**1、Prometheus基本使用**

**1.1 Prometheus介绍**

<https://prometheus.io>

<https://github.com/prometheus>

Prometheus（普罗米修斯）是一个最初在SoundCloud上构建的监控系统。自2012年成为社区开源项目，

拥有非常活跃的开发人员和用户社区。为强调开源及独立维护，Prometheus于2016年加入云原生云计算基

金会（CNCF），成为继Kubernetes之后的第二个托管项目。

Prometheus 特点：

• 多维数据模型：由度量名称和键值对标识的时间序列数据

• PromQL：一种灵活的查询语言，可以利用多维数据完成复杂的查询

• 不依赖分布式存储，单个服务器节点可直接工作

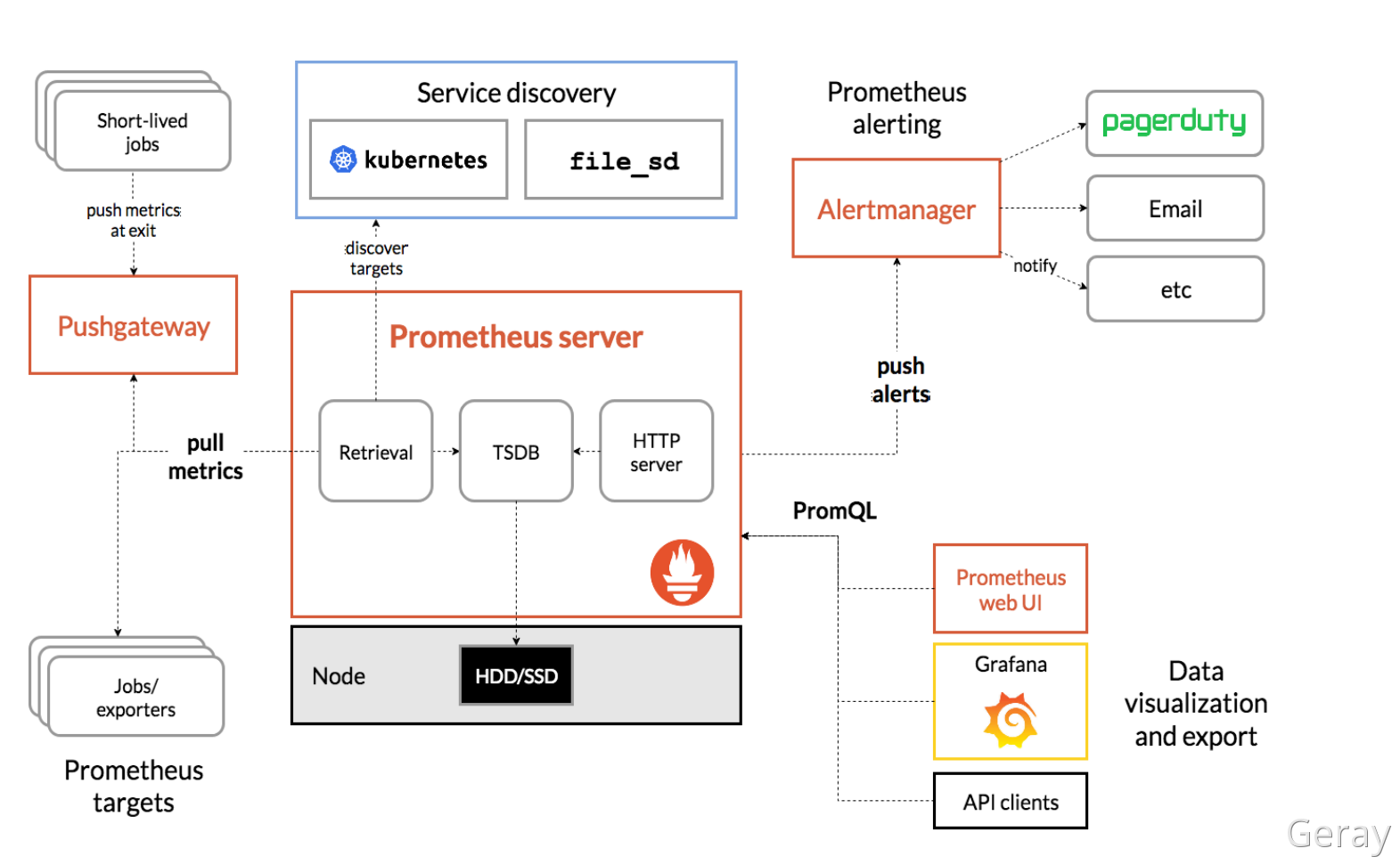
• 基于HTTP的pull方式采集时间序列数据

• 推送时间序列数据通过PushGateway组件支持

• 通过服务发现或静态配置发现目标

• 多种图形模式及仪表盘支持（grafana）

**1.2 Prometheus组件与架构**



• Prometheus Server：收集指标和存储时间序列数据，并提供查询接口

• ClientLibrary：客户端库

• Push Gateway：短期存储指标数据。主要用于临时性的任务

• Exporters：采集已有的第三方服务监控指标并暴露metrics

• Alertmanager：告警

• Web UI：简单的Web控制台

**1.3 Prometheus部署**

**1.3.1 部署方式**

• 二进制部署

<https://prometheus.io/docs/prometheus/latest/getting_started/>

• Docker部署

<https://prometheus.io/docs/prometheus/latest/installation/>

访问地址：<http://IP:9090>

./premetheus -h 命令行常用参数：

• --config.file="prometheus.yml" # 指定配置文件

• --web.listen-address= "0.0.0.0:9090" # 监听地址和端口

• --log.level=info # 日志级别

• --alertmanager.timeout=10s # 与报警组件的超时时间

• --storage.tsdb.path="data/ " # 数据目录

• --storage.tsdb.retention.time=15d # 数据保存时间，默认15天

* promtool 检查Prometheus语法

# 检查配置文件是否正确

./promtool check config ./prometheus.yaml

**1.3.2 使用systemd服务管理Prometheus**

配置为系统服务管理：

# vi /usr/lib/systemd/system/prometheus.service

[Unit]

Description=prometheus

[Service]

ExecStart=/opt/monitor/prometheus/prometheus --config.file=/opt/monitor/prometheus/prometheus.yml

ExecReload=/bin/kill -HUP $MAINPID

KillMode=process

Restart=on-failure

[Install]

WantedBy=multi-user.target

# systemctl daemon-reload

# systemctl start prometheus

# systemctl enable prometheus

**1.3.3 docker部署**

进入到加压目录或者配置文件目录

docker run \

-p 9090:9090 \

--name promethues \

-v $PWD/prometheus.yml:/etc/prometheus/prometheus.yml \

prom/prometheus

**1.4 Prometheus配置文件**

prometheus.yml

# 全局配置

global:

scrape\_interval: 15s # 采集数据时间间隔，默认15秒

evaluation\_interval: 15s # 评估告警规则时间间隔，默认1分钟

# scrape\_timeout # 采集数据超时时间，默认10秒

# 告警规则

alerting: # 告警配置

alertmanagers:

- static\_configs:

- targets:

# - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation\_interval'.

rule\_files: # 告警规则

# - "first\_rules.yml"

# - "second\_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:

# Here it's Prometheus itself.

scrape\_configs:

# 配置被监控端，称为target，每个target用job\_name分组管理，又分为静态配置和服务发现

- job\_name: 'prometheus' # 分组管理，又分为静态配置和服务发现

# metrics\_path defaults to '/metrics' # 暴露接口的地址

# scheme defaults to 'http'.

static\_configs:

- targets: ['localhost:9090']

* 每次向配置文件中配置后，重启服务开销比较大，可以使用kill热加载

# 检查配置文件是否正确

$ ./promtool check config ./prometheus.yml

Checking ./prometheus.yml

SUCCESS: 0 rule files found

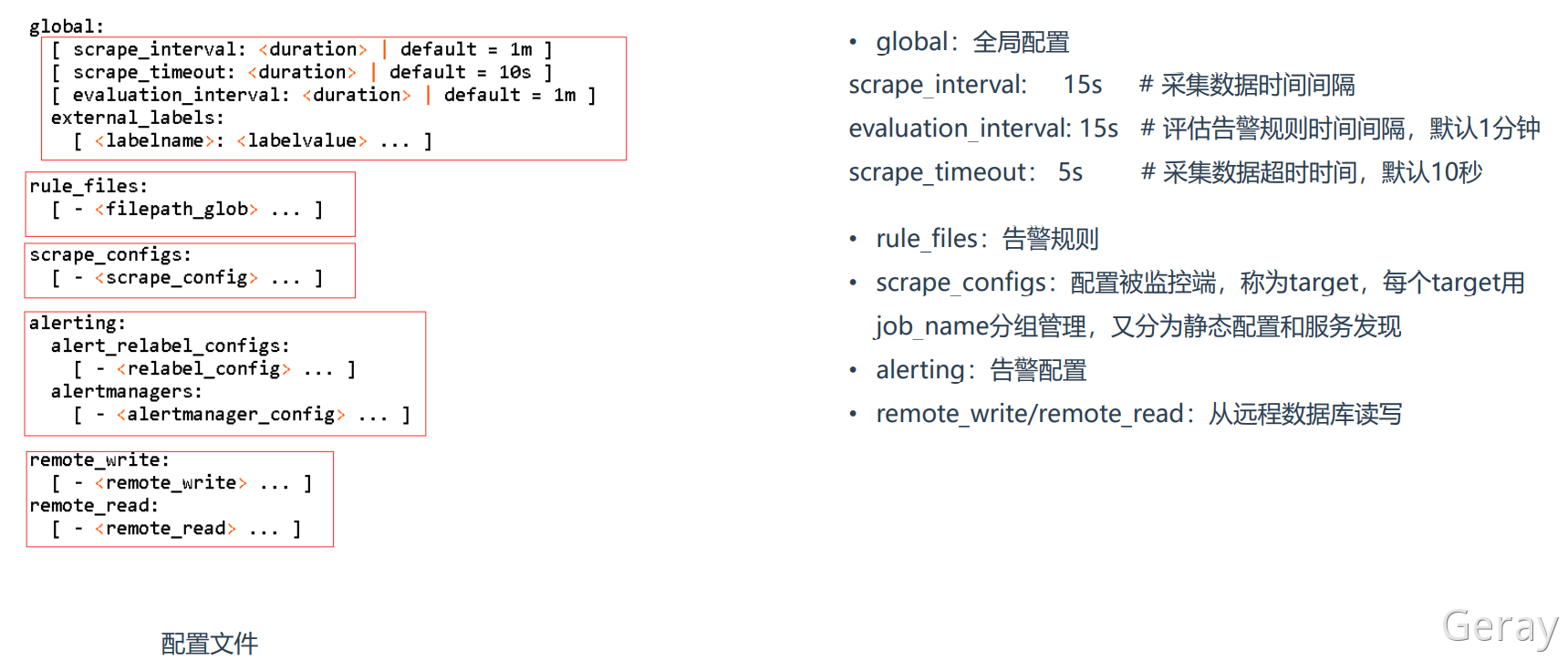
# 热加载配置文件

$ kill -HUP <pid>

**1.5 配置文件解析**

**1.5.1 配置文件**

参考文档：<https://prometheus.io/docs/prometheus/latest/configuration/configuration/>



• global：全局配置

scrape\_interval: 15s # 采集数据时间间隔

evaluation\_interval: 15s # 评估告警规则时间间隔，默认1分钟

scrape\_timeout： 5s # 采集数据超时时间，默认10秒

• rule\_files：告警规则

• scrape\_configs：配置被监控端，称为target，每个target用

job\_name分组管理，又分为静态配置和服务发现

• alerting：告警配置

• remote\_write/remote\_read：从远程数据库读写

**1.5.2 配置被监控端**



目标（targets）：被监控端

实例（Instances）：每个被监控端称为实例

作业（Job）：具有相同目标的实例集合称为作业

scrape\_configs:

- job\_name: 'prometheus'

metrics\_path: '/metrics'

scheme: 'http'.

static\_configs:

- targets: ['localhost:9090']

