console，所有的查询都基于数学运算公式；
6. 采用HTTP pull/push两种对应的数据采集传输方式，所有的数据采集都基本采用HTTP，而且分为pull/push两种方式去写/采集程序；
7. push方法灵活，push的这种采集方法可采集几乎任何形式的数据；
8. 本身自带图形调试，可方便运维进行调试，但最终还是要与其他图形化展示的插件进行结合（如grafana）；
9. 最精细的数据采样，prometheus理论上可以达到每秒采集，可自行定义频率。

prometheus的数据类型

prometheus本身是一个以进程的方式启动，之后以多进程和多线程的方式实现监控数据收集、计算、查询、更新和存储的一种C/S架构模型。

prometheus监控中对于采集过来的数据统一称为metrics数据，metric是一种对采样数据的总称（metrics并不代表某一种具体的数据格式，是一种对于试题计算单位的抽象）。

metrics的几种主要类型：

1. **Gauges**，最简单的度量指标，只有一个简单的返回值，或者叫瞬时状态，例如：要监控硬 容量或者内存的使用量，就应该使用**Gauges**或者**metrics**格式来度量（因为硬盘的容量或者内存的容量是随着时间的推移不断的瞬时变化的，这种变化没有规律，当前是多少采集回来的就是多少这是没有规律的）。
2. **counters**，计数器，从数据量0开始累积计算，在理想状态下**counters**只能永远不会下降，例如用户访问量只会增加或保持不变而不会减少（例如网卡发出的字节数、当日累计访问数）。
3. **Histograms**，统计数据分布情况，比如最小值，最大值，中间值，还有中位数，这一种特殊的**metrics**数据类型，代表一种近似的百分比估算数值，此种数据类型在实际使用中应用性较强（如：将**HTTP**的响应时间按比例予以显示，例如**0.5**秒以下的有多少，**0.5~1**秒的有多少等）。

Key-Value数据的形式：

prometheus的数据类型就是依赖于这种**metrics**的类型计算出来的，**metrics**是由**exporter**（例如**node_exporter**）在服务器上采集来的服务器上的**Key/Value**类型的**metrics**数据；当一个**exporter**被安装和运行在被监控的服务器上之后，使用简单的**curl**命令就可以看到**exporter**采集到的**metrics**数据的样子，以**key/value**的形式展现和保存

```
shell# curl localhost:9100/metrics
```

以#开头的即为对数据的注释。

exporter介绍

- **blackbox_exporter**
- **Haproxy_exporter**
- **Node_exporter**
- **Statsd_exporter**

