2021.3.8 构建 docker 容器监控系统

案例概述

Docker 作为目前十分出色的容器管理技术，得到大量企业的青睐，在生产环境中使用 Docker 容器部署服务及应用的场景越来越多。所以面对日益庞大的 docker 服务群应用，如何具有针对性的，有效的监控也变成了企业运维人员工作需求。

容器信息采集及监控的方案有很多，有 docker 自身的 docker stats 命令.Scout、Data Dog、Prometheus 等，本次为大家分享两款比较经典的容器开源监控组合方案 Cadvisor.+InfluxDBtGrafana 和 Cadxisor.+BrometheussGrafana。

一．Cadvisor+InfluxDB+Grafana

Cadvisor

Cadxisar.是 Google 用来监测单节点资源信息的监控工具。Cadvisor.提供了基础查询界面和 http 接口，方便其他组件如 Grafana 、Prometheus 等进行数据抓取。Cadvisor.可以对 Docker 主机上的资源及容器进行实时监控和性能数据采集，包括 CPU 使用情况、内存使用情况、网络吞吐量及文件系统使用情况等。Cadvisor.使用 Go 语言开发,利用 Linux 的 Cgroups 获取容器的资源使用信息。

Google 的 Kubernetes 中也默认地将其作为单节点的资源监控工具，各个节点默认会安

Gadvisor 产品特点

·可以展示主机个容器两个层次的监控数据

·可以展示历史变化数据

·监控指标齐全

·方便部署，有官方的 docker 镜像

·缺点：默认只在本地保存 1 分钟的数据，可以集成 influxDB 等第三方存储使用

由于 Cadvisor 提供的操作界面略显简陋，而且需要在不同页面之间跳转，并且只能监控一个 Host，这不免会让人质疑它的实用性。但 Cadvisor.的一个亮点是它可以将监控到的数据导出给第三方工具，由这些工具进一步加工处理。

我们可以把 Cadvisor.定位为一个监控数据收集器，收集和导出数据是它的强项，而非展示数据。

二．influxDB

InfluxDB.是一个由 InfluxData.开发的开源非关系型时序型数据库。它由 Go 写成，着力于高性能地查询与存储时序型数据。InfluxDB..被广泛应用于存储系统的监控数据，IoT 行业的实时数据等场景。同类型的数据库产品还有 Elasticsearch、Graphite 等。

InfluxDB.应用场景:性能监控，应用程序指标，物联网传感器数据和实时分析等的后端存储。

indluxDB 的主要功能

·基于时间序列，支持与时间有关的相关函数（如最大，最小。求和等）；
·可度量性：可以实时对大量数据进行计算
·基于事件：它支持任意的事件数据

influxDB 主要特点
·无结构（无模式）：可以是任意系列：
·支持扩展
·支持 min，max，sum，count，mean，median 等一系列函数，方便统计：
·原生的 HTTP 支持，内置 HTTP，API
·强大的 SQL 语法：
·自带管理界面，方便使用：

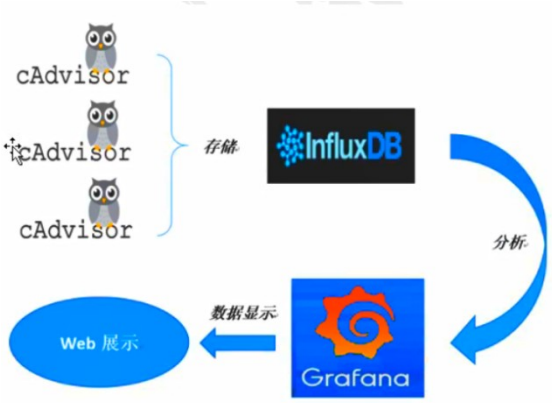
三．Grafana

Grafana 是一个可视化面被(Dashboard)工具,有着非常漂亮的图表和布局等展示功能，功能齐全的度量仪表盘和图形编辑器，支持 Graphite、zabbix、InfluxDB、Prometheus 和 QpenTS.DB 等组件作为数据源。

1.Grafana 主要特性
·灵活丰富的图形化选项
·可以混合多种风格
·支持白天和夜间模式
·支持多个数据源

四．监控组件架构

监控组件架构图：



提示：cadvisor 用户采集数据，influDB 用于数据库存储，Grafana 用于数据展示
监控组件架构部署：
1 创建自定义网络 monitor（自定义网络名称），用于后期容器加入此网络中；
2 创建 influxDB 容器，创建数据用户，数据库
3 创建 influxDB 容器；
4 创建 Grafana 容器，配置 grafana；

