hce-os下docker安装redis+nginx+oa系统+Prometheus+Grafana

HCE-OS 容器化部署OA系统

一，实验目的

该实验旨在帮助学员熟悉华为云欧拉操作系统（**HCE-OS**）下容器的创建和启动，通过完整的**OA**部署，让学员了解基础的**Dockerfile**编写，**Nginx**反向代理配置，部署Prometheus+Grafana**实现**Docker节点监控等内容。

二，docker安装配置

1，安装Docker

```
ip link add name docker0 type bridge
ip addr add dev docker0 10.0.0.1/8
```

2，安装Docker，预计三分钟。

dnf -y install docker

```
Installed:
  docker-engine-18.09.0.200-200.h14.17.11.hcev1.x86_64              libcgroup-0.42.2-1.h1.hcev1.x86_64

Complete!
```

3， 修改/etc/docker/daemon.json配置文件

```
vi /etc/docker/daemon.json
{
"registry-mirrors": [
"https://094f6e8c9700f3060f39c0043d3e15c0.mirror.swr.myhuaweicloud.com"
    ]
}
```

4，启动Docker。

```
systemctl restart docker
docker info
```

```
Registry Mirrors:
 https://094f6e8c9700f3060f39c0043d3e15c0.mirror.swr.myhuaweicloud.com/
Live Restore Enabled: true
```

三， 制作HCE-OS基础镜像

1，制作**Docker**镜像本地源

```
mkdir /hce-x86-server && yum -y -- installroot=/hce-x86-server/ install yum net-tools bash-completion iputils vim
```

2，拷贝系统环境变量到Docker镜像中

```
cp /etc/skel/.bash* /hce-x86-server/root && echo " " >/hce-x86-server/root/.bash_history
```

3，进入到镜像文件目录

```
cd /hce-x86-server
```

4，打包根目录

```
tar -zcvpf /root/hce-x86-server.tar --exclude=proc --exclude=sys --exclude=dev --exclude=run --exclude=boot .
```

```
./etc/crypto-policies/state/
./etc/crypto-policies/state/CURRENT.pol
./etc/crypto-policies/state/current
./etc/crypto-policies/local.d/
./etc/sysconfig/
./etc/sysconfig/run-parts
./etc/services
./etc/csh.cshrc
./etc/csh.login
./etc/bash_completion.d/
./etc/bash_completion.d/dnf/
./etc/bash_completion.d/copy_of__filedir
./lib
./srv/
```

5，导入镜像包至Docker

```
docker import /root/hce-x86-server.tar hce/hce-x86-server:202201
```

6， 查看镜像包

```
docker images|grep hce
```

```
[root@ecs-hce hce-x86-server]# docker images|grep hce
hce/hce-x86-server   202201              45a92c451488        47 seconds ago      511MB
```

7，测试镜像，启动容器

```
docker run -itd hce/hce-x86-server:202201 /bin/bash
```

```
[root@ecs-hce hce-x86-server]# docker run -itd  hce/hce-x86-server:202201 /bin/bash
398e6c78de89f7d5d1a071d28b611999220a40616410d82933261155abaad73d
```

四，部署redis服务

```
# 下载最新Redis镜像
cd ~ && docker pull redis

#  查看镜像
docker images

# 启动Redis容器
docker run -itd --name redis-sys -p 6379:6379 redis
docker ps

# 进入Redis容器
docker exec -it redis-sys /bin/bash

# 进入Redis客户端，出现以下截图后，Redis部署成功
```

```
redis-cli
```

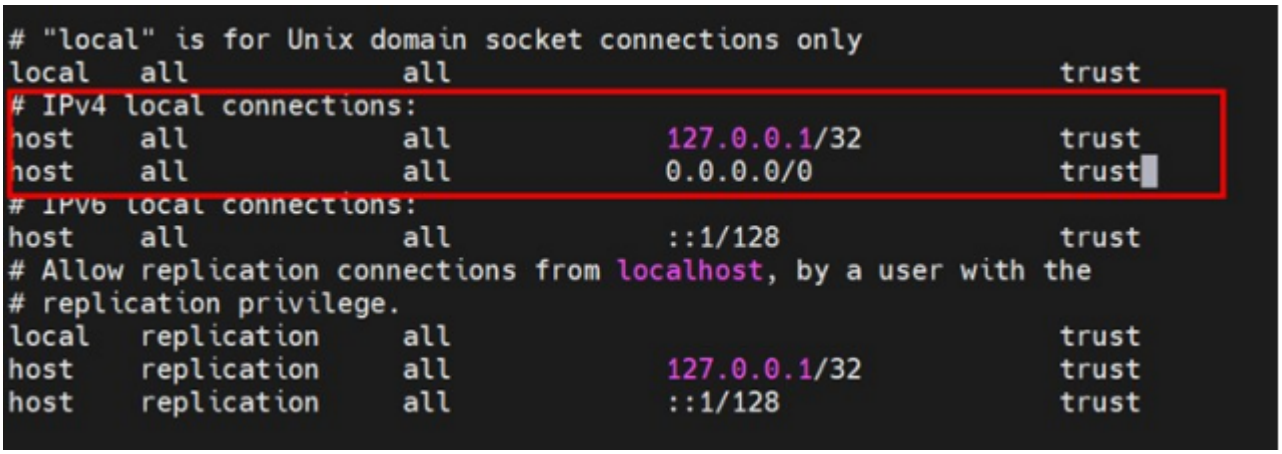
五，部署Postgresql 服务

1，pgsql容器创建

```
## 拉取Postgresql镜像
cd ~ &&docker pull postgres:13

## 步骤 2 查看镜像
docker images

## 步骤 3 创建容器
docker run -p 5432:5432 -itd --name postgres -- restart always -e POSTGRES_PASSWORD=123456 -e
ALLOW_IP_RANGE=0.0.0.0/0 -v
/home/postgres/data:/var/lib/postgresql -d postgres:13

##查看容器
docker ps

## 步骤 5 进入数据库容器
docker exec -it postgres bash

## 步骤 6 切换用户
su postgres

## 步骤 7 登录数据库，密码为123456
psql -U postgres -W

## 步骤 8 创建数据库
create database oasys;

## 步骤 9 退出数据库和容器
\q
exit
exit
```

2，修改pgsql数据库配置文件

```
## 步骤 10 拷贝pg_hba.conf文件至本机
sudo docker cp postgres:/var/lib/postgresql/data/pg_hba.conf /home

## 步骤 11 修改pg_hba.conf文件，在# IPv4 local connections:后添加以下内容，并保存
vi /home/pg_hba.conf
## #增加以下内容
host    all all 0.0.0.0/0  trust

## 步骤 12 将pg_hba.conf文件拷贝回容器
sudo docker cp /home/pg_hba.conf postgres:/var/lib/postgresql/data

## 步骤 13 进入容器
docker exec -it postgres bash

## 步骤 14 重启postgresql并使配置生效
su postgres
./usr/lib/postgresql/13/bin/pg_ctl restart
```



3，导入pgsql数据备份文件。

```
## 步骤 15 安装git
yum install -y git

## 步骤 16 创建code目录并进入
mkdir /home/code
cd /home/code

## 步骤 17 拉取代码及数据文件
git clone https://codehub.devcloud.cn-north-4.huaweicloud.com/oasys00001/oasys.git

## 步骤 18 拷贝数据文件至容器
sudo docker cp ./oasys/oasys-pgsql-data.sql postgres:/var/lib/postgresql/data
sudo docker cp ./oasys/oasys-pgsql-table.sql postgres:/var/lib/postgresql/data

## 步骤 19 进入容器
docker exec -it postgres bash

## 步骤 20 导入数据
psql -U postgres -d oasys -a -f /var/lib/postgresql/data/oasys-pgsql-table.sql
psql -U postgres -d oasys -a -f /var/lib/postgresql/data/oasys-pgsql-data.sql
```

