**Prometheus+Grafana+Es**

**监控F5设备及业务状态**

**实验记录**

版本:1.0

日期:2022-11-21

目录

目录 I

版本控制信息 III

第1章 容器化部署 4

1.1 Prometheus部署 4

1.2 Grafana部署 8

1.3 snmp-exporter部署 10

1.4 ES部署 12

1.5 Kibana部署 17

1.6 Logstash部署 20

1.7 node-exporter部署 35

1.8 cadvisor部署 36

1.9 portainer部署 37

第2章 Grafana初始化配置 39

2.1 添加数据源 39

2.1.1 添加pormetheus数据源： 39

2.1.2 添加Elasticsearch数据源： 41

2.2 导入Dashboard模板 43

2.2.1 导入Prometheus模板 43

2.2.2 导入Elasticsearch模板 44

2.2.3 模板对应的数据源 45

第3章 Prometheus监控F5设备 46

3.1 生成Snmp.yml文件 46

3.2 Prometheus配置Snmp监控 50

3.2.1 Prometheus.yml配置 50

3.2.2 Mib信息查看 52

3.3 Grafana可视化 53

3.3.1 Dashboard模板介绍 53

3.3.2 Grafana视图制作 55

3.3.2.1 Dashboard变量设置 55

3.3.2.2 Dashboard Panel制作示例 60

第4章 Grafana展示ES视图 63

4.1.1 Dashboard模板介绍 63

4.1.2 Grafana视图制作 65

4.1.2.1 Dashboard变量设置 65

4.1.2.2 Dashboard Panel制作示例 69

第5章 Prometheus监控宿主机和容器 71

5.1 宿主机监控Prometheus.yml配置 71

5.2 容器监控Prometheus.yml配置 72

5.3 导入宿主机和容器监控模板 72

第6章 其它部署容器使用 75

6.1 Portainer 75

附件：Grafana Dashboard模板文件 76

版本控制信息

* 文档属性

|  |  |
| --- | --- |
| 属性 | 内容 |
| 文档名称 | Prometheus+Grafana+Es实验记录 |
| 文档标题 | Prometheus+Grafana+Es实验记录 |
| 文档版本号 | 1.0 |
| 版本日期 | 2022-11-24 |
| 作者 | 胡易鹏 |
| 审核人 |  |

* 文档变更过程

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 更新日期 | 更新人 | 主要更新内容 |
| 1.0 | 2022-11-24 | 胡易鹏 | 1. 初稿,Prometheus、Grafana、ES容器安装 2. Prometheus+Snmp-exporter监控F5硬件状态 3. Grafana+ES展示F5业务视图 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 容器化部署

## Prometheus部署

1、创建宿主机配置文件目录和数据文件目录

#配置文件目录

mkdir -p /opt/prometheus

#将prometheus.yml文件提前拷贝至该文件目录

#数据文件目录，建议提前在该目录下挂载一个大容量硬盘

mkdir -p /var/prom/data

#数据文件目录需要赋予权限

chmod -R 777 /var/prom/data

2、宿主机创建prometheus.yml配置文件

#创建prometheus.yml配置文件

vi /opt/prometheus/prometheus.yml

#如下配置信息供参考

# my global config

global:

scrape\_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.

evaluation\_interval: 10s # 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"]

#编辑完成，保存退出

3、拉取镜像，并启动容器

#拉取镜像

docker pull prom/prometheus

#启动容器

docker run --name prometheus -d --restart always -p 9090:9090 \

-v /opt/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml \

-v /var/prom/data:/prometheus/data \

--log-opt max-size=100m --log-opt max-file=1 \

prom/prometheus \

--storage.tsdb.path=/prometheus/data \

--storage.tsdb.retention.time=30d \

--config.file=/etc/prometheus/prometheus.yml

#取消容器的自动启动

docker update --restart no prometheus

启动容器命令解释：

#--restart always，自动启动

#-v /opt/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml，将本地宿主机的/opt/prometheus/prometheus.yml文件映射为容器内/etc/prometheus/prometheus.yml配置文件。

注：本地的prometheus.yml文件需要提前在宿主机目录编辑完成

#-v /var/prome/data:/prometheus/data 将本地宿主机/var/prome/data数据目录映射为容器的数据目录:/prometheus/data。

注：本地宿主机配置文件及目录及需要提前创建好，数据目录设置对应访问权限

#--log-opt max-size=100m --log-opt max-file=1 设置日志文件大小为100m，日志文件保存数量为一个

#--storage.tsdb.path=/prometheus/data 设置数据存储目录

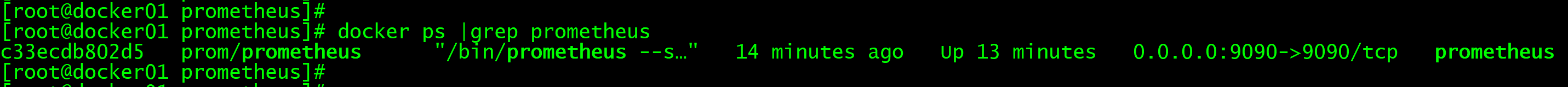
#--storage.tsdb.retention.time=30d 设置数据保存时间为30天

#--config.file=/etc/prometheus/prometheus.yml 设置配置文件路径的容器内目录

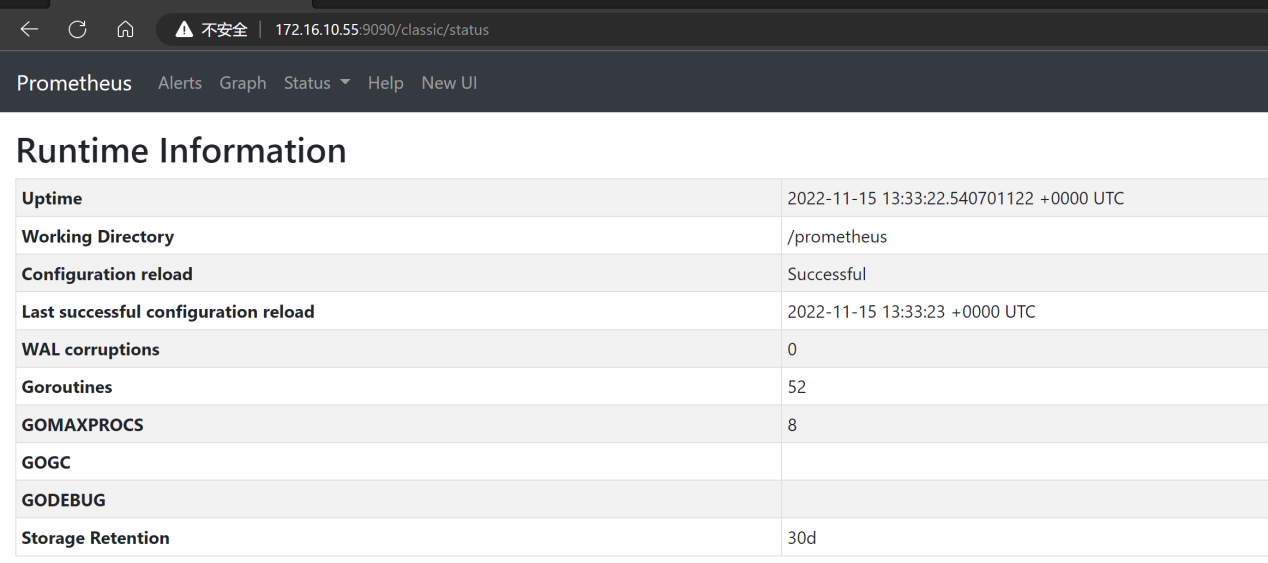
注：上述三个配置为容器内配置命令，需要放在最后面，我的理解是容器启动后才能执行。

3、查看容器运行状态

docker ps |grep prometheus



4、访问宿主机9090端口，访问运行中的prometheus。



## Grafana部署

1、创建宿主机数据文件目录

#数据文件目录

mkdir -p /var/grafana

#数据文件目录需要赋予权限

chmod -R 777 /var/grafana

2、拉取镜像，并启动容器

#拉取镜像

docker pull grafana/grafana

#启动容器

docker run --name grafana -d --restart always -p 3000:3000 \

-v /var/grafana:/var/lib/grafana \

--log-opt max-size=100m --log-opt max-file=1 \

grafana/grafana

#取消容器的自动启动

docker update --restart no grafana

启动容器命令解释：

#--restart always，自动启动

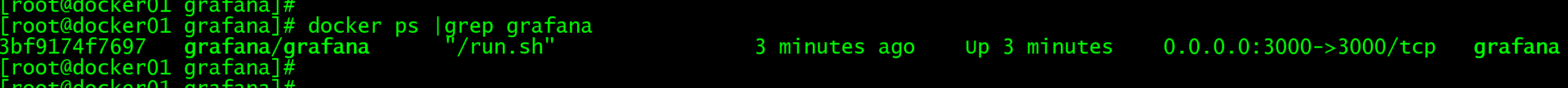
#-v /var/grafana:/var/lib/grafana，将本地宿主机/var/grafana数据目录映射为容器的数据目录:/var/lib/grafana。

注：本地宿主机数据目录需要提前创建好，数据目录设置对应访问权限

#--log-opt max-size=100m --log-opt max-file=1 设置日志文件大小为100m，日志文件保存数量为一个

3、查看容器运行状态

docker ps |grep grafana



1. 访问宿主机3000端口，访问运行中的grafana，初始默认用户名和密码为：admin/admin，初次登陆会提示修改密码



## snmp-exporter部署

1、创建宿主机配置文件目录和数据文件目录

#配置文件目录

mkdir -p /opt/snmp-exporter

#拷贝制作完成的snmp.yml文件至该目录，参考3.1小节生成snmnp.yml文件

2、拉取镜像，并启动容器

#拉取镜像

docker pull prom/snmp-exporter

#启动容器

docker run -d --name snmp-exporter --restart=always -p 9116:9116 \

-v /opt/snmp\_exporter/snmp.yml:/etc/snmp\_exporter/snmp.yml \

--log-opt max-size=100m --log-opt max-file=1 \

prom/snmp-exporter

#设置容器的自动启动

docker update --restart always snmp-exporter

启动容器命令解释：

#--restart always，自动启动

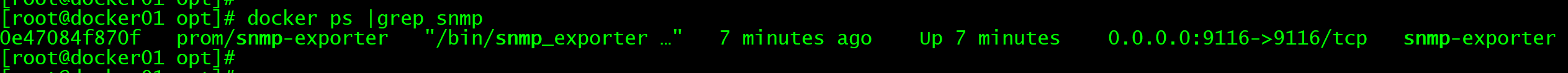
#-v /opt/snmp\_exporter/snmp.yml:/etc/snmp\_exporter/snmp.yml，将本地宿主机的/opt/snmp\_exporter/snmp.yml文件映射为容器内的/etc/snmp\_exporter/snmp.yml配置文件。

注：本地的snmp.yml文件需要提前生成并拷贝至宿主机目录（参考3.1小节生成snmnp.yml文件）。

#--log-opt max-size=100m --log-opt max-file=1 设置日志文件大小为100m，日志文件保存数量为一个。

3、查看容器运行状态

docker ps |grep snmp



4、访问宿主机9116端口，访问运行中的snmp exporter



## ES部署

1. 调整服务器虚拟内存大小

#调整虚拟内存大小

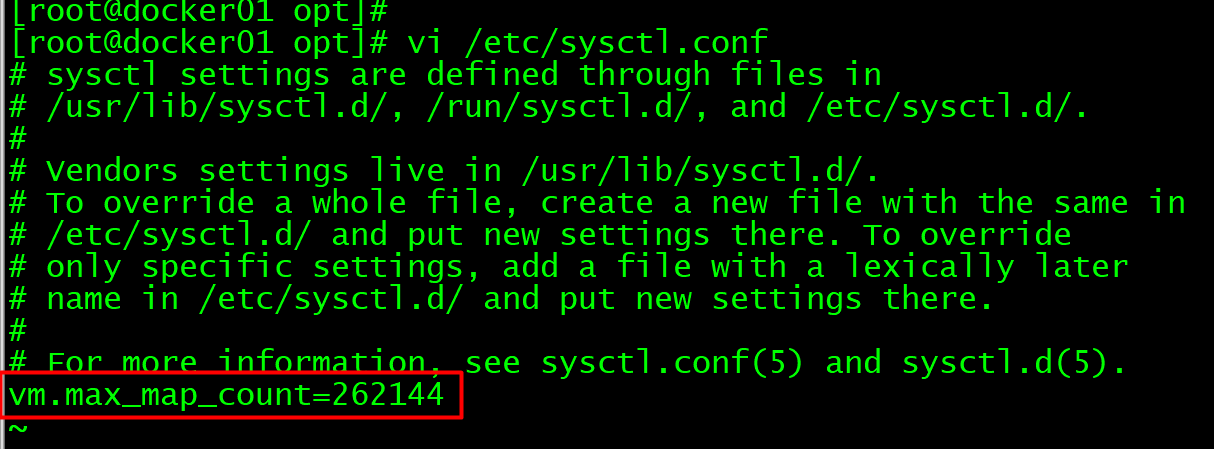
vi /etc/sysctl.conf

#在最后一行添加

vm.max\_map\_count=262144

#执行如下命令 立即生效

/sbin/sysctl -p



1. 创建宿主机配置文件目录和数据文件目录

#配置文件目录

mkdir -p /opt/es

#创建数据文件目录，建议提前在数据目录下挂载一个大容量硬盘

/var/es-data/es-data1

/var/es-data/es-data1

#数据文件目录需要赋予权限

chmod -R 777 /var/es-data

1. 宿主机创建elasticsearch.yml配置文件

#创建elasticsearch.yml配置文件

vi /opt/es/elasticsearch.yml

#如下配置信息供参考

#node名称

node.name: es-node-1

#

#数据存储目录

path.data: /usr/share/elasticsearch/data/es-data1,/usr/share/elasticsearch/data/es-data2

#

#网络地址

network.host: 0.0.0.0

#

#HTTP 端口

http.port: 9200

#

cluster.initial\_master\_nodes: ["es-node-1"]

#

#启用认证

xpack.security.enabled: true

xpack.security.transport.ssl.enabled: true

#

http.cors.enabled: true

http.cors.allow-origin: "\*"

http.cors.allow-headers: Authorization,X-Requested-With,Content-Length,Content-Type

#编辑完成，保存退出

4、拉取镜像，并启动容器

#拉取镜像，测试使用7.17.7版本

docker pull docker.elastic.co/elasticsearch/elasticsearch:7.17.7

#启动容器

docker run --name es -d \

-v /opt/es/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml \

-v /var/es-data:/usr/share/elasticsearch/data \

--log-opt max-size=100m --log-opt max-file=1 \

-e ES\_JAVA\_OPTS="-Xms2g -Xmx2g" \

-p 9200:9200 -p 9300:9300 \

docker.elastic.co/elasticsearch/elasticsearch:7.17.7

#设置容器的自动启动

docker update --restart always es

启动容器命令解释：

#--restart always，自动启动

#-v /opt/es/elasticsearch.yml:/usr/share/elasticsearch/config/elasticsearch.yml，将本地宿主机的 /opt/es/elasticsearch.yml文件映射为容器内/usr/share/elasticsearch/config/elasticsearch.yml配置文件。

注：本地的elasticsearch.yml需要提前在宿主机目录编辑完成

#-v /var/es-data:/usr/share/elasticsearch/data， 将本地宿主机的/var/es-data目录映射为容器内/usr/share/elasticsearch/data数据目录。

注：本地宿主机目录需要提前创建好，数据目录设置对应访问权限。

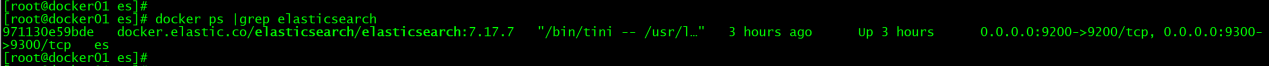
#--log-opt max-size=100m --log-opt max-file=1，设置日志文件大小为100m，日志文件保存数量为一个。

#-e ES\_JAVA\_OPTS="-Xms2g -Xmx2g"，设置数据为实际内存的一半，

注：该参数貌似不能动态修改，想要创建后再修改，最简单的方式是把容器停止、删除，再使用新的参数再建一个。

5、查看容器运行状态

docker ps |grep elasticsearch



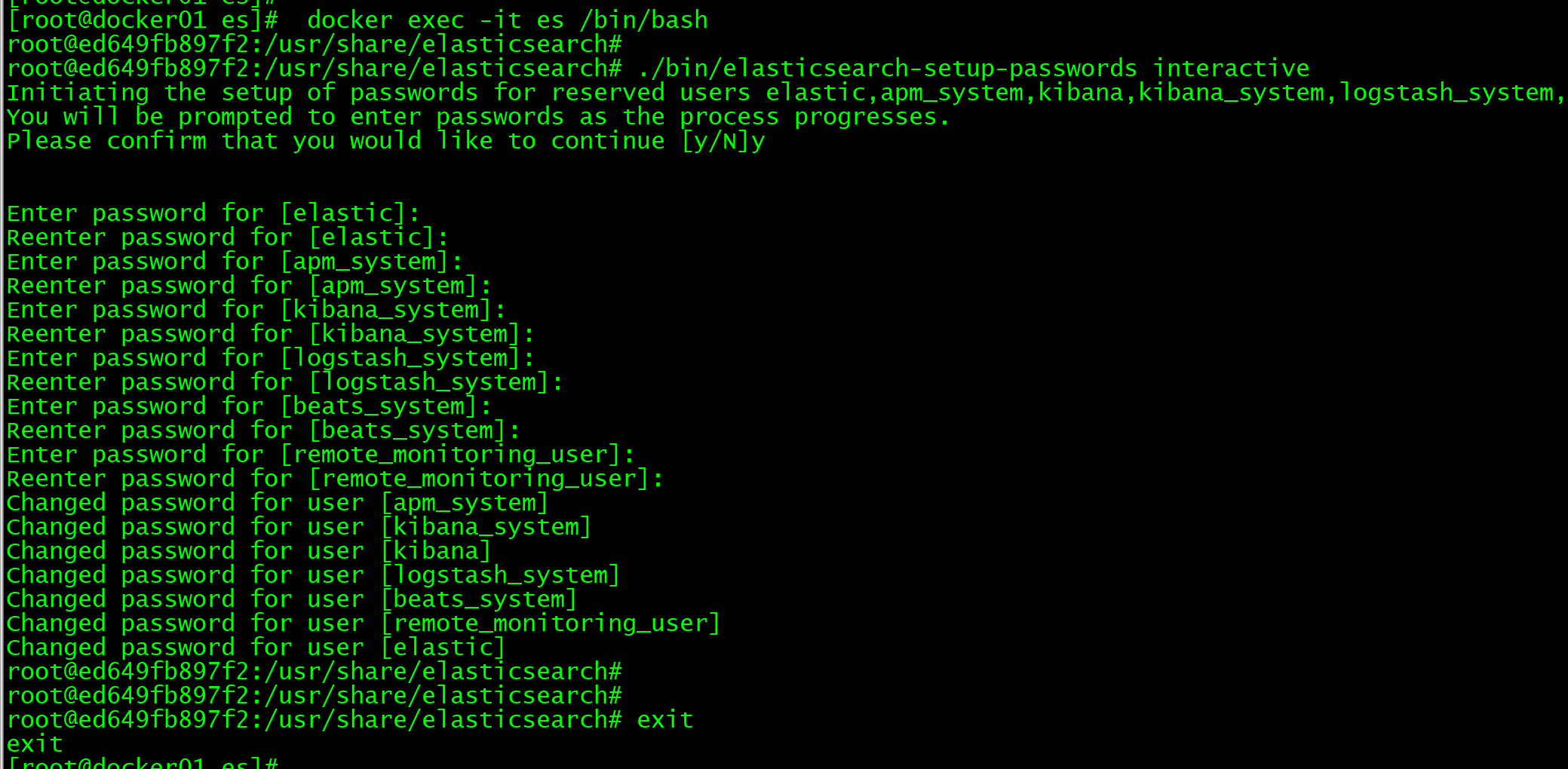
6、设置es密码

#进入容器

docker exec -it es /bin/bash

#设置密码

./bin/elasticsearch-setup-passwords interactive



7、访问宿主机9200端口，访问运行中的es



## Kibana部署

1、创建宿主机配置文件目录

#配置文件目录

mkdir -p /opt/kibana

#拷贝配置好的kibana.yml文件至该目录

2、宿主机创建elasticsearch.yml配置文件

#创建kibana.yml配置文件

vi /opt/kibana/kibana.yml

#如下配置信息供参考

#kibana端口

server.port: 5601

#kibana地址

server.host: "0.0.0.0"

#es地址,根据实际情况修改

elasticsearch.hosts: "http://192.168.101.190:9200"

#index设置

kibana.index: ".kibana"

monitoring.ui.container.elasticsearch.enabled: true

elasticsearch.requestTimeout: 60000

#es用户名密码

elasticsearch.username: "elastic"

elasticsearch.password: "default"