五．开始部署

安装部署 docker-ce（不过多演示）

1.导入镜像

```
[root@server03 ~]# docker load < influxdb.tar   #导入 infulxDB 镜像
[root@server03 ~]# docker load < grafana.tar   #导入 grafana 镜像
[root@server03 ~]# docker load < cadvisor.tar   #导入 cadvisor 镜像
[root@server04 ~]# docker images
REPOSITORY          TAG       IMAGE ID       CREATED         SIZE
grafana/grafana     latest    651ff2dc930f   3 months ago    187MB
google/cadvisor     latest    eb1210707573   2 years ago     69.6MB
tutum/influxdb      latest    c061e5808198   4 years ago     290MB
```

2. 创建自定义网络

```
[root@server04 ~]# docker network  create monitor
bf3d1c84ffe726b807ee61b99e23f8f95aa40e0d43ec95c29915bf7aaac9f5ed
[root@server04 ~]# docker network  ls
NETWORK ID     NAME      DRIVER    SCOPE
9c7634b1127c   bridge    bridge    local
6bdce343ebb5   host      host      local
bf3d1c84ffe7   monitor   bridge    local
b9fb94199bb7   none      null      local
```

3. 定义并开启一个容器

```
[root@server04 ~]# docker run -d --name influxdb --net monitor -p 8083:8083 -p 8086:8086 tutum/influxdb
5845a093b3d98b2cdd2514d3da7d5bd5e9954a1410bea1a5b89e62d9ef9e73d1
```

选项解释：--net monitor #加入到 monitor 网段中

8086 端口 #是 infulxDB 的数据端口

8083 端口 #是 infulxDB 的后台控制端口

--name influxdb #表示容器的新名字

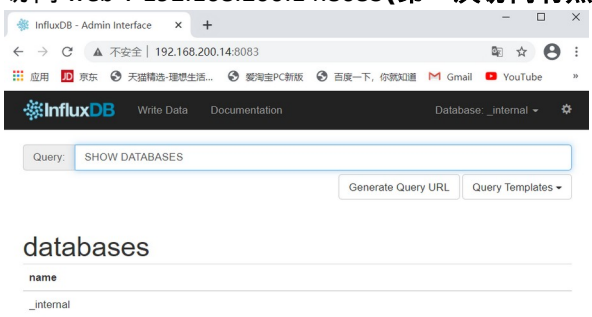
4. 查看容器的状态

5. [root@server04 ~]# docker ps -a #查看容器的状态

```
CONTAINER ID   IMAGE             COMMAND     CREATED          STATUS          PORTS
                                 NAMES
5845a093b3d9   tutum/influxdb    "/run.sh"   4 minutes ago    Up 4 minutes    0.0.0.0:8083->8083/tcp, 0.0
.0.0:8086->8086/tcp   influxdb
```

注意：up 状态

访问 web：192.168.200.14:8083**(第一次访问有点慢)**

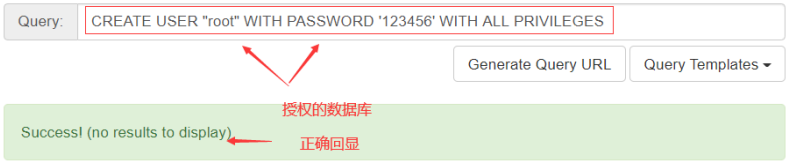


在界面授权一个数据库及创建数据表，后面会用到

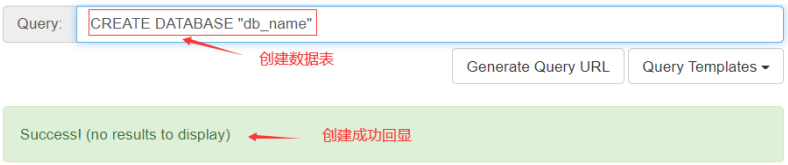
CREATE USER "root" WITH PASSWORD '123456' WITH ALL PRIVILEGES

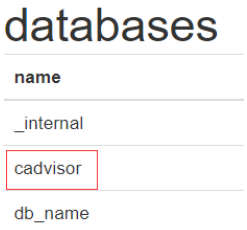
注释：用户：root

密码：123456



创建一个数据表： CREATE DATABASE "cadvisor"





5.创建 **Gadvisor**

```
docker run -d --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --volume=/var/lib/docker/:/var/lib/docker:ro --net monitor --publish=8080:8080  --name=cadvisor google/cadvisor -storage_driver=influxdb -storage_driver_db=cadvisor -storage_driver_host=influxdb:8086
```

