肖祥

基于Prometheus的Pushgateway实战

一、Pushgateway 简介

Pushgateway 是 Prometheus 生态中一个重要工具，使用它的原因主要是：

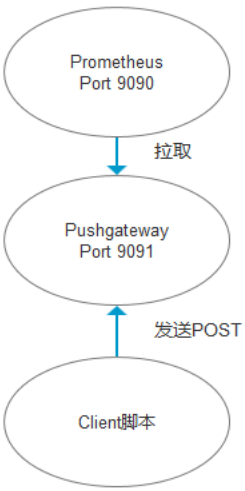
- Prometheus 采用 pull 模式，可能由于不在一个子网或者防火墙原因，导致 Prometheus 无法直接拉取各个 target 数据。
- 在监控业务数据的时候，需要将不同数据汇总, 由 Prometheus 统一收集。

由于以上原因，不得不使用 pushgateway，但在使用之前，有必要了解一下它的一些弊端：

- 将多个节点数据汇总到 pushgateway, 如果 pushgateway 挂了，受影响比多个 target 大。
- Prometheus 拉取状态 up 只针对 pushgateway, 无法做到对每个节点有效。
- Pushgateway 可以持久化推送给它的所有监控数据。

因此，即使你的监控已经下线，prometheus 还会拉取到旧的监控数据，需要手动清理 pushgateway 不要的数据。

拓扑图如下：



二、基于Docker 安装

使用 prom/pushgateway 的 Docker 镜像

```
docker pull prom/pushgateway
```

公告

凛冬将至,群狼可活,孤狼

独立博客:
http://www.py3study.com

昵称：肖祥
园龄：1年2个月
粉丝：131
关注：12
+加关注

< 2019年5月

日	一	二	三	四
28	29	30	1	2
5	6	7	8	9
12	13	14	15	16
19	20	21	22	23
26	27	28	29	30
2	3	4	5	6

搜索

ELK(5)

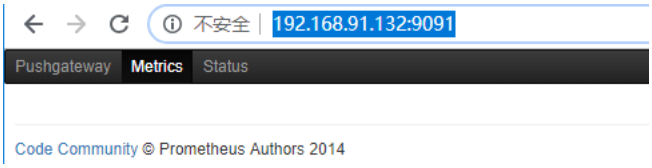
接下来启动Push Gateway：

```
docker run -d \
  --name=pg \
  -p 9091:9091 \
  prom/pushgateway
```

访问url：

```
http://192.168.91.132:9091/
```

效果如下：



在上一篇文章 https://www.cnblogs.com/xiao987334176/p/9930517.html 中，已经搭建好了Prometheus

要使Push Gateway正常工作，必须要在prometheus中配置对应的job才行

修改配置文件

```
vim /opt/prometheus/prometheus.yml
```

添加Push Gateway，完整内容如下：

```
global:
  scrape_interval:     60s
  evaluation_interval: 60s

scrape_configs:
  - job_name: prometheus
    static_configs:
      - targets: ['localhost:9090']
        labels:
          instance: prometheus

  - job_name: linux
    static_configs:
      - targets: ['192.168.91.132:9100']
        labels:
          instance: localhost

  - job_name: pushgateway
    static_configs:
      - targets: ['192.168.91.132:9091']
        labels:
          instance: pushgateway
```

由于prometheus.yml是外部加载的，docker在前面已经后台运行了。无法及时生效！

使用 docker ps 命令查看当前docker进程

```
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
NAMES
```


docker(4)

随笔分类

python 全栈开发(152)

python 运维开发(54)

随笔档案

2019年5月 (7)

2019年4月 (1)

2019年2月 (1)

2019年1月 (13)

2018年12月 (12)

2018年11月 (18)

2018年10月 (8)

2018年9月 (22)

2018年8月 (21)

2018年7月 (21)

2018年6月 (22)