#编辑完成，保存退出

3、拉取镜像，并启动容器

#拉取镜像

docker pull docker.elastic.co/kibana/kibana:7.17.7

#启动容器

docker run --name kibana -d \

-v /opt/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml \

--log-opt max-size=100m --log-opt max-file=1 \

-p 5601:5601 docker.elastic.co/kibana/kibana:7.17.7

#启用容器的自动启动

docker update --restart always kibana

启动容器命令解释：

#--restart always，自动启动

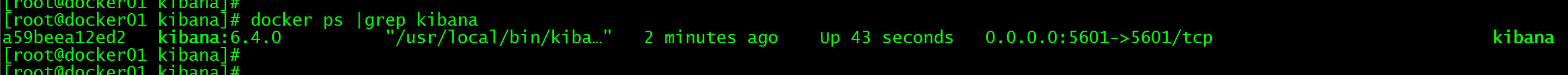
#-v /opt/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml，将本地宿主机的/opt/kibana/kibana.yml文件映射为容器内/usr/share/kibana/config/kibana.yml配置文件。

注：本地的kibana.yml文件需要提前在宿主机目录编辑完成。

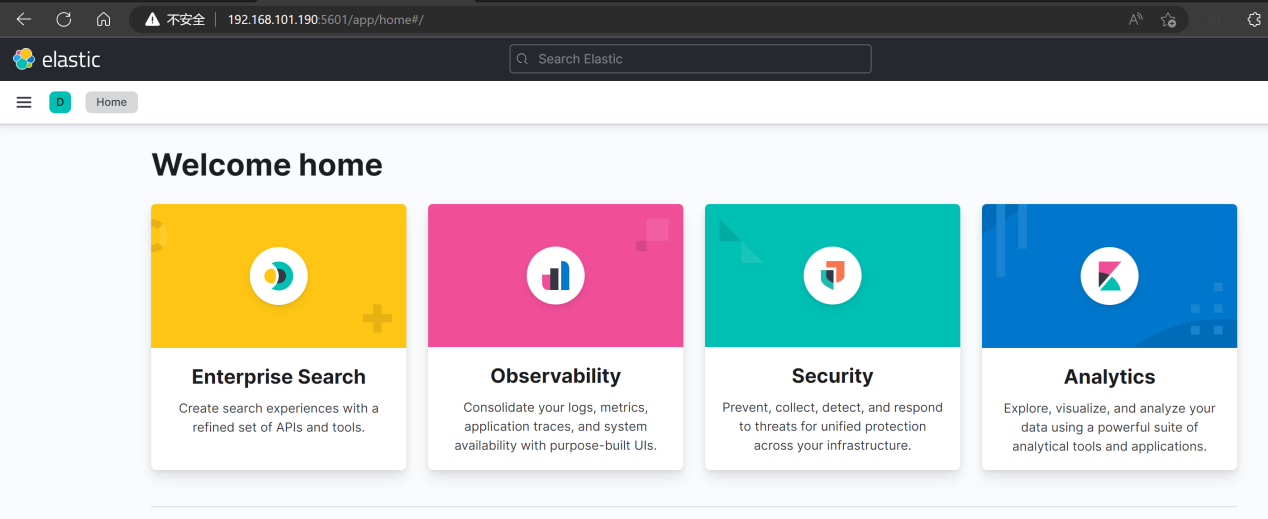
#--log-opt max-size=100m --log-opt max-file=1 设置日志文件大小为100m，日志文件保存数量为一个。

4、查看容器运行状态

docker ps |grep kibana



1. 访问宿主机5601端口，输入用户名和密码访问运行中的kibana



## Logstash部署

1、创建宿主机配置文件目录

#配置文件目录

mkdir -p /opt/logstash

2、宿主机创建logstash配置文件

**HTTP业务Logstash配置文件**

#创建F5 HTTP业务logstash配置文件

vi /opt/logstash/f5-http.conf

#如下配置信息供参考

input {

tcp {

port => 8514

type => "f5-http"

}

}

filter {

if [type] == "f5-http" and [message] =~ ' '{

grok {

match => { "message" => "%{IP:clientip} %{DATA:xff} \[%{HTTPDATE:timestamp}\] %{IP:virtual\_ip} %{DATA:virtual\_name} \"%{DATA:path}\" \"(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})\" %{NUMBER:response:int} %{NUMBER:bytes:int} %{NUMBER:response\_ms:int} %{QS:referrer} %{QS:agent}"}

}

} else if [type] == "f5-http" {

grok {

match => { "message" => "%{IP:clientip} %{DATA:xff} \[%{HTTPDATE:timestamp}\] %{IP:virtual\_ip} %{DATA:virtual\_name} %{DATA:virtual\_pool\_name} %{DATA:server} %{NUMBER:server\_port} \"%{DATA:path}\" \"(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})\" %{NUMBER:response:int} %{NUMBER:bytes:int} %{NUMBER:response\_ms:int} %{QS:referrer} %{QS:agent}"}

}

}

if [xff] {

mutate {

gsub => [

"xff", ",.\*", ""

]

}

geoip {

source => "xff"

target => "geoip"

}

} else {

geoip {

source => "clientip"

target => "geoip"

}

}

}

output {

if [type] == "f5-http" {

elasticsearch {

hosts => ["192.168.101.190:9200"] #ES IP地址和端口，根据实际情况修改

index => "logstash-f5-http-%{+YYYY.MM.dd}"

user => "elastic"

password => "xxxx" #ES密码，根据实际密码修改

}

}

}

#编辑完成，保存退出

**DNS业务Logstash配置文件**

#创建F5 DNS业务logstash配置文件

vi /opt/logstash/f5-dns.conf

#如下配置信息供参考

input {

tcp {

port => 8513

type => 'f5-dns'

}

}

filter {

if [type] == 'f5-dns' and [message] =~ 'query:' {

grok {

match => { "message" => "%{TIMESTAMP\_ISO8601:requesttime} %{HOSTNAME:F5hostname} qid %{NUMBER:queryid} from %{IP:clientip}#%{POSINT:clientport}: view %{GREEDYDATA:viewname}: query: %{DATA:queryname} IN %{GREEDYDATA:querytype} %{DATA:unknow} \(%{IP:listenervs}\%%{NUMBER:routedomain}\)" }

}

geoip {

source => "clientip"

target => "geoip"

}

} else if [type] == 'f5-dns' and [message] =~ '; '{

grok {

match => { "message" => "%{TIMESTAMP\_ISO8601:responsetime} %{HOSTNAME:F5hostname} qid %{NUMBER:queryid} to %{IP:clientip}#%{POSINT:clientport}: \[%{WORD:responsecode} %{GREEDYDATA:responseflag}\] response: %{HOSTNAME:responsename}\. %{NUMBER:ttl} IN %{GREEDYDATA:answer} %{IP:recordip}" }

add\_field => [ "iswideip", "no" ]

}

geoip {

source => "clientip"

target => "geoipresponse"

}

} else if [type] == 'f5-dns' and [message] =~ ': empty' {

grok {

match => { "message" => "%{TIMESTAMP\_ISO8601:responsetime} %{HOSTNAME:F5hostname} qid %{NUMBER:queryid} to %{IP:clientip}#%{POSINT:clientport}: \[%{WORD:responsecode} %{GREEDYDATA:responseflag}\] response: %{GREEDYDATA:answer}" }

add\_field => [ "emptyresponse", "yes" ]

}

geoip {

source => "clientip"

target => "geoipresponse"

}

} else if [type] == 'f5-dns' {

grok {

match => { "message" => "%{TIMESTAMP\_ISO8601:responsetime} %{HOSTNAME:F5hostname} qid %{NUMBER:queryid} to %{IP:clientip}#%{POSINT:clientport}: \[%{WORD:responsecode} %{GREEDYDATA:responseflag}\] response: %{HOSTNAME:responsename}\. %{NUMBER:ttl} IN %{GREEDYDATA:answer} %{IP:recordip}" }

}

geoip {

source => "clientip"

target => "geoipresponse"

}

}

}

output {

if [type] == "f5-dns" {

elasticsearch {

hosts => ["192.168.101.190:9200"] #ES IP地址和端口，根据实际情况修改

index => "logstash-f5-dns-%{+YYYY.MM.dd}"

user => "elastic"

password => "xxxx" #ES密码，根据实际密码修改

}

}

}

#编辑完成，保存退出

**TCP-L4业务Logstash配置文件**

#创建F5 TCP L4 业务logstash配置文件

vi /opt/logstash/f5-tcpl4.conf

#如下配置信息供参考

input {

tcp {

port => 8515

type => 'f5-tcpl4'

}

}

filter {

if [message] =~ 'init-delay' {

grok {

match => { "message" => "%{DATA:virtual\_name} %{IP:clientip} %{NUMBER:clientport} %{IP:vsip} %{NUMBER:vsport} %{IP:snatip} %{NUMBER:snatport} %{IP:memberip} %{NUMBER:memberport} cdnumber %{NUMBER:cdnumber:int} init-delay %{NUMBER:init-delay:int}" }

}

}

if [message] =~ 'HS-delay' {

grok {

match => { "message" => "%{DATA:virtual\_name} %{IP:clientip} %{NUMBER:clientport} %{IP:vsip} %{NUMBER:vsport} %{IP:snatip} %{NUMBER:snatport} %{IP:memberip} %{NUMBER:memberport} HS-delay %{NUMBER:hs-delay:int}" }

}

}

if [message] =~ 'svr-pkts-delay' {

grok {

match => { "message" => "%{DATA:virtual\_name} %{IP:clientip} %{NUMBER:clientport} %{IP:vsip} %{NUMBER:vsport} %{IP:snatip} %{NUMBER:snatport} %{IP:memberip} %{NUMBER:memberport} svr-pkts-delay %{NUMBER:svr-pkts-delay:int}" }

}

}

geoip {

source => "clientip"

target => "geoip"

}

}

output {

if [type] == "f5-tcpl4" {

elasticsearch {

hosts => ["192.168.101.190:9200"] #ES IP地址和端口，根据实际情况修改

index => "logstash-f5-tcpl4-%{+YYYY.MM.dd}"

user => "elastic"

password => "xxxx" #ES密码，根据实际密码修改

}

}

}

#编辑完成，保存退出

**AWAF业务Logstash配置文件**

#创建F5 AWAF业务logstash配置文件

vi /opt/logstash/f5-awaf.conf

#如下配置信息供参考

input {

tcp {

port => 8516

type => "f5-asm"

}

tcp {

port => 8516

type => "f5-bot"

}

tcp {

port => 8516

type => "f5-dos"

}

}

filter {

if [type] == "f5-asm" {

mutate {

split => ["message"," ASM:"]

add\_field => { "tmp\_log" => "%{[message][1]}" }

}

kv {

source => "tmp\_log"

field\_split => ","

}

if [xff] {

mutate {

gsub => [

"xff", ",.\*", ""

]

}

geoip {

source => "x\_forwarded\_for\_header\_value"

target => "geoip"

}

} else {

geoip {

source => "ip\_client"

target => "geoip"

}

}

}

if [type] == "f5-bot" {

kv {

source => "message"

field\_split => ","

}

if [xff] {

mutate {

gsub => [

"xff", ",.\*", ""

]

}

geoip {

source => "x\_forwarded\_for\_header\_value"

target => "geoip"

}

} else {

geoip {

source => "ip\_client"

target => "geoip"

}

}

}

if [type] == "f5-dos" {

kv {

source => "message"

field\_split => ","

}

mutate {

remove\_field => [ "message" ]

}

if [xff] {

mutate {

gsub => [

"xff", ",.\*", ""

]

}

geoip {

source => "x\_forwarded\_for\_header\_value"

target => "geoip"

}

} else {

geoip {

source => "ip\_client"

target => "geoip"

}

}

}

}

output {

if [type] == "f5-asm" {

elasticsearch {

hosts => ["192.168.101.190:9200"] #ES IP地址和端口，根据实际情况修改

index => "logstash-asm-%{+YYYY.MM.dd}"

user => "elastic"

password => "xxxx" #ES密码，根据实际密码修改

}

}

if [type] == "f5-bot" {

elasticsearch {

hosts => ["192.168.101.190:9200"] #ES IP地址和端口，根据实际情况修改

index => "logstash-bot-%{+YYYY.MM.dd}"

user => "elastic"

password => "xxxx" #ES密码，根据实际密码修改

}

}

if [type] == "f5-dos" {

elasticsearch {

hosts => ["192.168.101.190:9200"] #ES IP地址和端口，根据实际情况修改

index => "logstash-dos-%{+YYYY.MM.dd}"

user => "elastic"

password => "xxxx" #ES密码，根据实际密码修改

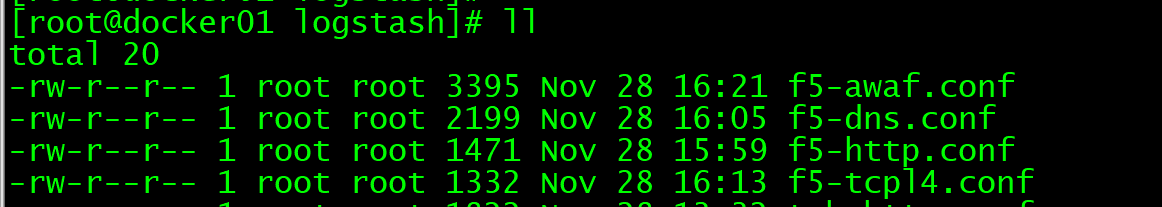
}

}

}

#编辑完成，保存退出

注：共四个.conf文件，分别用于展示F5的http业务、dns解析、tcp业务、awaf安全防护四个视图的展示



2、拉取镜像，并启动容器

#拉取镜像

docker pull docker.elastic.co/logstash/logstash:7.17.7

#启动容器

注：需要接收什么类型的业务日志就启用一个该接收该类型日志的Logstash容器。

#http业务视力logstash

docker run --name logs-f5-http -d \

-v /opt/logstash/f5-http.conf:/usr/share/logstash/pipeline/f5-http.conf \

-e LS\_JAVA\_OPTS="-Xms512m -Xmx512m" \

--log-opt max-size=100m --log-opt max-file=1 \

-p 8514:8514 \

docker.elastic.co/logstash/logstash:7.17.7

#dns业务视力logstash

docker run --name logs-f5-dns -d \

-v /opt/logstash/f5-dns.conf:/usr/share/logstash/pipeline/f5-dns.conf \

-e LS\_JAVA\_OPTS="-Xms512m -Xmx512m" \

--log-opt max-size=100m --log-opt max-file=1 \

-p 8513:8513 \

docker.elastic.co/logstash/logstash:7.17.7

#tcp业务视力logstash

docker run --name logs-f5-tcpl4 -d \

-v /opt/logstash/f5-tcpl4.conf:/usr/share/logstash/pipeline/f5-tcpl4.conf \

-e LS\_JAVA\_OPTS="-Xms512m -Xmx512m" \

--log-opt max-size=100m --log-opt max-file=1 \

-p 8515:8515 \

docker.elastic.co/logstash/logstash:7.17.7

#awaf业务视力logstash

docker run --name logs-f5-awaf -d \

-v /opt/logstash/f5-awaf.conf:/usr/share/logstash/pipeline/f5-awaf.conf \

-e LS\_JAVA\_OPTS="-Xms512m -Xmx512m" \

--log-opt max-size=100m --log-opt max-file=1 \

-p 8516:8516 \

docker.elastic.co/logstash/logstash:7.17.7

#启用容器的自动启动

docker update --restart always logs-\*

启动容器命令解释：

#--restart always，自动启动

#-v /opt/logstash/f5-\*.conf:/usr/share/logstash/pipeline/f5-\*.conf，将本地宿主机的/opt/logstash/f5-\*.conf文件映射为容器内/usr/share/logstash/pipeline/f5-\*.conf配置文件。

注：上述文件需要提前拷贝至宿主机目录。

#--log-opt max-size=100m --log-opt max-file=1 设置日志文件大小为100m，日志文件保存数量为一个。

#-e LS\_JAVA\_OPTS="-Xms512m -Xmx512m"，设置数据为实际内存的一半，

注：该参数貌似不能动态修改，想要创建后再修改，最简单的方式是把容器停止、删除，再使用新的参数再建一个

3、查看容器运行状态

docker ps |grep logstash



## node-exporter部署

1、拉取镜像，并启动容器

#拉取镜像

docker pull prom/node-exporter

#启动容器

docker run --name=node-exporter -d -p 9100:9100 \

-v /proc:/host/proc \

-v /sys:/host/sys \

-v /:/rootfs \

--log-opt max-size=100m --log-opt max-file=1 \

prom/node-exporter \

--path.procfs /host/proc \

--path.sysfs /host/sys \

--path.rootfs=/rootfs \

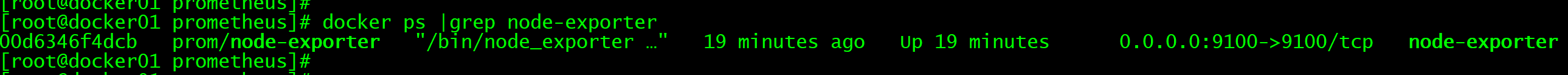
--collector.filesystem.mount-points-exclude="^/(sys|proc|dev|host|etc)($|/)"

#启用容器的自动启动

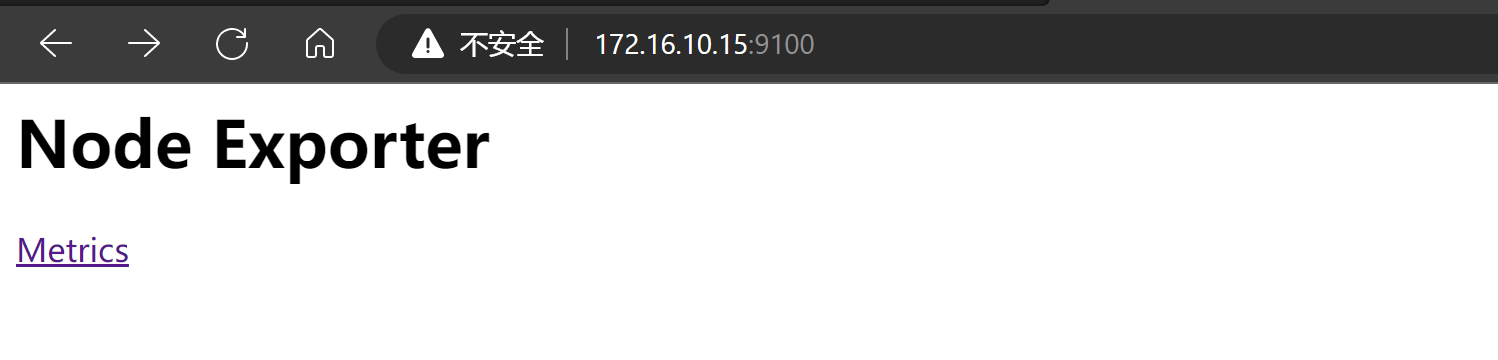
docker update --restart always node-exporter

2、查看容器运行状态

docker ps |grep node-exporter



3、访问宿主机9100端口，访问运行中的node-exporter



## cadvisor部署

1、拉取镜像，并启动容器

#拉取镜像

docker pull google/cadvisor

#使用如下命令启动正式容器

docker run \

--volume=/:/rootfs:ro \

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

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

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

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

--publish=8080:8080 \

--log-opt max-size=100m --log-opt max-file=1 \

--detach=true \

--name=cadvisor \

--privileged=true \

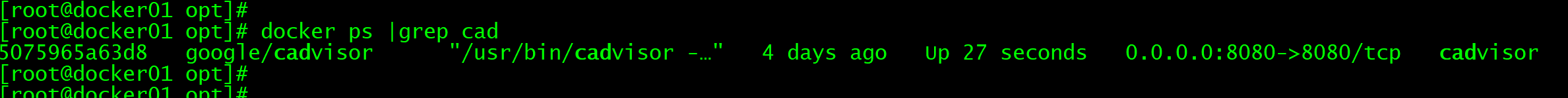
google/cadvisor

#启用容器的自动启动

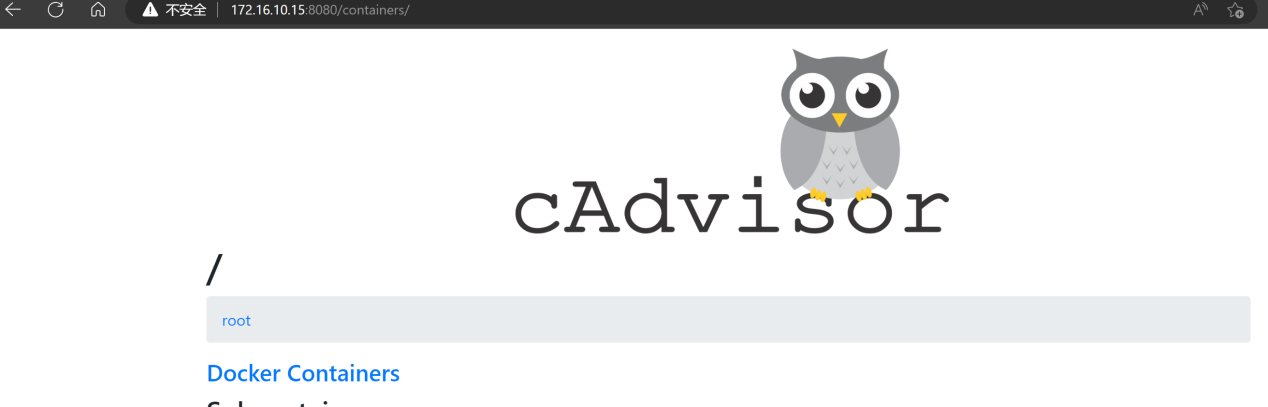
docker update --restart always cadvisor

3、查看容器运行状态

docker ps |grep cad



1. 访问宿主机8080端口，访问运行中的cadvisor



## portainer部署

1、拉取镜像，并启动容器

#拉取镜像

docker pull portainer/portainer-ce:latest

#使用如下命令启动正式容器

docker run -d \

-p 9000:9000 \

--name portainer \

--restart=always \

-v /var/run/docker.sock:/var/run/docker.sock \

-v portainer:/data \

--log-opt max-size=100m --log-opt max-file=1 \

portainer/portainer-ce:latest

#启用容器的自动启动

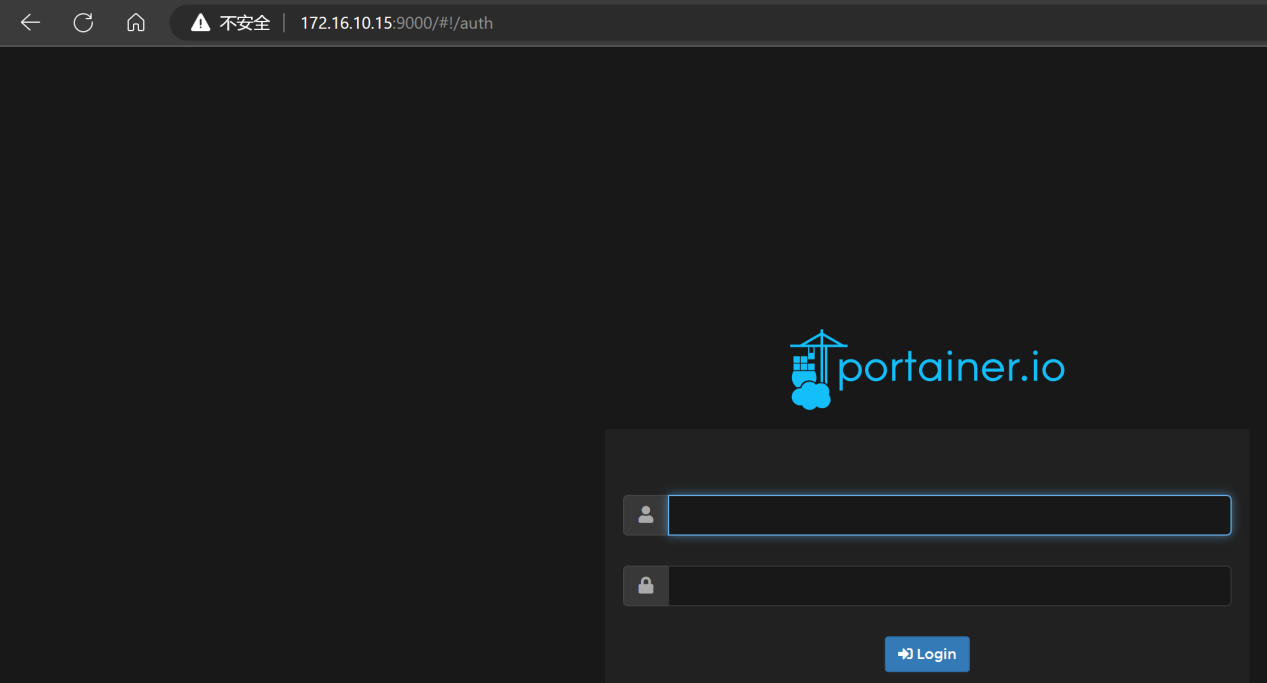
docker update --restart always portainer