大多数**exporters**下载后，就提供了启动的命令，一般直接运行，带上一些参数即可。

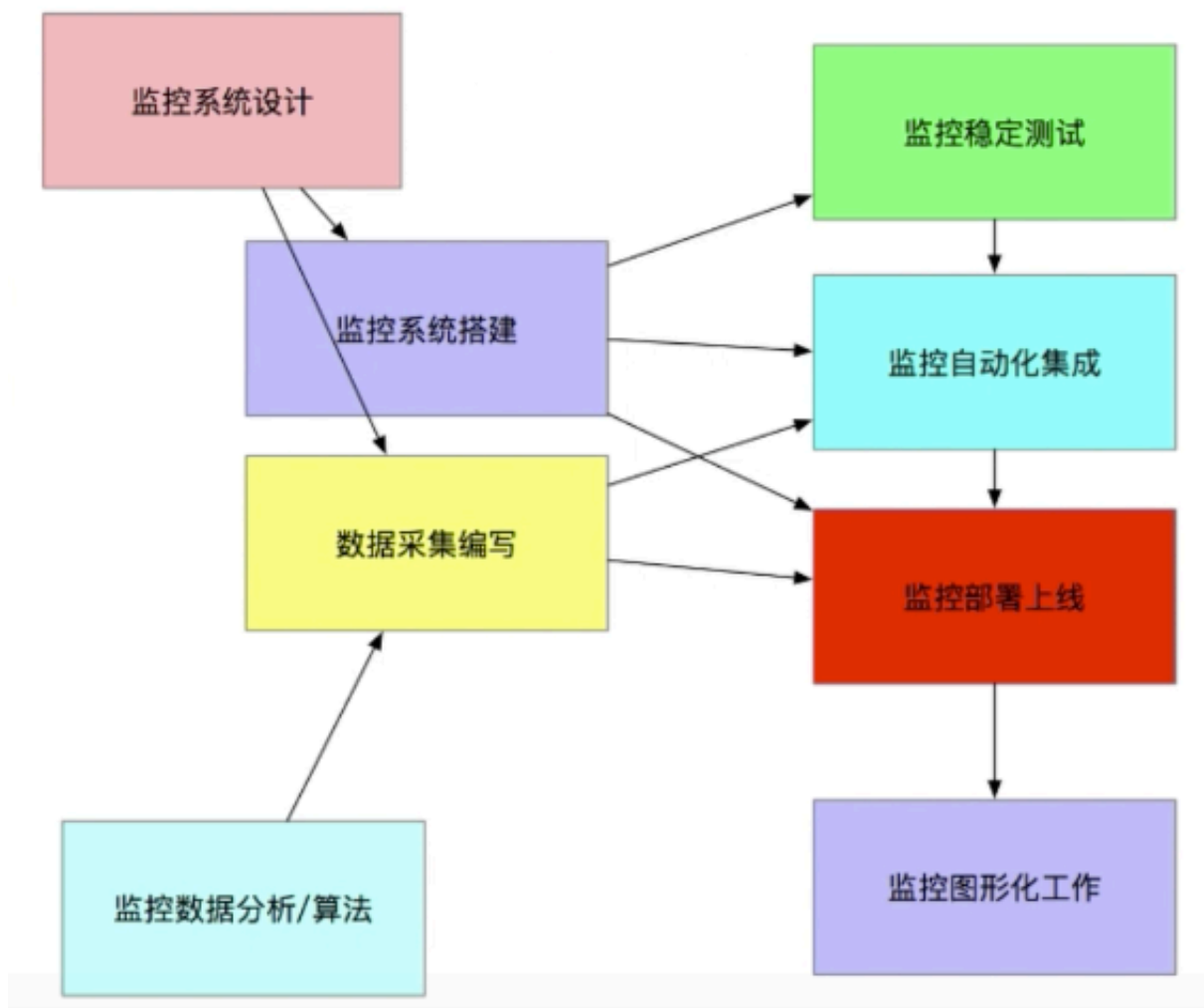
最常用的即为**node_exporter**，几乎可以把Linux系统中和系统相关的监控数据全都抓取出来。

pushgateway

exporter是安装在被监控的服务器上运行在后台，然后自动采集系统数据，本身又是一个HTTP_server可以被prometheus服务器定时去HTTP GET取得数据，属于pull的形式，pushgateway就是相反的过程，pushgateway安装在客户端或服务端均可，pushgateway本身也是一个http服务器，运维通过写自己的脚本程序抓自己想要的监控数据，然后摄像头到pushgateway上再由pushgateway推送到prometheus服务端。

- exporter虽然采集类型已经很丰富了，但我们仍需要很多自制的监控数据；
- exporter由于数据类型采集量大，而很多数据在实际应用中用不到，而使用pushgateway是定义一项数据就可以采集一种，可以大量节省资源。

Prometheus安装与基本使用



prometheus安装步骤

```
shell> chronyc -c makestep
shell> wget -P /root
https://github.com/prometheus/prometheus/releases/download/v2.17.1/prometheus-2.17.1.linux-amd64.tar.gz
```

```

shell> tar axf /root/prometheus-2.17.1.linux-amd64.tar.gz -C
/usr/local
shell> cd /usr/local
shell> tar axf prometheus-2.17.1.linux-amd64 prometheus
shell> cd /usr/local/prometheus
shell> vim startup.sh
    /usr/local/prometheus/prometheus --
config.file="/usr/local/prometheus/prometheus.yml" --web.listen-
address="0.0.0.0:9090" --web.read-timeout=5m --web.max-connections=10
--storage.tsdb.retention=15d --storage.tsdb.path="/us
r/local/prometheus/data" --query.max-concurrency=20 --query.timeout=2m
shell> chmod +x startup.sh

shell> wget -P /root http://local-
yum.ycigilink.local/Softwares/daemonize/daemonize-release-1.7.8.tar.gz
shell> tar axf /root/daemonize-release-1.7.8.tar.gz -C /usr/local/src
shell> cd /usr/local/src/daemonize-release-1.7.8
shell> ./configure && make && make install

shell> daemonize -c /usr/local/prometheus
/usr/local/prometheus/startup.sh