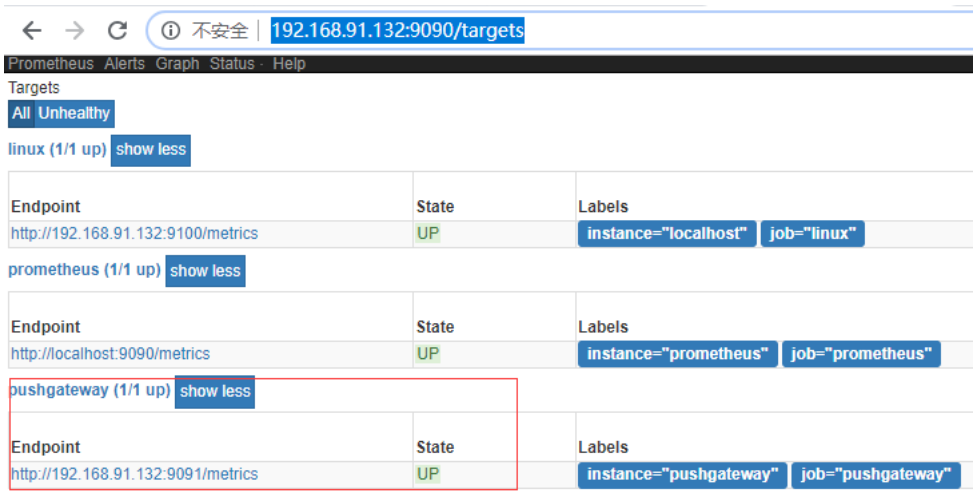
2018年5月 (23)

```
59ae7d9c8c3a        prom/prometheus      "/bin/prometheus -..."   16 minutes ago      Up 16 minutes
0.0.0.0:9090->9090/tcp   awesome_mcnulty
d907d0240018        prom/pushgateway     "/bin/pushgateway"       36 minutes ago      Up 36 minutes
0.0.0.0:9091->9091/tcp   pg
6b06f3b354cb        grafana/grafana      "/run.sh"                About an hour ago   Up About an hour
0.0.0.0:3000->3000/tcp   grafana3
62a0f435ea08        prom/node-exporter   "/bin/node_exporter"     2 hours ago         Up 2 hours
happy_galileo
```

重启prometheus的docker容器

```
docker restart 59ae7d9c8c3a
```

访问targets，等待1分钟，等待pushgateway状态为UP



三、数据管理

正常情况我们会使用 Client SDK 推送数据到 pushgateway, 但是我们还可以通过 API 来管理, 例如：

shell脚本

向 {job="some_job"} 添加单条数据：

```
echo "some_metric 3.14" | curl --data-binary @- http://pushgateway.example.org:9091/metrics/job/some_job
```

--data-binary 表示发送二进制数据，注意：它是使用POST方式发送的！

添加更多更复杂数据，通常数据会带上 instance, 表示来源位置：

```
cat <<EOF | curl --data-binary @- http://pushgateway.example.org:9091/metrics/job/some_job/instance/some_instanc
# TYPE some_metric counter
some_metric{label="val1"} 42
# TYPE another_metric gauge
# HELP another_metric Just an example.
another_metric 2398.283
EOF
```

注意：必须是指定的格式才行！

2. Re:python 全栈开发，Da
介绍,变量,if,while)

希望坚持更新，你是我们的榜

3. Re:python 全栈开发，Da
作业讲解,模块搜索路径,编译
件,包以及包的import和from
范)

感谢感谢!

4. Re:python 全栈开发，Da
之间的交互,类命名空间与对象
名空间,类的组合用法)

@黑色彩虹 内容是重点啊，么
啊...

5. Re:python 全栈开发，Da
器,生成器)

讲的太棒了!

阅读排行榜

1. 基于Prometheus的Push
战(4485)

2. Python Elasticsearch ap

删除某个组下的某实例的所有数据：

```
curl -X DELETE http://pushgateway.example.org:9091/metrics/job/some_job/instance/some_instance
```

删除某个组下的所有数据：

```
curl -X DELETE http://pushgateway.example.org:9091/metrics/job/some_job
```

可以发现 pushgateway 中的数据我们通常按照 job 和 instance 分组分类，所以这两个参数不可缺少。

因为 Prometheus 配置 pushgateway 的时候，也会指定 job 和 instance，但是它只表示 pushgateway 实例，不能真正表达收集数据的含义。所以在 prometheus 中配置 pushgateway 的时候，需要添加 honor_labels: true 参数， 从而避免收集数据本身的 job 和 instance 被覆盖。

注意，为了防止 pushgateway 重启或意外挂掉，导致数据丢失，我们可以通过 -persistence.file 和 -persistence.interval 参数将数据持久化下来。

本文参考链接：

https://songjiayang.gitbooks.io/prometheus/content/pushgateway/how.html

python脚本

安装模块

```
pip3 install flask
pip3 install prometheus_client
```

Metrics

Prometheus提供4种类型Metrics：Counter, Gauge, Summary和Histogram

Counter

Counter可以增长，并且在程序重启的时候会被重设为0，常被用于任务个数，总处理时间，错误个数等只增不减的指标。

示例代码：

```
import prometheus_client
from prometheus_client import Counter
from prometheus_client.core import CollectorRegistry
from flask import Response, Flask

app = Flask(__name__)

requests_total = Counter("request_count", "Total request cout of the host")

@app.route("/metrics")
def requests_count():
    requests_total.inc()
    # requests_total.inc(2)
    return Response(prometheus_client.generate_latest(requests_total),
                    mimetype="text/plain")

@app.route('/')
def index():
```