3、查看容器运行状态

docker ps |grep porta



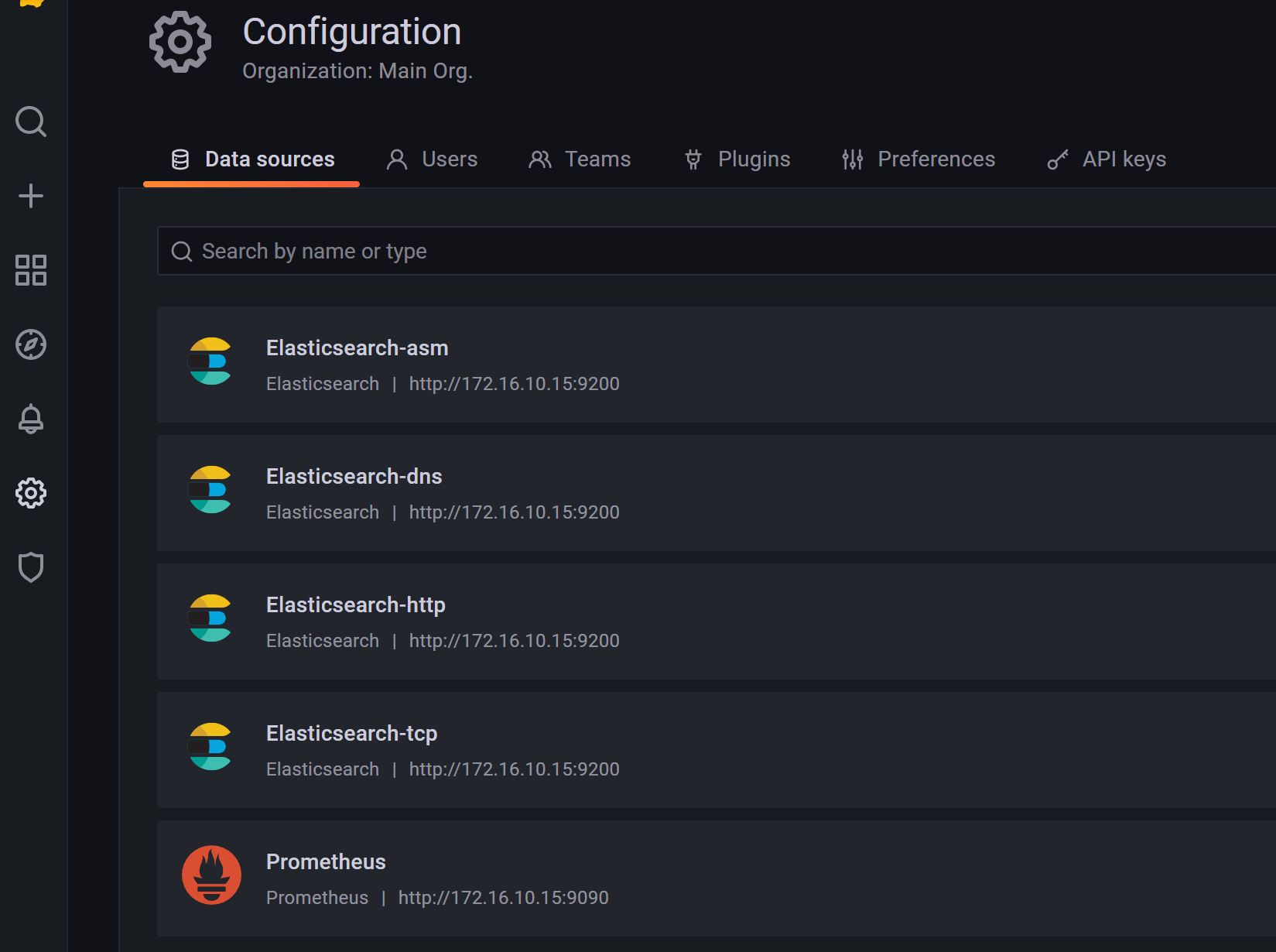
1. 访问宿主机9000端口，访问运行中的portainer



# Grafana初始化配置

## 添加数据源

数据源有两类（Prometheus和Elasticsearch）共五个，分别如下：

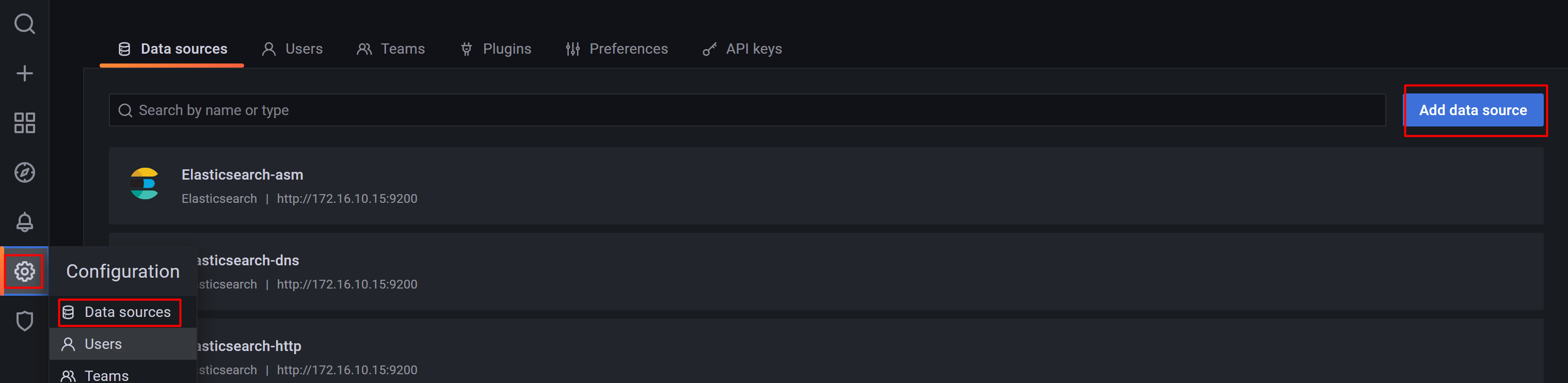


Prometheus数据源用于监控F5硬件设备和宿主机及容器资源使用；Elasticsearch数据用于监控基于F5 HSL的http\dns\tcp\asm四类业务日志。