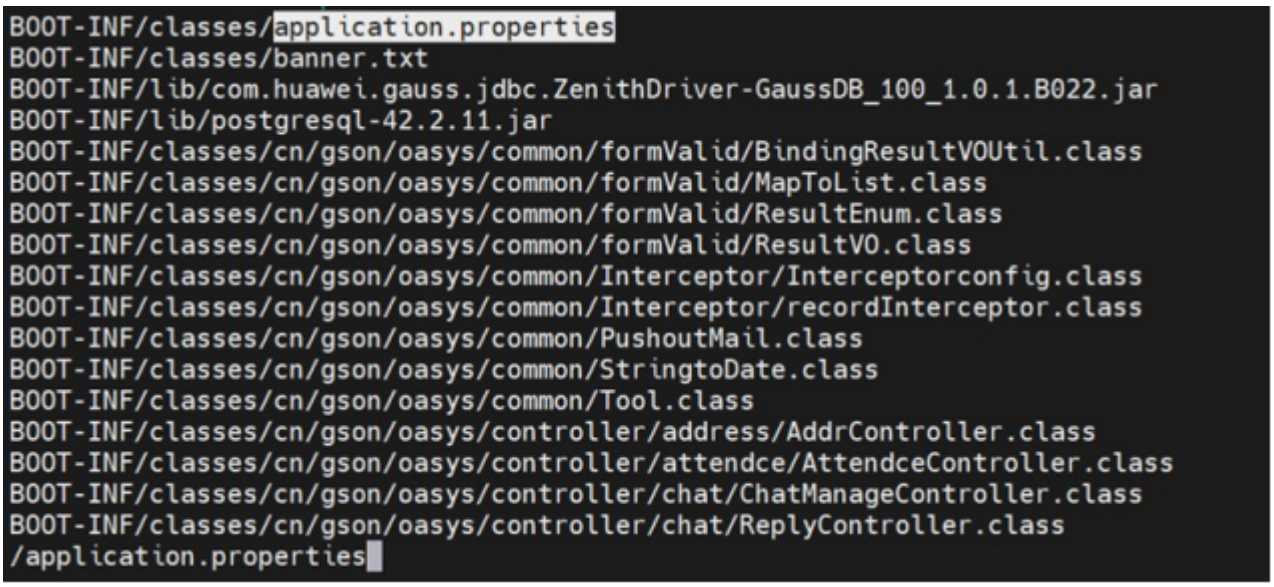
六，部署JAVA项目。

```


## 新建目录并进入
 mkdir /usr/java && cd /usr/java

## 下载JDK和HCE镜像包
wget https://sandbox-expriment-files.obs.cn-north- 1.myhuaweicloud.com/20220411/jdk-8u321-linux-
x64.tar.gz

## 步骤 3 解压JDK
tar -zxvf jdk-8u321-linux-x64.tar.gz

## 步骤 4 修改目录名称
mv jdk1.8.0_321 jdk1.8

## 步骤 5 获取Java程序包
wget https://sandbox-expriment-files.obs.cn- north-1.myhuaweicloud.com/20220412/oasys-0.0.1-
SNAPSHOT.jar
```

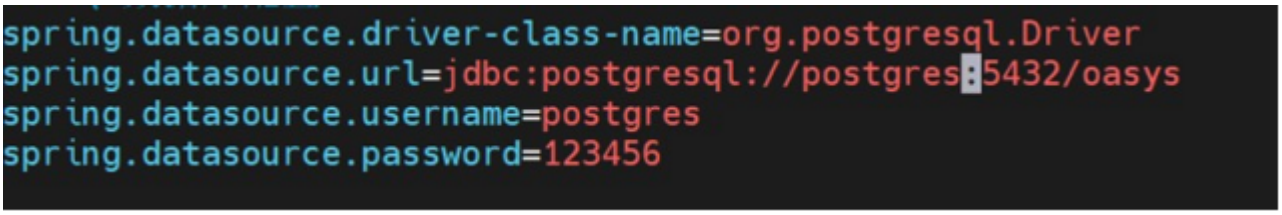
七，docker网络与镜像打包

```
## 步骤 6 创建Docker网络
docker network create oa-net

## 步骤 7 将Postgres容器加入oa-net
docker network connect oa-net postgres

## 步骤 8 将Redis容器加入oa-net
docker network connect oa-net redis-sys

## 步骤 9 修改jar包配置文件，使用vim打开jar包
vim oasys-0.0.1-SNAPSHOT.jar
```

步骤 10 输入 /application.properties 搜索该文件，并敲回车键2次进入该文件

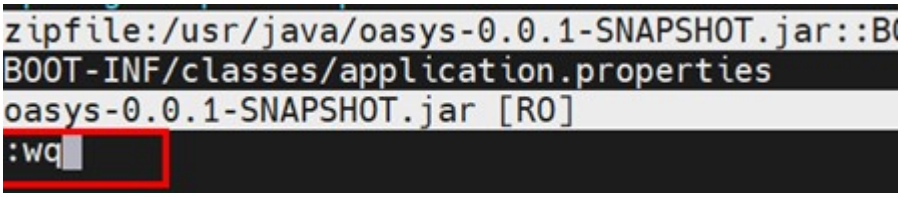


步骤 11 修改spring.datasource.url地址为jdbc:postgresql://postgres:5432/oasys



步骤 12 修改spring.redis.host 地址为redis-sys

步骤 13 按“esc”键，并输入“:wq”保存



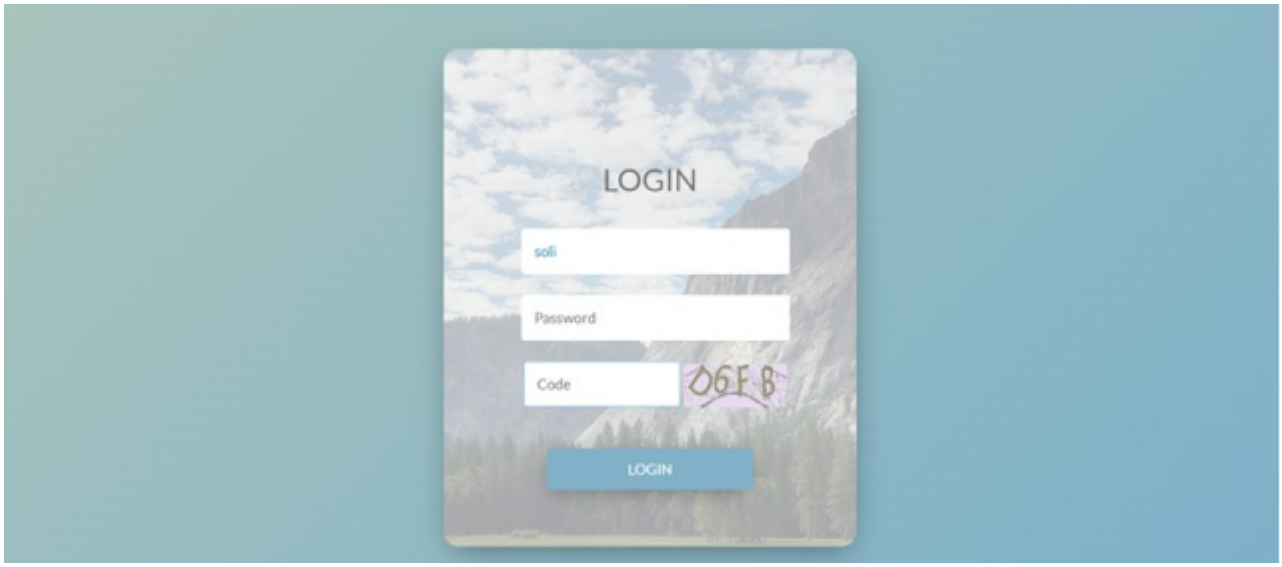
```
## 步骤 14 按“esc”键，并输入“:q”退出
## 步骤 15 创建Dockerfile
vim Dockerfile

## 步骤 16 输入“i”,并增加以下内容，之后按“esc”键，并输入“:wq”保存退出

FROM hce/hce-x86-server:202201
WORKDIR /home
COPY jdk1.8 /home/java
COPY oasys-0.0.1-SNAPSHOT.jar /home ENV JAVA_HOME=/home/java
ENV PATH=$JAVA_HOME/bin:$PATH
ENV CLASSPATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tool.jar
EXPOSE 8088
CMD [ "java","-jar","oasys-0.0.1-SNAPSHOT.jar"]

## 步骤 17 输入以下命令，创建docker镜像
docker build -t hce/hce_java_oa:202201 .
```

```
Step 8/9 : EXPOSE 8088
 ---> Running in d33b7f4f8390
Removing intermediate container d33b7f4f8390
 ---> 12aac32980c3
Step 9/9 : CMD [ "java","-jar","oasys-0.0.1-SNAPSHOT.jar"]
 ---> Running in c5de434522a2
Removing intermediate container c5de434522a2
 ---> 8754edbaa8c9
Successfully built 8754edbaa8c9
Successfully tagged hce/hce_java_oa:202201
```

```
## 步骤 18 查看镜像
docker images

## 步骤 19 启动镜像
docker run -itd --name=oa-java --network=oa-net -p 8088:8088 hce/hce_java_oa:202201
```

```
## 步骤 21 进入镜像
docker exec -it oa-java /bin/bash

## 步骤 21 查看java进程，并退出容器
jps
exit
```