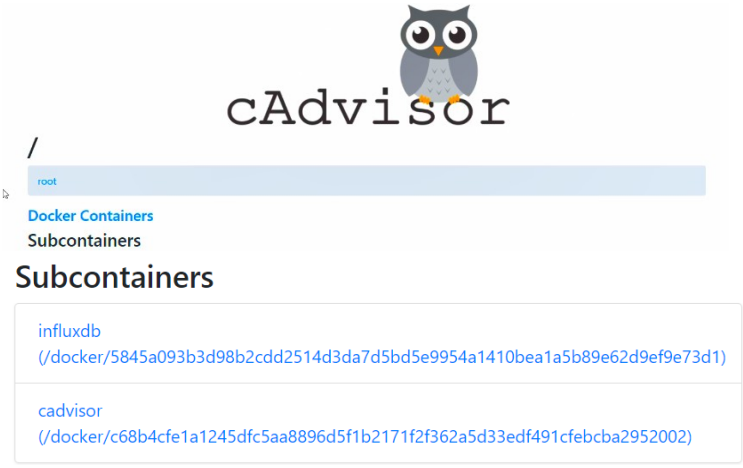
参数说明：

- -d：后台运行此容器；
- --name：启运容器分配名字 Cadvisor ；
- --net：把容器加入到新的网络 monitor；
- -p：映射端口 8080；
- --mout：把宿主机的相文目录绑定到容器中，这些目录都是 Cadvisor 需要采集的目录文件和监控内容；
- -storage_driver：需要指定 Cadvisor 的存储驱动、数据库主机、数据库名；
- google/Cadvisor ：通过 Cadvisor 这个镜像来运行容器，默认会在 docker 官方仓库把镜像 pull 下来；

6. [root@server04 ~]# docker ps -a #查看容器的状态

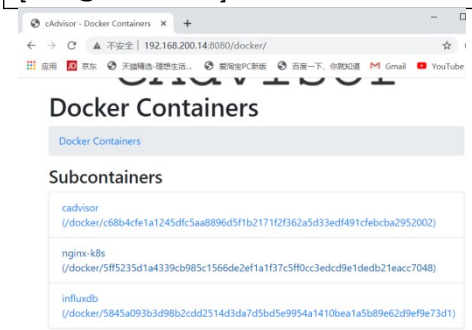
```
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS
          NAMES
c68b4cfe1a12   google/cadvisor    "/usr/bin/cadvisor -… "   7 seconds ago    Up 3 seconds
          cadvisor
5845a093b3d9   tutum/influxdb     "/run.sh"                19 minutes ago   Up 19 minutes
->8086/tcp   influxdb
```

访问 web：192.168.200.14:8080(第一次访问有点慢)



·做一个小测试：拉取一个 nginx 镜像

```
[root@server04 ~]# docker pull nginx    #拉取一个 nginx 镜像
[root@server04 ~]# docker run  -itd --name nginx-k8s  -p 8000:80 nginx  #开启容器器
```



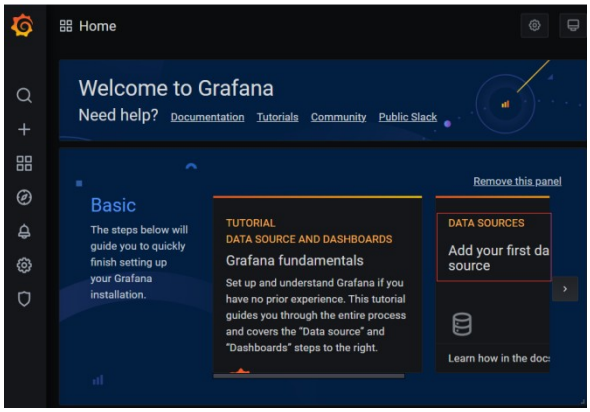
·启动 grafana 容器

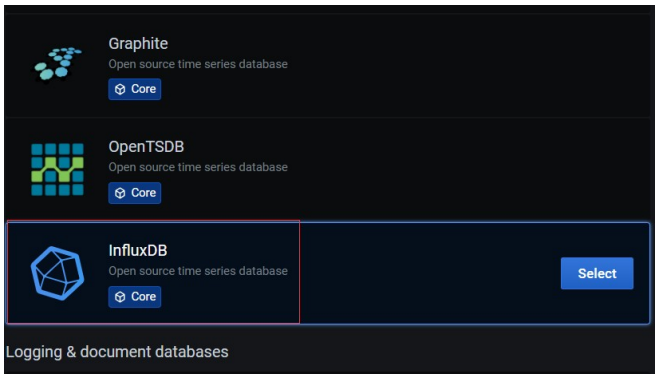
```
[root@server04 ~]# docker run -d --name grafana --net monitor -p 3000:3000 grafana/grafana
#启动 grafana 容器器
[root@server04 ~]# docker ps -a  #查看状态
```