数据源添加方式如下：

### 添加pormetheus数据源：

点击Configuration\Data source，点击Add data source，添加数据源。

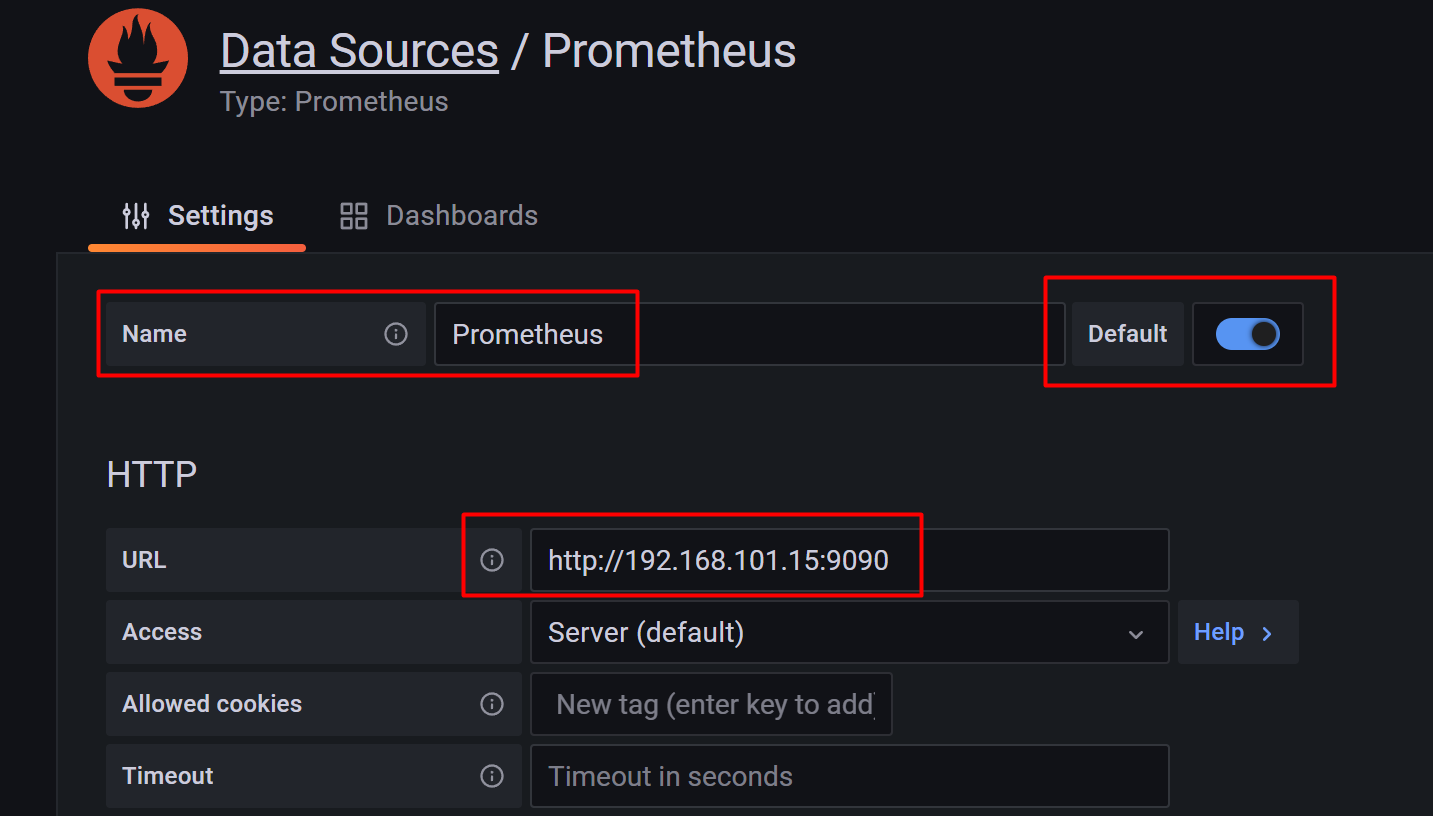


选择Prometheus，点击Select。

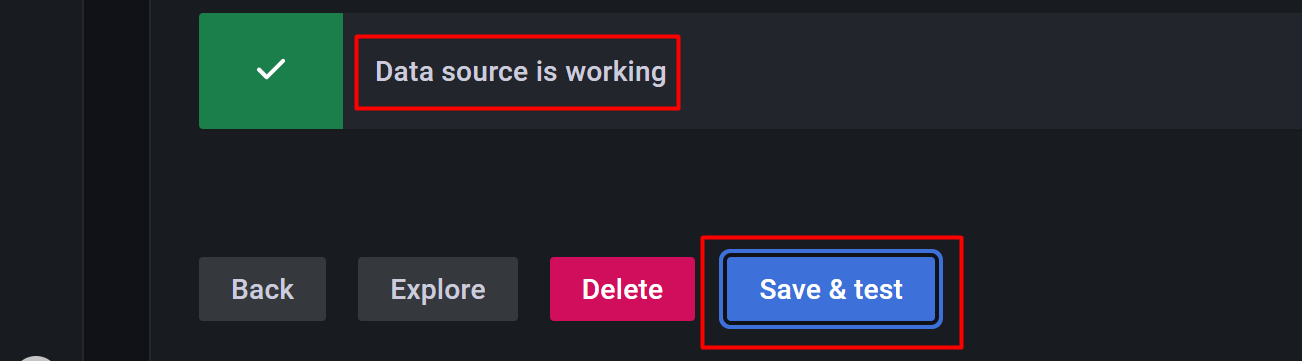


Name：配置数据源名称，Default：设置为默认数据源

URL：输入prometheus的访问url。

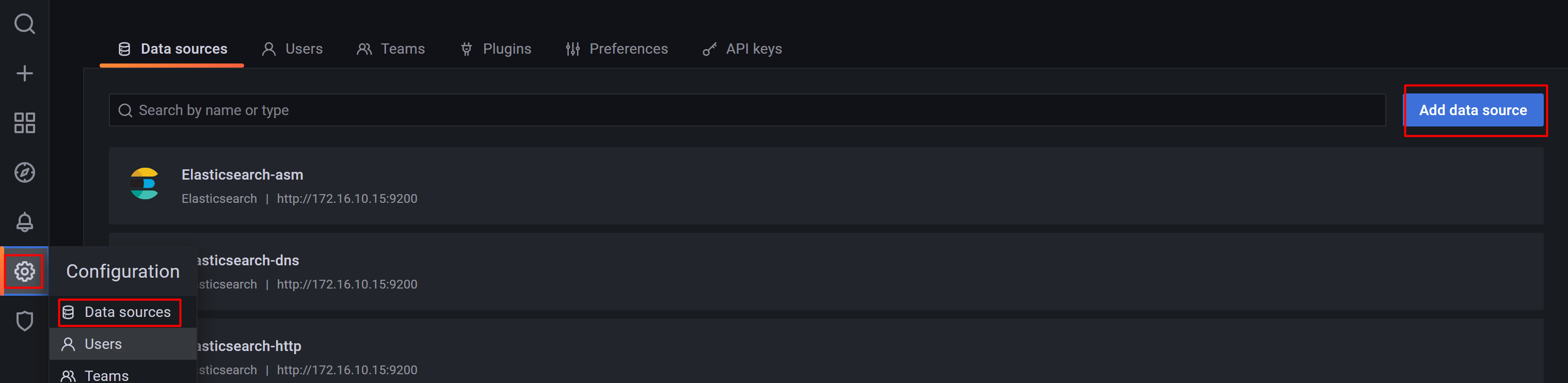


下拉至底部，点击Save & test，出现Data source is working，证明添加成功，prometheus也工作正常。

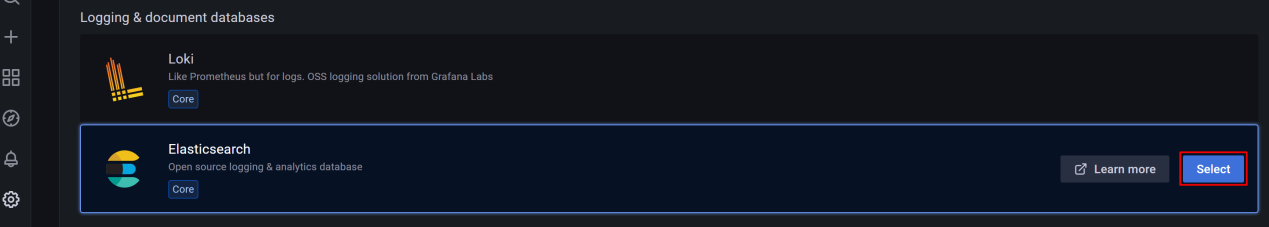


### 添加Elasticsearch数据源：

点击Configuration\Data source，点击Add data source，添加数据源。

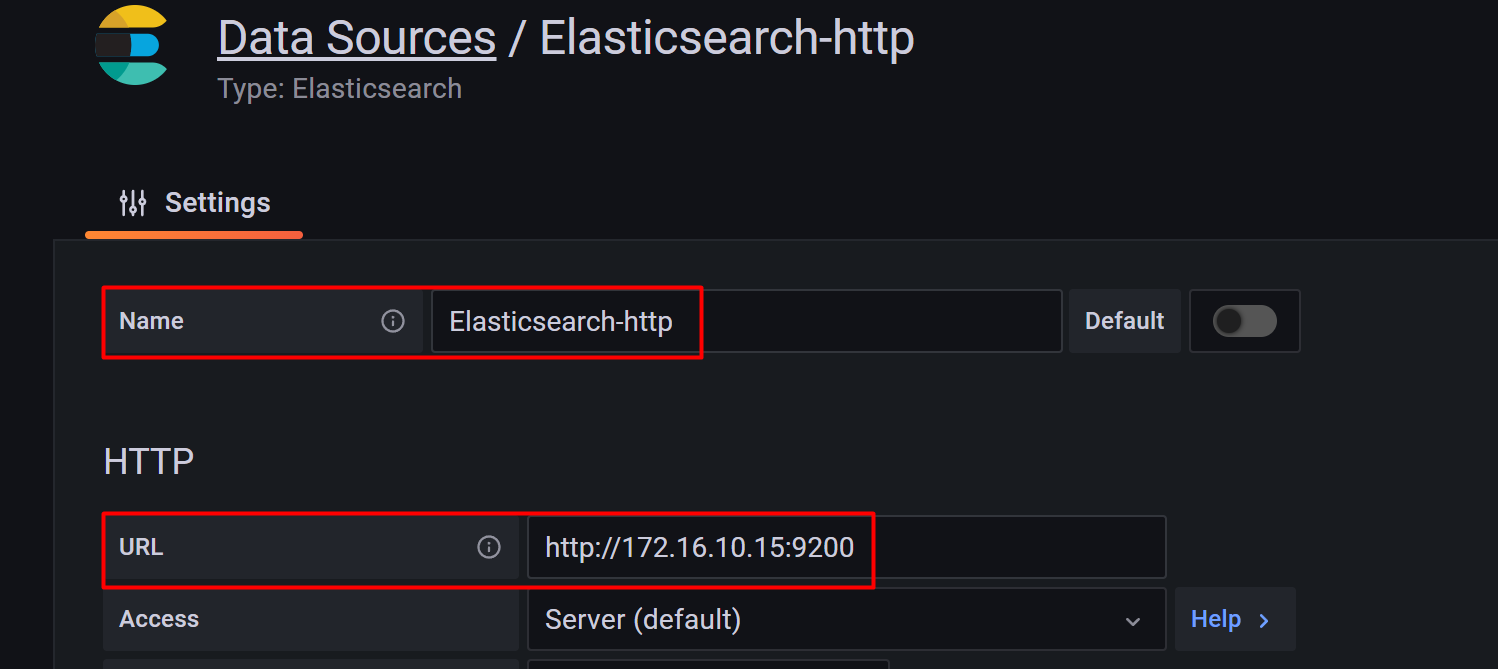


选择Elasticsearch，点击Select。



Name：配置数据源名称，Default：是否设置为默认数据源

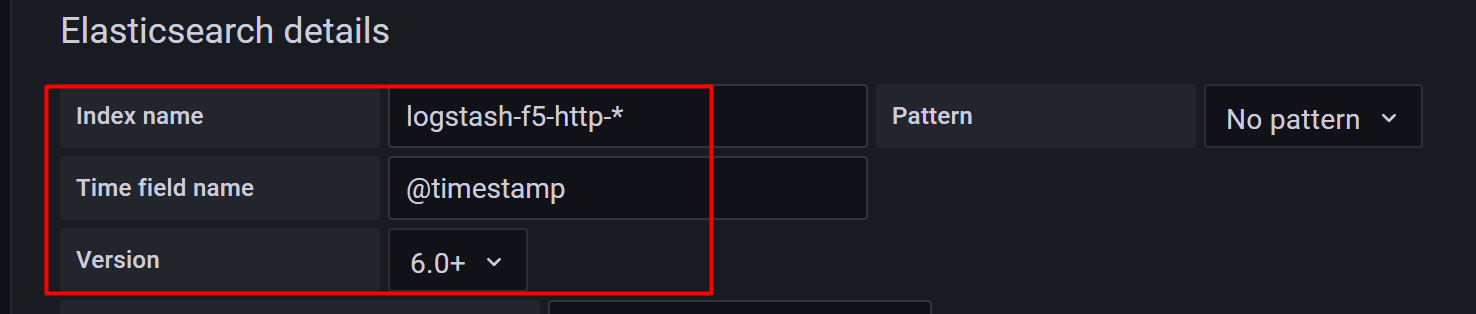
URL：输入Elasticsearch的访问url



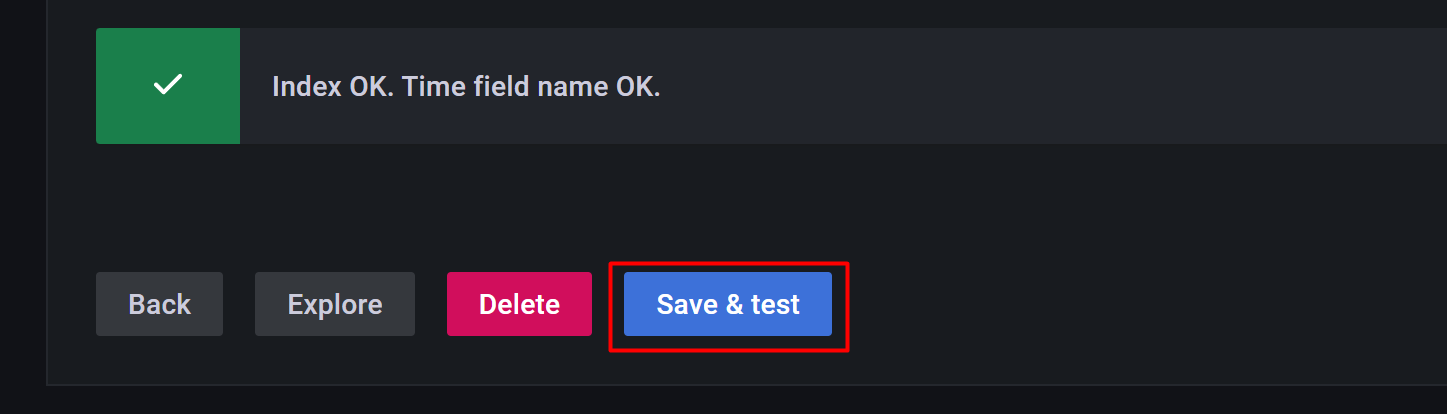
Index name：配置es索引名称

Time field name：选择@timestamp

Version：选择6.0+（使用版本为6.4.0）



下拉至底部，点击Save & test，Index OK.Time field name OK，证明添加成功，Elasticsearch也工作正常。

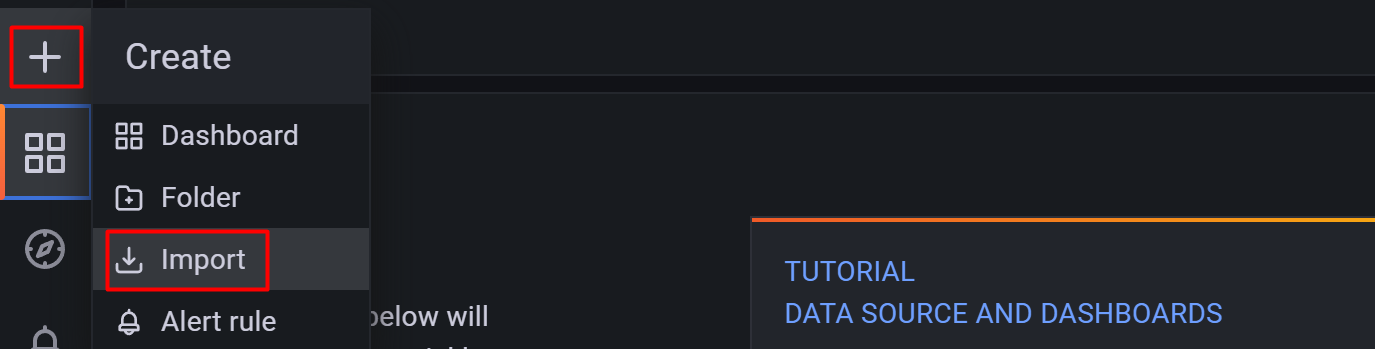


参照上述方法，继续添加其它业务类型的Elasticsearch（dns\asm\tcp）数据源。

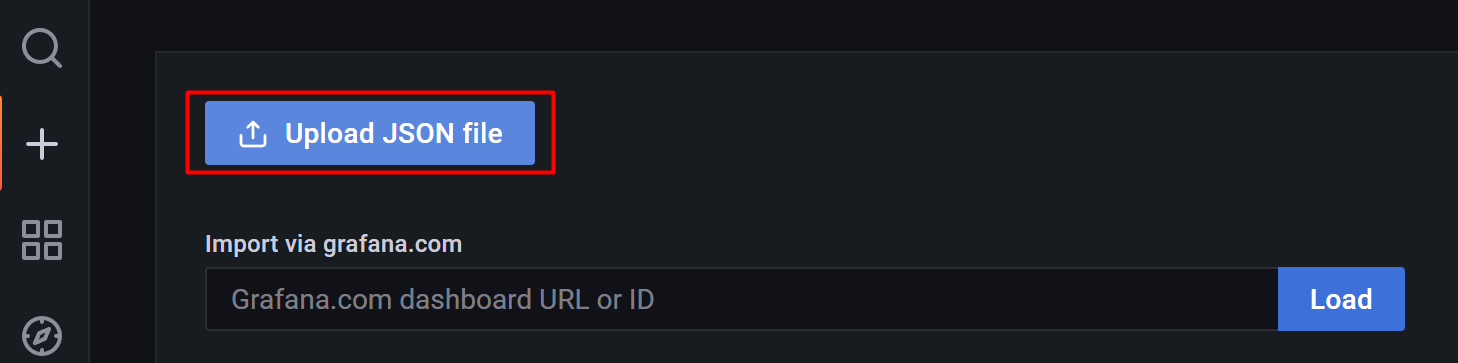
## 导入Dashboard模板

### 导入Prometheus模板

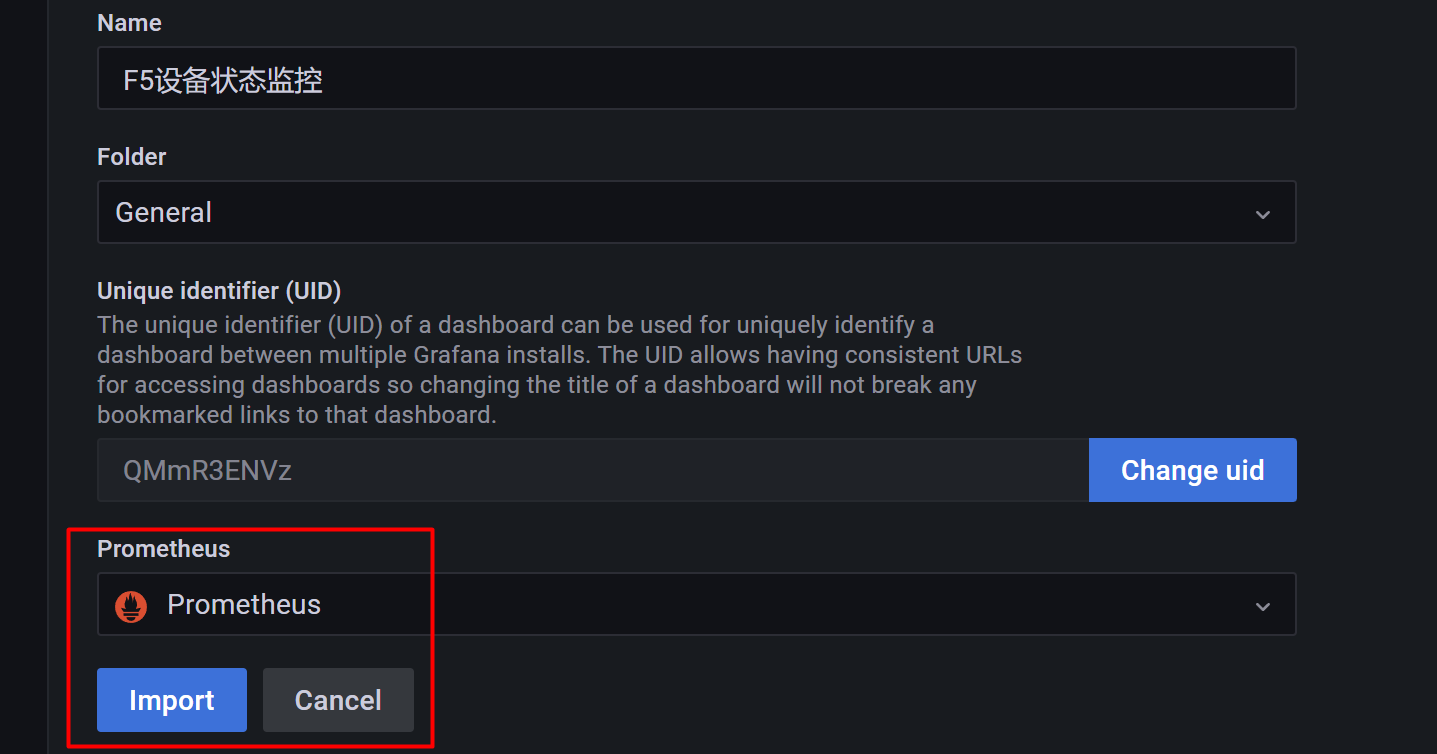
点击 +，点击Import进入Dashboard导入界面。



点击Upload JSON file，选择需要导入的dashboard模板文件，并点击Load。

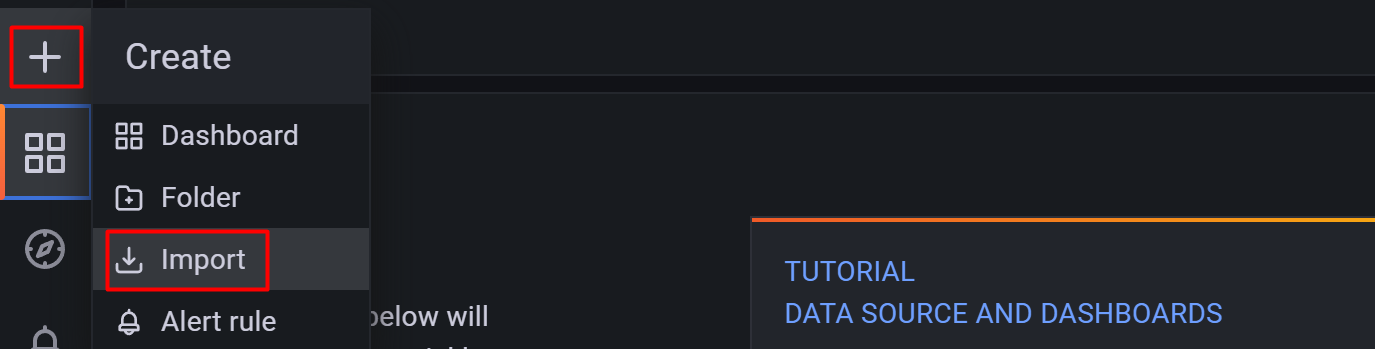


下方Select a Prometheus data source选择Prometheus，并点击Import导入。

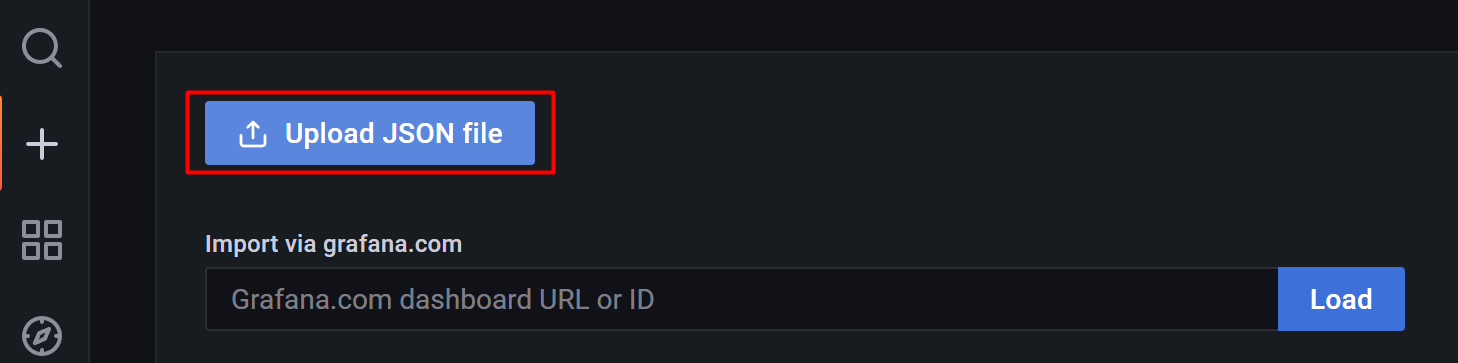


### 导入Elasticsearch模板

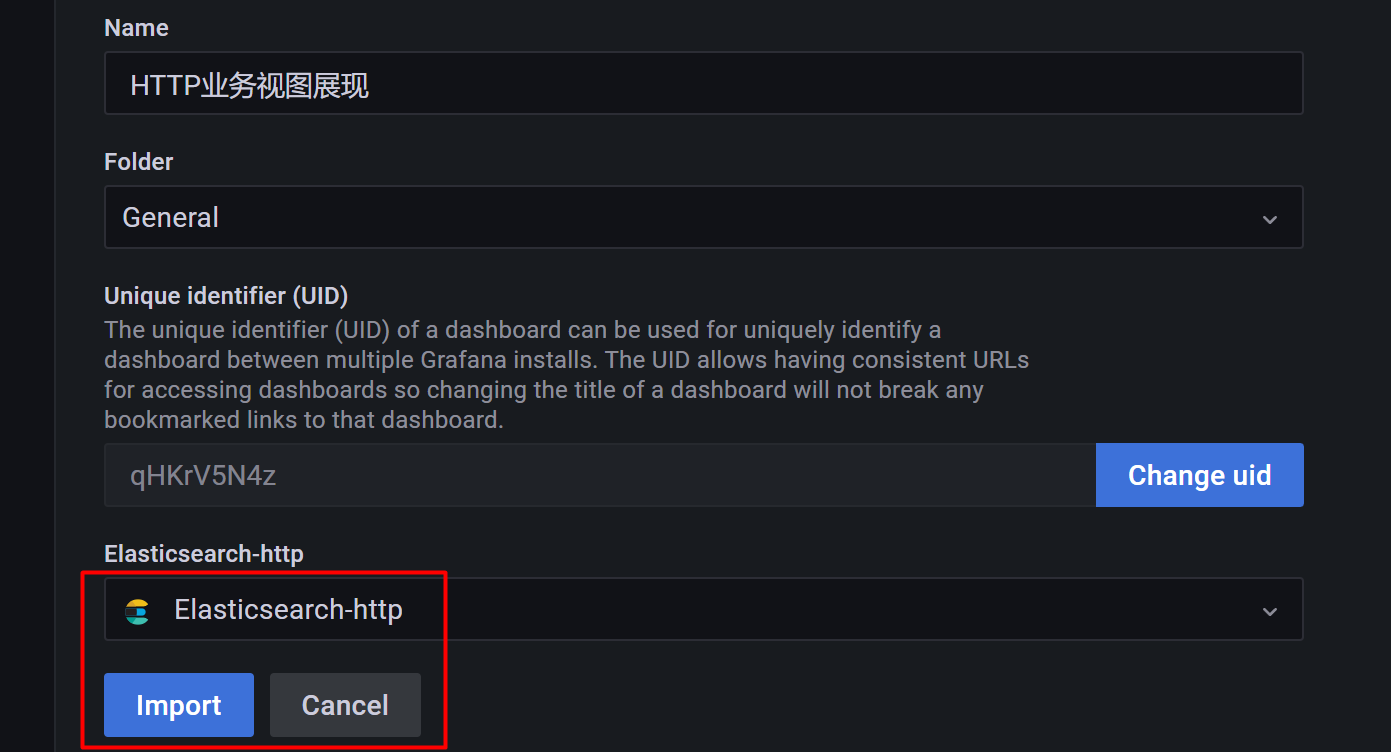
点击 +，点击Import进入Dashboard导入界面。



点击Upload JSON file，选择需要导入的dashboard模板文件，并点击Load。



下方Select a Prometheus data source选择Elasticsearch-http，并点击Import导入。



### 模板对应的数据源

|  |  |  |
| --- | --- | --- |
| **Dashboard模板名称** | **Grafana数据源** | **功能** |
| F5设备状态监控 | Prometheus | 监控F5设备硬件状态 |
| Container&Node性能监控 | Prometheus | 监控宿主机硬件状态 |
| HTTP业务视图展现 | Elasticsearch-http | 展示F5 HTTP业务态势 |
| DNS解析视图展现 | Elasticsearch-dns | 展示F5 DNS服务态势 |
| TCP-L4延时视图展现 | Elasticsearch-tcp | 展示F5 L4业务建链态势 |
| 安全防护态势视图展现 | Elasticsearch-asm | 展示F5 Awaf安全防护态势 |

# Prometheus监控F5设备

## 生成Snmp.yml文件

Prometheus可以通过snmp-exporter模块监控F5设备，snmp-exporter模块依赖于snmp的mib文件，需要使用snmp\_exporter工具生产snmp.yml文件，我们示例启用一个centos容器来完成所有的工作，具体步骤如下：

1. 运行centos容器

#拉取centos7镜像

docker pull centos:7

#启动容器

docker run -dit --name centos -v /root:/root censos:7

1. 安装依赖包和下载相关组件

#进入容器命令行

docker exec -it centos /bin/sh

#安装依赖包和epel源

yum install -y gcc make net-snmp net-snmp-utils net-snmp-libs net-snmp-devel epel-release

#验证snmp walk可用，需要提前配置F5的snmp

snmpwalk -v 2c -c f5-demo 192.168.101.60

#安装golang和git

yum install -y golang

yum install git -y

#修改go环境变量，添加代理

go env -w GO111MODULE=on

go env -w GOPROXY=https://goproxy.cn,direct

#git拉取snmp\_exporter文件

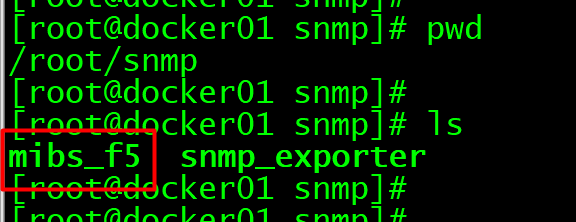
mkdir -p /root/snmp

cd /root/snmp/

git clone https://github.com/prometheus/snmp\_exporter

1. 编译snmp.yml文件

#下载f5的mib文件，并解压缩后拷贝至一个文件夹，并上传至宿主机/root/snmp目录下



#进入容器命令行

docker exec -it centos /bin/sh

#设置环境变量

export MIBDIRS=/root/snmp/mibs\_f5

注：mibs\_f5就是mib文件所在的目录,如果多个目录路径中间用冒号隔开，/root/snmp/mibs\_f501:/root/snmp/mibs\_f502