步骤 22 浏览器访问oa，输入eip:8088访问，账号soli，密码123456（eip可在华为云控制台获取）



八， 部署Nginx并配置反向代理

```
## 步骤 1 下载**nginx**镜像
docker pull nginx

## 步骤 2 查看nginx镜像
docker images|grep nginx

## 步骤 3 启动nginx镜像
docker run -d --network=oa-net --name=nginx -p 80:80 nginx

## 步骤 4 查看启动的nginx镜像
docker ps

## 步骤 5 在浏览器，输入eip:80，访问nginx
```

```
步骤 6 进入nginx容器
docker exec -it nginx /bin/bash

## 步骤 7 进入nginx的conf.d目录
cd /etc/nginx/conf.d/

## 步骤 8 更新apt资源列表，等待更新完成
apt-get update

## 步骤 9 下载vim工具
apt-get install -y vim

## 步骤 10 打开defalut.conf文件
vim default.conf

## 步骤 11 输入i进入编辑模式，在location节点下进行编辑后，保存退出
```

```
#增加proxy-pass配置proxy_pass http://eip:8088;
#将root ,index节点注释,后添加以下内容。

proxy_pass http://eip:8088;
```

```
server {
    listen       80;
    listen  [::]:80;
    server_name  localhost;

    #charset koi8-r;
    #access_log  /var/log/nginx/host.access.log  main;

    location / {
       # root   /usr/share/nginx/html;
       # index  index.html index.htm;
     proxy_pass http://121.[redacted].73:8088;
    }
```

步骤 12 验证文件是否配置正常

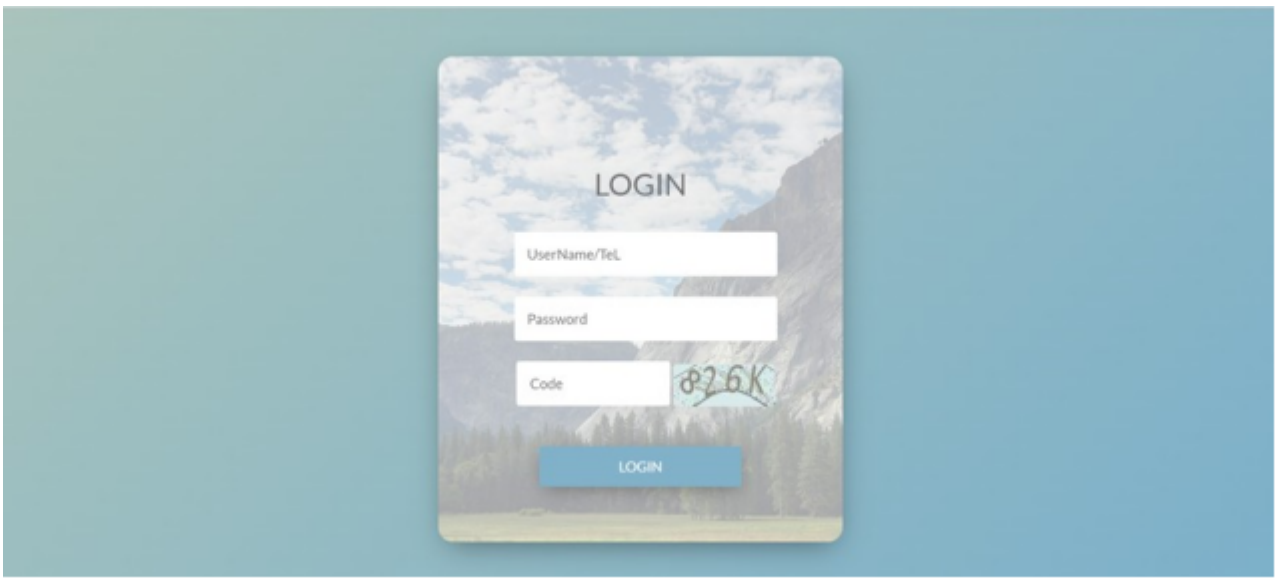
nginx -t

```
root@13ccfd09e06c:/etc/nginx/conf.d# nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@13ccfd09e06c:/etc/nginx/conf.d#
```