**1.5.3 监控指标数据模型**

数据模型：

• Prometheus将所有数据存储为时间序列；

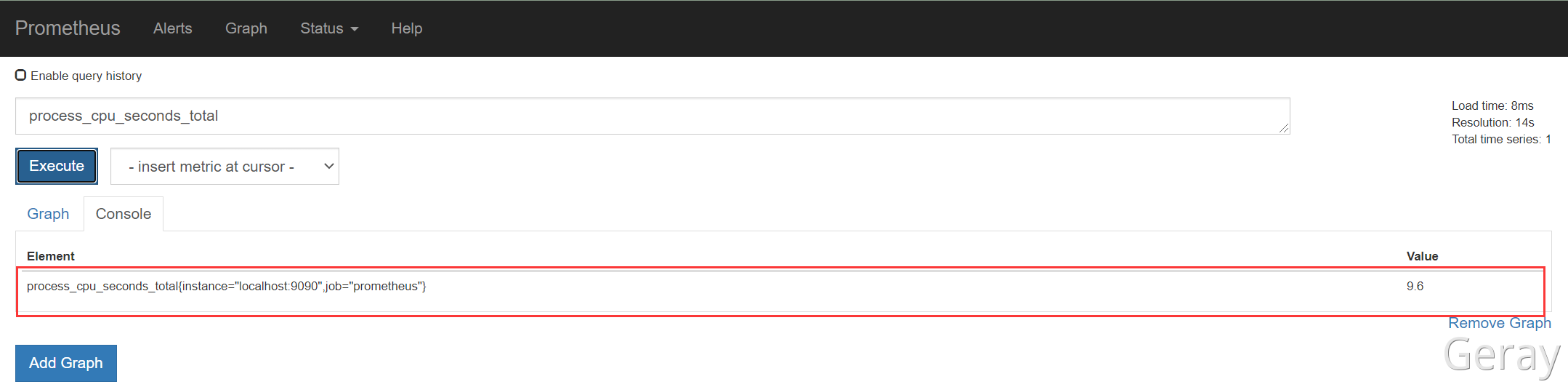
• 具有相同度量名称以及标签属于同一个指标；

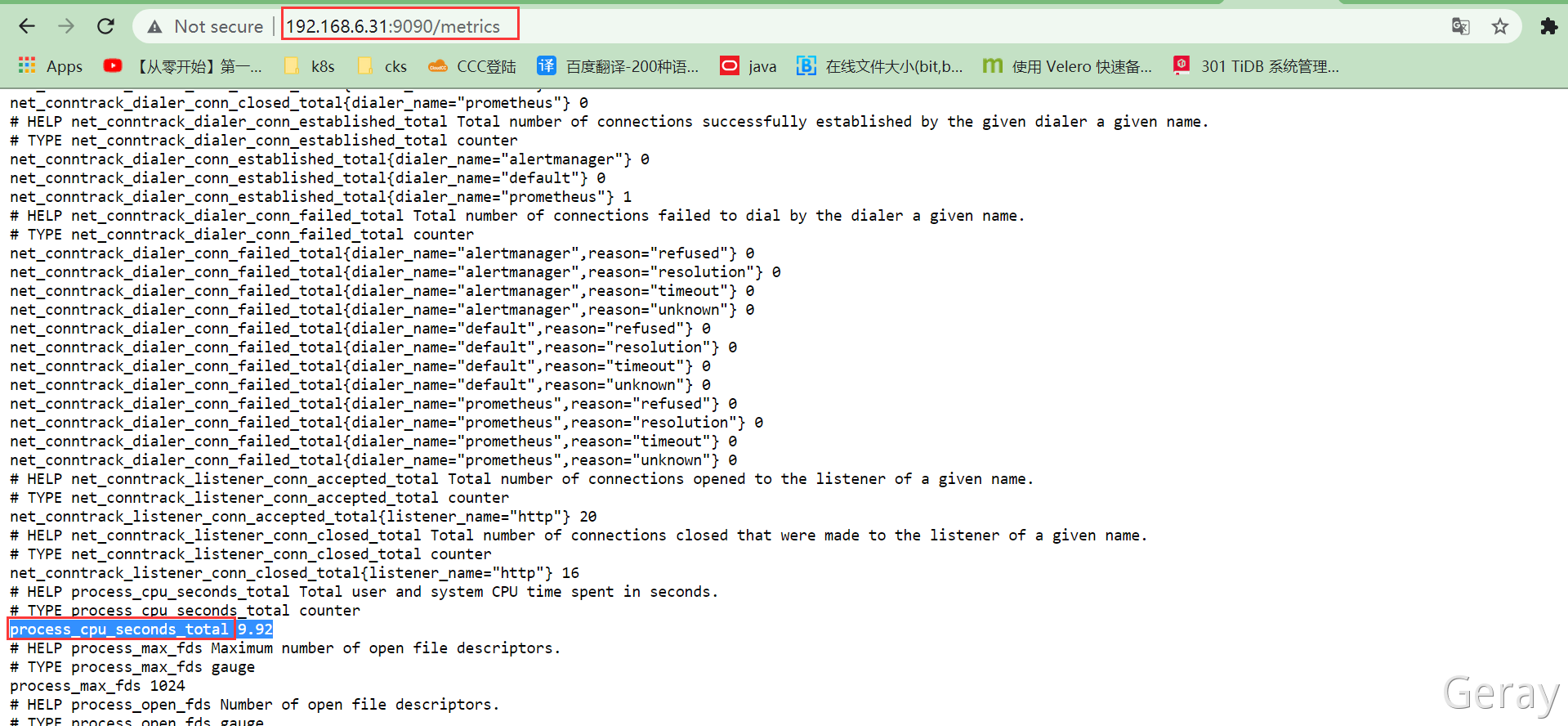
• 每个时间序列都由度量标准名称和一组键值对（称为标签）唯一标识，

通过标签查询指定指标。

指标格式：

<metric name>{<label name>=<label value>,...}



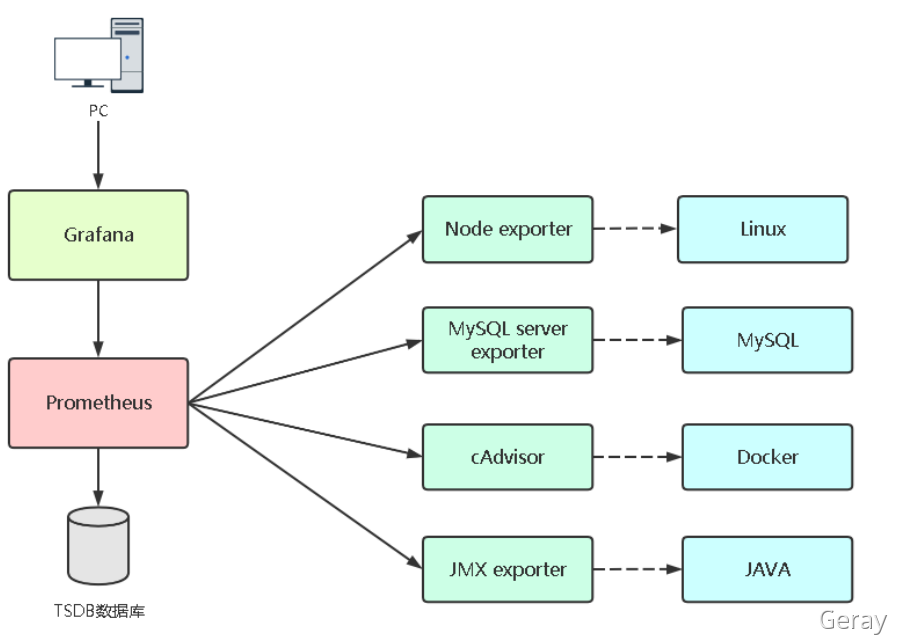


**2、监控案例**

**2.1 如何监控服务**

exporter列表：

<https://prometheus.io/docs/instrumenting/exporters>



如果要想监控，前提是能获取被监控端指标数据，并且这个数据格式必须遵循Prometheus数据模型，这样才能识别和采集，一般使用exporter提供监控指标数据。

**2.2 Grafana部署**

Grafana是一个开源的度量分析和可视化系统。

部署文档：<https://grafana.com/grafana/download>

访问地址：<http://IP:3000>

用户名/密码：admin/admin

* 第一次需要重置密码

配置为系统服务管理：

# vi /usr/lib/systemd/system/grafana.service

[Unit]

Description=grafana

[Service]

ExecStart=/opt/monitor/grafana/bin/grafana-server -homepath=/opt/monitor/grafana

ExecReload=/bin/kill -HUP $MAINPID

KillMode=process

Restart=on-failure

[Install]

WantedBy=multi-user.target

# systemctl daemon-reload

# systemctl start grafana

# systemctl enable grafana

Grafana只用于展示数据，但这个数据从哪里来？

需要你根据提供数据的服务选择，支持的数据源如下：（选择Prometheus，只需要指定URL即可）

**2.3 监控Linux服务**

使用文档：<https://prometheus.io/docs/guides/node-exporter/>

GitHub：<https://github.com/prometheus/node_exporter>

node\_exporter：用于监控Linux系统的指标采集器。

常用指标：

• CPU

• 内存

• 硬盘

• 网络流量

• 文件描述符

• 系统负载

• 系统服务

数据接口：http://IP:9100

**使用systemd服务管理：**

配置为系统服务管理：

# vi /usr/lib/systemd/system/node\_exporter.service

[Unit]

Description=node\_exporter

[Service]

ExecStart=/usr/local/node\_exporter/node\_exporter

ExecReload=/bin/kill -HUP $MAINPID

KillMode=process

Restart=on-failure

[Install]

WantedBy=multi-user.target

# systemctl daemon-reload

# systemctl start node\_exporter

# systemctl enable node\_exporter

**2.3.1 添加Prometheus配置**

# 添加监控

- job\_name: 'k8s-host'

metrics\_path: '/metrics'

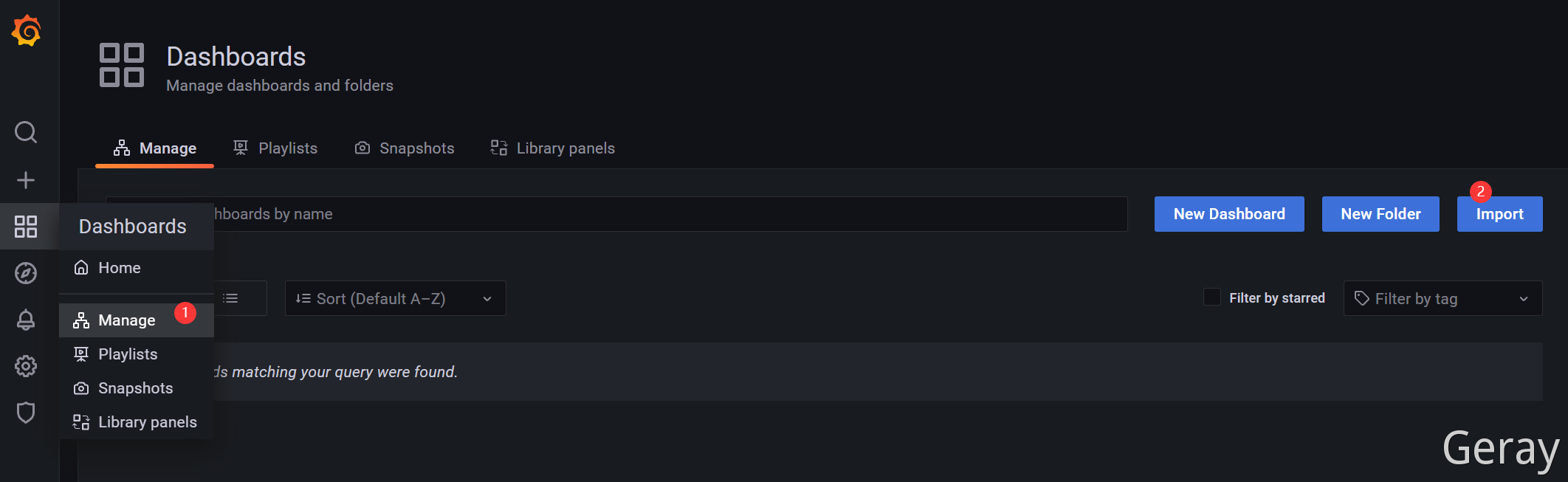
scheme: 'http'

static\_configs:

- targets: ['192.168.6.31:9100','192.168.6.32:9100']