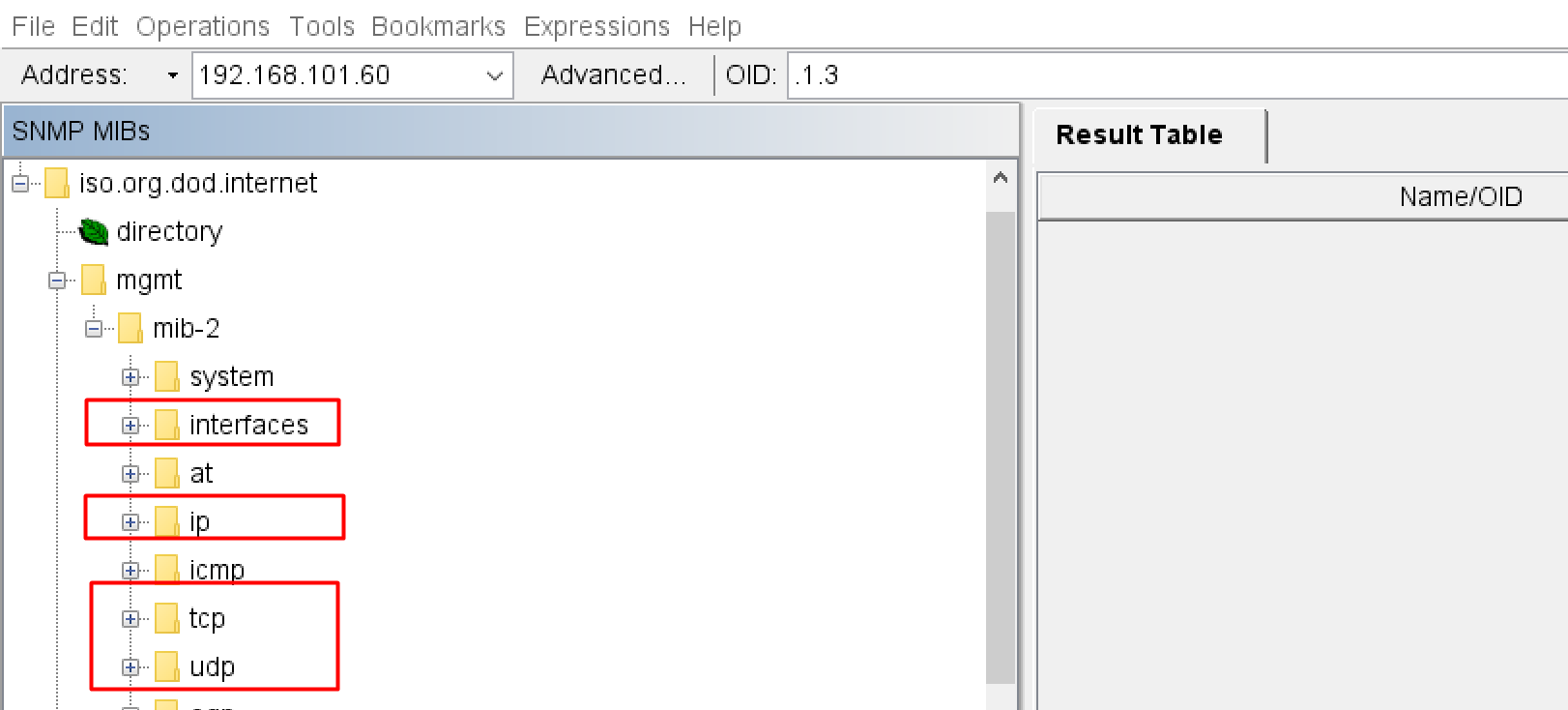
#进入snmp\_exporter/generator目录，开始进行编译

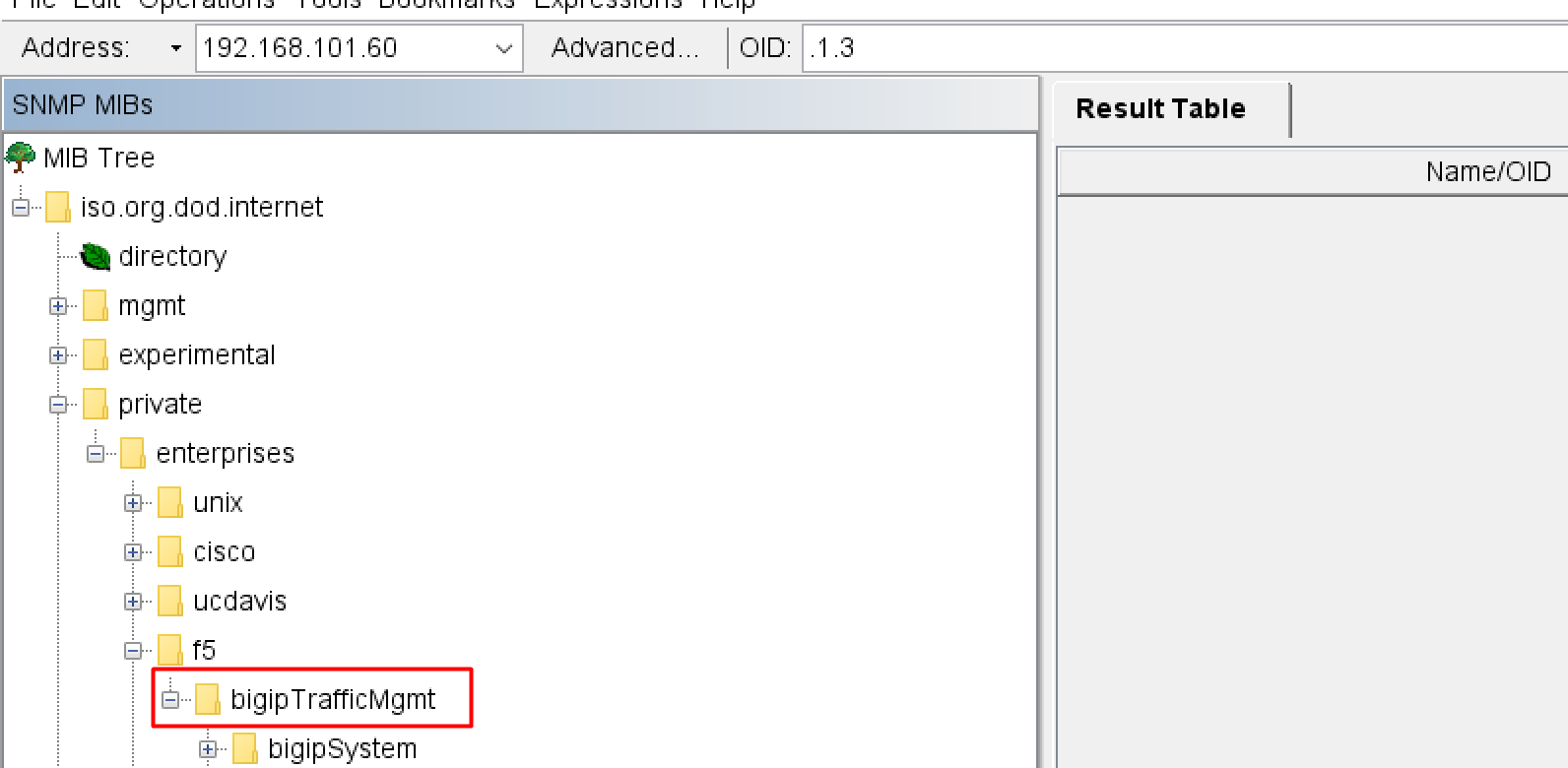
cd /root/snmp/snmp\_exporter/generator

go build

#利用MIB Browser浏览器浏览的信息信息编辑/root/snmp/snmp\_exporter/generator目录下的generator.yml文件

#Browser浏览器浏览的信息信息如下：





以下为只获取上图红框中的mib信息的编辑示例

modules:

# F5-MIB With F5 Device.

f5\_mib:

walk:

- bigipTrafficMgmt

- system

- interfaces

- ip

- tcp

- udp

lookups:

- source\_indexes: [ifIndex]

lookup: ifAlias

- source\_indexes: [ifIndex]

# Uis OID to avoid conflict with PaloAlto PAN-COMMON-MIB.

lookup: 1.3.6.1.2.1.2.2.1.2 # ifDescr

- source\_indexes: [ifIndex]

# Use OID to avoid conflict with Netscaler NS-ROOT-MIB.

lookup: 1.3.6.1.2.1.31.1.1.1.1 # ifName

overrides:

ifAlias:

ignore: true # Lookup metric

ifDescr:

ignore: true # Lookup metric

ifName:

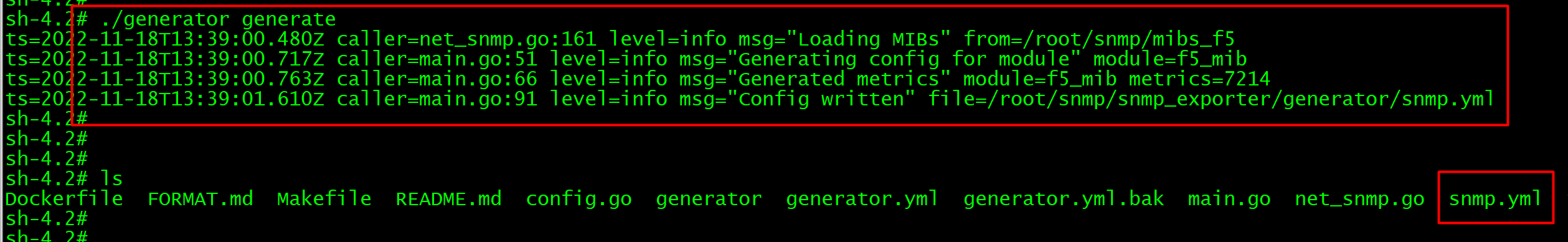
ignore: true # Lookup metric

ifType:

type: EnumAsInfo

#编辑完成，使用如下命令生成snmp.yml文件

./generator generate



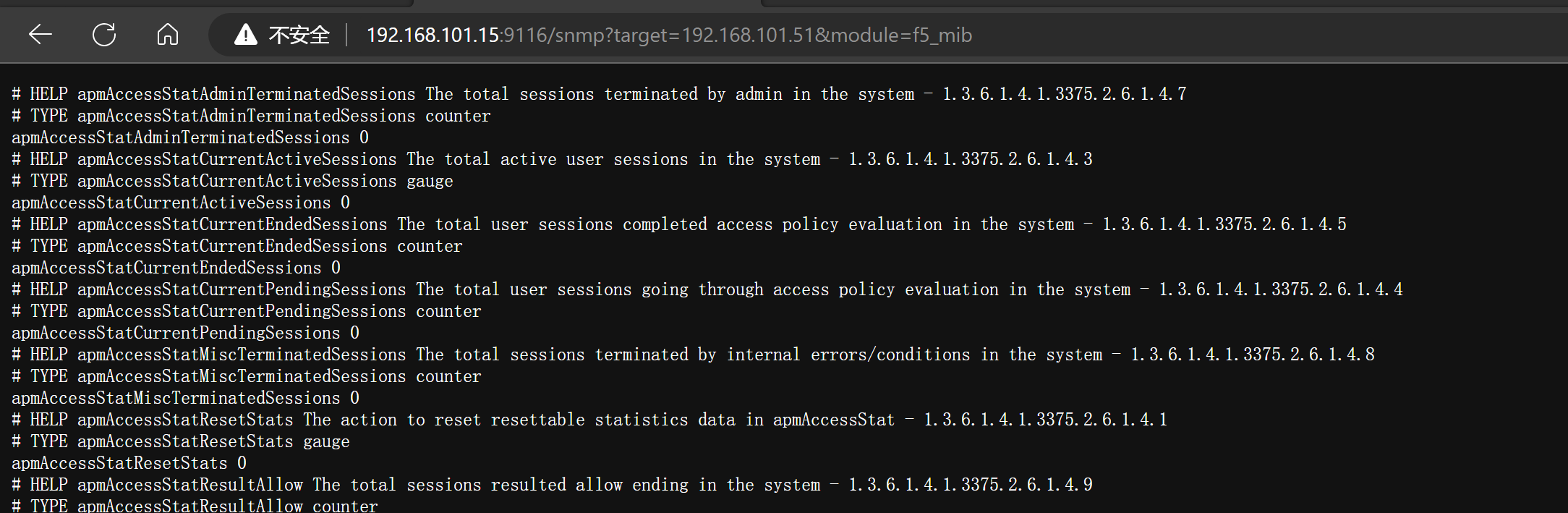
#将生成的snmp.yml文件拷贝至/opt/snmp\_exporter,并启动snmp-exporter容器

1. 验证snmp-exporter容器是否正常

访问snmp-exporter的9116端口，输入F5 IP及modules名，点击提提交。



正常获取到F5的oid信息。



## Prometheus配置Snmp监控

### Prometheus.yml配置

#编辑prometheus.yml文件添加snmp监控的job

vi /opt/prom/prometheus.yml

- job\_name: "snmp"

scrape\_interval: 30s #刷新间隔默认10s

scrape\_timeout: 30s #超时时间，snmp\_exporter刷数据慢修改大一点

static\_configs:

- targets:

- 192.168.101.51 #F5设备IP地址

- 192.168.101.52

- 192.168.101.60

- 10.1.10.100

metrics\_path: /snmp

params:

module: [f5\_mib] #generator.yml自定义文件的模块名

relabel\_configs:

- source\_labels: [\_\_address\_\_]

target\_label: \_\_param\_target

- source\_labels: [\_\_param\_target]

target\_label: instance

- target\_label: \_\_address\_\_

replacement: 172.16.10.15:9116 # snmp\_exporter 服务IP地址

#编辑完成，重启Prometheus容器

docker restart prometheus

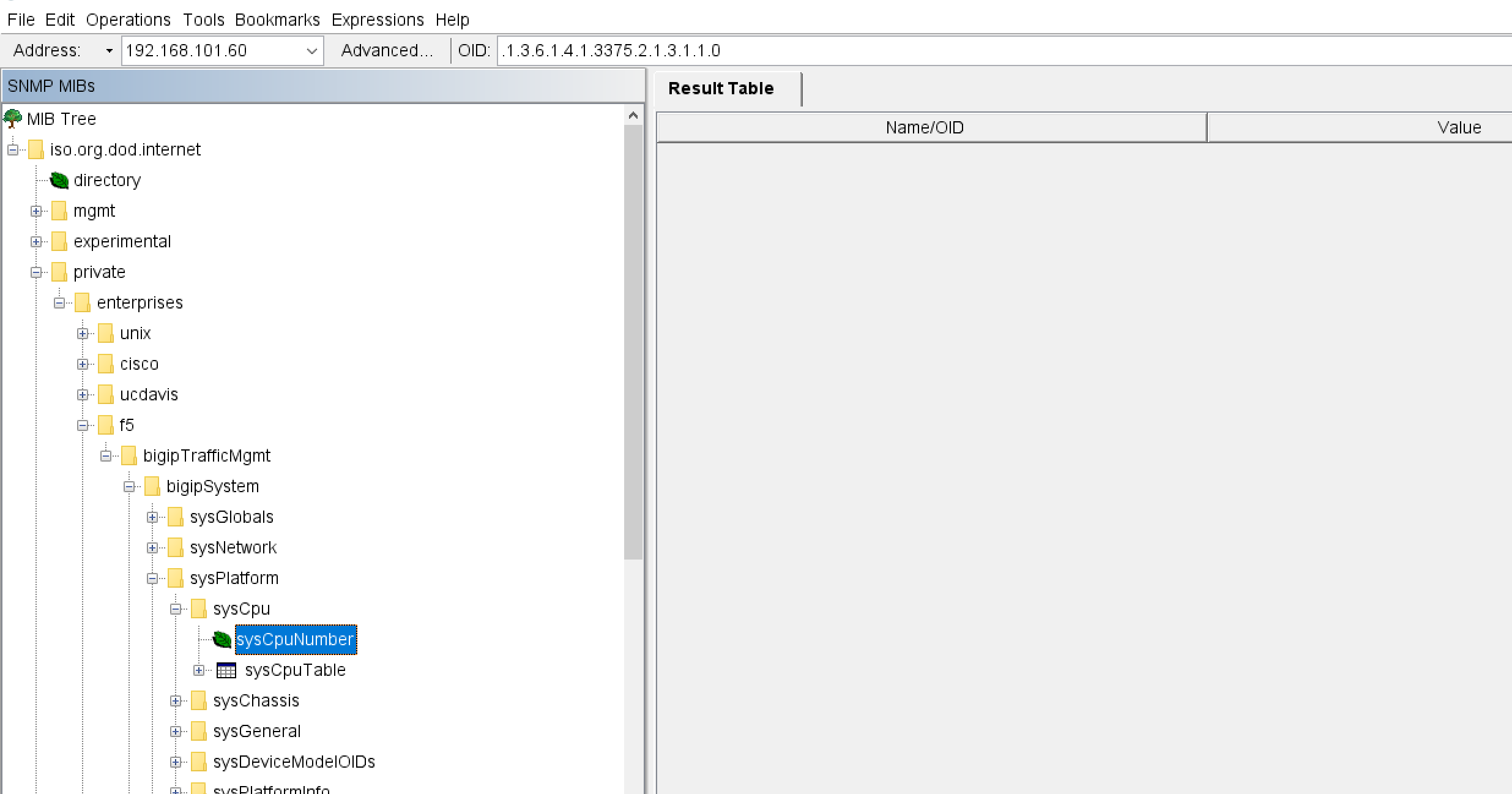
访问http://172.16.10.15:9090/targets查看当前添加设备的SNMP状态



### Mib信息查看

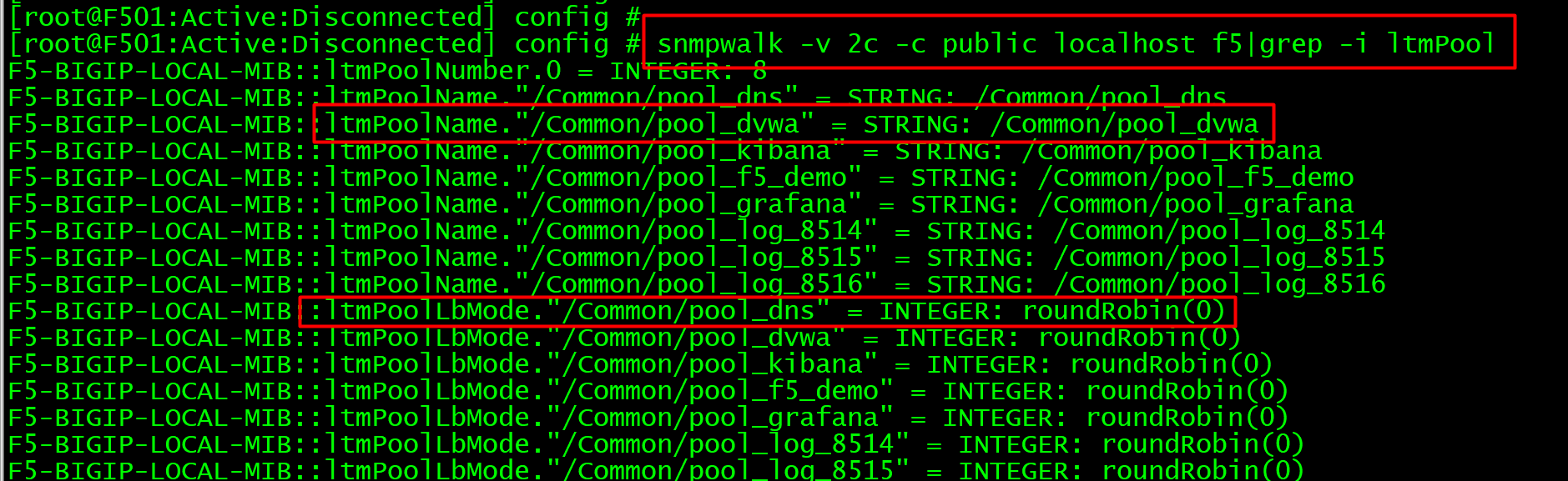
F5设备监控是通过prometheus的snmp-exporter模块，实时读取F5设备对应功能的snmp mib信息，并根据采集到的值进行展示。

F5设备的mib信息可以通过MIB Browser进行查看：



也可以通过snmpwalk命令进行查看，如下是过滤所有包括ltmPool字段的snmp mib信息

snmpwalk -v 2c -c public localhost f5|grep -i ltmPool



## Grafana可视化

### Dashboard模板介绍

F5设备状态监控Dashboard模板分别从硬件状态、接口状态、Virtual\_Server状态、Pool&Member状态四个维度展示F5的设备状态，并设置变量进行选择和查询