2. python 全栈开发，Day7 组件-forms组件)(4)

3. python 全栈开发，Day6 简介)(2)

4. kafka查看消费数据(2)

5. python 全栈开发，Day8 子评论,后台管理,富文本编辑r,bs4模块)(2)

推荐排行榜

1. python 全栈开发，Day1 绍,变量,if,while)(3)

2. python 全栈开发，Day3 tr切片,str常用操作方法,for循

3. python 全栈开发，Day1 成器)(2)

4. python 全栈开发，Day2 口类,多态,鸭子类型)(2)

5. python 全栈开发，Day1 用,闭包,装饰器初识,带参数以的装饰器)(2)

```
    requests_total.inc()
    return "Hello World"

if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

运行该脚本，访问youhost:5000/metrics

```
# HELP request_count Total request cout of the host
# TYPE request_count counter
request_count 3.0
```

Gauge

Gauge与Counter类似，唯一不同的是Gauge数值可以减少，常被用于温度、利用率等指标。

示例代码：

```
import random
import prometheus_client
from prometheus_client import Gauge
from flask import Response, Flask

app = Flask(__name__)

random_value = Gauge("random_value", "Random value of the request")

@app.route("/metrics")
def r_value():
    random_value.set(random.randint(0, 10))
    return Response(prometheus_client.generate_latest(random_value),
                    mimetype="text/plain")


if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

运行该脚本，访问youhost:5000/metrics

```
# HELP random_value Random value of the request
# TYPE random_value gauge
random_value 3.0
```

Summary/Histogram

Summary/Histogram概念比较复杂，一般exporter很难用到，暂且不说。

PLUS

LABELS

使用labels来区分metric的特征

示例代码：

```
from prometheus_client import Counter
```

```
c = Counter('requests_total', 'HTTP requests total', ['method', 'clientip'])

c.labels('get', '127.0.0.1').inc()
c.labels('post', '192.168.0.1').inc(3)
c.labels(method="get", clientip="192.168.0.1").inc()
```

REGISTRY

示例代码：

```
from prometheus_client import Counter, Gauge
from prometheus_client.core import CollectorRegistry

REGISTRY = CollectorRegistry(auto_describe=False)

requests_total = Counter("request_count", "Total request cout of the host", registry=REGISTRY)
random_value = Gauge("random_value", "Random value of the request", registry=REGISTRY)
```

本文参考链接：

https://blog.csdn.net/huochen1994/article/details/76263078

举例(网卡流量)