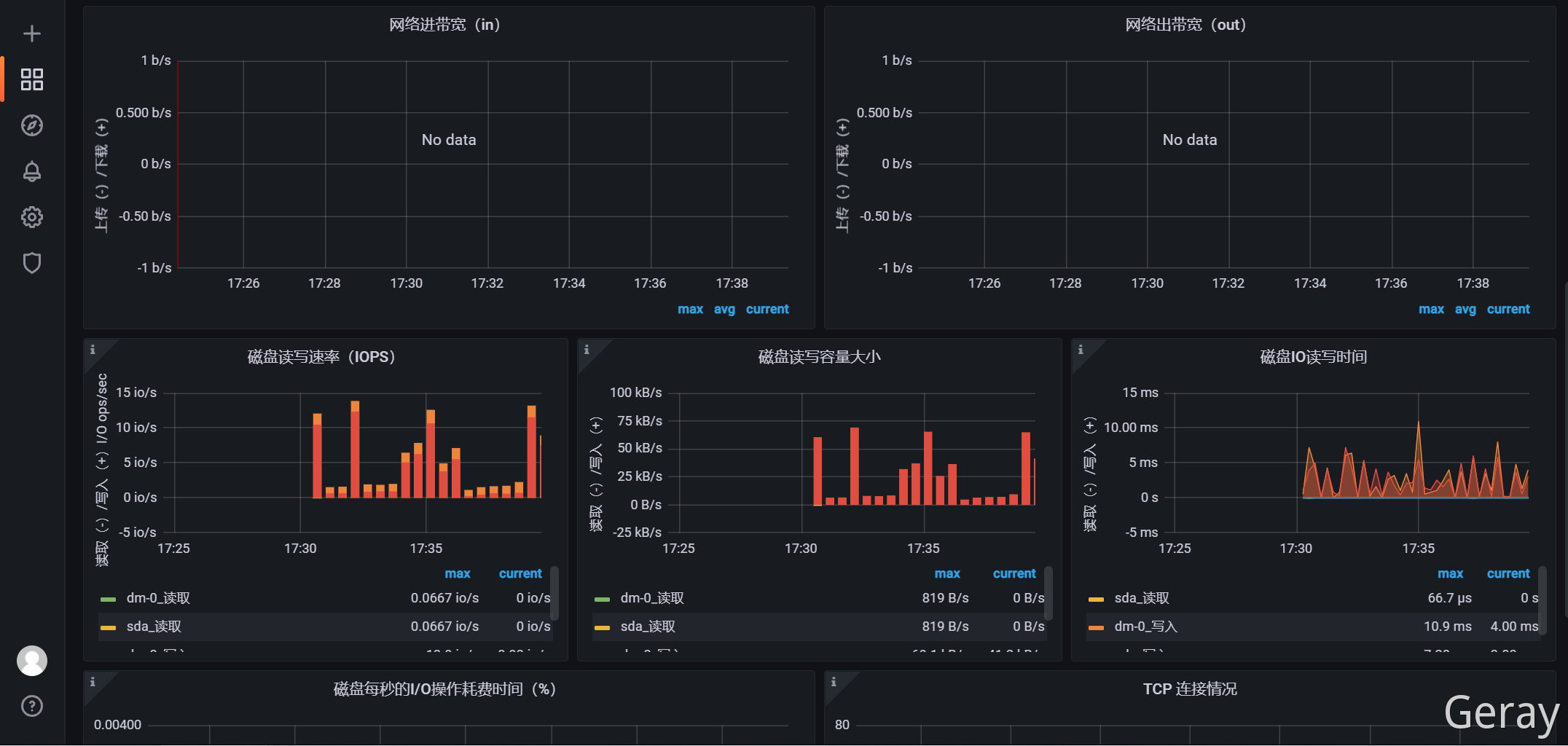
**2.3.2 添加grafana仪表盘（仪表盘ID： 9276）**

<https://grafana.com/grafana/dashboards/>



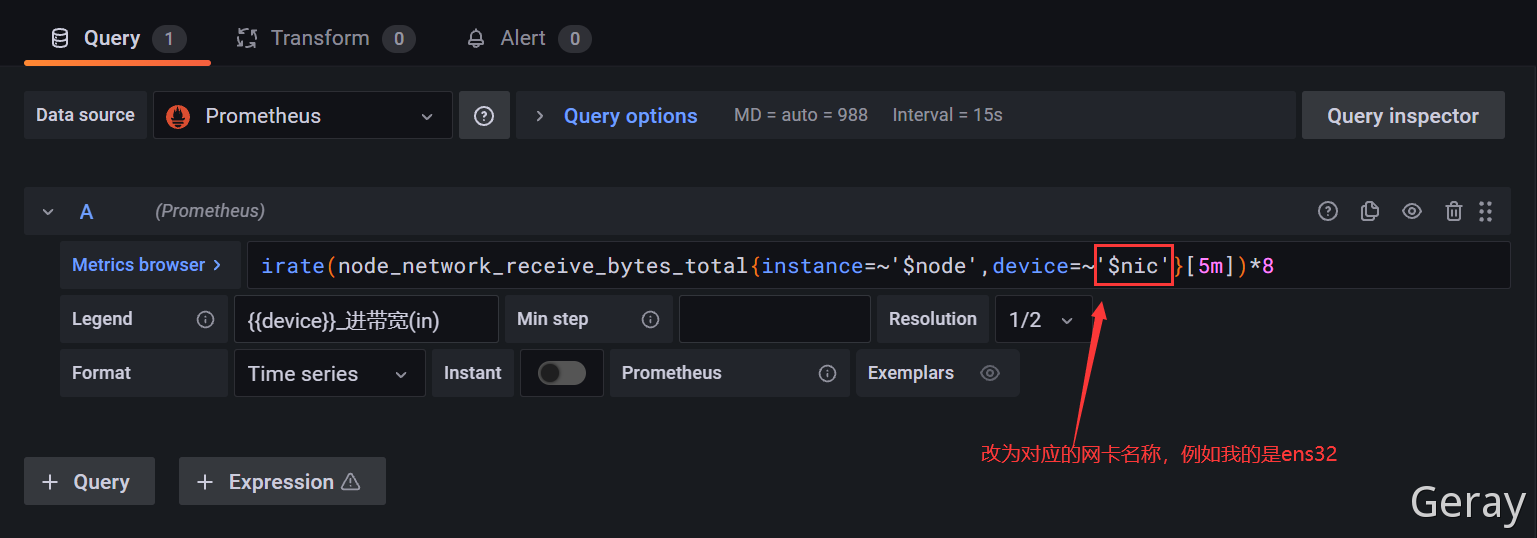
仪表盘ID： 9276

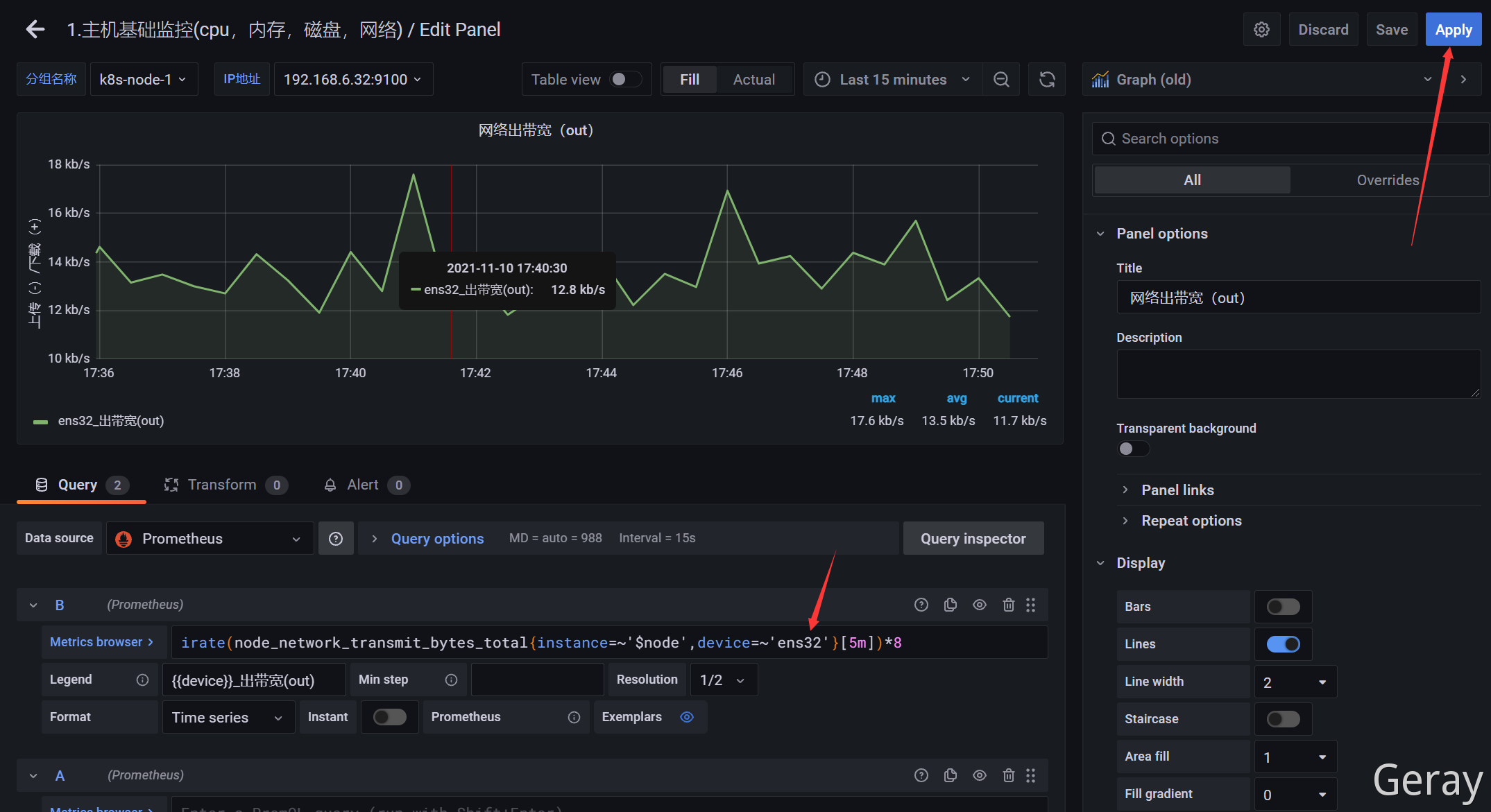




**2.3.3 网络无数据 - 监控指标的设备名称不合适**

例如我的是ens32网卡名称





**2.3.4 启用HTTP认证暴露指标接口**

加密：

# 安装加密工具

apt install apache2-utils -y

yum install httpd-tools –y

$ htpasswd -nBC 12 '' | tr -d ':\n'

New password:

Re-type new password:

$2y$12$8K7NFVCXqygXELVsX4mtqO2dHfoxb3CTkU2qkB/P5jAY7JgoGRLC2

* $2y$12$8K7NFVCXqygXELVsX4mtqO2dHfoxb3CTkU2qkB/P5jAY7JgoGRLC2 （123456加密后的内容）

# 启用HTTP认证

$ vi /usr/local/node\_exporter/config.yml

basic\_auth\_users:

# 用户名: 密码

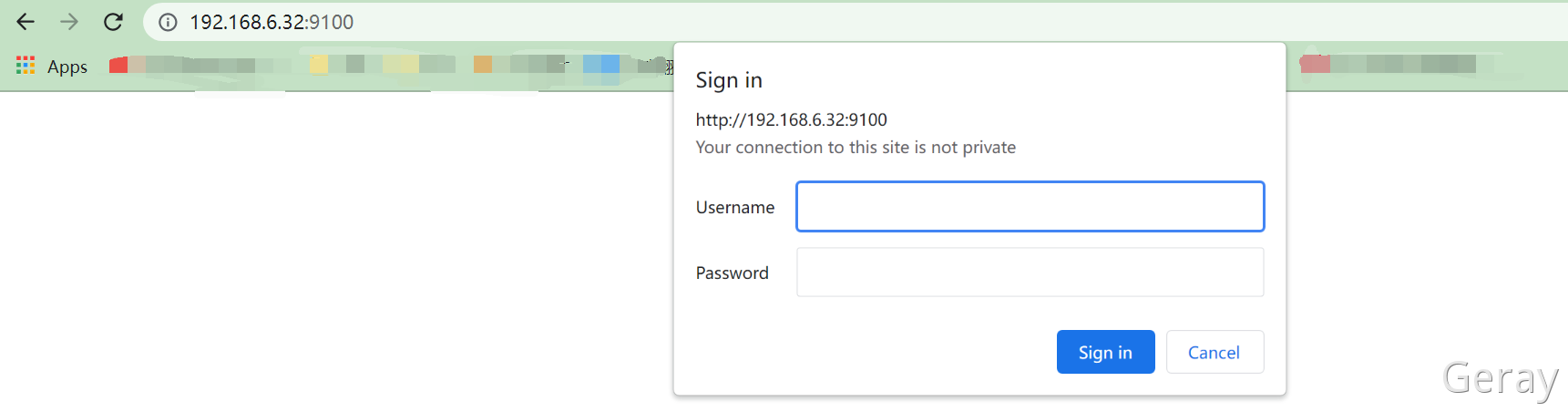
prometheus: $2y$12$8K7NFVCXqygXELVsX4mtqO2dHfoxb3CTkU2qkB/P5jAY7JgoGRLC2

指定ExecStart参数，使node\_export加载该配置：

ExecStart=/usr/local/node\_exporter/node\_exporter --web.config=/usr/local/node\_exporter/config.yml

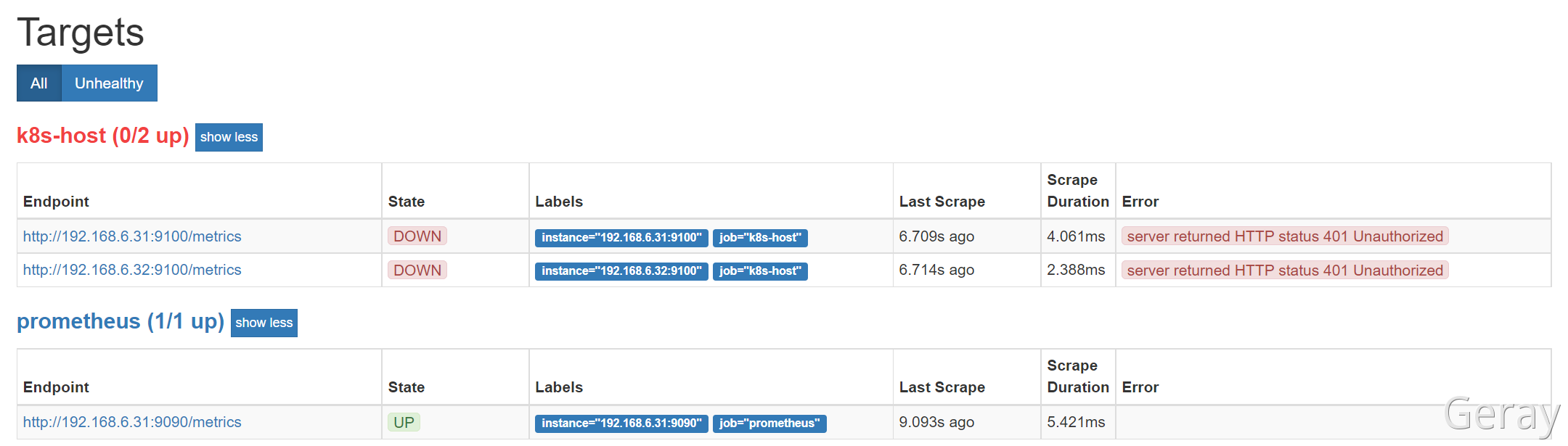
systemctl daemon-reload

systemctl restart node\_exporter.service



* 用户名密码就是config.yml配置文件中的（prometheus/123456）

**Prometheus监控添加认证**



- job\_name: 'k8s-host'

metrics\_path: '/metrics'

scheme: 'http'

basic\_auth: # 添加认证

username: prometheus

password: 123456

static\_configs:

- targets: ['192.168.6.31:9100','192.168.6.32:9100']

**2.4 监控监控系统服务运行状态**

使用node\_exporter，添加相关服务启动参数

# 可以使用一下命令查看参数

./node\_exporter --help

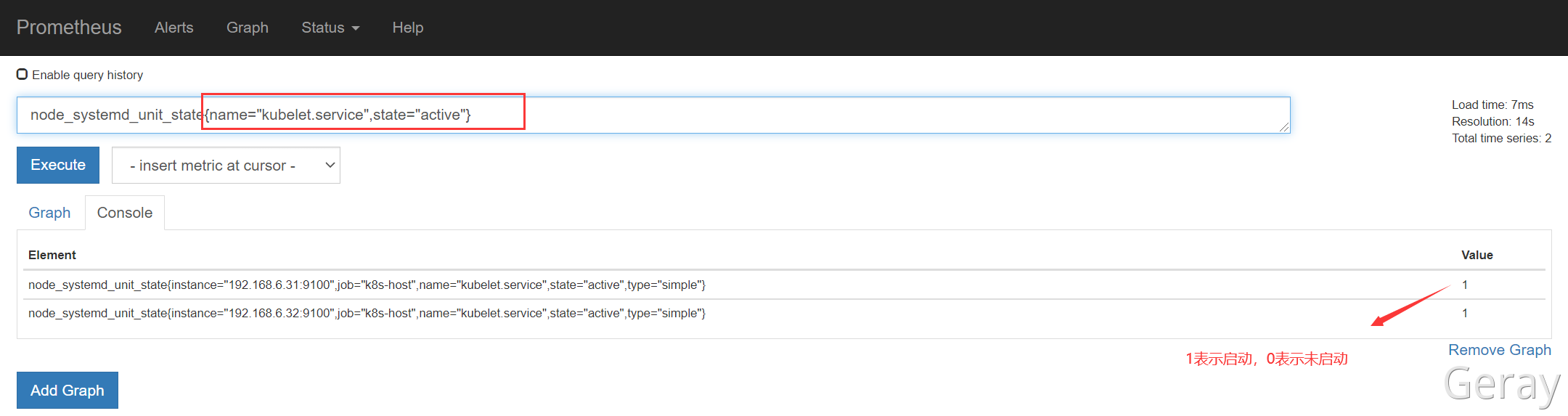
ExecStart=/usr/local/node\_exporter/node\_exporter --web.config=/usr/local/node\_exporter/config.yml --collector.systemd --collector.systemd.unit-whitelist=(docker|sshd|nginx|kubelet).service