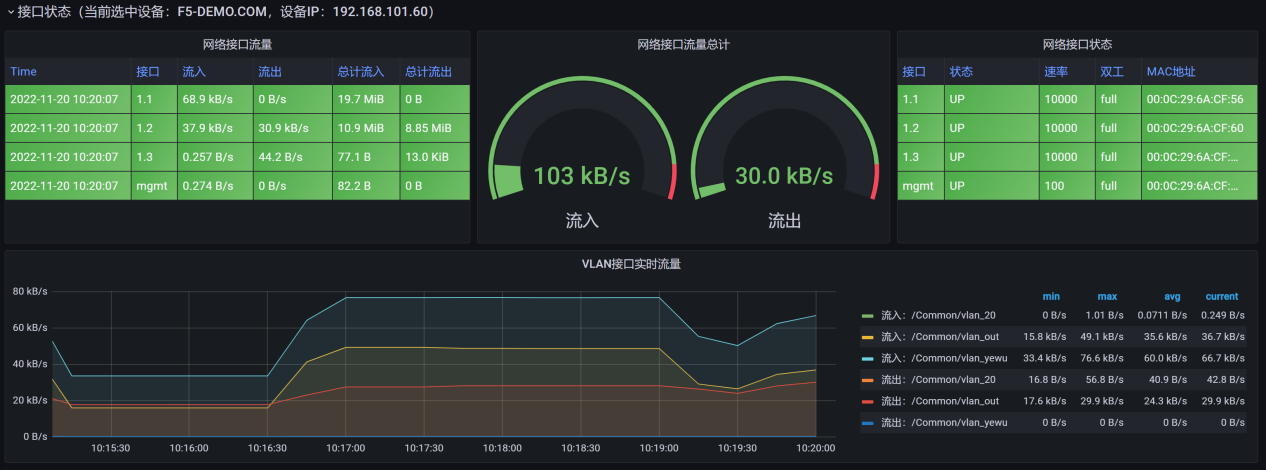
1. 硬件状态

硬件状态分别展示了设备工作状态、主备机同步状态、CPU核心使用、内存使用、CPU全局使用、开机时间、TMM CPU使用等各项指标



1. 接口状态

接口状态分别展示了网络接口流量、网络接口流量总计、网络接口状态、VLAN接口实时流量等各项指标。



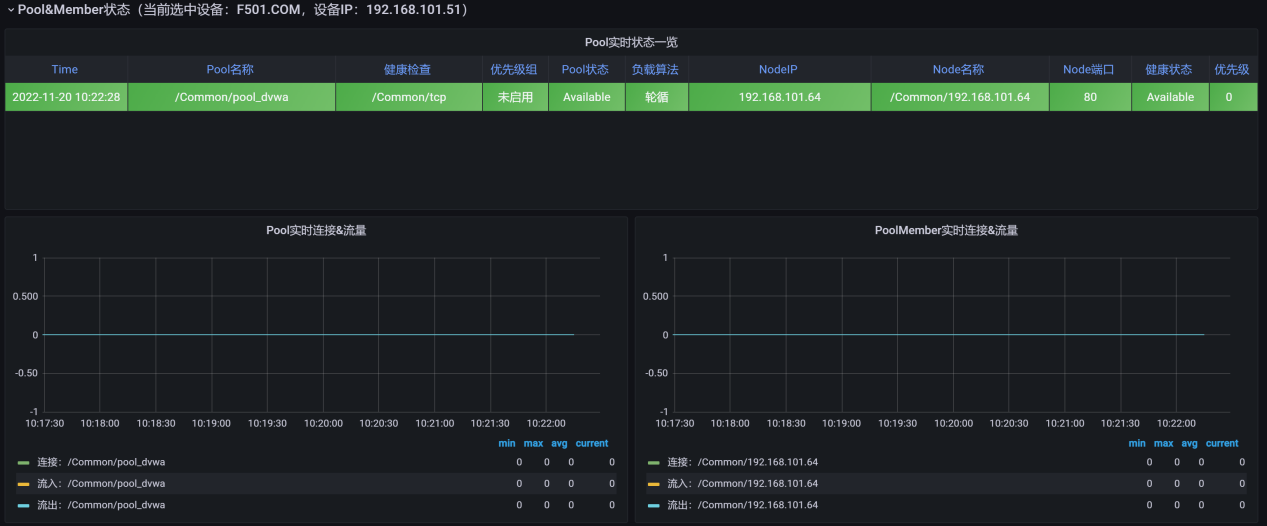
1. Virtual\_Server状态

Virtual\_Server状态分别展示了VS实时配置状态一览、VS实时访问流量、VS实时访问连接等各项指标



1. Pool\_Member状态

Pool\_Member状态分别展示了这Pool实时配置状态一览、Pool实时连接&流量、PoolMember实时连接&流量等各项指标

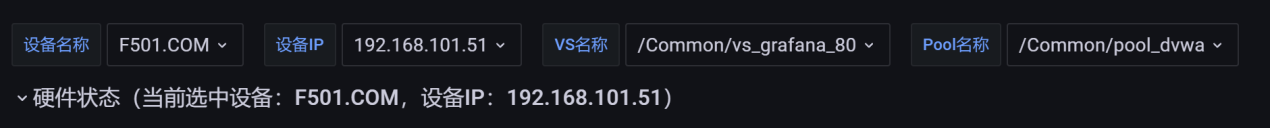


### Grafana视图制作

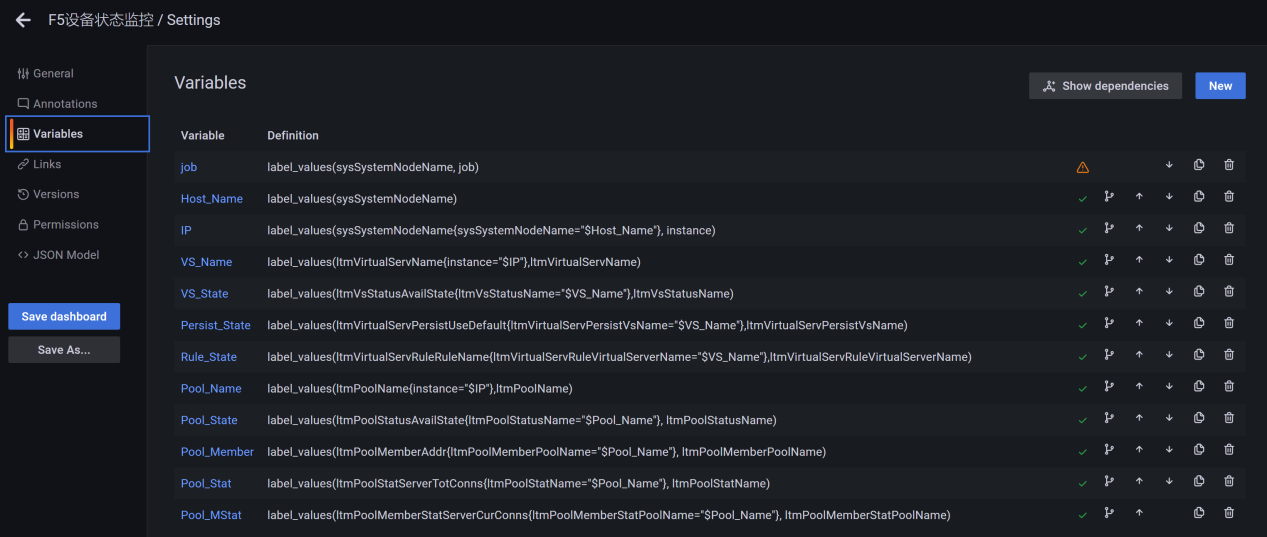
#### Dashboard变量设置

Grafana的Variables变量提供了用户和面板交互，并动态刷新面板的功能，变量的下拉菜单显示在面板的顶部，这样改变下拉菜单的值，即可改变变量的值，而且所有使用该变量的仪表板都会随着变量的改变而改变。

如下图中，设置变量使用后，我们选择要查看的设备名称，这台设备的IP和其上部署的VS和Pool也一并被调出，下方硬件状态列也会同步提示当前查看的设备名和IP。



Dashboard模板共设置了如下变量，其中Host\_Name、IP、VS\_Name、Pool\_Name未做隐藏并设置了别名，在Dashboard上方提供过滤选择，其它变量做了隐藏处理，各变量之间根据需要进行了关联处理。



以下挑选部分变量介绍变量的设置方法：

**变量设置原始数据查看**

查看原始数据，sysSystemNodeName，获取到的信息有instance、job、sysSystemNodeName三个。



**job变量设置如下：**

现在我们需要将job名作为变量取出来

变量名称：job

Type：Query

Hide：Varible #设置为该变量隐藏不展示

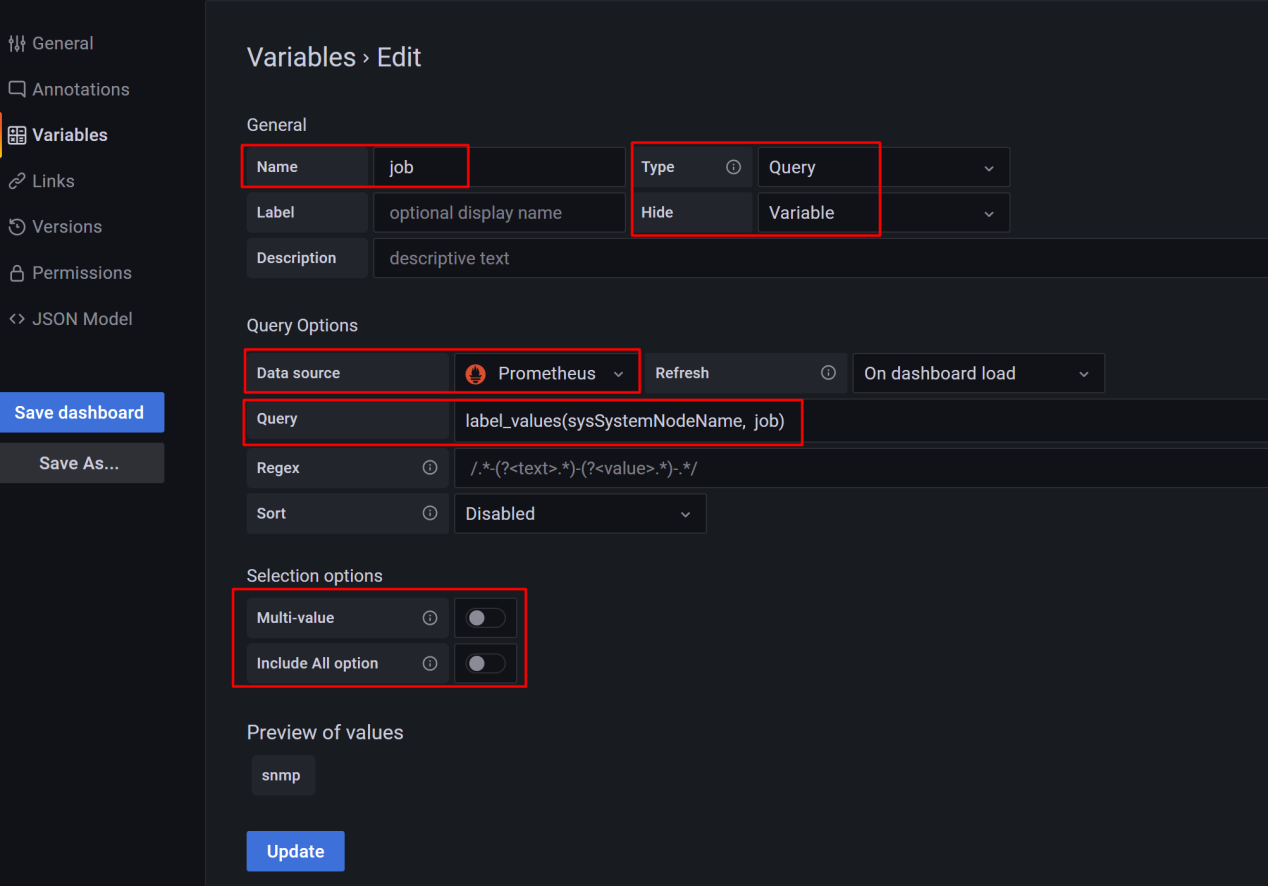
Data source：Promethues #数据源设置为prometheus

Query：label\_values(sysSystemNodeName, job)

#通过查询sysSystemNodeName的结果，并将结果中job的值取出作为变量的值

Multi-value：是否包含可选择的多个值

Include All option：是否包含可选择的所有值



**Host\_Name变量设置如下：**

现在我们需要将sysSystemNodeName的值作为变量取出来

变量名称：Host\_Name

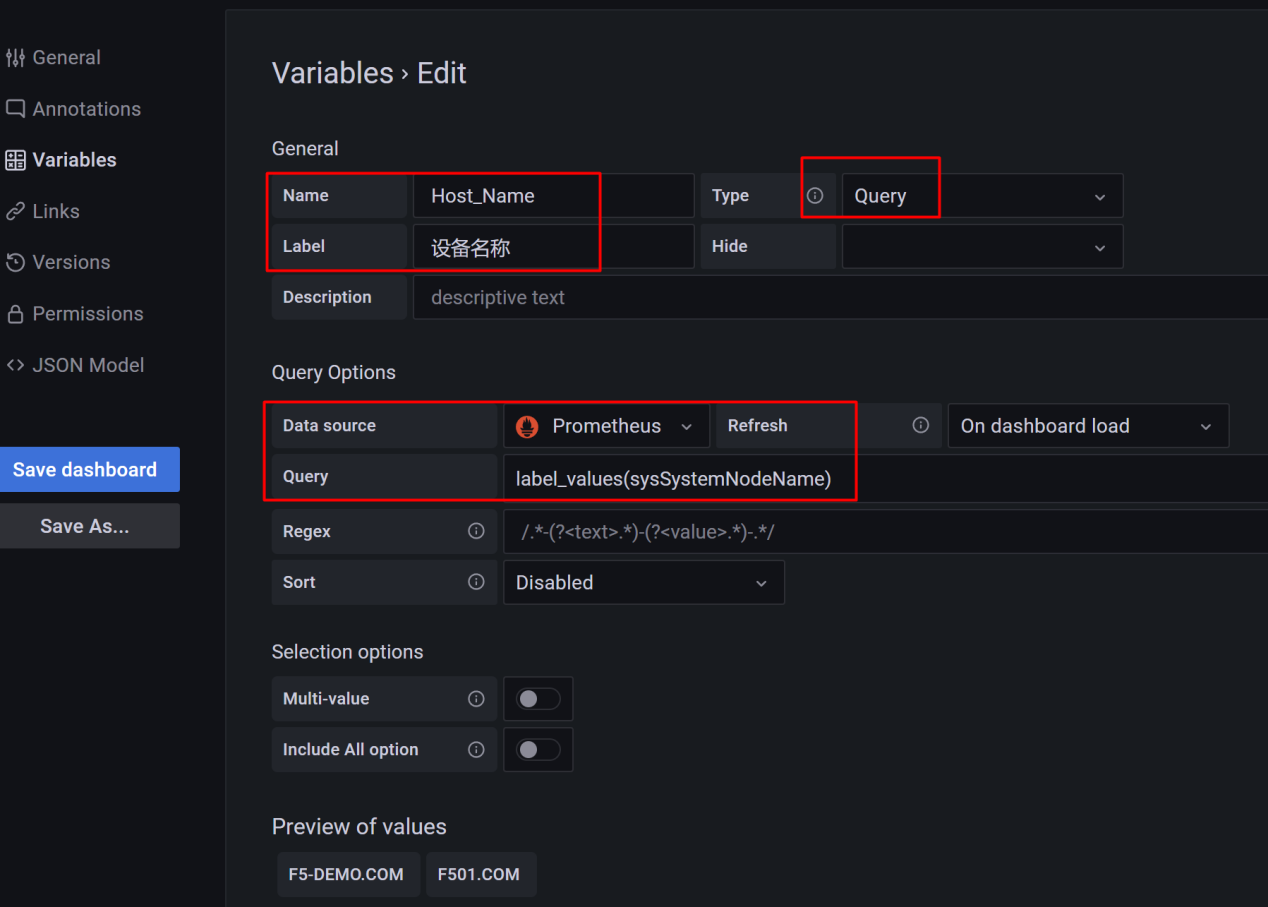
Lable：设备名称 #设置变量标签名，Dashboard上面的显示名称

Type：Query

Data source：Promethues #数据源设置为prometheus

Query：label\_values(sysSystemNodeName)

#通过查询sysSystemNodeName的结果，并将结果中sysSystemNodeName的值取出作为变量的值



**IP变量设置如下：**

现在我们需要将instance的值作为变量取出来，关与Host\_Name变量关联。

变量名称：IP

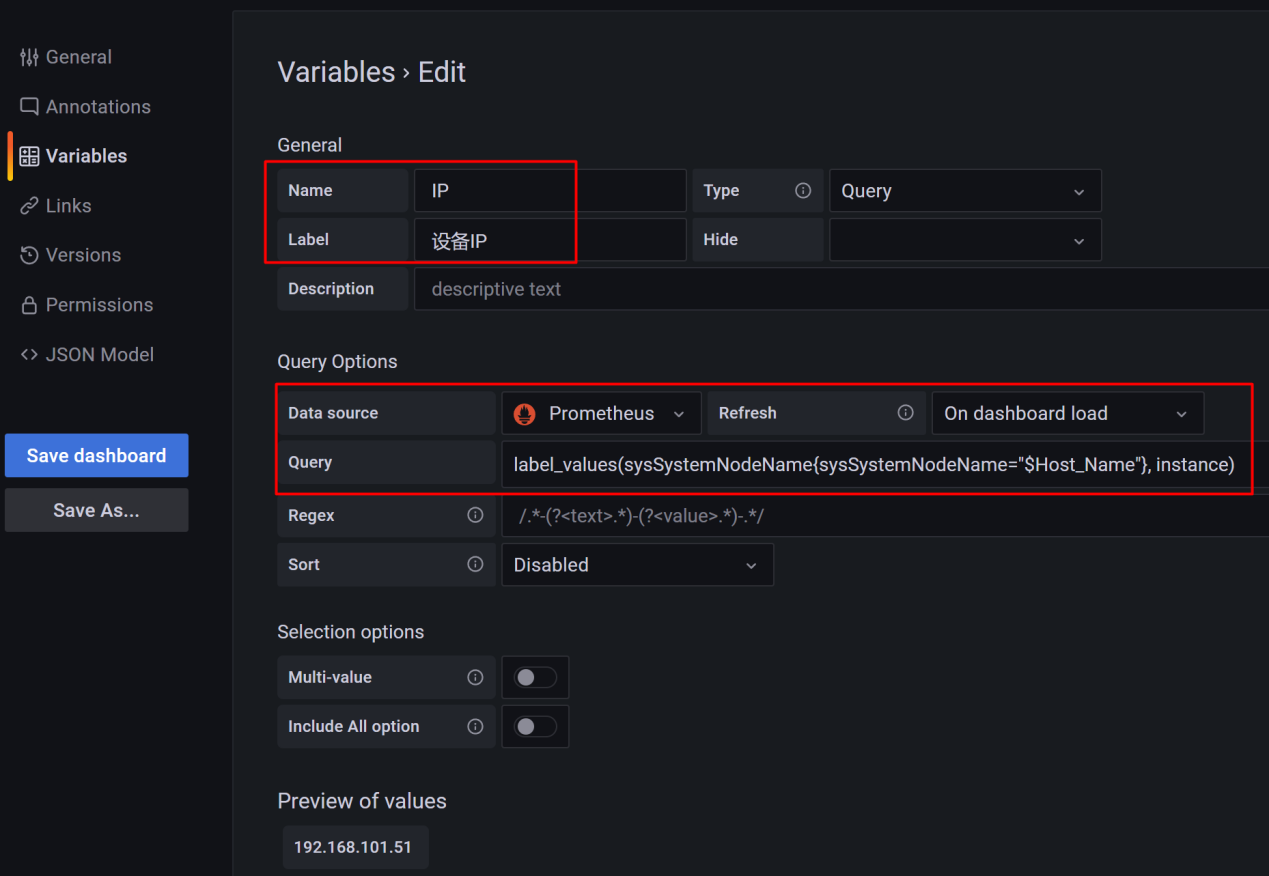
Lable：设备IP #设置变量标签名，Dashboard上面的显示名称

Type：Query

Data source：Promethues #数据源设置为prometheus

Query：label\_values(sysSystemNodeName{sysSystemNodeName="$Host\_Name"}, instance)

#通过查询sysSystemNodeName的结果，并将结果中instance的值取出作为变量的值，同时该变量设置同Host\_Name的变量进行了关联，意思是该instance的变量选择，依赖于Host\_Name变量的选择，设置格式参考{}中内容。



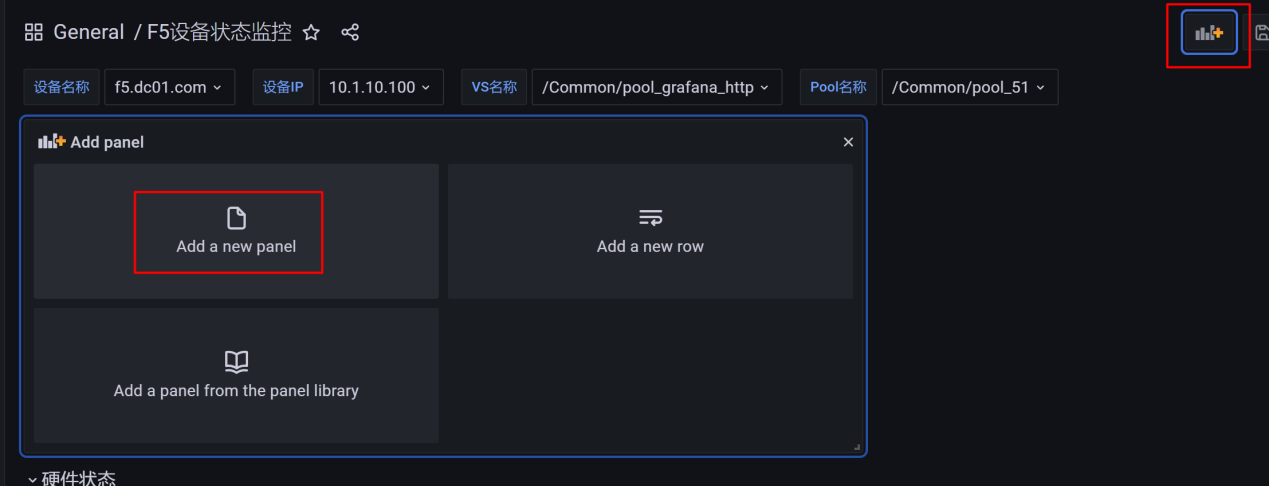
**变量关联效果如下：**

设备名称选择后，后面的设备IP也会变为对应的IP，因为后面的VS名称变量和Pool名称也做了关联，其上可选择的VS和Pool均为该台设备配置有的VS和Pool



#### Dashboard Panel制作示例

Dashboard panel就是Dashboard展示的一个个视图，可以通过点击添加一个新的panel。



以CPU核心使用监控为例：

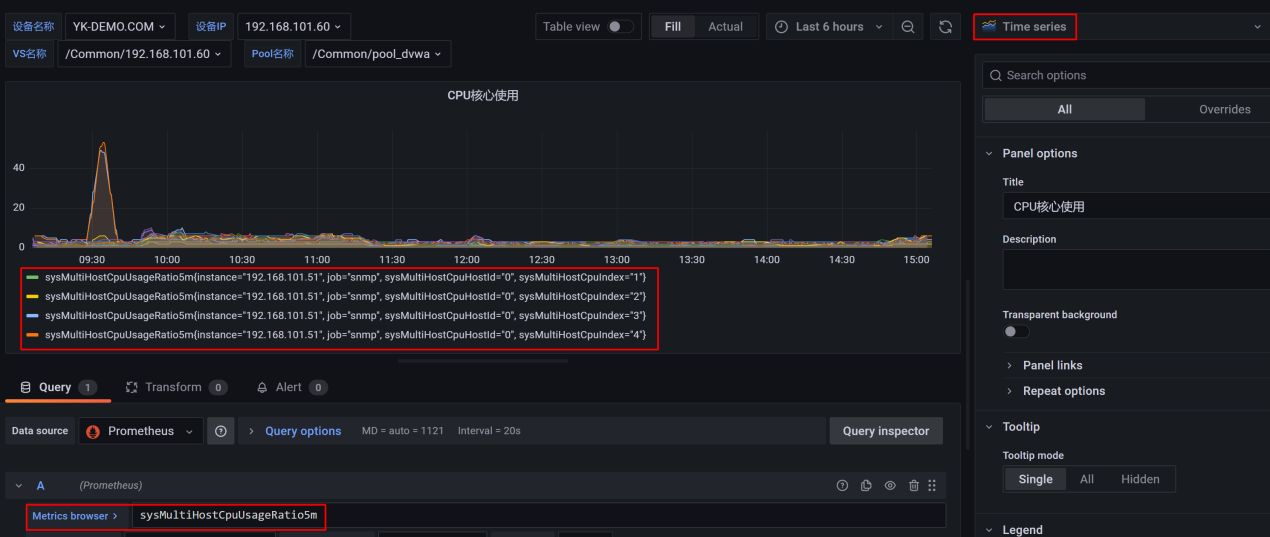
Metrics browser配置sysMultiHostCpuUsageRatio5m查看获取到的信息CPU各核心使用。