```

prometheus主配置文件

```

# my global config
global:
    scrape_interval:     15s # Set the scrape interval to every 15
seconds. Default is every 1 minute.
    evaluation_interval: 15s # Evaluate rules every 15 seconds. The
default is every 1 minute.
    # 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 'evaluation_interval'.
rule_files:
    # - "first_rules.yml"
    # - "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'

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

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

Global: 定义全局配置;

scrape_interval: 定义抓取采样数据的时间间隔, 默认每隔15s去被监控主机上采样一次。

evaluation_interval: 监控数据规则的评估频率, 主要用于监控阈值的评估。

scrape_configs: 定义prometheus自身的配置。

Job_name: 定义监控任务。

targets: 定义被监控的机器, 此处是一个数组, 可在该数组中定义多个被监控主机。

服务发现

以Consul为例, prometheus是通过配置文件来给prometheus定义被监控的项目和被监控节点,

```
- job_name: 'prometheus'
  static_configs:
    - targets: ['prometheus.server:9090']
- job_name: 'pushgateway'
  static_configs:
    - targets: ['localhost:9091']
```

job_name: 对应一个任务名称, 然后在这个“job_name”之下具体定义要被监控的节点以及节点上具体的端口信息等。

如果prometheus结合了consul这种服务发现软件, prometheus的配置文件就不再需要人工进行定义, 而是可以自动发现集群中有哪些新的机器, 以及新的机器上出现了哪些新的服务可以被监控。

prometheus采集客户端

prometheus主要有两种采集方式:

- pull, 主动拉取的形式; 即被监控机器先安装好各类已有的exporters, exporters

以守护进程的方式运行并开始采集数据，而exporter本身也是一个http_server，可以对http请求作出响应返回数据。

- push，被动推送的形式；在被监控机器（或其他机器）上安装好pushgateway插件，然后将运维开发的各种脚本把监控数据组织成k/v或metrics的形式发送给pushgateway，再由pushgateway推送给prometheus。

prometheus常用函数

increase(): 在prometheus中是用于针对Counter这种持续增长的数值，截取一段时间的增量。