使用PromSQL过滤查询想要的服务指标

# 获取服务名称是kubelet，状态为active的指标

node\_systemd\_unit\_state{name="kubelet.service",state="active"}

* 后面的Value值可以看到是否处于启动状态



**2.5 监控Docker服务**

项目地址：<https://github.com/google/cadvisor>

cAdvisor （Container Advisor） ：用于收集正在运行的容器资源使用和性能信息。

Docker部署cAdvisor：

docker run -d \

--volume=/:/rootfs:ro \

--volume=/var/run:/var/run:ro \

--volume=/sys:/sys:ro \

--volume=/var/lib/docker/:/var/lib/docker:ro \

--volume=/dev/disk/:/dev/disk:ro \

--publish=8080:8080 \

--detach=true \

--name=cadvisor \

google/cadvisor:latest

* 访问地址：ip:8080

**2.5.1 添加Prometheus配置**

- job\_name: 'docker'

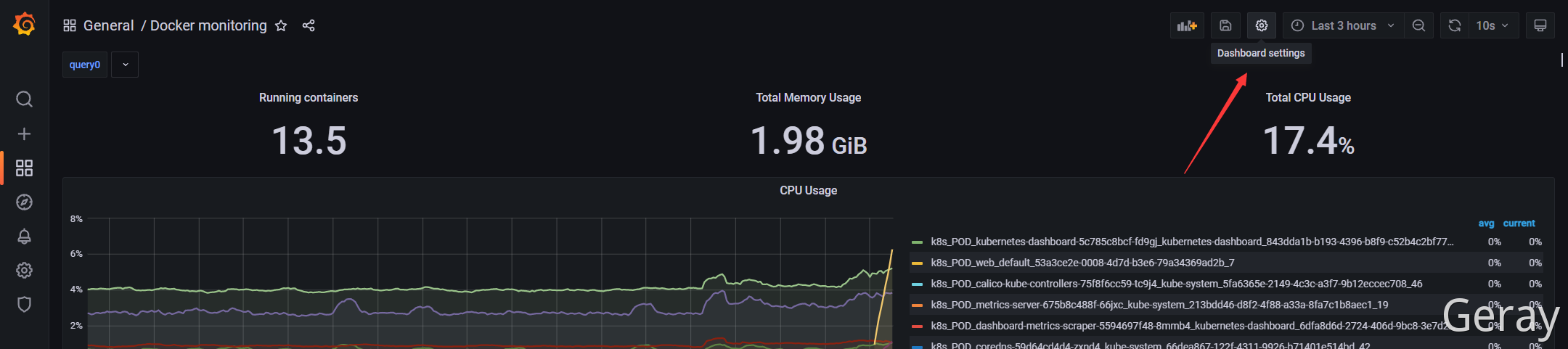
static\_configs:

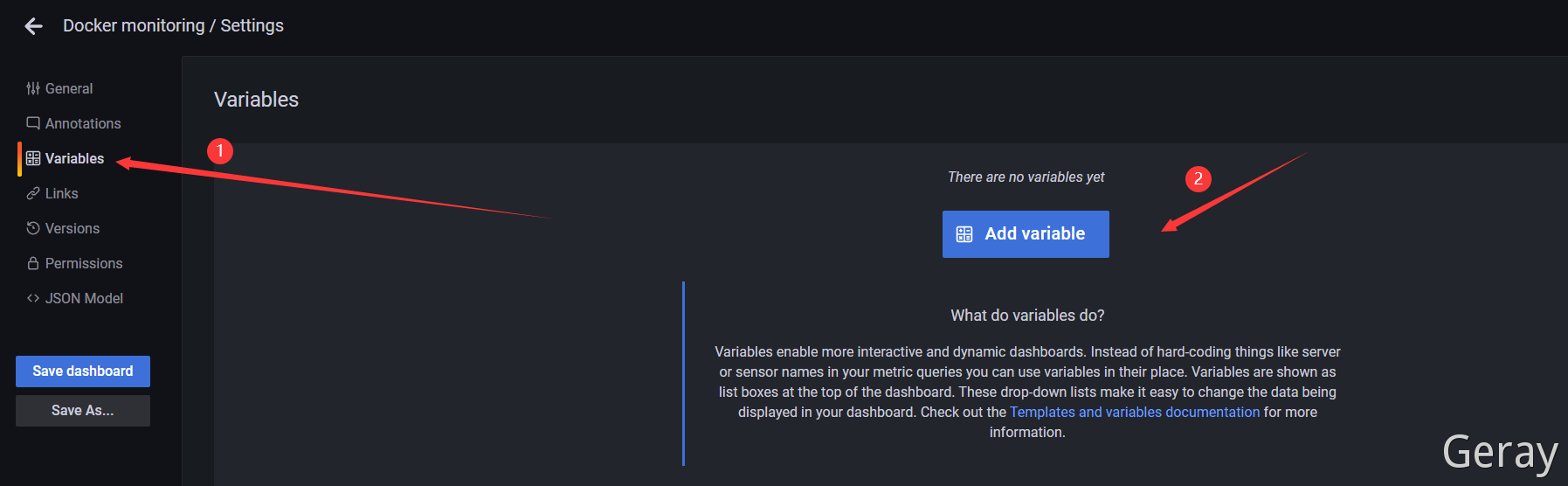
- targets: ['192.168.6.31:8080']