步骤 14 重启Nginx容器

docker restart nginx

步骤 15 输入**eip:80/logins**，访问**OA**系统



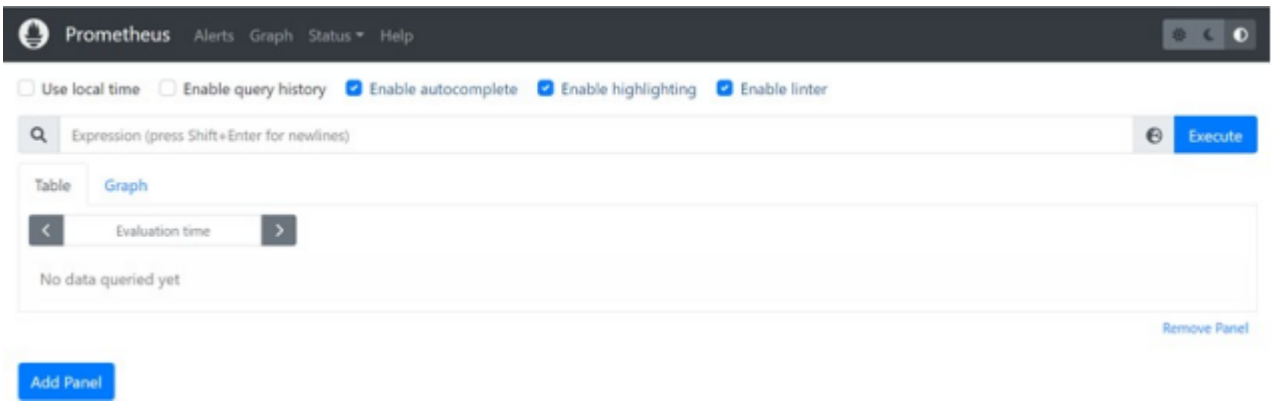
步骤 16 输入账号soli，密码123456 登录OA

九，部署监控平台

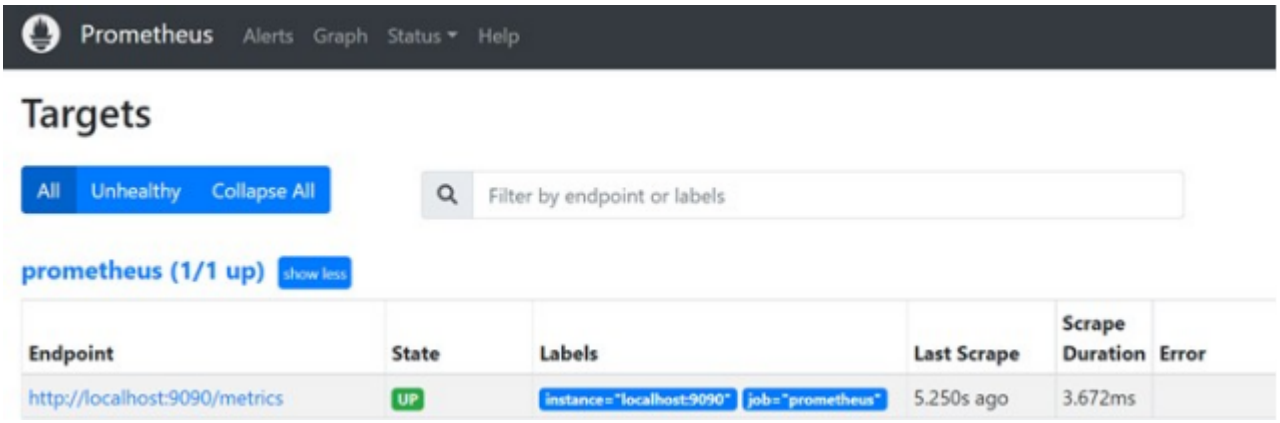
1，部署Prometheus

```
## 步骤 1 下载**Prometheus**镜像
cd ~ && docker pull prom/prometheus

## 步骤 2 查看Prometheus镜像
docker images|grep prometheus

## 步骤 3 启动Prometheus容器
docker run -id --name=prom --network=oa-net -p 9090:9090 prom/prometheus
```

步骤 4 验证**Prometheus**，在浏览器输入**EIP:9090**访问Prometheus



步骤 5 在Status选项下，点击Targets标签，进入数据页面。



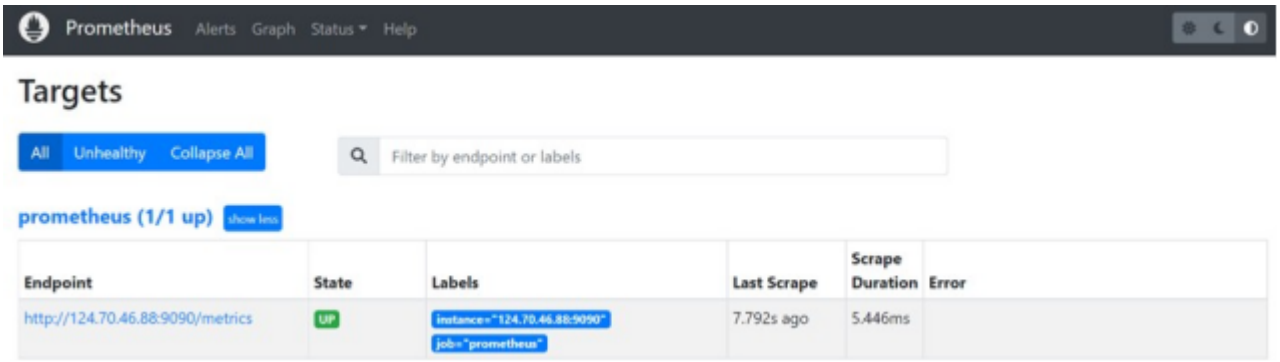
步骤 6 点击Endpoint下的链接，此时会发现页面无法访问


```
## 步骤 7 回到命令行，创建prometheus文件夹
mkdir prometheus

## 步骤 8 将Prometheus配置文件拷贝至宿主机
docker cp prom:/etc/prometheus/prometheus.yml ./prometheus

## 步骤 9 使用vim编写配置文件
vim prometheus/prometheus.yml

## 步骤 10 修改targets对应的值为eip:9090，并且保存退出
```

```
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: ["124.[redacted]88:9090"]
```

```
步骤 11 停止prom容器
docker stop prom

## 步骤 12 删除prom容器
docker rm prom

## 步骤 13 重新启动Prometheus容器
docker run -idt --name=prom --network=oa-net -p 9090:9090 -v
/root/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml prom/prometheus
```

步骤 14 通过EIP:9090访问Prometheus，并进入Status选项下Targets页面

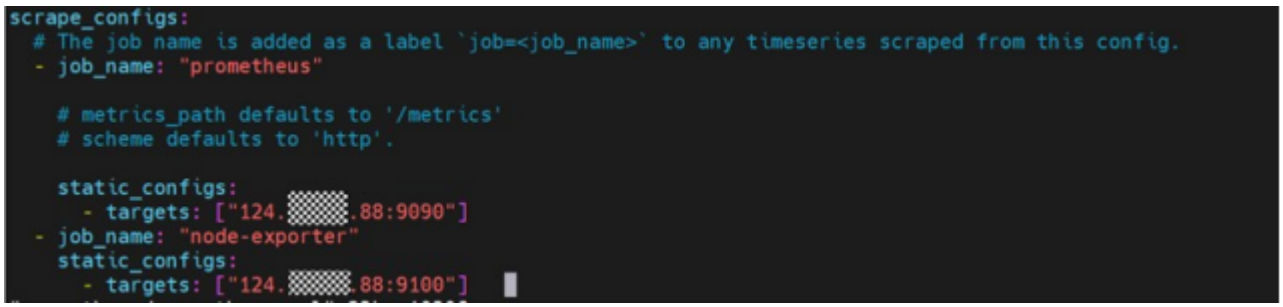


步骤 15 点击Endpoint标签下的链接，可以查看到Prometheus收集到的数据

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 2.8756e-05
go_gc_duration_seconds{quantile="0.25"} 4.4435e-05
go_gc_duration_seconds{quantile="0.5"} 5.3737e-05
go_gc_duration_seconds{quantile="0.75"} 7.355e-05
go_gc_duration_seconds{quantile="1"} 0.00014393
go_gc_duration_seconds_sum 0.001895696
go_gc_duration_seconds_count 31
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 8
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.637088e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 6.7739576e+07
```

2, 部署node-exporter

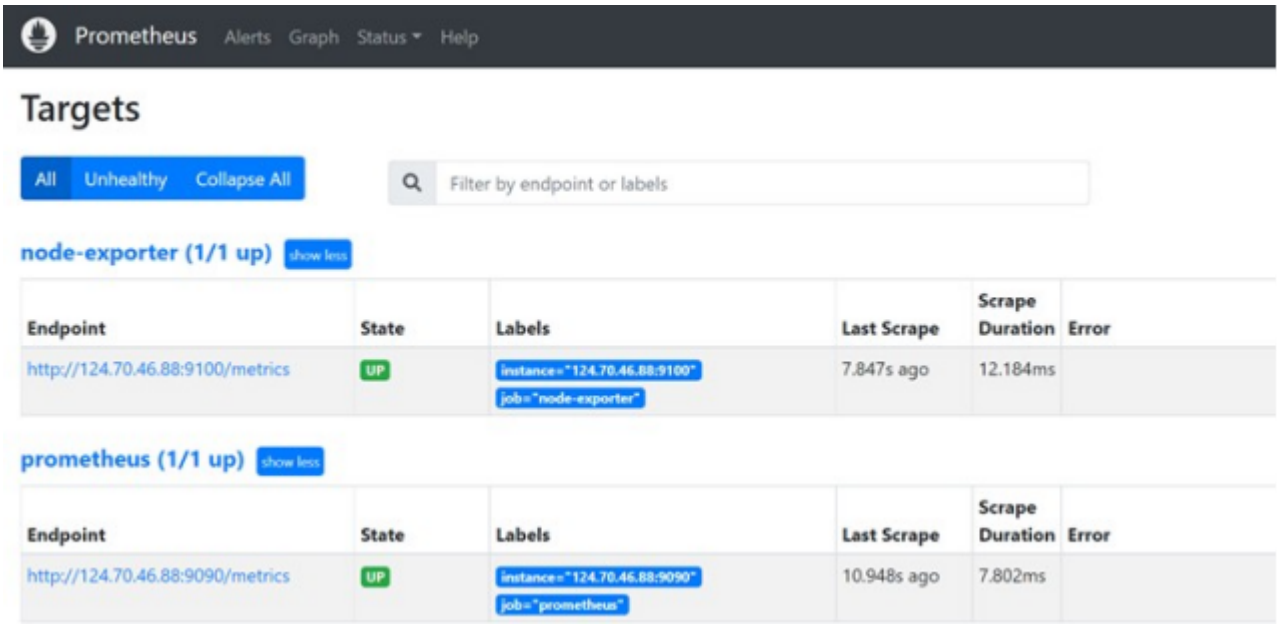
```
## 步骤 1 下载node-exporter镜像
docker pull prom/node-exporter

