Grafana + Prometheus + Node_exporter 监控资源

目录

1 Linux 下进行安装

1.1 下载地址

https://grafana.com/grafana/download

https://prometheus.io/download/

下载 grafana：

yum install -y https://dl.grafana.com/enterprise/release/grafana-enterprise-9.2.6-1.x86_64.rpm

下载 prometheus：

wget https://github.com/prometheus/prometheus/releases/download/v2.37.8/prometheus-2.37.8.linux-amd64.tar.gz

下载 node_exporter：

wget https://github.com/prometheus/node_exporter/releases/download/v1.5.0/node_exporter-1.5.0.linux-amd64.tar.gz

2 安装/解压

2.1 安装 grafana

yum install grafana-enterprise-9.2.6-1.x86_64.rpm

或

rpm -Uvh (--nodeps) grafana-enterprise-9.2.6-1.x86_64.rpm

2.2 安装 prometheus

tar -xzvf prometheus-2.37.8.linux-amd64.tar.gz

2.3 安装 node_exporter

tar -xzvf node_exporter-1.5.0.linux-amd64.tar.gz

3 修改配置

3.1 grafana

使用 vim /etc/grafana/grafana.ini,

将[server]->;http_addr= 的 ; 去掉

3.2 prometheus

使用 vim prometheus/Prometheus.yml

在 scrape_configs->static_configs 中添加需要监控的 ip:端口，例：192.168.3.5:9100，用逗号分隔

4 启动

4.1 启动 grafana

手动启动：systemctl start grafana-server

设置开机自启动：systemctl enable grafana-server

4.2 prometheus

后台启动：在 prometheus 安装目录下执行

nohup ./prometheus &

添加启动服务

vim /usr/lib/systemd/system/prometheus.service

```
[Unit]
Description= Prometheus
After=network.target
[Service]
Type=simple
User=prometheus
#此处是 Prometheus 所在路径以及数据所在路径
ExecStart=/usr/local/Prometheus/prometheus-2.37.8.linux-
amd64/prometheus --
config.file=/usr/local/Prometheus/prometheus-
2.37.8.linux-amd64/prometheus.yml --
storage.tsdb.path=/data/prometheus/data
ExecReload=/bin/kill -HUP $MAINPID
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

设置开机自启动

```
systemctl daemon-reload
systemctl enable prometheus.service
systemctl start prometheus.service
```

4.3 node_exporter

后台启动：在 node_exporter 安装目录下执行

```
nohup ./node_exporter &
```

```
vim /etc/systemd/system/node_exporter.service
[Unit]
Description=Prometheus Node Exporter
After=network.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/bin/node_exporter

[Install]
WantedBy=multi-user.target
```

设置开机自启动

```
systemctl daemon-reload
systemctl enable node_exporter
systemctl start node_exporter
```

5 访问

访问 grafana

输入 Linux ip:3000，例：192.168.3.5:3000

访问 prometheus

输入 Linux ip:9090，例：192.168.3.5:9090

访问 node_exporter

输入 Linux ip:9100，例：192.168.3.5:9100

6 添加自定义监控

6.1 pushgateway

6.1.1 下载地址

https://prometheus.io/download/

wget https://github.com/prometheus/pushgateway/releases/download/v1.4.3/pushgateway-1.4.3.linux-amd64.tar.gz

6.1.2 安装

tar -xzvf https://github.com/prometheus/pushgateway/release

s/download/v1.4.3/pushgateway-1.4.3.linux-amd64.tar.gz

6.1.3 启动

```
nohup ./pushgateway &
```

6.1.4 编写 shell 脚本

```
#!/usr/bin/bash

instance_name=`hostname -f|cut -d'.' -f1`   #截取主机名

if [ ${instance_name} == "localhost" ];then
  echo "Must FQDN hostname"      #要求主机名不能是localhost，不要主机名区别不了
  exit 1
fi

#定义 key
label_wait="count_netstat_wait_connections"
#定义 value
count_netstat_wait_connections=`netstat  -an|grep -i wait|wc -l`

echo
```

```
"${label_wait}:${count_netstat_wait_connections}"

#推送数据给 pushgateway
echo "${label_wait} ${count_netstat_wait_connections}"|curl --data-binary @- http://192.168.3.35:9091/metrics/job/pushgateway/instance/${instance_name}


#定义 key
label_wait="count_coredump"
#定义 value
count_coredump=`ls -lrt /var/lib/systemd/coredump|grep "^-"|wc -l`

echo "${label_wait}:${count_coredump}"

#推送数据给 pushgateway
echo "${label_wait} ${count_coredump}"|curl --data-binary @- http://192.168.3.35:9091/metrics/job/pushgateway/instance/${instance_name}
```

6.1.5 定义定时任务

Crontab -e

每五秒执行一次脚本，输入

```
*/1 * * * * sleep 5 && sh /root/shell_scripts/pushgateway_shell.sh
```

6.2 Python 代码

6.2.1 Python 连接 Linux 执行命令监控数据

```
# -*- encoding: utf-8 -*-
# Author: Komorebi
# Date: 2023/7/23 12:37
# Describe: Prometheus monitor server port
import random

import prometheus_client
from prometheus_client import Gauge
from prometheus_client.core import CollectorRegistry
from flask import Response, Flask

from utils.connLinux import ConnLinux, connLinux

app = Flask(__name__, static_url_path="/main")
# 实例化 REGISTRY
registry = CollectorRegistry(auto_describe=False)
gauge = Gauge(
    name="Server_port",
    documentation="monitor server port status.",
    labelnames=["sertype", "host", "port"],
    registry=registry
```

```
)


@app.route("/metrics")
def requests_count():
    result = ConnLinux().exec_command("ls -l /root/shell_scripts|grep '^-'|wc -l")

    # 模拟多个值传入
    rows = [
        {"sertype": "zookeeper", "host": "192.168.1.22", "port": "2181", "status": result},
        {"sertype": "zookeeper", "host": "192.168.1.33", "port": "2181", "status": random.randint(10, 30)},
        {"sertype": "zookeeper", "host": "192.168.1.44", "port": "2181", "status": random.randint(15, 35)},
        {"sertype": "mysql", "host": "192.168.1.88", "port": "3306", "status": random.randint(5, 25)},
        {"sertype": "mysql", "host": "192.168.1.99", "port": "3306", "status": random.randint(20, 40)}
    ]

    for row in rows:
        sertype = "".join(row.get("sertype"))
        ip = "".join(row.get("host"))
        port = "".join(row.get("port"))
        status = row.get("status")
```

```
        gauge.labels(sertype, ip, port).set(status)
    return
Response(prometheus_client.generate_latest(regist
ry), mimetype="text/plain")


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=31672, debug=True)
```