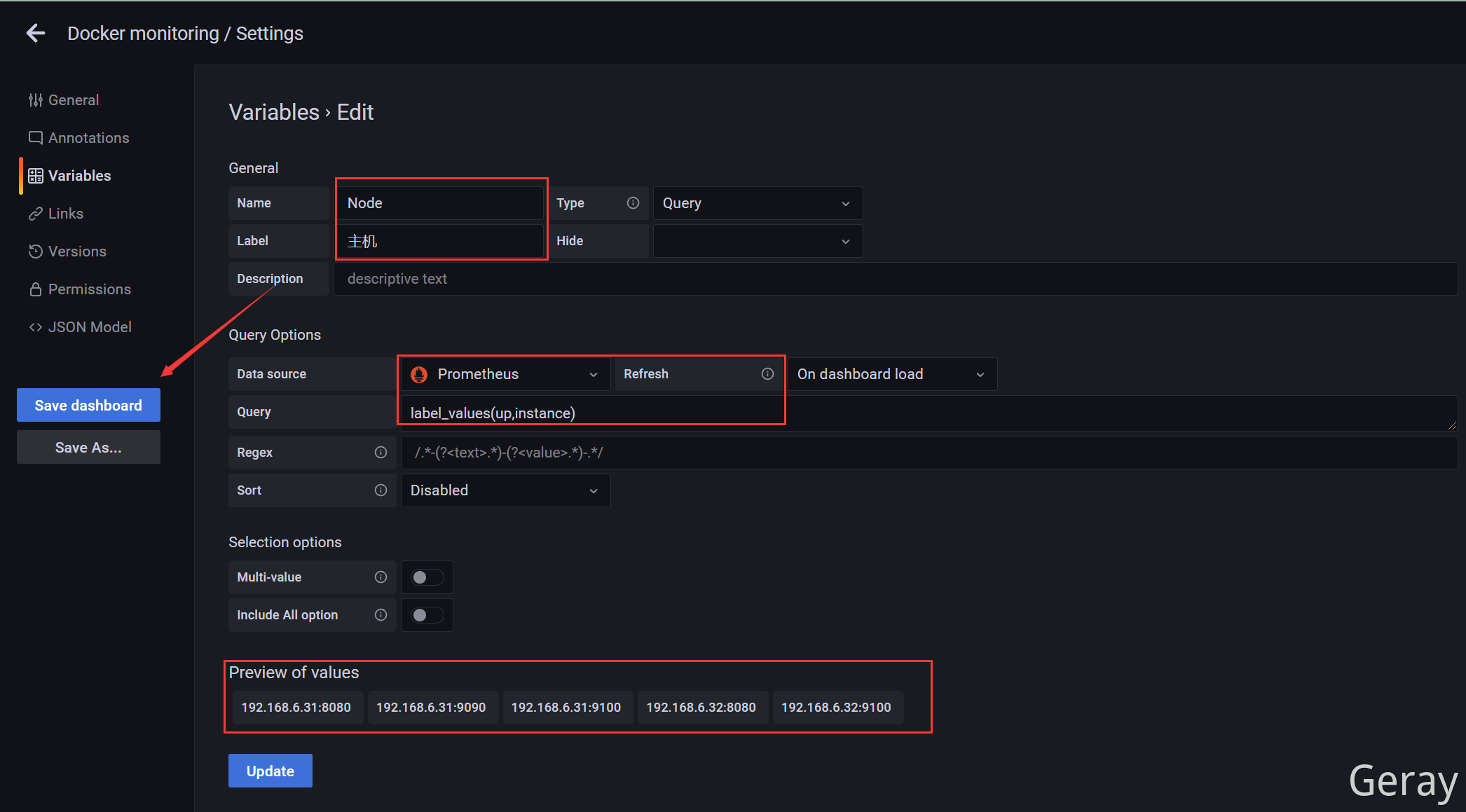
**2.5.2 添加grafana仪表盘（仪表盘ID：193）**

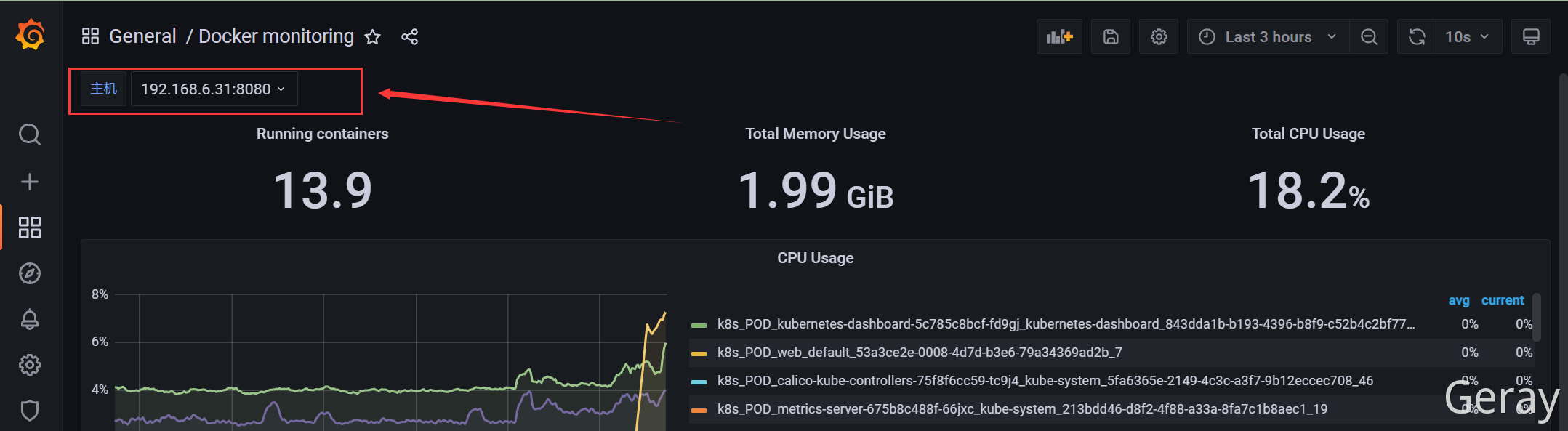


**2.5.3 添加多个docker导航栏**

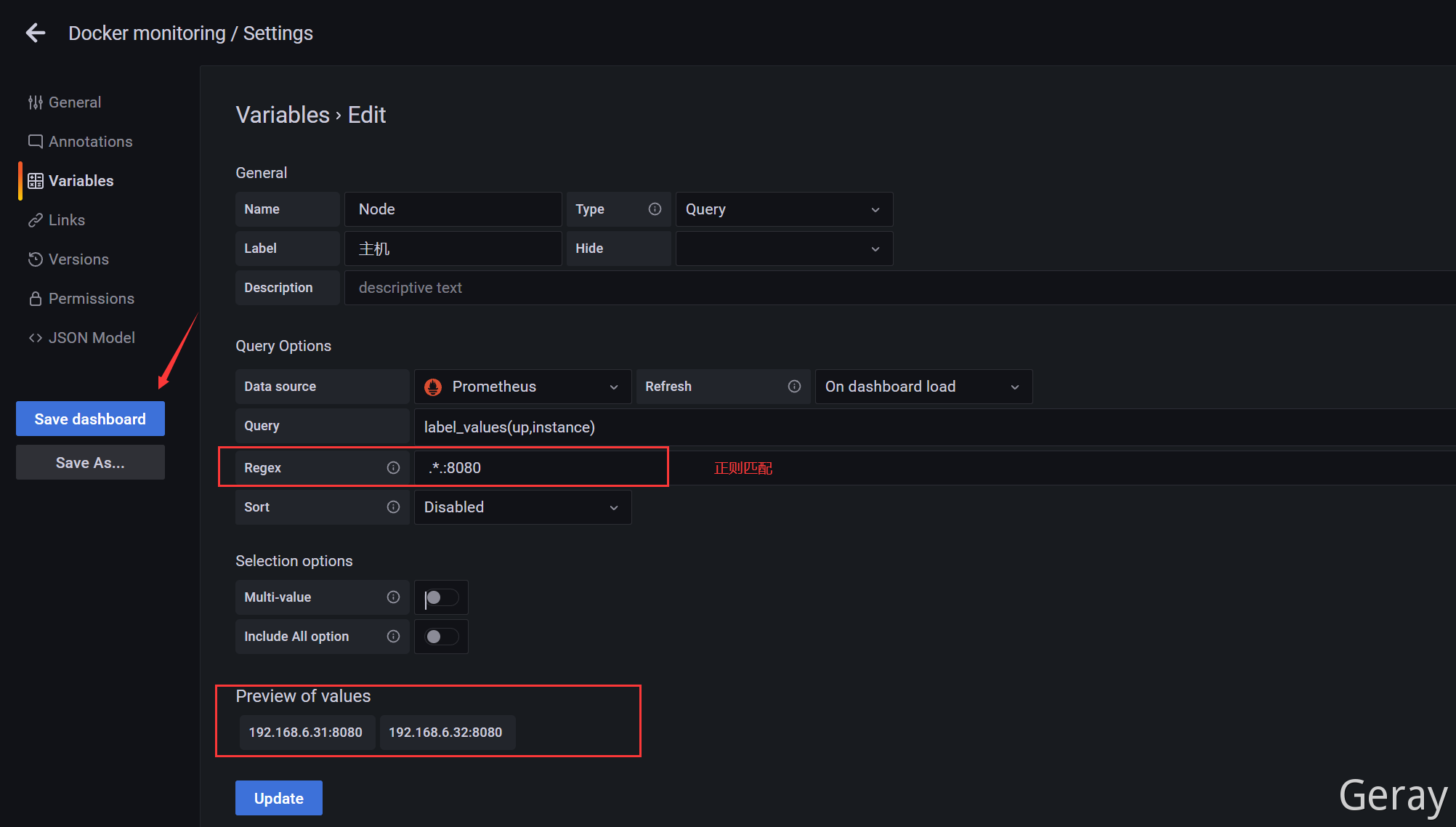








**使用正则只匹配对应监控指标**



**使监控对应的node指标，而不是所有的**

**添加过滤指标**



**2.6 监控MySQL服务**

项目地址：<https://github.com/prometheus/mysqld_exporter>

mysql\_exporter：用于收集MySQL性能信息。监听端口：9104

# docker run -d --name db -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 mysql:5.7

# docker exec -it db bash

mysql -uroot -p123456

# 设置exporter用户权限权限，查看进程，主从复制等

mysql> grant PROCESS, REPLICATION CLIENT, SELECT ON \*.\* to 'exporter'@'%' identified by '123456';

# 配置文件获取mysql服务信息

# vi /usr/local/mysql\_exporter/.my.cnf

[client]

user=exporter

password=123456

# ./mysqld\_exporter --config.my-cnf=.my.cnf

使用systemd服务管理：

# vi /etc/systemd/system//mysqld\_exporter.service

[Unit]

Description=mysqld\_exporter

[Service]

ExecStart=/usr/local/mysqld\_exporter/mysqld\_exporter --config.my-cnf="/usr/local/mysqld\_exporter/.my.cnf"

ExecReload=/bin/kill -HUP $MAINPID

KillMode=process

Restart=on-failure

[Install]

WantedBy=multi-user.target

**2.6.1 添加Prometheus配置**

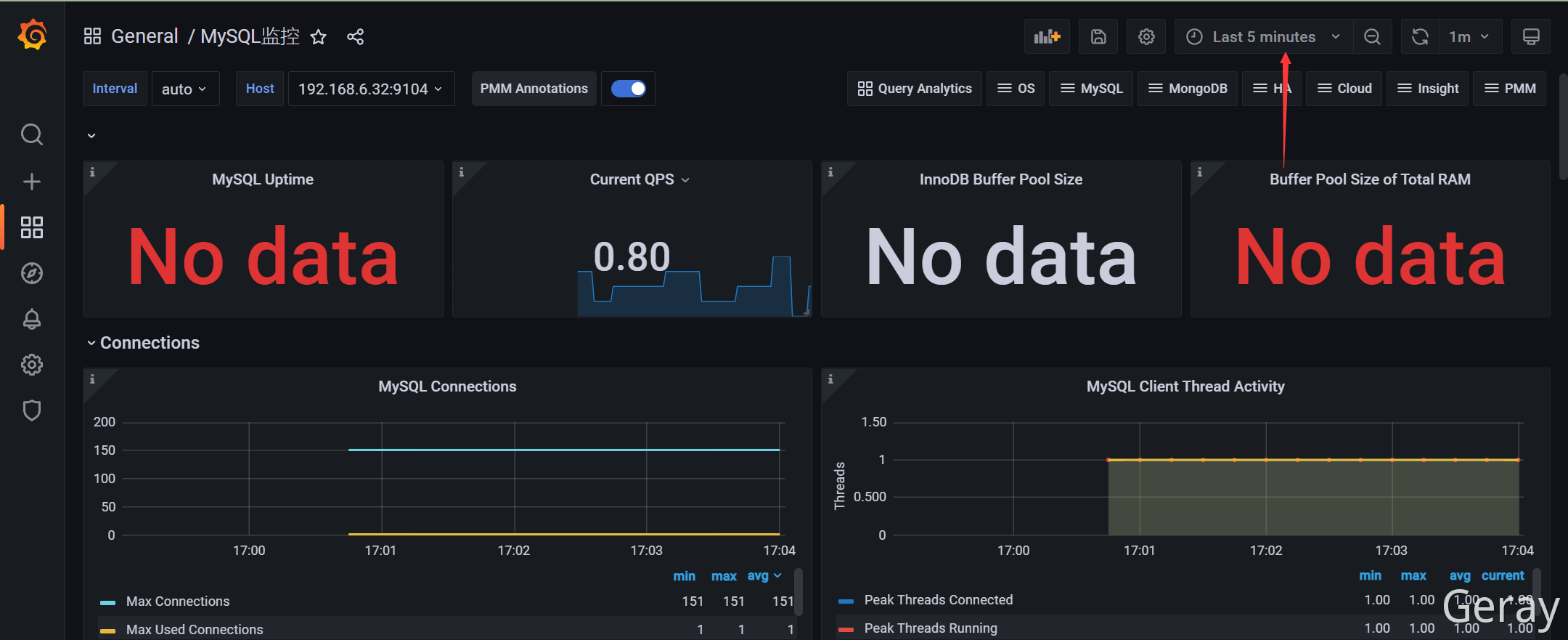
- job\_name: 'mysql'

static\_configs:

- targets: [''192.168.6.32:9104']

**2.6.2 添加grafana仪表盘（仪表盘ID：7362）**

* 设置时间，默认时间太长，可能无数据



**3、Prometheus自动化监控**

**3.1 Prometheus服务发现概述**

Prometheus添加被监控端支持两种方式：

* 静态配置：手动配置
* 服务发现：动态发现需要监控的Target实例

支持服务发现的来源：

• azure\_sd\_configs

• consul\_sd\_configs

• dns\_sd\_configs

• ec2\_sd\_configs

• openstack\_sd\_configs

• file\_sd\_configs

• gce\_sd\_configs

• kubernetes\_sd\_configs

• marathon\_sd\_configs

• nerve\_sd\_configs

• serverset\_sd\_configs

• triton\_sd\_configs

**3.2 基于文件的服务发现**

**3.2.1 添加Prometheus配置**

# 启用基于文件的服务发现：

- job\_name: 'file\_sd'

file\_sd\_configs:

- files: ['/opt/monitor/prometheus/sd\_config/\*.yml']

refresh\_interval: 5s # 每隔5秒检查一次

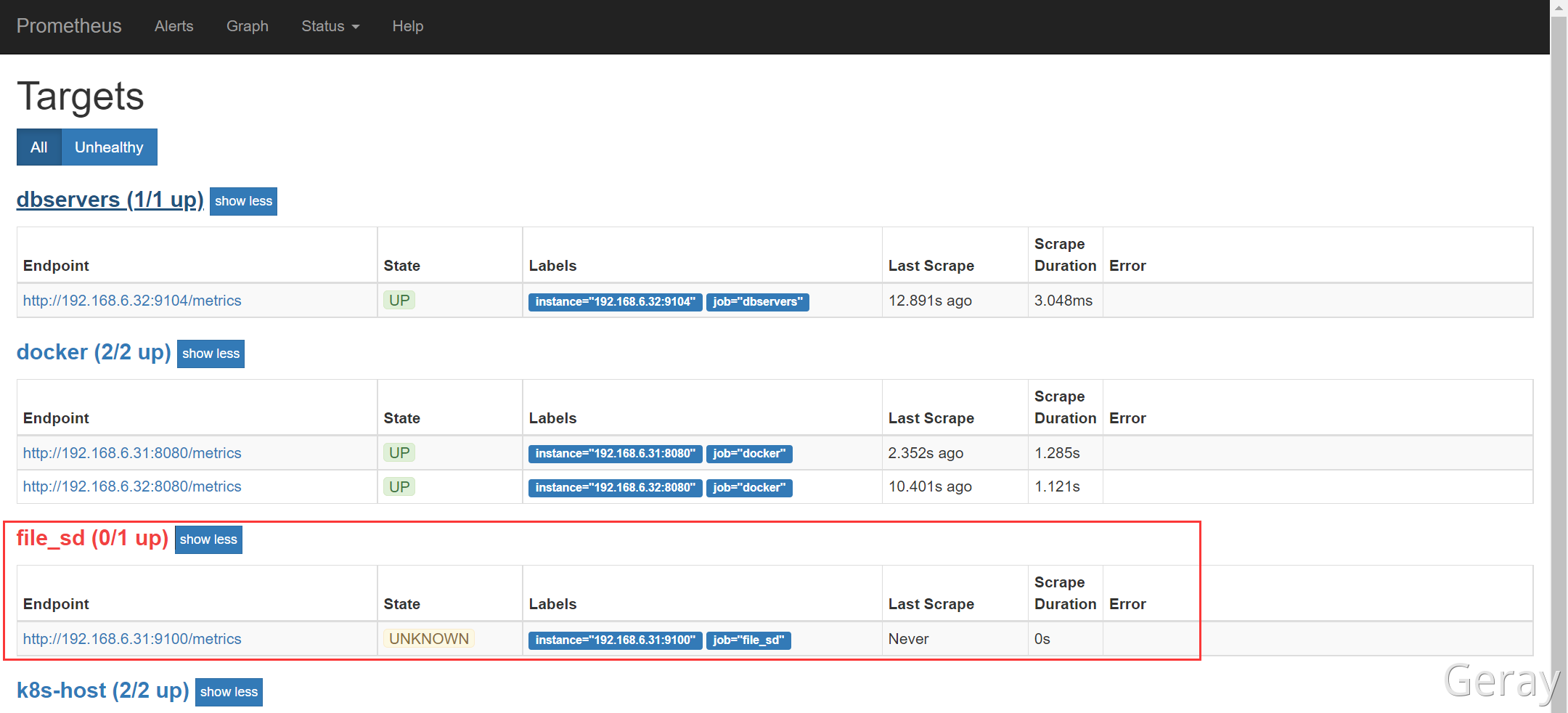
**3.2.2 添加被监控端**

添加被监控端：

cd /opt/monitor/prometheus/sd\_config/

vi test.yml

- targets: ['192.168.6.31:9100']



* 由于该服务使用了用户名密码认证，所有获取不到数据可以添加认证信息

- job\_name: 'file\_sd'

basic\_auth: # 添加认证

username: prometheus

password: 123456

file\_sd\_configs:

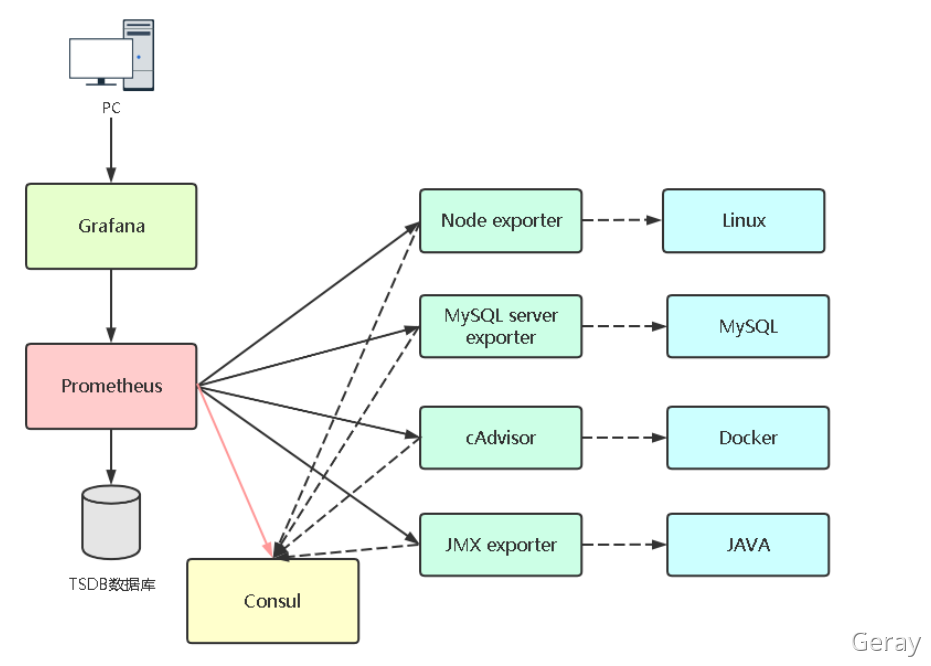
- files: ['/opt/monitor/prometheus/sd\_config/\*.yml']