## 步骤 2 启动node-exporter
docker run -id --name=node-exporter --network=oa- net -p 9100:9100 prom/node-exporter

## 步骤 3 修改Prometheus配置文件
vim prometheus/prometheus.yml

## 步骤 4 增加以下内容，并且保存退出
- job_name: "node-exporter"
  static_configs:
    - targets: ["EIP:9100"]
```



步骤 5 重新启动prometheus

docker restart prom

步骤 6 通过EIP:9090访问Prometheus，在Status—Targets页面下，可以看到新配置的node-exporter数据源。



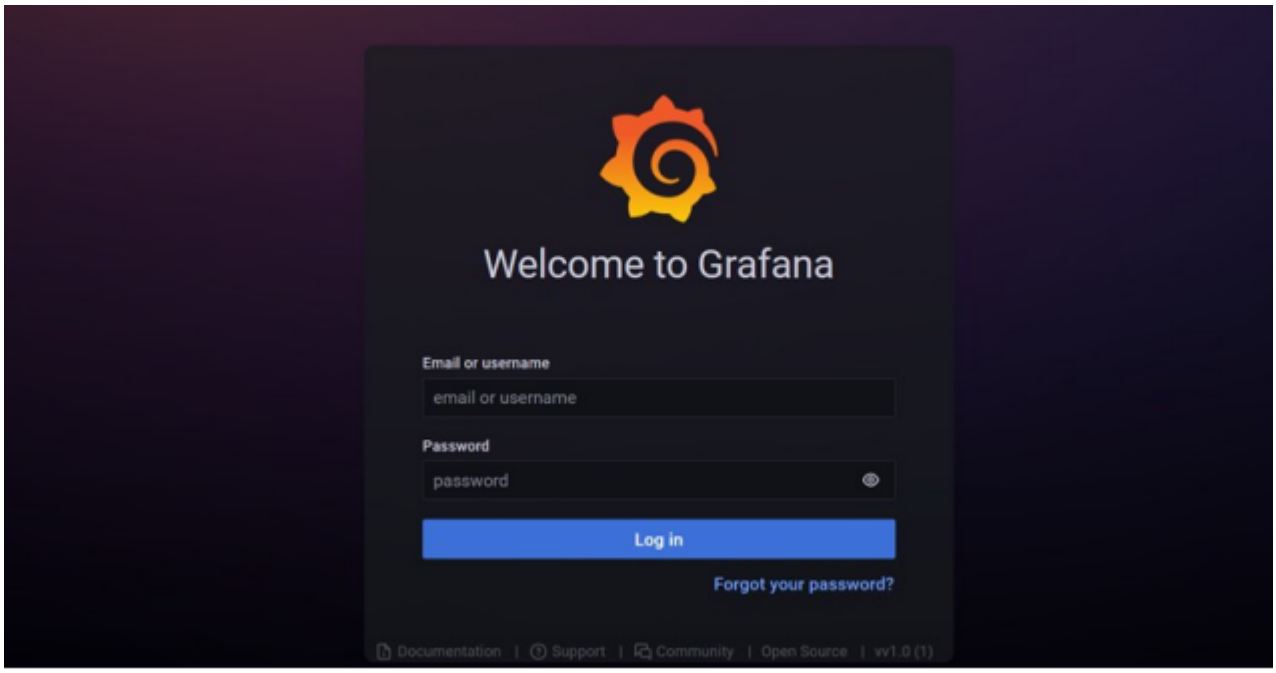
步骤 7 点击node-exporter的链接，查看当前节点的监控数据

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 2.6824e-05
go_gc_duration_seconds{quantile="0.25"} 4.3215e-05
go_gc_duration_seconds{quantile="0.5"} 5.7054e-05
go_gc_duration_seconds{quantile="0.75"} 8.178e-05
go_gc_duration_seconds{quantile="1"} 0.000207997
go_gc_duration_seconds_sum 0.024617291
go_gc_duration_seconds_count 330
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.640384e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 7.40425728e+08
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.541303e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 7.911797e+06
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 1.4744789879386341e-05
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 5.281328e+06
```

3，部署Grafana并创建监控仪表盘

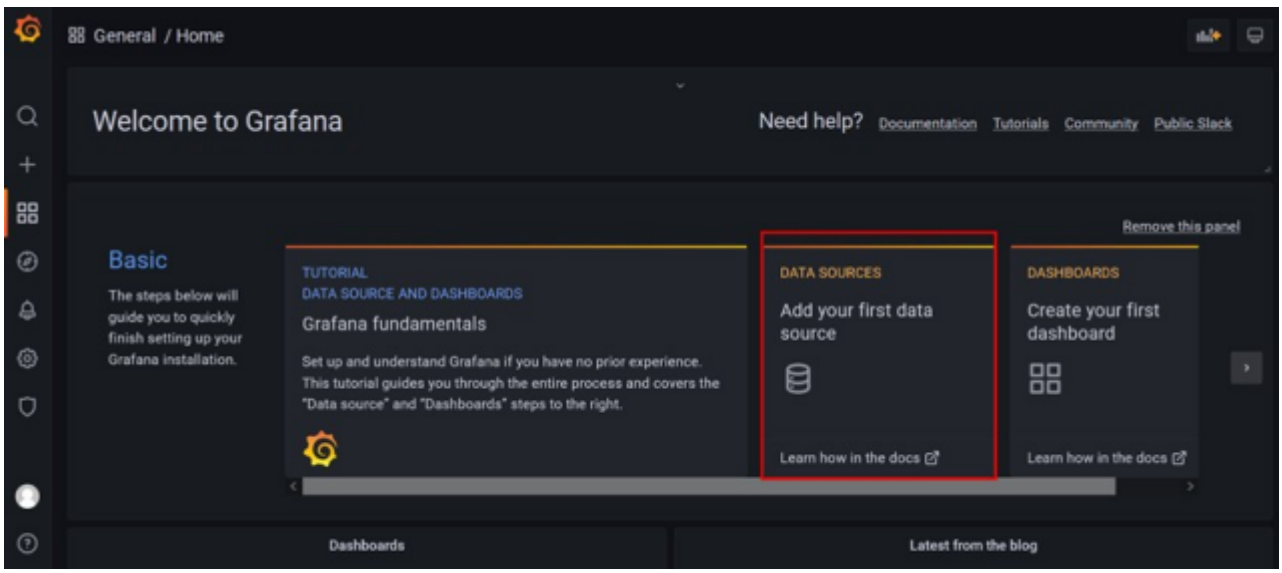
```
## 下载Grafana镜像
docker pull grafana/grafana

## 步骤 2 查看Grafana镜像
docker images grafana/grafana

## 步骤 3 启动Grafana容器
docker run -d --name grafana -p 3000:3000 grafana/grafana
```