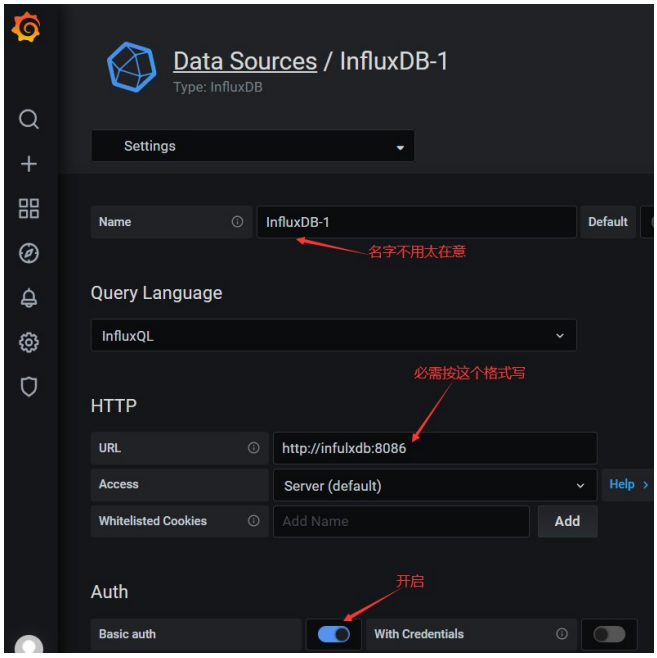
访问 web：192.168.200.14:3000

用户：admin（第一次登录要求更改密码）

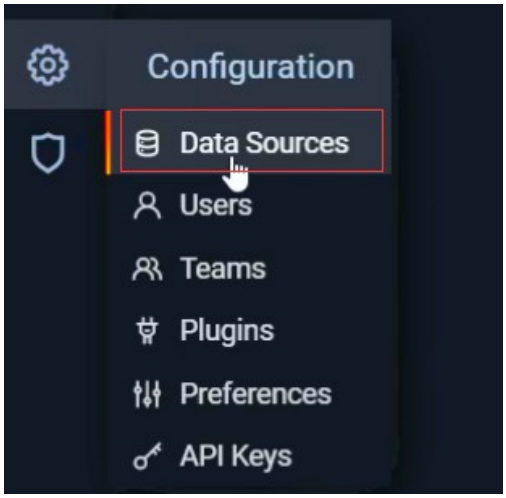
密码：admin



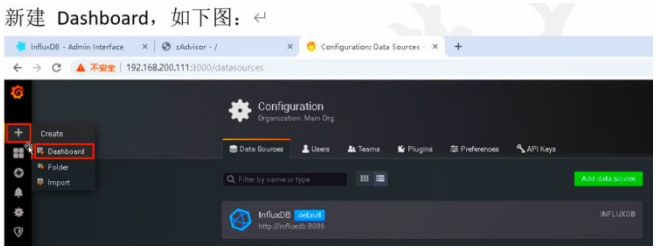

Graphite
Open source time series database
Core
OpenTSDB
Open source time series database
Core
InfluxDB
Open source time series database
Core
Select
Logging & document databases

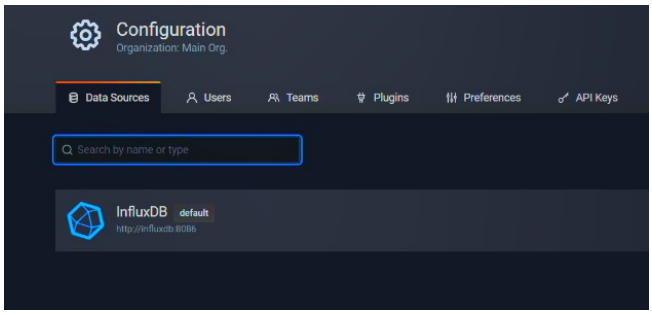

Data Sources / InfluxDB-1
Type: InfluxDB
Settings
Name
InfluxDB-1
Default
名字不用太在意
Query Language
InfluxQL
必需按这个格式写
HTTP
URL
http://infulxdb:8086
Access
Server (default)
Help
Whitelisted Cookies
Add Name
Add
Auth
开启
Basic auth
With Credentials