refresh\_interval: 5s # 每隔5秒检查一次



**3.3 基于consul的服务发现**

Consul是一个分布式的服务发现和键/值存储系统。



**3.3.1 部署consul**

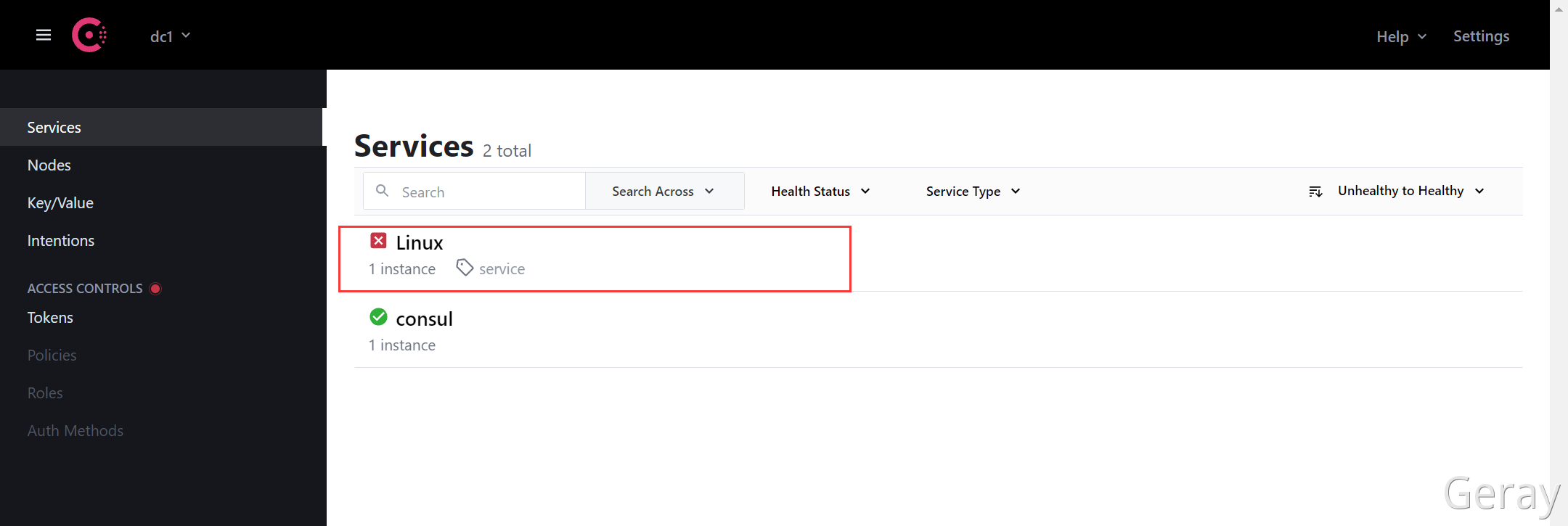
Docker部署Consul：

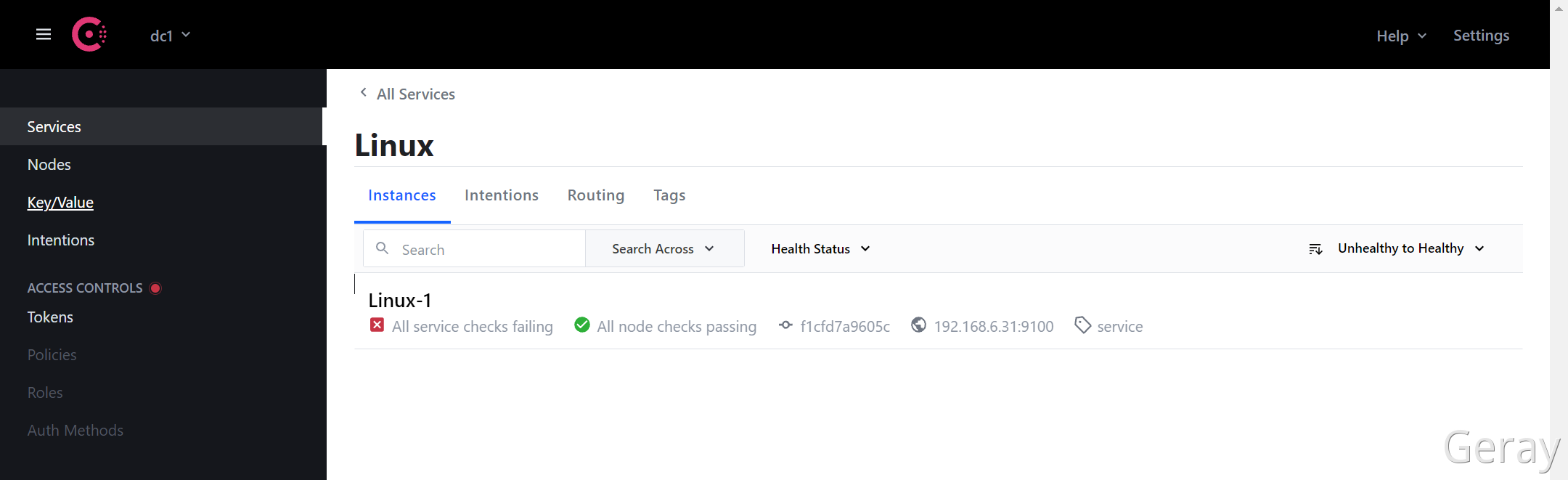
docker run --name consul -d -p 8500:8500 consul

向Consul注册服务：

curl -X PUT -d '{"id": "Linux-1","name": "Linux","address": "192.168.6.31","port": 9100,"tags": ["service"],"checks": [{"http": "http://192.168.6.31:9100","interval": "5s"}]}' http://192.168.6.31:8500/v1/agent/service/register

* id 相当于Prometheus配置文件中的一个实例，也就是一个监控单元
* name 相当于一个job，也就是分组
* tags 也就是注册的一个服务标签
* checks 健康检查
* 最后是一个借口地址





* 由于9100启用了认证所以显示红叉

**3.3.2 添加Prometheus配置 - 启用consul服务发现**

启用基于consul的服务发现：

- job\_name: 'consul\_sd'

basic\_auth: # 添加认证

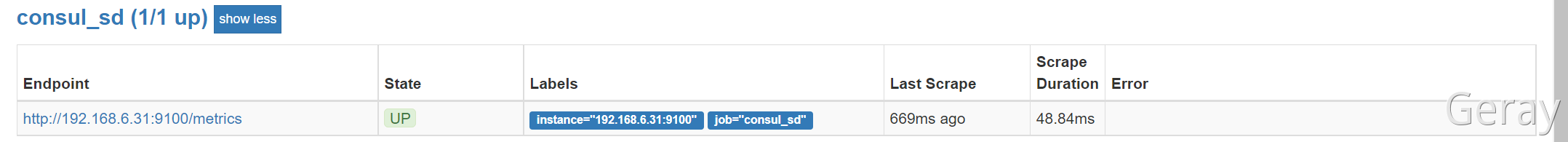
username: prometheus

password: 123456

consul\_sd\_configs:

- server: 192.168.6.31:8500

services: ['Linux']

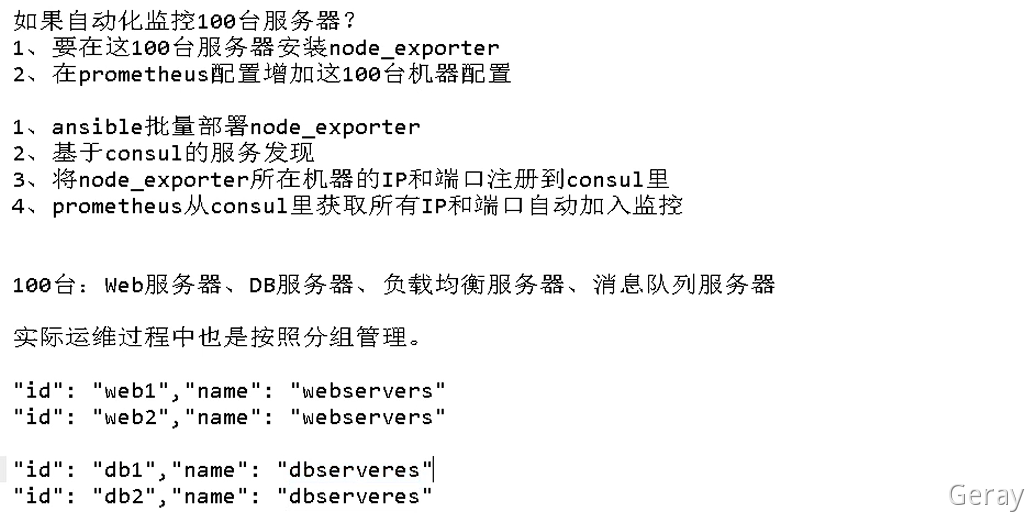


添加Linux-2

curl -X PUT -d '{"id": "Linux-2","name": "Linux","address": "192.168.6.32","port": 9100,"tags": ["service"],"checks": [{"http": "http://192.168.6.32:9100","interval": "5s"}]}' http://192.168.6.31:8500/v1/agent/service/register

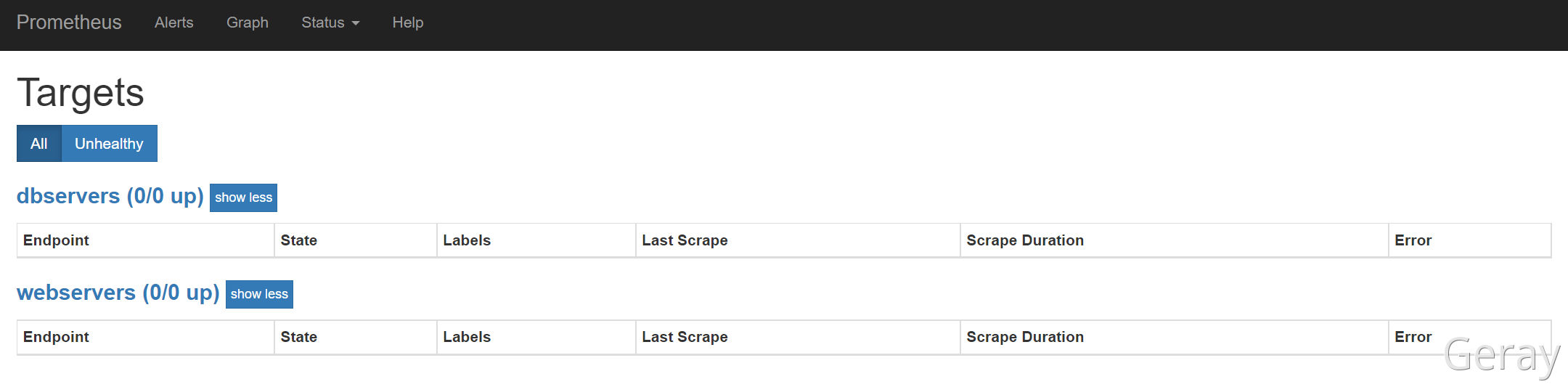


**3.4 Ansible+Consul实现100台主机自动监控**





*注：环境已清理*



修改Prometheus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: 'webservers'

consul\_sd\_configs:

- server: 192.168.6.31:8500

services: ['webservers']

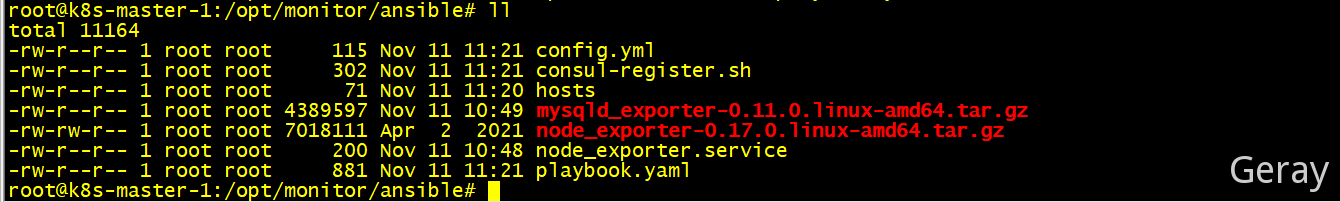
- job\_name: 'dbservers'

consul\_sd\_configs:

- server: 192.168.6.31:8500

services: ['dbservers']

**3.4.1 准备环境清单**



清理之前环境

systemctl disable node\_exporter.service

systemctl stop node\_exporter.service

rm -rf /etc/systemd/system/node\_exporter.service

rm -rf /usr/local/node\_exporter

systed服务管理node\_exporter

vi /opt/monitor/ansible/node\_exporter.service

[Unit]

Description=node\_exporter

[Service]

ExecStart=/usr/local/node\_exporter/node\_exporter

ExecReload=/bin/kill -HUP $MAINPID

KillMode=process

Restart=on-failure

[Install]

WantedBy=multi-user.target

vi mysqld\_exporter.service

[Unit]

Description=mysqld\_exporter

[Service]

ExecStart=/usr/local/mysqld\_exporter/mysqld\_exporter --config.my-cnf="/usr/local/mysqld\_exporter/.my.cnf"

ExecReload=/bin/kill -HUP $MAINPID

KillMode=process

Restart=on-failure

[Install]

WantedBy=multi-user.target

主机清单

vi /opt/monitor/ansible/hosts

[webservers]

192.168.6.31 name=web1

[dbservers]

192.168.6.32 name=db1

* 两个组中有相同的ip会导致出现问题

consul-register.sh

vi /opt/monitor/ansible/consul-register.sh

#!/bin/bash

service\_name=$1

instance\_id=$2

ip=$3

port=$4

echo $service\_name

echo $instance\_id

echo $ip

echo $port

curl -X PUT -d '{"id": "'"$instance\_id"'","name": "'"$service\_name"'","address": "'"$ip"'","port": '"$port"',"tags": ["service"],"checks": [{"http": "http://'"$ip"':'"$port"'","interval": "5s"}]}' http://192.168.6.31:8500/v1/agent/service/register

认证文件

vi /opt/monitor/ansible/config.yml

basic\_auth\_users:

# 用户名: 密码

prometheus: $2y$12$8K7NFVCXqygXELVsX4mtqO2dHfoxb3CTkU2qkB/P5jAY7JgoGRLC2

playbook.yaml

vi /opt/monitor/ansible/playbook.yaml

- hosts: webservers

# remote\_user: root

gather\_facts: no

# 定义变量

vars:

port: 9100

tasks:

- name: 推送二进制文件

unarchive: src=node\_exporter-0.17.0.linux-amd64.tar.gz dest=/usr/local

- name: 重命名

shell: |

cd /usr/local/

if [ ! -d node\_exporter ];then

mv node\_exporter-0.17.0.linux-amd64 node\_exporter

fi

# - name: 推送认证配置文件

# copy: src=config.yml dest=/usr/local/node\_exporter

- name: 推送systemd服务管理配置

copy: src=node\_exporter.service dest=/etc/systemd/system/

- name: 启动服务

systemd: name=node\_exporter state=started enabled=yes daemon\_reload=yes

- name: 推送注册脚本

copy: src=consul-register.sh dest=/usr/local/bin/

- name: 注册当前节点

# 服务名 实例名 ip port

shell: /bin/bash /usr/local/bin/consul-register.sh {{ group\_names[0] }} {{ name }} {{ inventory\_hostname }} {{ port }}

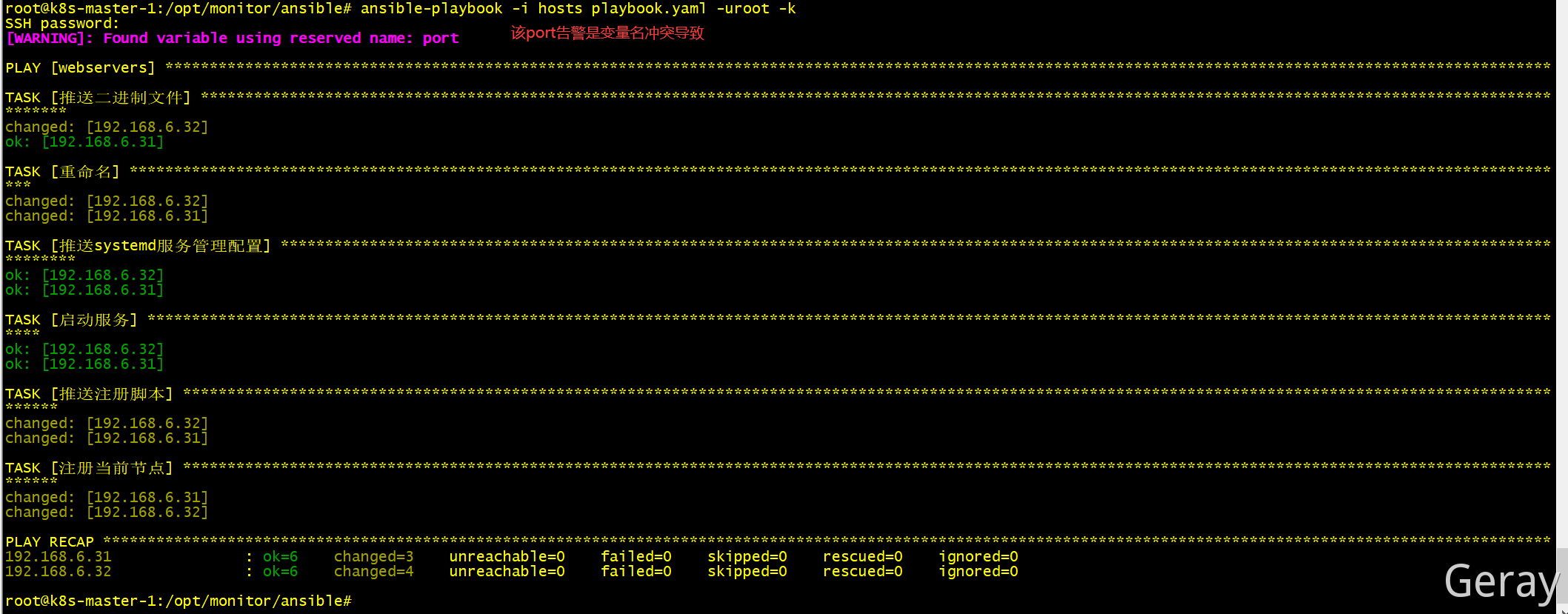
* mysqld\_exporter-0.11.0.linux-amd64.tar.gz 暂未使用
* {{ group\_names[0] }} 返回的是- hosts: webservers值
* {{ name }} 返回的是hosts清单中的name
* {{ inventory\_hostname }} 当前的ip
* {{ port }} 使用的是变量port

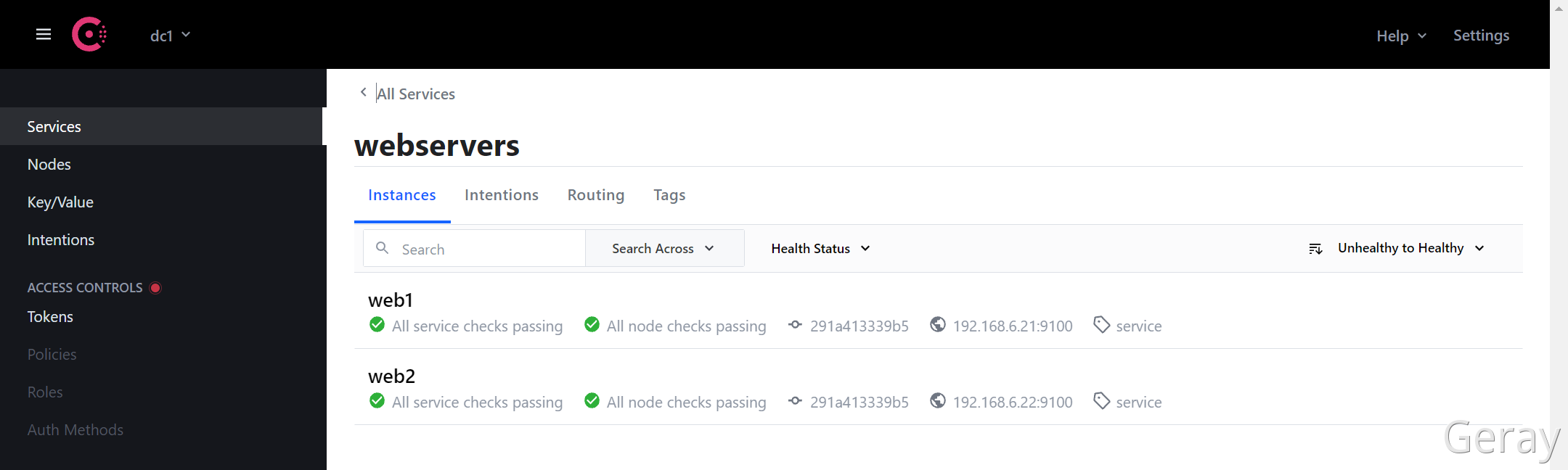
**3.4.2 检查playbook清单，并执行**

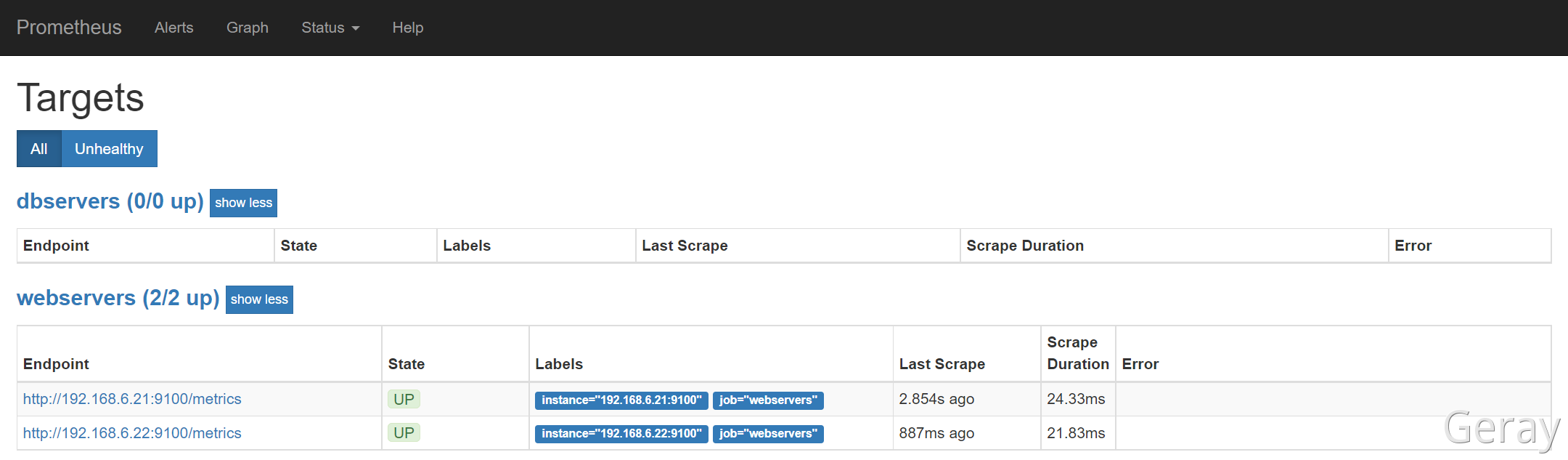
ansible-playbook -v --syntax-check playbook.yaml -i hosts

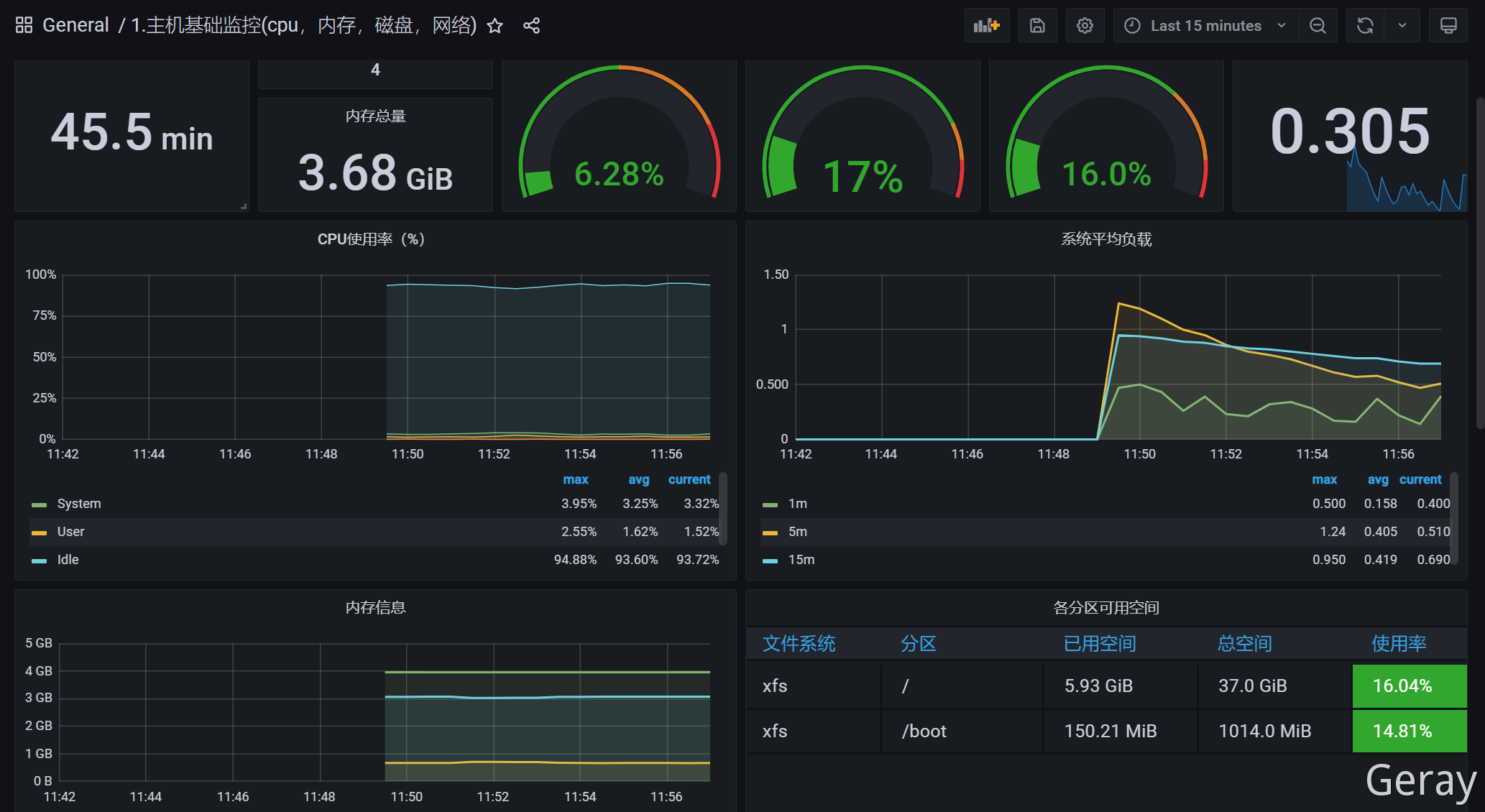
执行

ansible-playbook -i hosts playbook.yaml -uroot -k









**3.5 监控kubernetes**

* ubuntu测试没有添加正确的token.k8s导致找不到监控指标

参考文档：<https://prometheus.io/docs/prometheus/latest/configuration/configuration/#kubernetes_sd_config>