```
increase(node_cpu_seconds_total[1m])
```

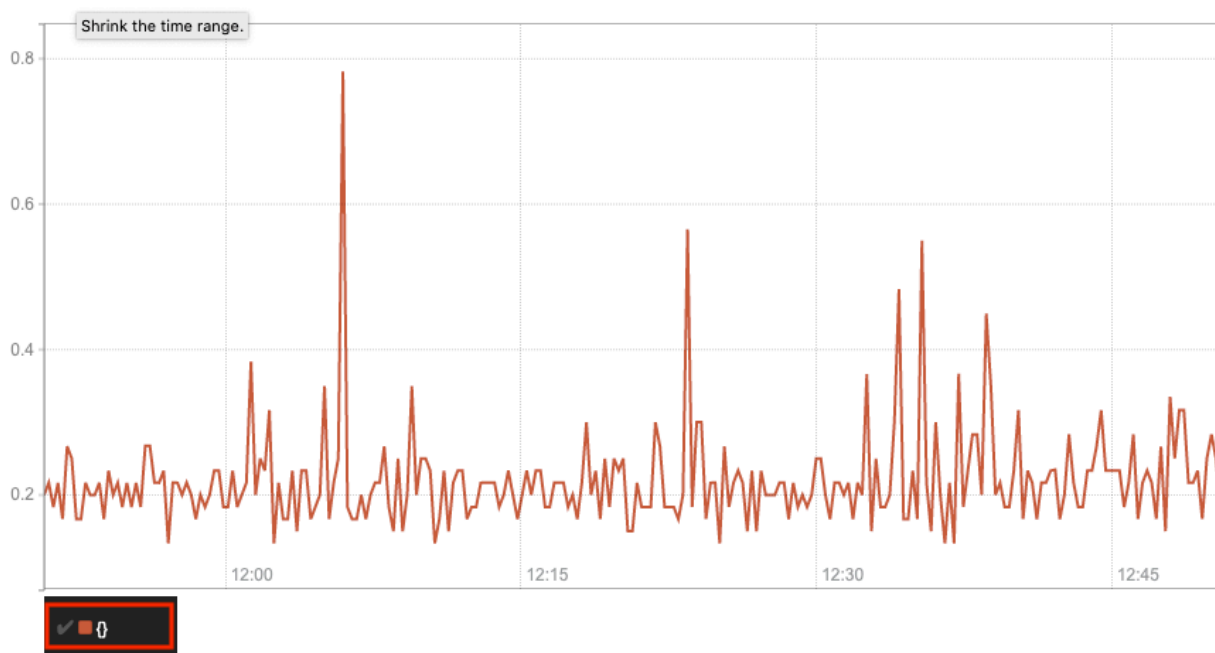
sum(): 对函数内的数值进行求和，但会对匹配的所有机器的该指标数据进行加和。

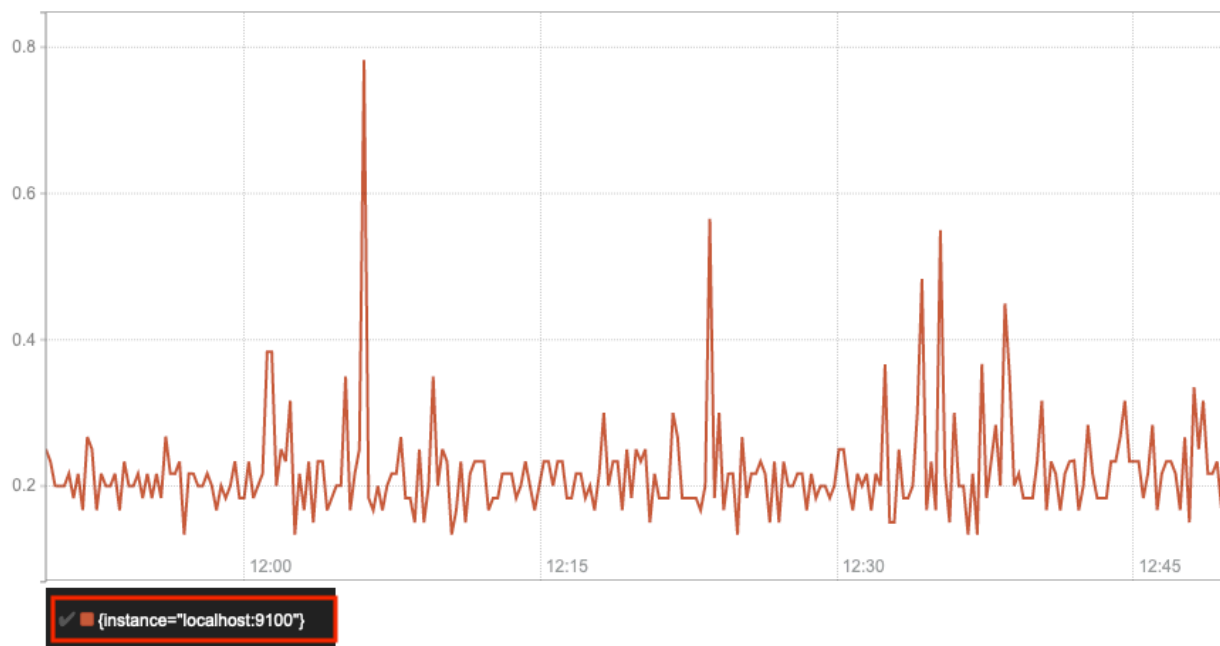
```
sum(increase(node_cpu_seconds_total{mode="idle"}[30s]))
```