步骤 4 通过EIP:3000访问Grafana，初始账号为admin，密码为 admin

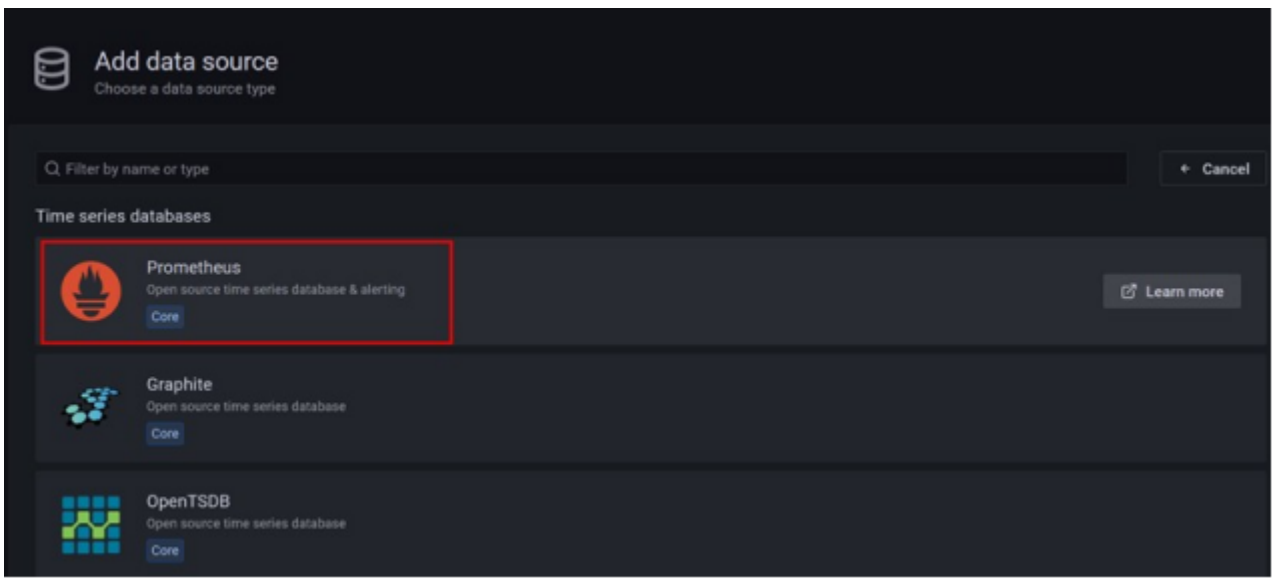


步骤 5 登录后可自行重置初始密码

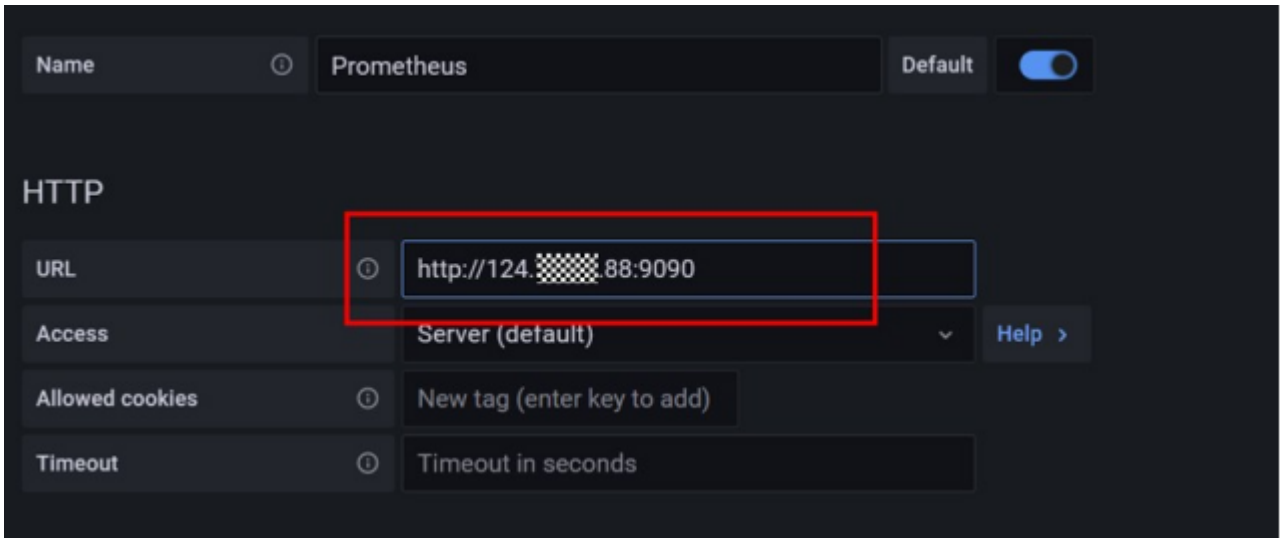
步骤 6 登录后，DATA SOURCE，增加数据源



步骤 7 数据源类型选择Prometheus

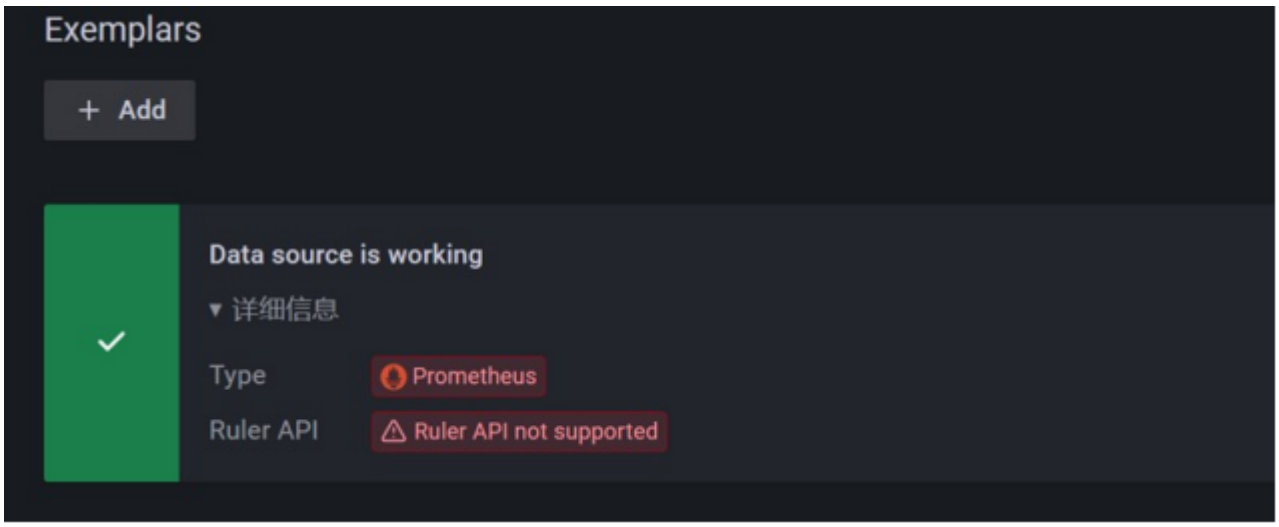


步骤 8 配置URL为http://eip:9090

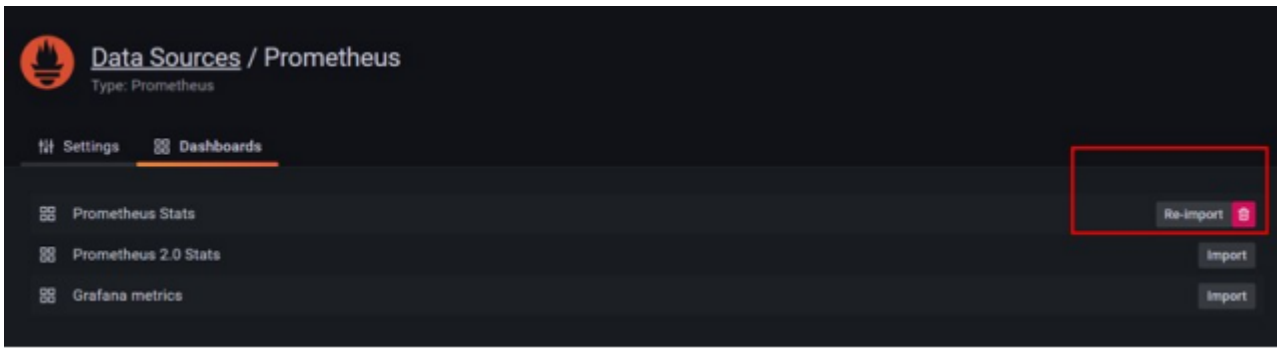


步骤 9 点击Save&test按钮，保存配置

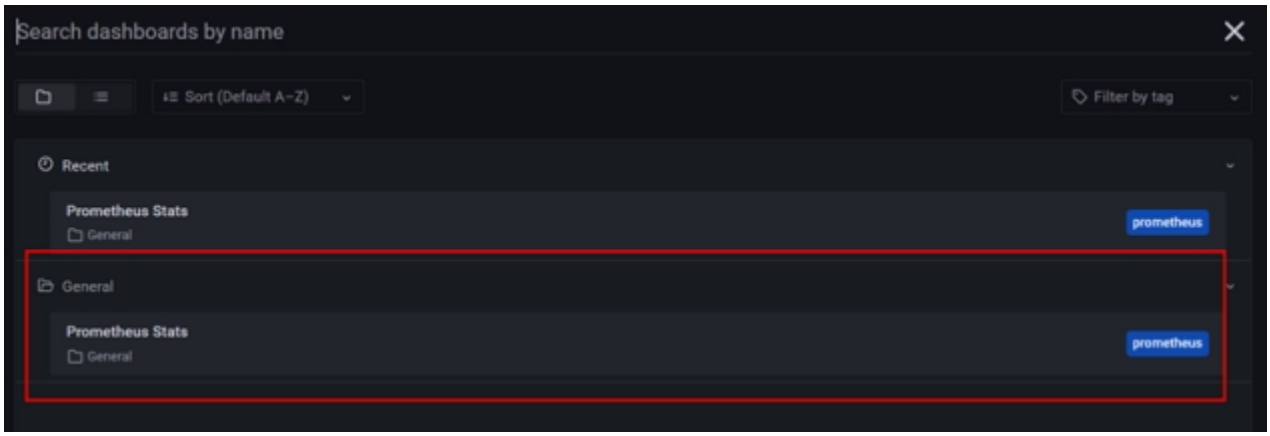
步骤 10 出现Data source is working 标签后，表示配置成功