**3.5.1 监控k8s概述**

**监控指标**

Kubernetes本身监控

• Node资源利用率

• Node数量

• Pods数量（Node）

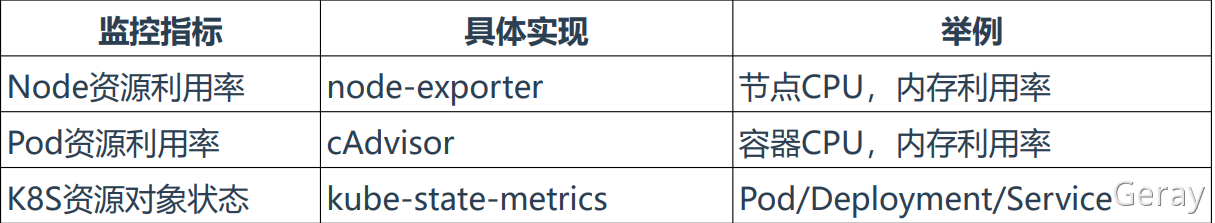
• 资源对象状态

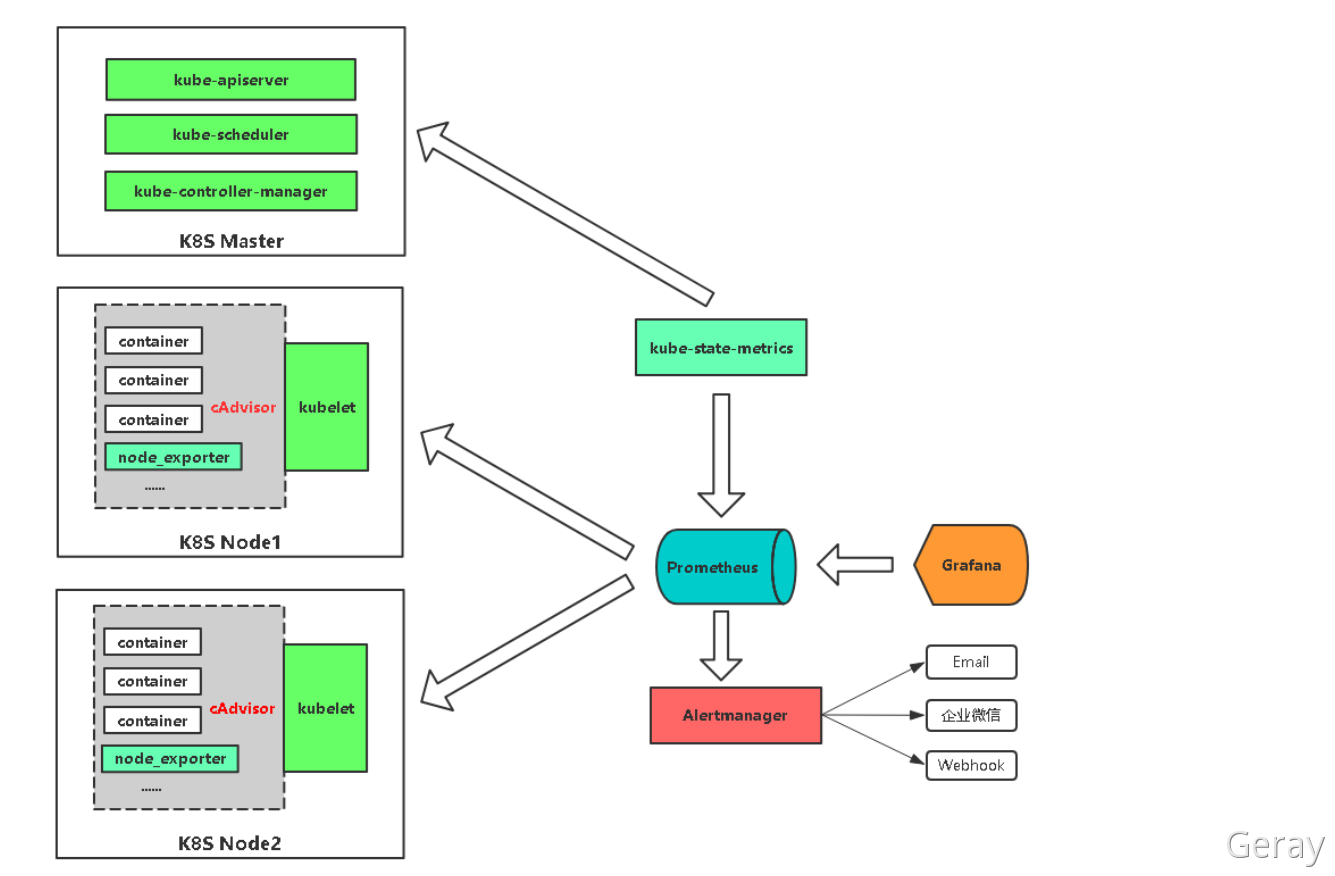
Pod监控

• Pod数量（项目）

• 容器资源利用率

• 应用程序







参考文档：<https://prometheus.io/docs/prometheus/latest/configuration/configuration/#kubernetes_sd_config>

**3.5.2 配置k8s服务发现 - 监控pod**

**1）授权**

[📎rbac.yaml](https://www.yuque.com/attachments/yuque/0/2021/yaml/8425422/1636699742637-c43bf75b-88b7-41b9-b3f0-556738c194f6.yaml)

kubectl apply -f rbac.yaml

**2）获取Token并保存到文件**

kubectl get sa prometheus -n kube-system -o yaml

kubectl describe secret <prometheus-token-xxx | secret名称> -n kube-system

# 例如

kubectl describe secret prometheus-token-ghtr9 -n kube-system | grep token: | awk '{print $2}'

* 将结果保存到指定文件中，token.k8s（Prometheus需要该token获取信息）

3、创建Job和kubeconfig\_sd\_configs - 添加Prometheus配置

[📎prometheus.yml](https://www.yuque.com/attachments/yuque/0/2021/yml/8425422/1636700438924-503fa883-7ca9-4a71-988d-f8b9629e79e0.yml)

4、Grafana导入仪表盘

[📎K8S集群资源监控-2020.json](https://www.yuque.com/attachments/yuque/0/2021/json/8425422/1636710309479-957ff891-b9b4-49b8-87e8-9db157a51af1.json)

**3.5.3 配置k8s服务发现 - 监控资源对象**

**1、部署kube-state-metrics**

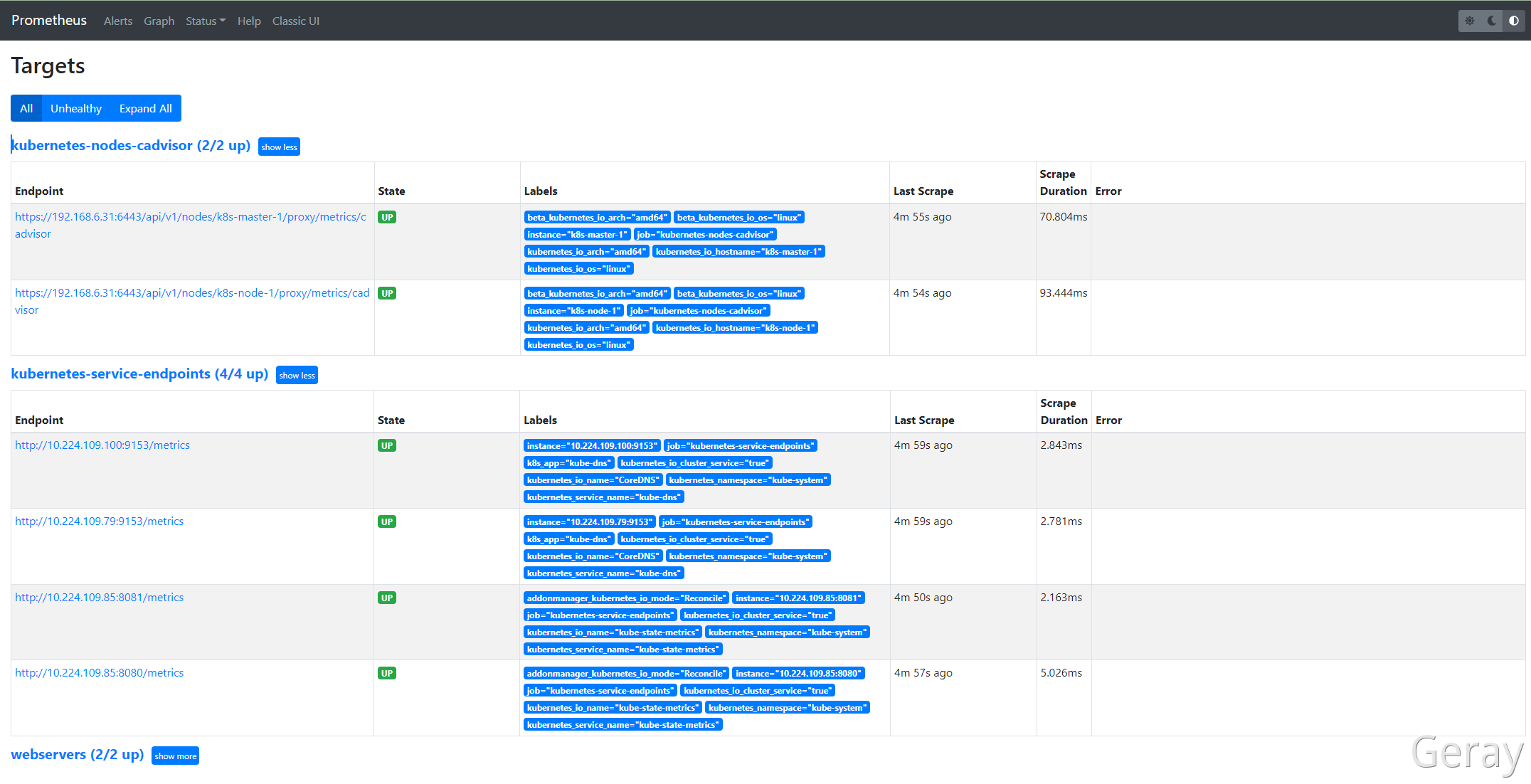
[📎kube-state-metrics.yaml](https://www.yuque.com/attachments/yuque/0/2021/yaml/8425422/1636710284421-2fb76eac-64e1-47b3-96a5-e84e56560795.yaml)

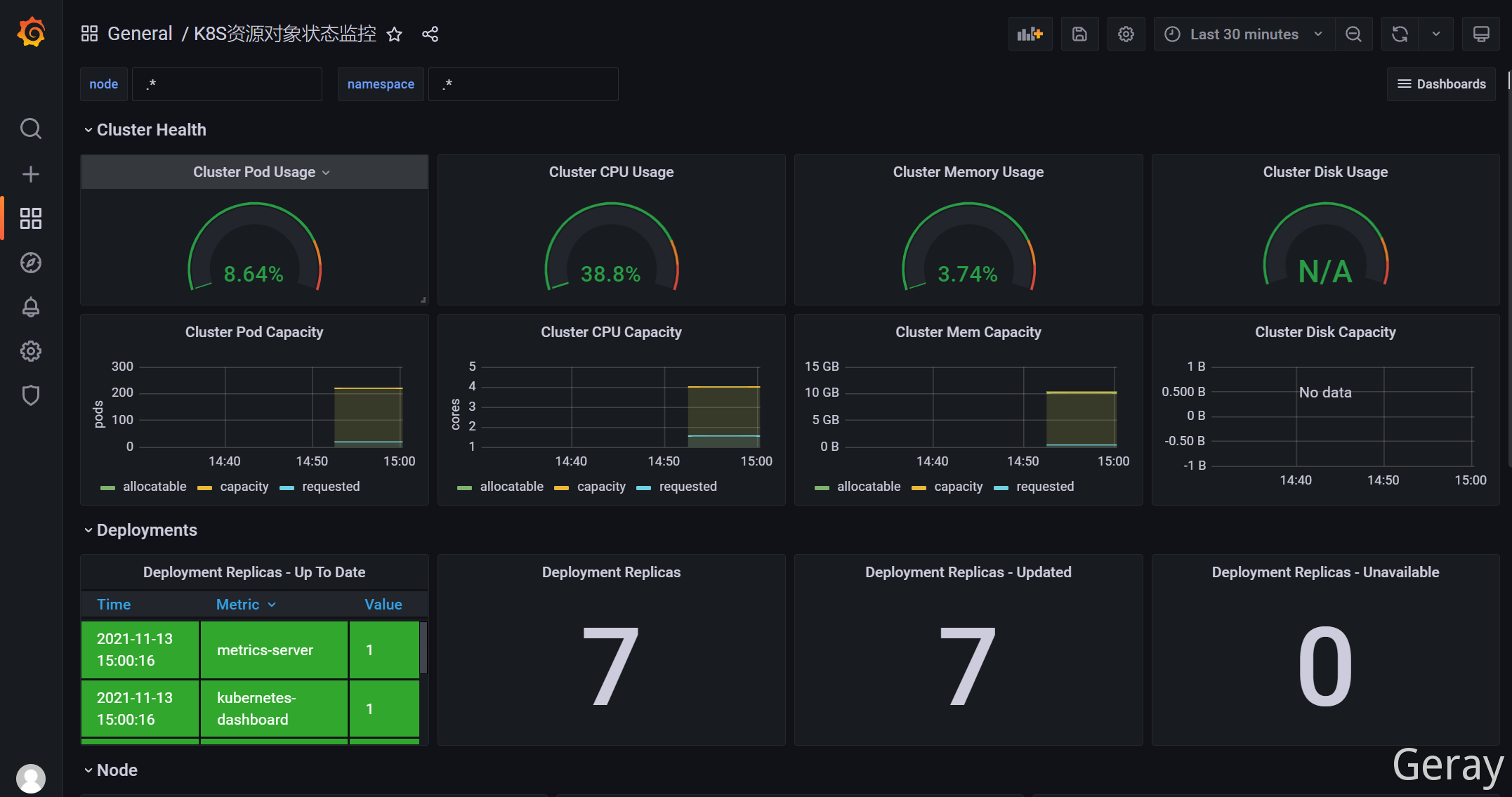
curl 10.224.109.84:8080/metrics 检查是否可以被监听到

**2、添加prometheus配置**

[📎prometheus.yml](https://www.yuque.com/attachments/yuque/0/2021/yml/8425422/1636786587164-9dbfc293-e287-4d40-8924-0857442e00bf.yml)

**3、Grafana导入仪表盘**[📎K8S资源对象状态监控-2020.json](https://www.yuque.com/attachments/yuque/0/2021/json/8425422/1636710321732-7146dfda-7b0f-498c-a56f-d7f8ab6474c4.json)



****

**3.5.4 集群外部署问题解决**

* 以上Prometheus都是在k8s集群中部署的，所以不存在访问pod内部ip问题
* 集群之外部署会存在PodIP无法获取问题

**1、开启路由转发**

* 这里是同网段，可以直接添加路由表
* 如果是跨网段，需要在上层添加路由表

# Prometheus节点添加路由表

ip route add 10.244.0.0/16 via 192.168.31.71 dev ens33

# 查看添加的路由表

ip route

* 192.168.31.71 是k8s集群master地址