如上图，获取到的直观信息有

instance： 设备IP

job： Prometheus的job

sysMultiHostCpuIndex： CPU核心



鼠标移至图中线条上，可以查看当前的某一核心的使用率



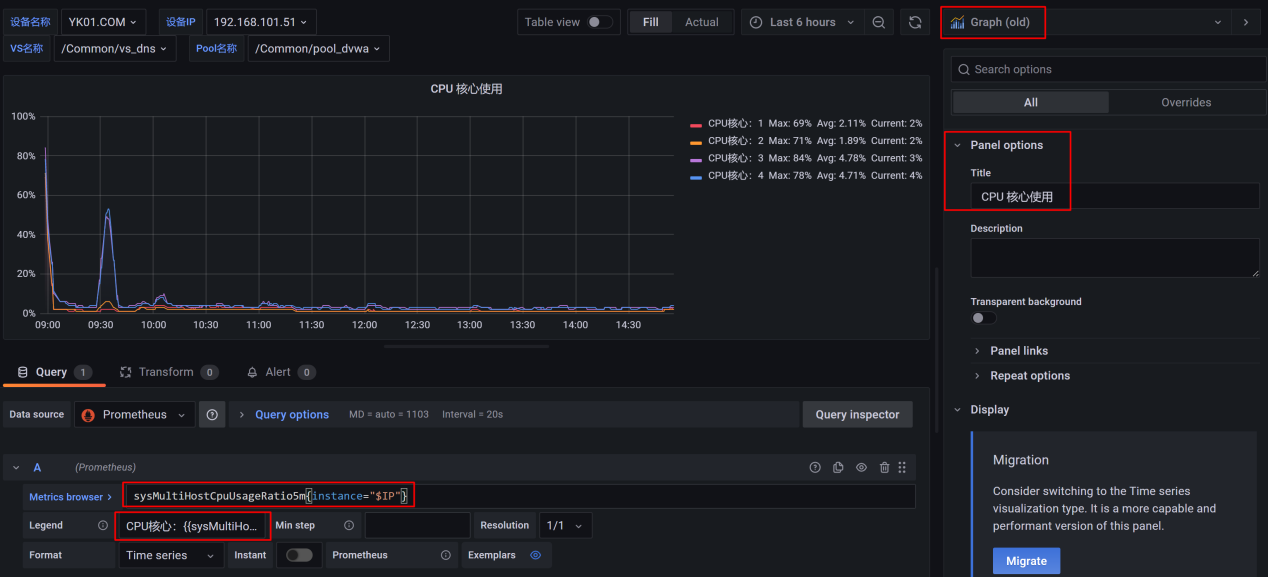
定制化设置：

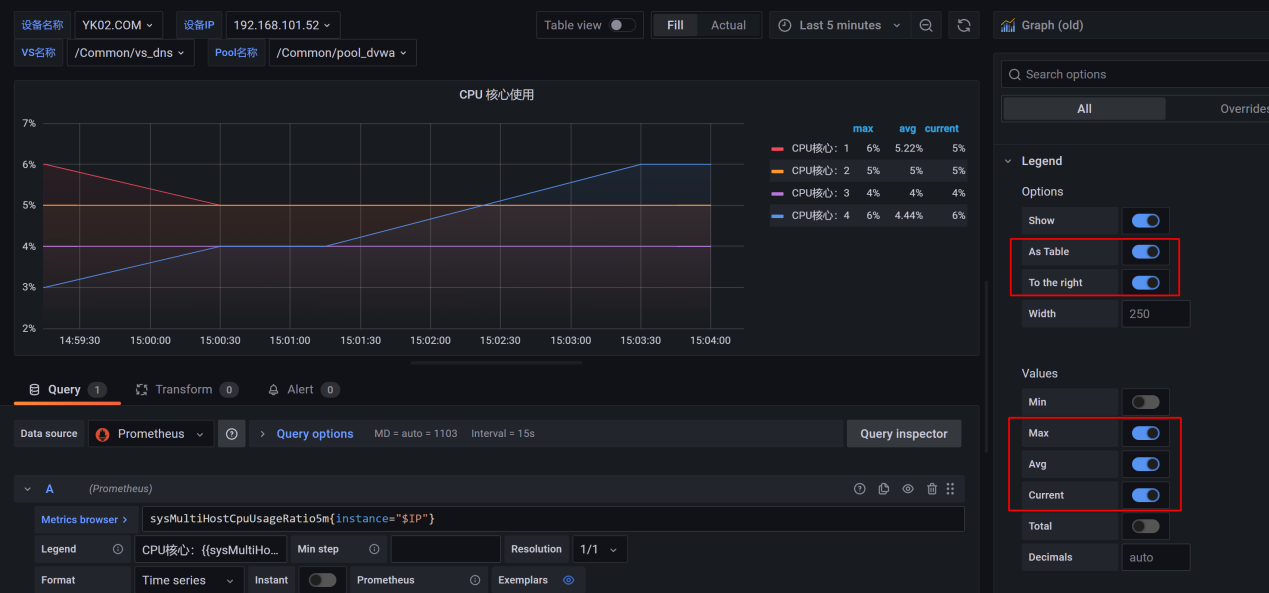
图表选择：Graph(old)

Panel option Title：名称显示为“CPU核心使用”

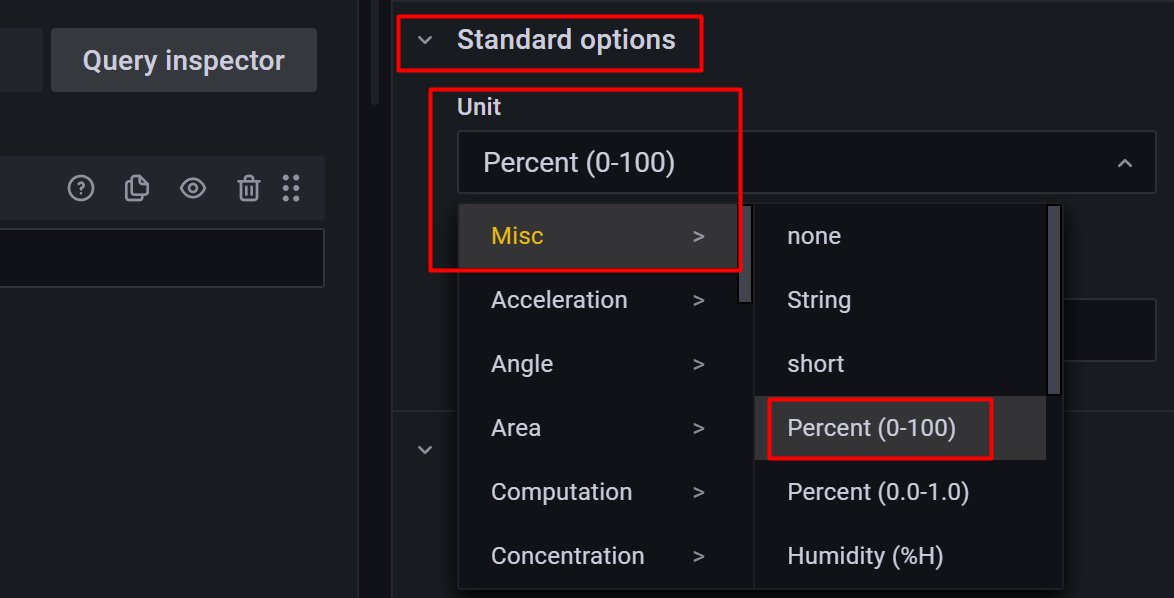
Metrics browser：sysMultiHostCpuUsageRatio5m{instance="$IP"}，{}引用变量。

Legend：CPU核心：{{sysMultiHostCpuIndex}}，显示内容为CPU核心{{核心}}

Legend设置启用As Table，靠右显示，显示Max（最大）、Avg（平均）、Current（当前）共三个使用数值。



Standard options设置显示百分比。



# Grafana展示ES视图

### Dashboard模板介绍

Elasticsearch共有四个数据源，共四个模板。分别对应基于F5 HSL的dns\http\tcp\asm四类业务日志。F5和ES配置请参考林静大神的giuhup

https://github.com/myf5/f5-elk-demo/



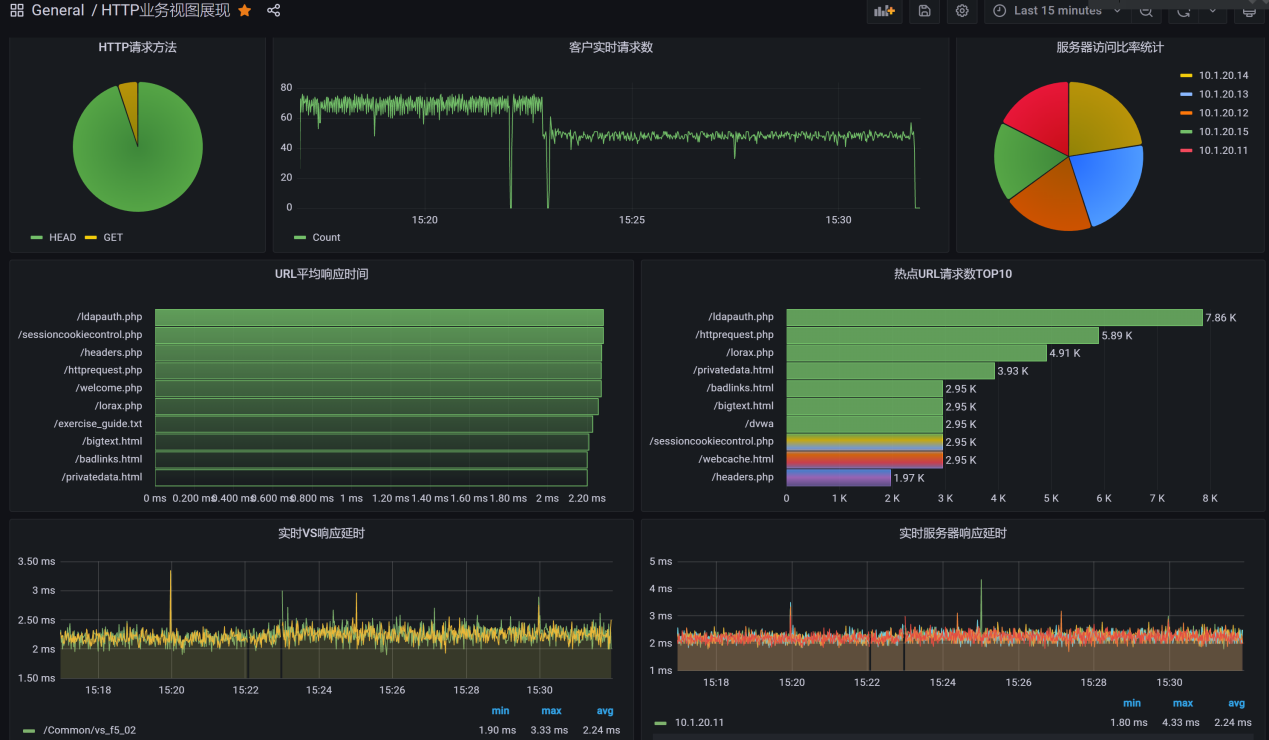
1. DNS业务视图展现

实时获取F5 DNS解析数据，展视DNS解析视图



1. HTTP业务视图展现

实时获取F5 HTTP业务访问态势，展示HTTP业务访问视图



1. TCP-L4延时视图展现

实时获取F5 Performance L4访问TCP建链信息，展示TCP连接视图



1. 安全防护态势展现

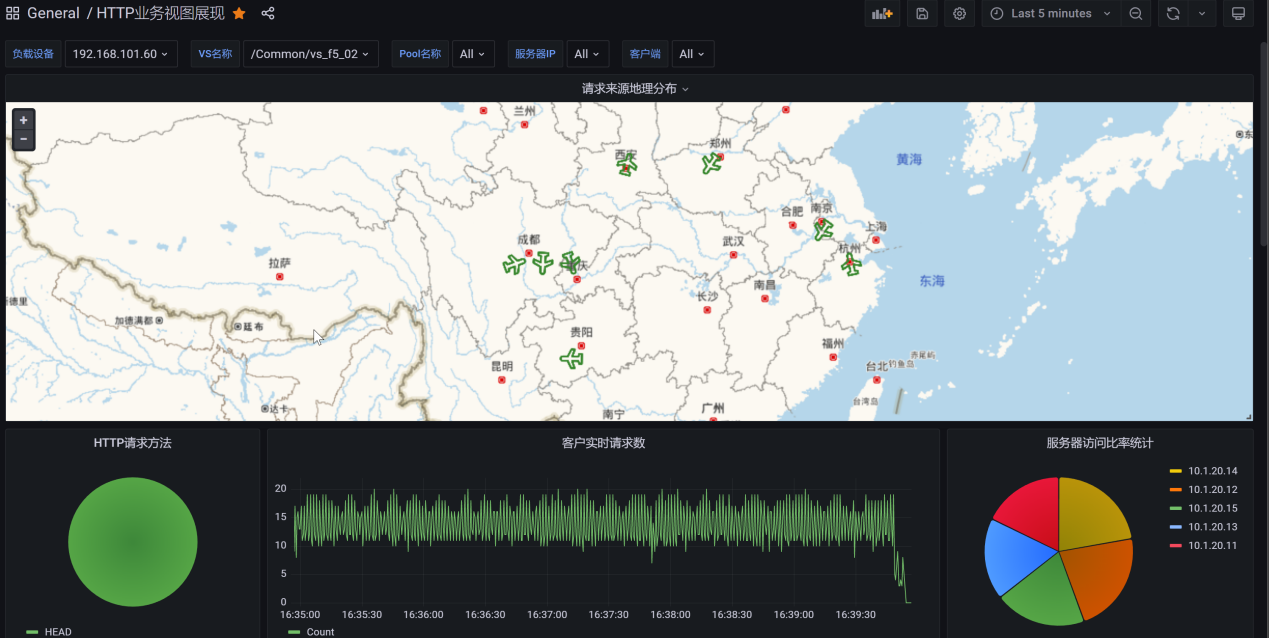
实时获取F5 Awaf安全事件信息，展示安全防护态势



### Grafana视图制作

#### Dashboard变量设置

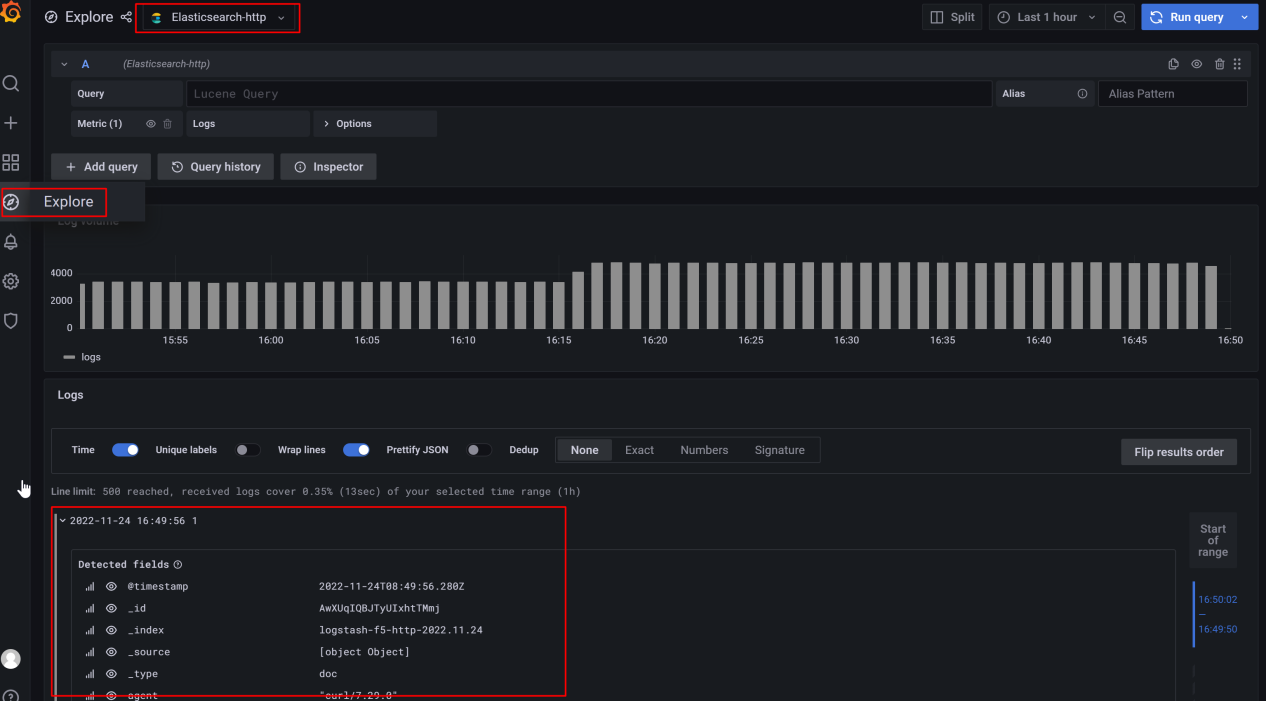
与F5设备监控类似，也在Dashboard中设置了变量，如下图中，设置变量使用后，我们选择要查看的负载设备名称，这台设备的IP和其上部署的VS和Pool也一并被调出