先访问这篇文章《python 获取网卡实时流量》：

http://www.py3study.com/Article/details/id/347.html

下面这段python脚本，主要是参考上面文章的基础上修改的

发送本机网卡流量

```
import prometheus_client
from prometheus_client import Counter
from prometheus_client import Gauge
from prometheus_client.core import CollectorRegistry
import psutil
import time
import requests
import socket

def get_key():
    key_info = psutil.net_io_counters(pernic=True).keys()

    recv = {}
    sent = {}

    for key in key_info:
        recv.setdefault(key, psutil.net_io_counters(pernic=True).get(key).bytes_recv)
        sent.setdefault(key, psutil.net_io_counters(pernic=True).get(key).bytes_sent)

    return key_info, recv, sent


def get_rate(func):
    import time
```

```
    key_info, old_recv, old_sent = func()

    time.sleep(1)

    key_info, now_recv, now_sent = func()

    net_in = {}
    net_out = {}

    for key in key_info:
        # float('%.2f' % a)
        # net_in.setdefault(key, float('%.2f' %((now_recv.get(key) - old_recv.get(key)) / 1024)))
        # net_out.setdefault(key, float('%.2f' %((now_sent.get(key) - old_sent.get(key)) / 1024)))

        # 计算流量
        net_in.setdefault(key, now_recv.get(key) - old_recv.get(key))
        net_out.setdefault(key, now_sent.get(key) - old_sent.get(key))

    return key_info, net_in, net_out

# def get_host_ip():
#     """
#     查询本机ip地址,针对单网卡
#     :return: ip
#     """
#     try:
#         s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
#         s.connect(('8.8.8.8', 80))
#         ip = s.getsockname()[0]
#     finally:
#         s.close()
#         return ip

# 打印多网卡 mac 和 ip 信息
def PrintNetIfAddr():
    dic = psutil.net_if_addrs()
    net_dic = {}
    net_dic['no_ip'] = []  # 无ip的网卡列表
    for adapter in dic:
        snicList = dic[adapter]
        mac = '无 mac 地址'
        ipv4 = '无 ipv4 地址'
        ipv6 = '无 ipv6 地址'
        for snic in snicList:
            if snic.family.name in {'AF_LINK', 'AF_PACKET'}:
                mac = snic.address
            elif snic.family.name == 'AF_INET':
                ipv4 = snic.address
            elif snic.family.name == 'AF_INET6':
                ipv6 = snic.address
        # print('%s, %s, %s, %s' % (adapter, mac, ipv4, ipv6))

        # 判断网卡名不在net_dic中时,并且网卡不是lo
        if adapter not in net_dic and adapter != 'lo':
            if not ipv4.startswith("无"):  # 判断ip地址不是以无开头
                net_dic[adapter] = ipv4  # 增加键值对
            else:
                net_dic['no_ip'].append(adapter)  # 无ip的网卡

    # print(net_dic)
    return net_dic

key_info, net_in, net_out = get_rate(get_key)

# ip=get_host_ip()  # 本机ip
```

```
hostname = socket.gethostname() # 主机名

REGISTRY = CollectorRegistry(auto_describe=False)
input = Gauge("network_traffic_input", hostname,['adapter_name','unit','ip','instance'],registry=REGISTRY)  # 流入
output = Gauge("network_traffic_output", hostname,['adapter_name','unit','ip','instance'],registry=REGISTRY)  # 流出


for key in key_info:
    net_addr = PrintNetIfAddr()
    # 判断网卡不是lo(回环网卡)以及 不是无ip的网卡
    if key != 'lo' and  key not in net_addr['no_ip']:
        # 流入和流出
        input.labels(ip=net_addr[key],adapter_name=key, unit="Byte",instance=hostname).inc(net_in.get(key))
        output.labels(ip=net_addr[key],adapter_name=key, unit="Byte",instance=hostname).inc(net_out.get(key))

requests.post("http://192.168.91.132:9091/metrics/job/network_traffic",data=prometheus_client.generate_latest(REGISTRY))
print("发送了一次网卡流量数据")
```

执行脚本，它会发送1次数据给Push Gateway

取到的流量没有除以1024，所以默认是字节

注意：发送的链接，约定成俗的格式如下：

```
http://Pushgateway地址:9091/metrics/job/监控项目
```

比如监控etcd，地址就是这样的

```
http://Pushgateway地址:9091/metrics/job/etcd
```

必须使用POST方式发送数据！

代码解释

关键代码，就是这几行

```
REGISTRY = CollectorRegistry(auto_describe=False)
input = Gauge("network_traffic_input", hostname,['adapter_name','unit','ip','instance'],registry=REGISTRY)  # 流入
output = Gauge("network_traffic_output", hostname,['adapter_name','unit','ip','instance'],registry=REGISTRY)  # 流出

input.labels(ip=net_addr[key],adapter_name=key, unit="Byte",instance=hostname).inc(net_in.get(key))
output.labels(ip=net_addr[key],adapter_name=key, unit="Byte",instance=hostname).inc(net_out.get(key))
```

1、自定义的指标收集类都必须到CollectorRegistry进行注册， 指标数据通过CollectorRegistry类的方法或者函数，返回给Prometheus.
2、CollectorRegistry必须提供register()和unregister()函数，一个指标收集器可以注册多个CollectorRegistry.
3、客户端库必须是线程安全的

代码第一行，声明了CollectorRegistry

input和output是流入流出的流量。Metrics使用的是Gauge

```
input = Gauge("network_traffic_input", hostname,['adapter_name','unit','ip','instance'],registry=REGISTRY)  # 流
```

network_traffic_input表示键值，它必须唯一。因为在grafana图表中，要用这个键值绘制图表。

"" 为空，它其实对应的是描述信息。为了避免数据冗长，一般不写它。

['adapter_name','unit','ip','instance'] ，它是一个列表，里面每一个元素都是labels，它是用来区分metric的特征

registry=REGISTRY 把数据注册到REGISTRY中

```
input.labels(ip=net_addr[key],adapter_name=key, unit="Byte",instance=hostname).inc(net_in.get(key))
```