步骤 11 回到页面最上方，点击Dashboards，选择Prometheus Stats，点击import

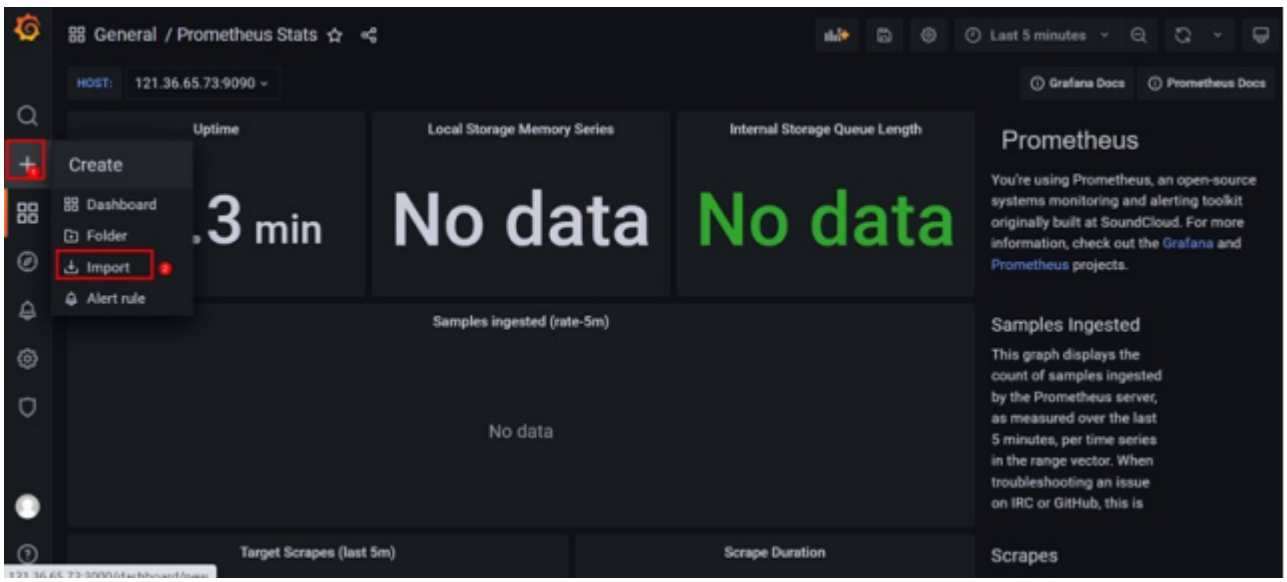


步骤 12 点击图标，进入Home页面，点击上方Home标签，进入Home页面，点击Prometheus Stats，进入仪表盘

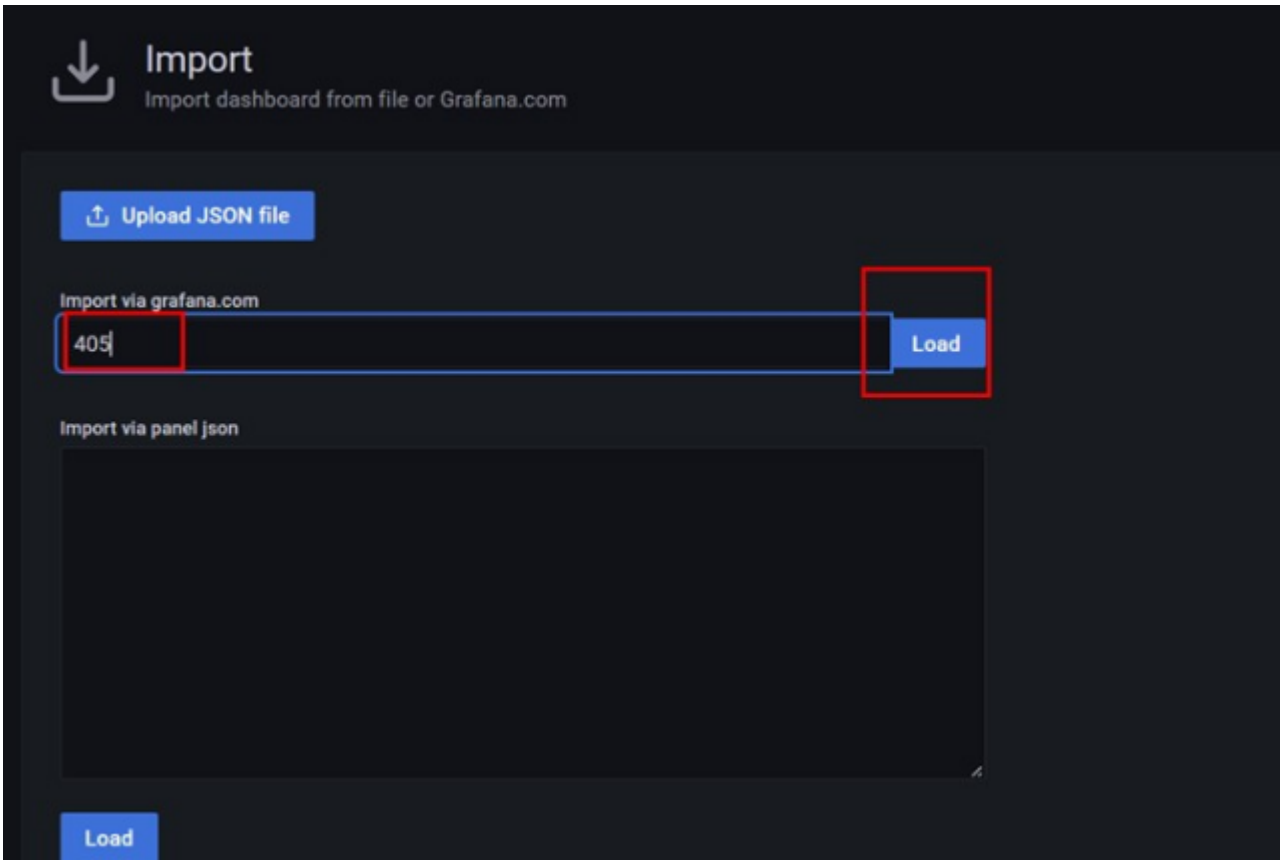


步骤 13 查看仪表盘

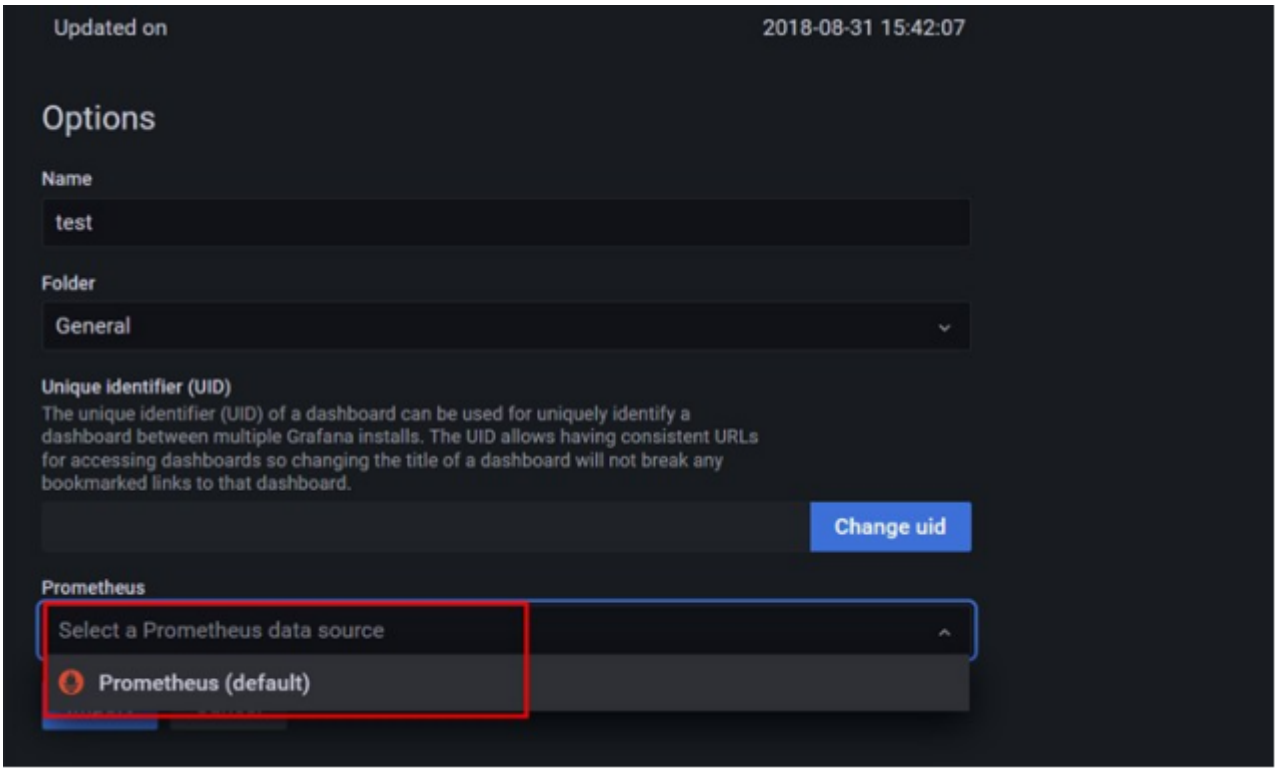
步骤 14 点击左侧加号图标，点击import选项，导入仪表盘



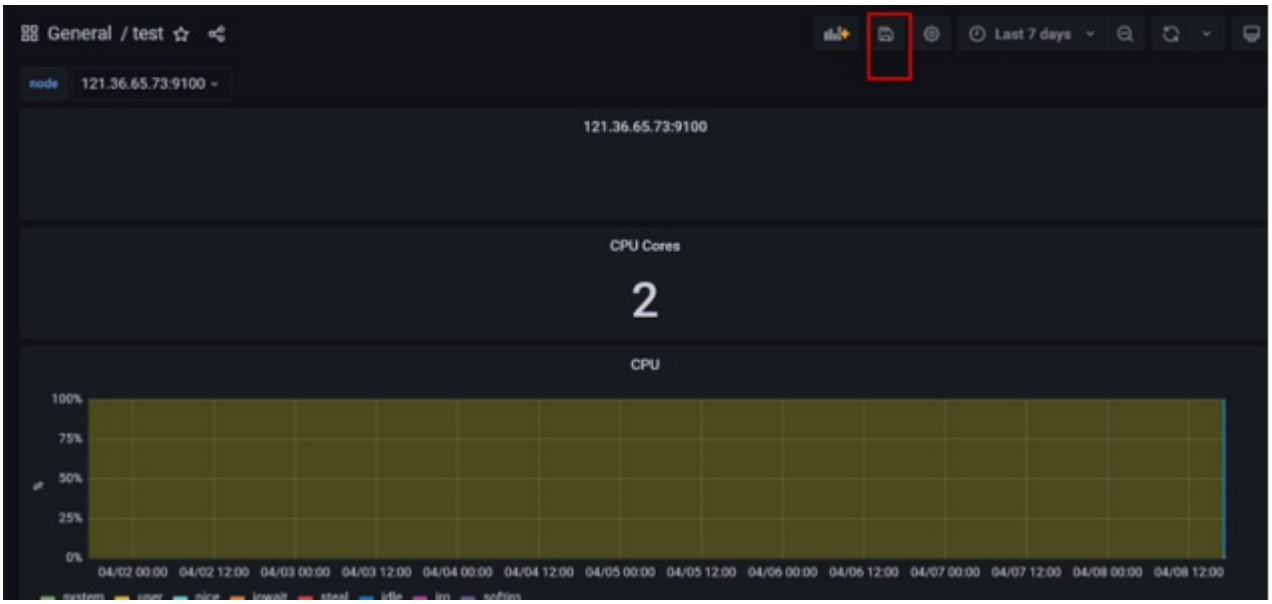
步骤 15 输入ID为405点击右侧load按钮



步骤 16 选择数据源为Prometheus，并点击import按钮



步骤 17 查看仪表盘，点击右上方保存按钮保存


General / test
Last 7 days
node
121.36.65.73:9100
121.36.65.73:9100
CPU Cores
2
CPU
100%
75%
50%
25%
0%
04/02 00:00
04/02 12:00
04/03 00:00
04/03 12:00
04/04 00:00
04/04 12:00
04/05 00:00
04/05 12:00
04/06 00:00
04/06 12:00
04/07 00:00
04/07 12:00
04/08 00:00
04/08 12:00