Elasticsearch和Prometheus因为数据源不同，所以变量的配置参数也不同，以下挑选部分变量介绍变量的设置方法：

**Elasticsearch原始数据查看**

可以点击，选择Explore，数据源选择Elasticsearch-\*查看Elasticsearch原始数据。



**负载设备变量设置如下：**

现在我们需要将F5的设备IP作为负载设备选择的变量进行设置

Name：host #变量名称

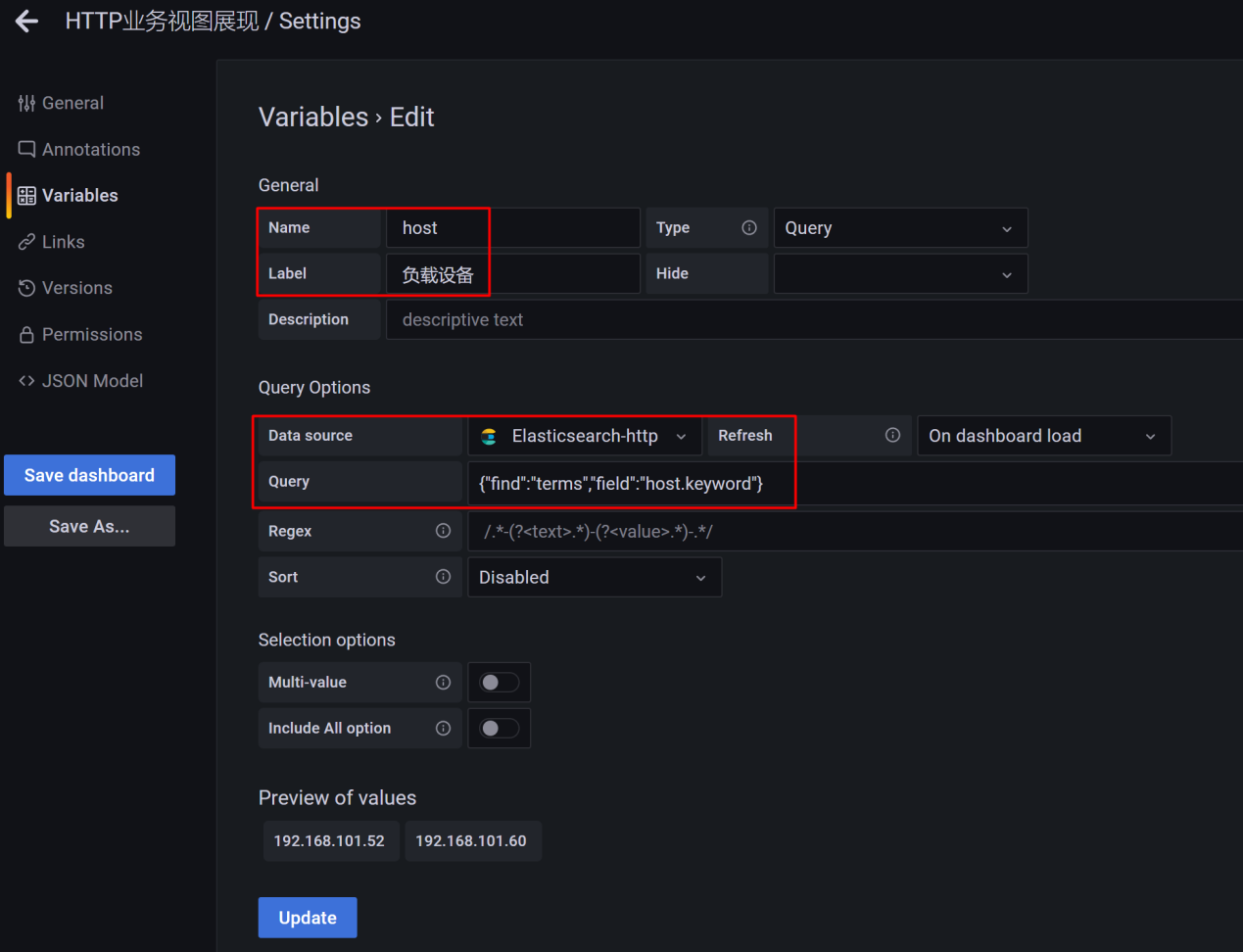
Type：Query

Lable：负载均衡 #该变量在Dashboard中的显示名称

Data source：Elasticsearch-http #数据源设置为Elasticsearch-http

Query：{"find":"terms","field":"host.keyword"}

#通过查找host.keyword关键字的结果，并将结果中的值取出作为变量的值



**VS名称变量设置如下：**

现在我们需要将VS Name的值作为变量取出来，关与负载设备变量关联。

变量名称：vs

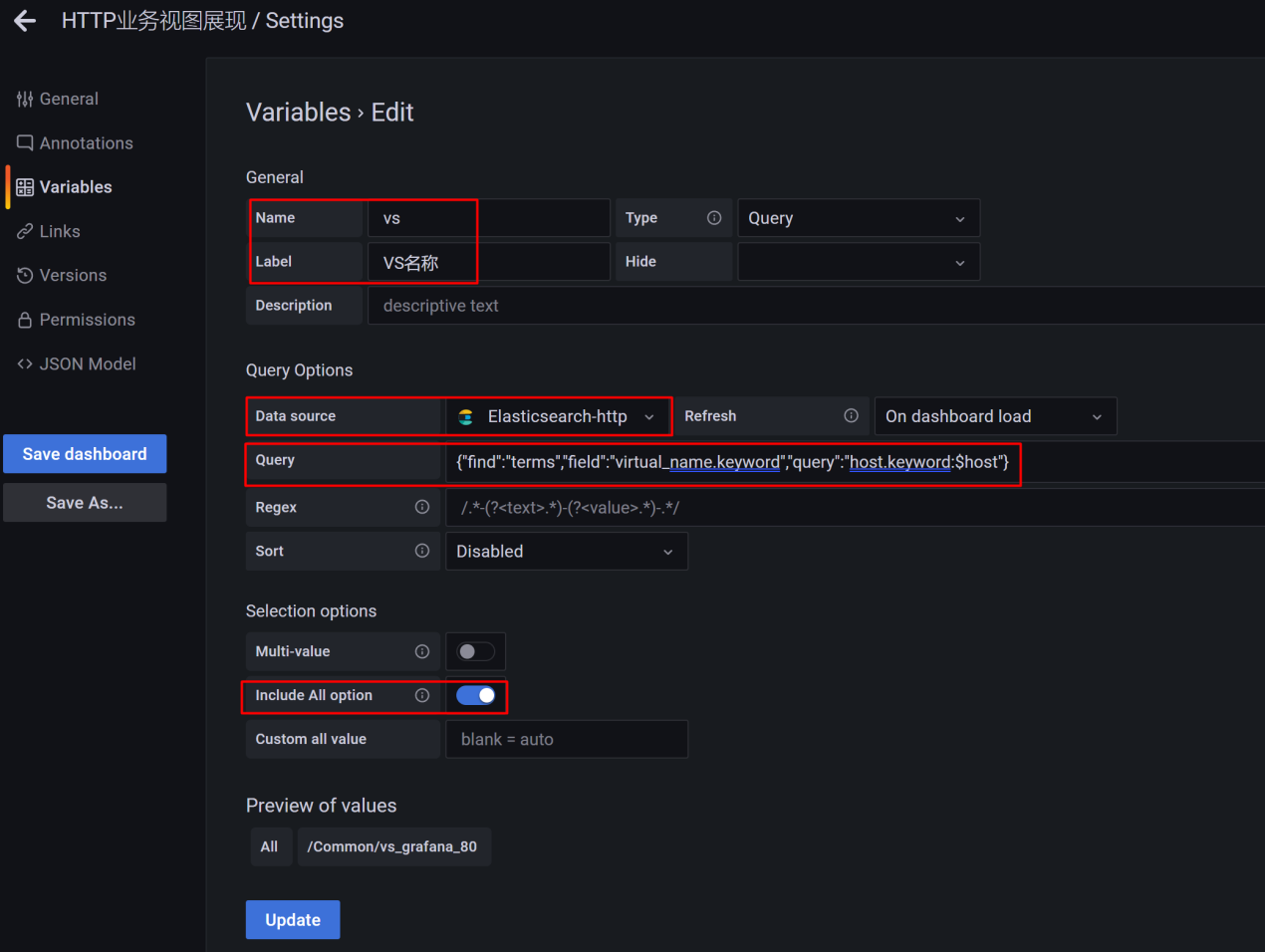
Lable：VS名称 #设置变量标签名，Dashboard上面的显示名称

Type：Query

Data source：Elasticsearch-http #数据源设置为Elasticsearch-http

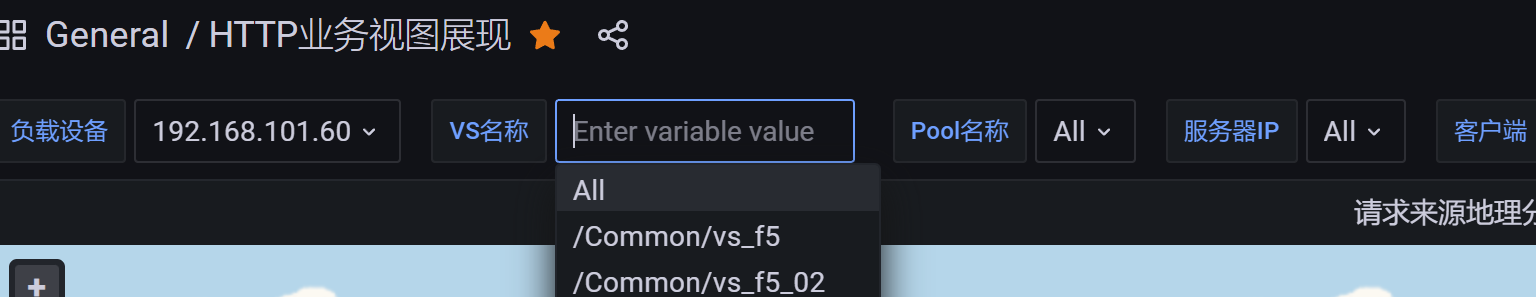
Query：{"find":"terms","field":"virtual\_name.keyword","query":"host.keyword:$host"}

#通过查找virtual\_name.keyword关键字的结果，并将结果中的值取出作为变量的值，同时该变量关联名为host的变量“,"query":"host.keyword:$host"”。



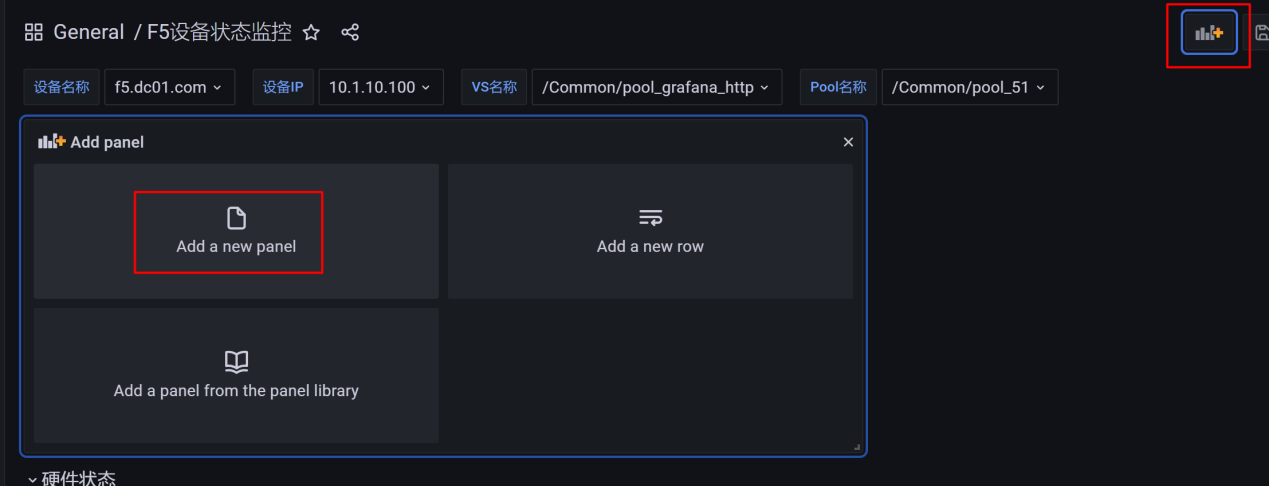
**变量关联效果如下：**

负载设备选择后，后面的VS名称也会自动调用该设备上部署的VS，因为后面的Pool名称变量和服务器IP变量也做了关联，其上可选择的Pool和服务器均为该台设备配置有的VS和Pool以及服务器。

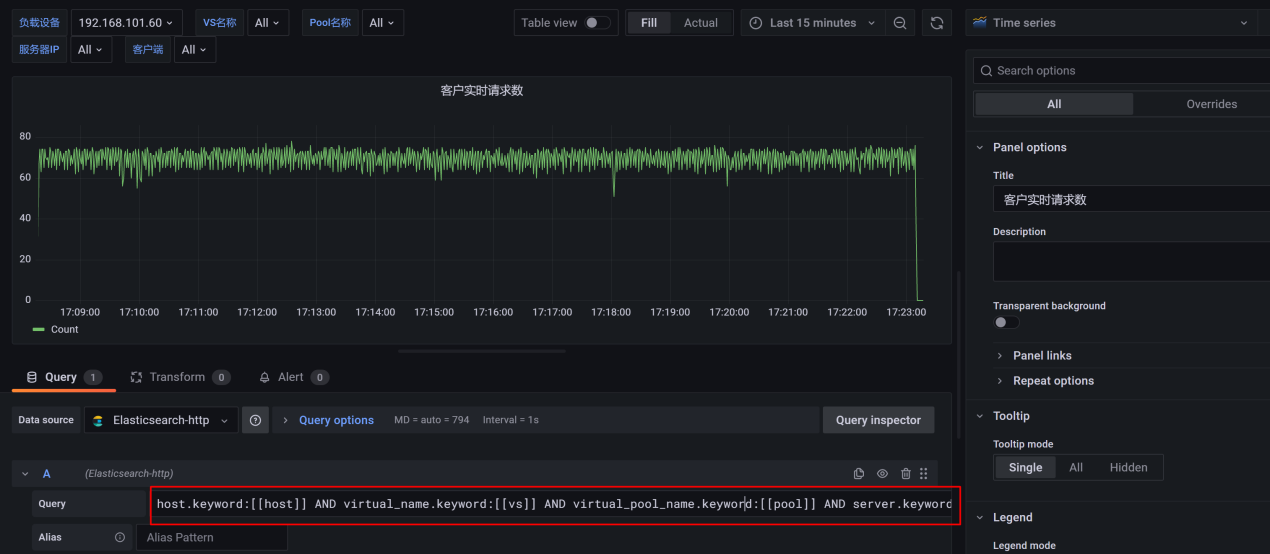


#### Dashboard Panel制作示例

Dashboard panel就是Dashboard展示的一个个视图，可以通过点击添加一个新的panel。



panel设置参考kibana，设置对应的关键字的筛选即可，这里介绍一下调用变量的用法，es数据变量调用是在Query项中配置，使用lucence语法，参考下图，该pancel调用了host\vs\pool\server\client变量，格式参考如下，每个变量中间需要使用大写的AND。  
host.keyword:[[host]] AND virtual\_name.keyword:[[vs]] AND virtual\_pool\_name.keyword:[[pool]] AND server.keyword:[[server]] AND clientip.keyword:[[client]]



# Prometheus监控宿主机和容器

Prometheus监控宿主机和容器使用node-exporter和cadvisor组件完成，参考容器部署章节，启用两个容器即可

## 宿主机监控Prometheus.yml配置

#编辑prometheus.yml文件添加node\_exporter监控的job

- job\_name: 'node\_exporter'

static\_configs:

- targets:

- '192.168.101.15:9100' #运行node-exporter的服务器

- '192.168.101.31:9100'

relabel\_configs:

- source\_labels: [\_\_address\_\_]

target\_label: addr

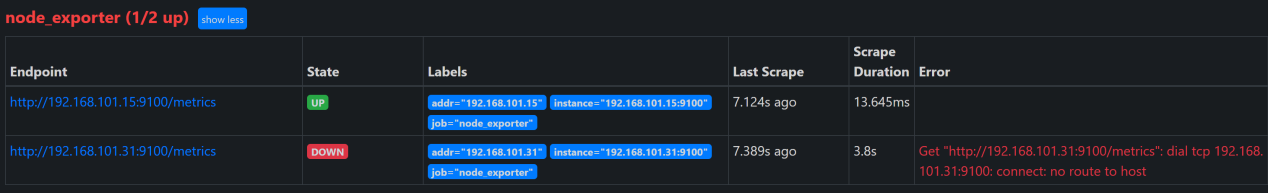
regex: (.\*):(.\*)

replacement: $1

#编辑完成，重启Prometheus容器

docker restart prometheus

访问http://172.16.10.15:9909/targets查看当前添加设备的node-exporter状态



## 容器监控Prometheus.yml配置

#编辑prometheus.yml文件添加node\_exporter监控的job

- job\_name: 'docker'

static\_configs:

- targets:

- '192.168.101.15:8080' #运行cadvisor的服务器

- '192.168.101.31:8080'

relabel\_configs:

- source\_labels: [\_\_address\_\_]

target\_label: addr

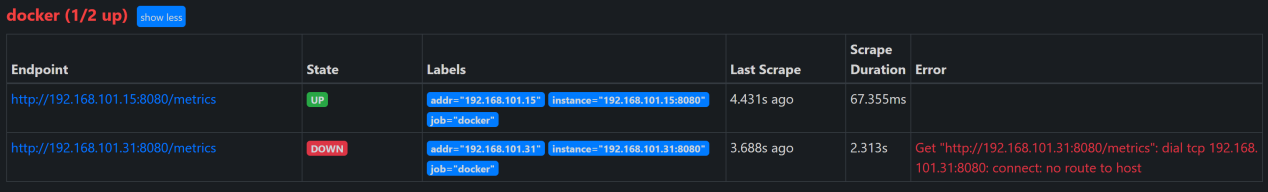
regex: (.\*):(.\*)

replacement: $1

#编辑完成，重启Prometheus容器

docker restart prometheus

访问http://172.16.10.15:9909/targets查看当前添加设备的容器监控状态



## 导入宿主机和容器监控模板

参考模板导入，导入模板（该模板为官网下载）

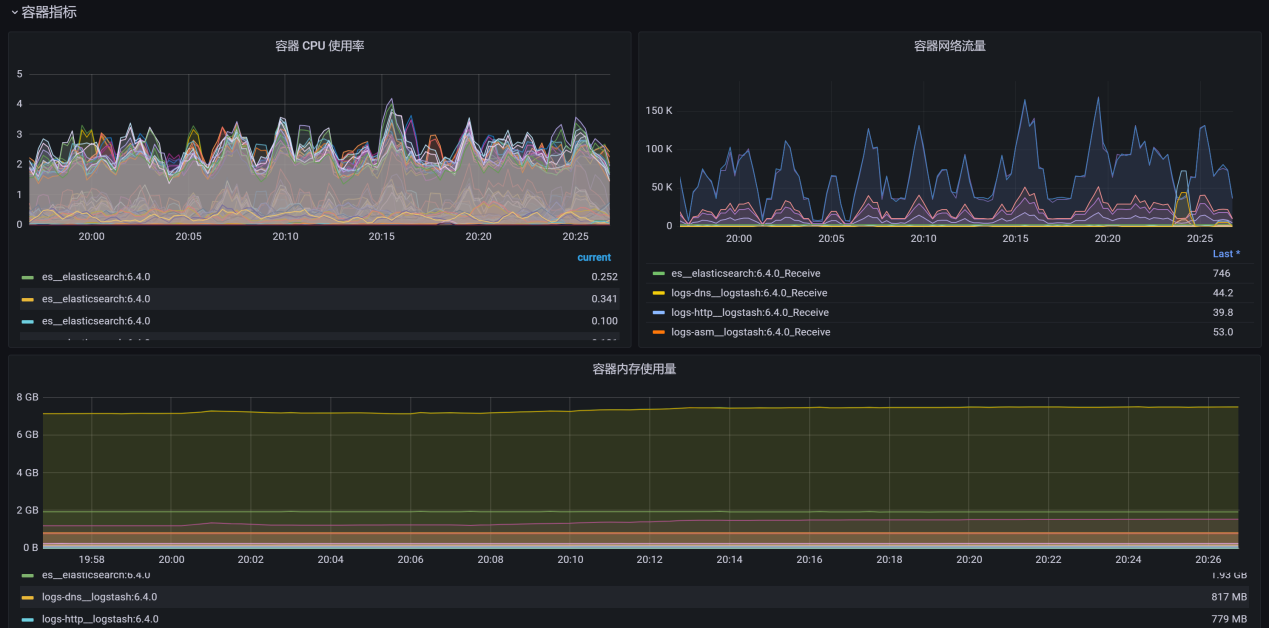
该模板共设置了主机指标、容器指标、容器详细信息三类的监控，并设置了变量，可以自主选择想要查看的宿主机或者容器。



主机指标监控内容如下：



容器指标监控内容如下：



容器详细信息监控内容如下：



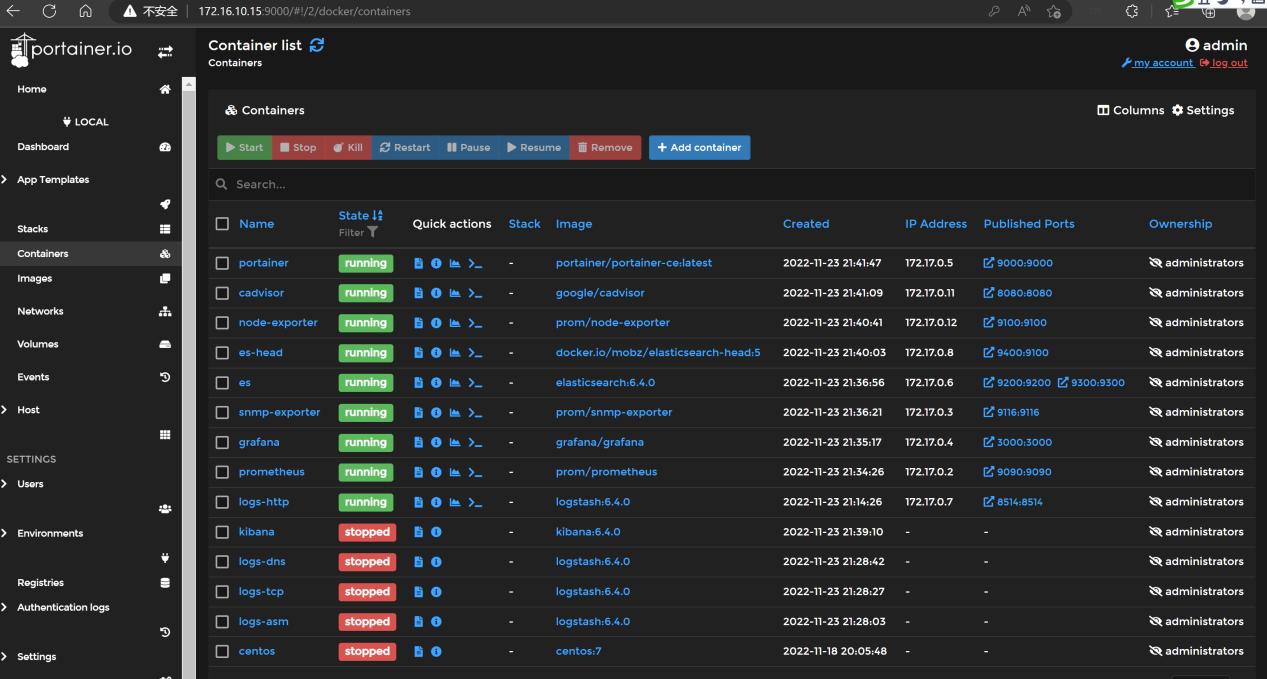
# 其它部署容器使用

## Portainer

Portainer是一个可视化的容器镜像的图形管理工具，可以通过web界面的方式管理容器。

使用方式如下：

登陆portainer容器的映射端口，即可对现有的容器进行管理操作。



# 附件：Grafana Dashboard模板文件