这里定义了input的labels，括号里面有3个键值对。**注意：这3个键值对必须在['adapter_name','unit','ip'] 列表中。**

如果labels中要增加键值对，那么上面的列表中，也要增加对应的元素。否则会报错！

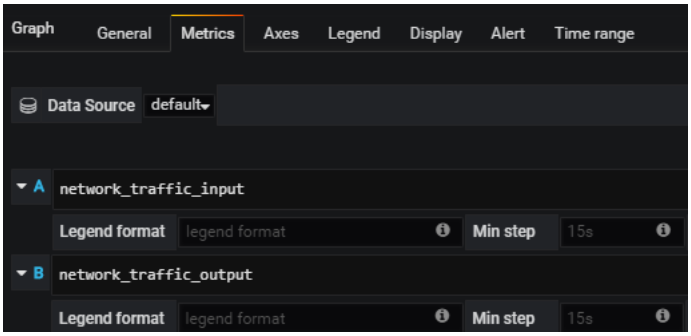
inc表示具体值。它对应的是input

刷新Push Gateway页面

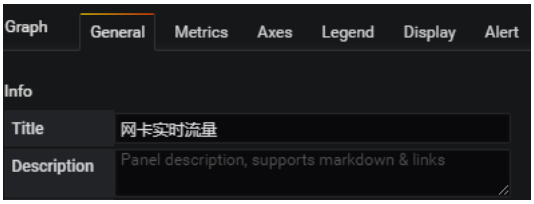
展开数据，这里就是流入流出的数据了

进入grafana页面，新建一个图表

添加网络 流入和流出指标



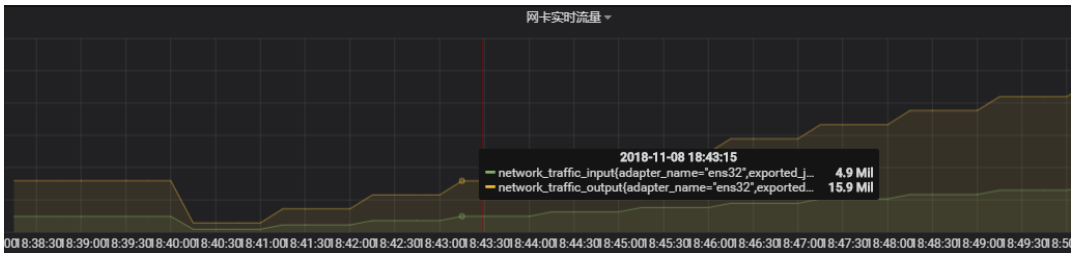
更改标题



设置liunx任务计划，每分钟执行一次

```
* * * * * python3 /opt/test.py
```

效果如下：



如果服务器没有流量的话，可以造点流量

写一个脚本，持续访问某张图片

```
import requests
while True:
    requests.get("http://192.168.91.128/Netraffic/dt.jpg")
    print('正在访问图片')
```

如果需要监控Mysql，参考这篇文章

https://www.jianshu.com/p/27b979554ef8

注意：它使用的是用flask暴露了一个Metrics，用来给Prometheus提供数据。

那么就需要在 Prometheus的配置文件中，添加对应的job才能收集到数据。

它会定期访问暴露的http链接，获取数据。

总结：

使用Prometheus监控，有2中方式

1. 暴露http方式的Metrics，注意：需要在Prometheus的配置文件中添加job

2. 主动发送数据到Pushgateway，注意：只需要添加一个Pushgateway就可以了。它相当于一个API，无论有多少个服务器，发送到统一的地址。

生产环境中，一般使用Pushgateway，简单，也不需要修改Prometheus的配置文件！

分类： python 运维开发

标签： 运维开发

肖祥
关注 - 12
粉丝 - 131
+加关注

« 上一篇：基于docker 搭建Prometheus+Grafana

» 下一篇：python 多线程删除MySQL表

posted @ 2018-11-09 11:07 肖祥 阅读(4486) 评论(0) 编辑 收藏

相关博文：
· prometheus+grafana安装部署（centos6.8）
· cAdvisor+Prometheus+Grafana监控docker
· prometheus安装
· Prometheus+Grafana+Altermanager钉钉报警
· 2-prometheus各组件安装

最新新闻：
· 哈勃发现一个正在靠近银河系的星系
· 腾讯云的『提速』前行
· 华为视频宣布与腾讯视频达成多方面合作
· 中国移动16亿元投资芒果超媒 成为第二大股东
· 一箭60星发射升空！美国准备跳过5G，直接升级到6G?
» 更多新闻...