```
( 1- sum(increase(node_cpu_seconds_total{mode="idle"}[30s]))/sum(increase(node_cpu_seconds_total[30s])) ) * 100
```

by(instance): 把sum加和到一起的数值按照指定的一个方式进行拆分，“instance”即表示机器名。

```
( 1- sum(increase(node_cpu_seconds_total{mode="idle"}[30s]))by(instance) /sum(increase(node_cpu_seconds_total[30s]))by(instance)) * 100
```



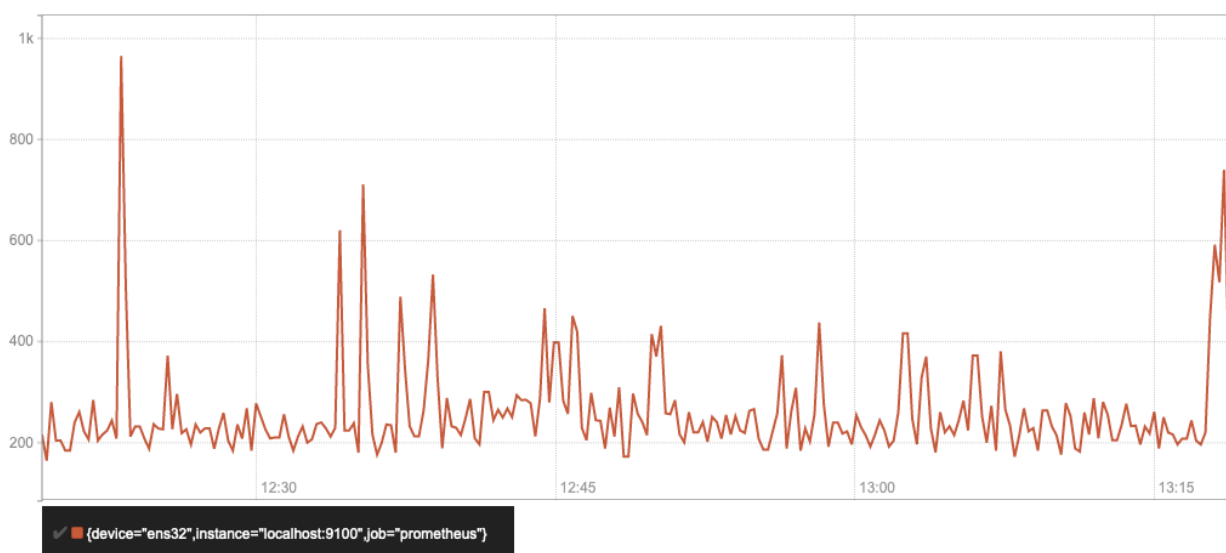


`exported_instance="xxx"`: 用于定义展示出的匹配标签的数据。

`exported_instance=~"xxx"`: 用于匹配正则表达式。

`rate()`: 专门用于搭配`counter`类型的数据使用的函数，其功能是按照设置的一个时间段，取出`counter`在这个时间段中平均每秒的变化。

`rate(node_network_receive_bytes_total{device="ens32"}[30s])`



如需获取概要数据可使用`increase()`函数，而如果获取精细数据则应该使用`rate()`函数进行处理。

Grafana

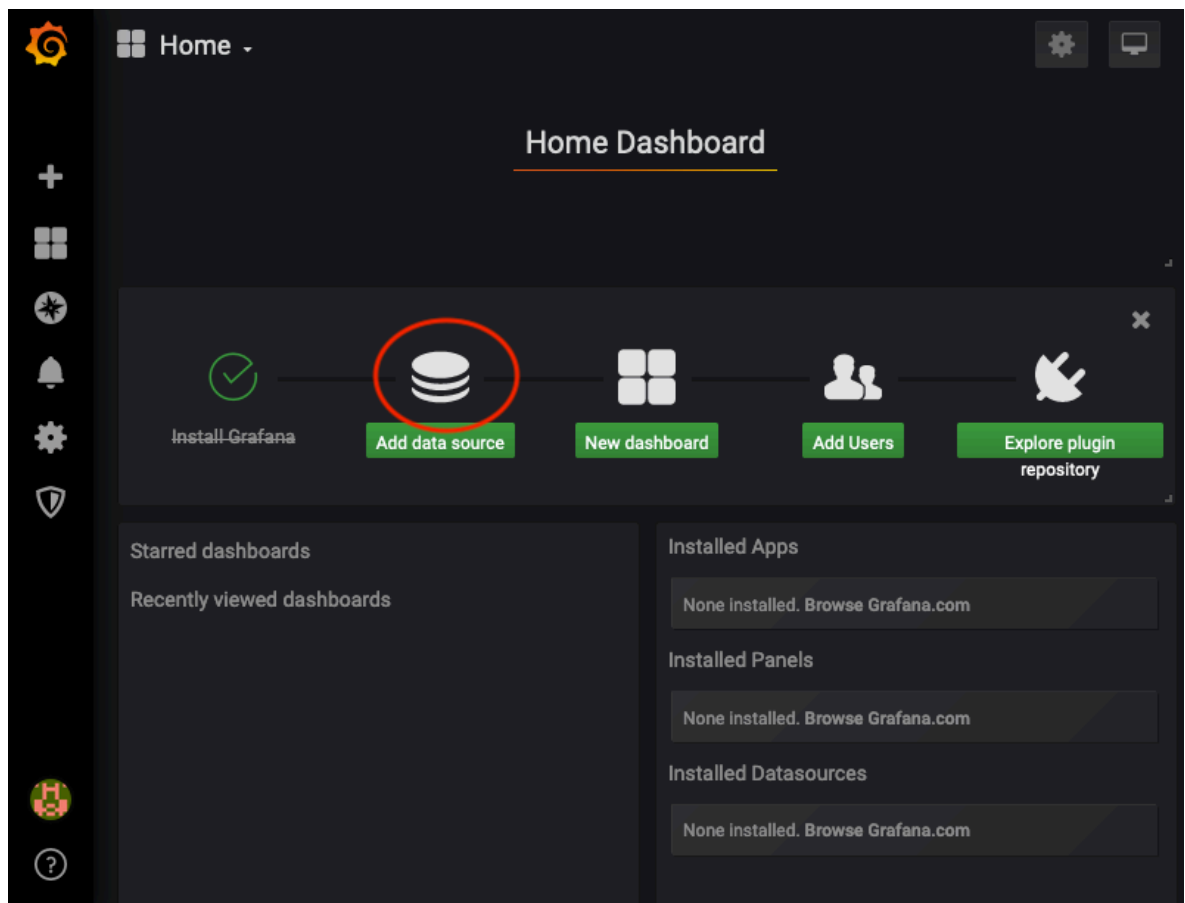
添加数据源










1. 安装并启动grafana服务。


```
shell> yum -y localinstall ./grafana-6.5.2-1.x86_64.rpm  
shell> systemctl start grafana-server.service  
shell> systemctl enable grafana-server.service
```

启动服务后grafana监听在3000的端口。

2. 初次登陆grafana有初始密码（admin/admin），但登陆后需要修改密码。
3. 添加数据源。






Add data source


Choose a data source type

[Cancel](#)


Time series databases




Prometheus
Open source time series database & alerting

[Learn more](#) 


Select




Graphite
Open source time series database



OpenTSDB
Open source time series database



InfluxDB
Open source time series database


Name ⓘtest_prometheusDefault 


HTTP


URL ⓘhttp://test4.ycigilink.local:9090


Whitelisted Cookies ⓘAdd Name Add


Auth


Basic auth 

With Credentials ⓘ 

TLS Client Auth 

With CA Cert ⓘ 

Skip TLS Verify 

Forward OAuth Identity ⓘ 

Scrape interval ⓘ

Query timeout ⓘ60s

HTTP Method ⓘChoose ▾

Misc

Custom query parameters ⓘExample: max_source_resolution=5m&timeout=10

Save & Test Delete Back

添加完成后点击首页即可看到数据源的位置已被标识为删除线。