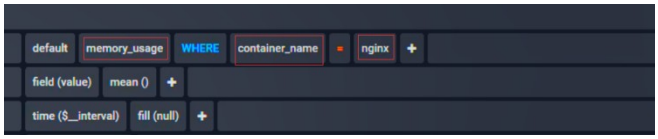

InfluxDB Details
授权的用户
Database
cadvisor
User
root
授权的用户
Password
用户密码
HTTP Method
Choose

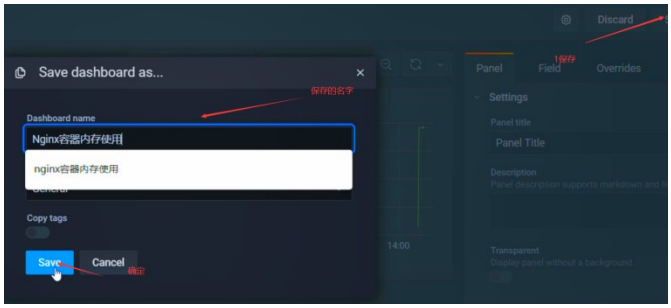

Configuration
Data Sources
Users
Teams
Plugins
Preferences
API Keys

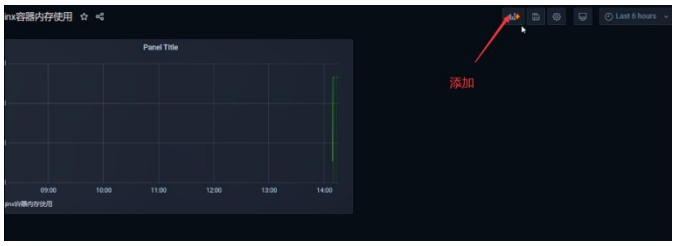
新建 Dashboard，如下图：↵

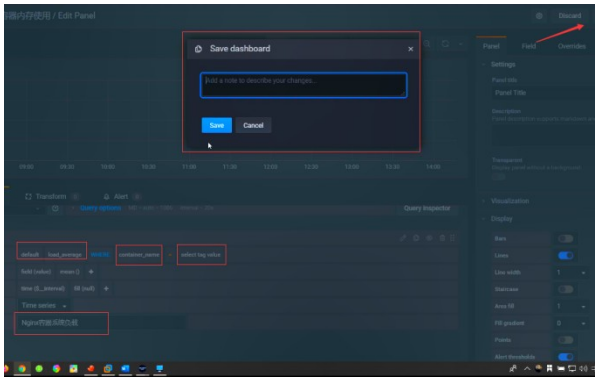












到这里 Cadvisor+InfluxDB+Grafana 容器监控系统就部署完成了，至于其它 grafana 的监控项配置不重点介绍。大家如果感兴趣可以参考一些官方资料都是图形化的界面操作。

Cadvisor+Prometheneus+Grafana

2.1,cadvisor 产品介绍

Cadvisor.是 Google 开源的一款用于展示和分析容器运行状态的可视化工具。通过在主机上运行 Cadvisor.用户可以轻松的获取到当前主机上容器的运行统计信息,并以图表的形式向用户展示。

2.2 部署 cadvisor

被监控主机上部署 cadvisor

```
[root@server04 ~]# docker rm -f $(docker ps -qa)   #删除上一次的实验环境（防止环境冲突）
[root@server04 ~]# docker rm -f $(docker ps -qa)
d6d7763a4434
5ff5235d1a43
c68b4cfe1a12
5845a093b3d9
[root@server04 ~]#
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       #部署 cadvisor 容器
```

2.3，访问 cadvisor 页面：192.168.200.14:8080

·prometheus 产品介绍

Prometheus 是一个最初在 SoundCloud 上构建的开源系统监视和警报工具包。自 2012 年成立以来，很多公司和组织都采用了 Prometheus，该项目拥有非常活跃的开发者和用户社区。 它现在是一个独立的开源项目，可以独立于任何公司进行维护。 为了强调这一点，并阐明项目的治理结构，Prometheus 于 2016 年加入 Cloud Native Computing Foundation，作为继 Kubernetes 之后的第二个托管项目。↵

prometheus 的主要特征：

·支持多维度数据模型

·promQL，一种灵活的查询语言

·不依赖分布式存储，单个服务器节点是自治的

·以 HTTP 方式，通过 pull 模型拉取时间序列数据

·支持通过中间网关推送时间序列数据

·通过服务发现或者静态配置，来发现目标服务对象

·支持多种多样的图表和界面展示

2.4，部署 prometheus

1）准备好镜像

```
[root@server04 ~]# docker load < prometheus.tar    #导入镜像
[root@server04 ~]# docker images              #查看导入的镜像
```

2）准备配置文件

```
[root@server04 ~]# vi  /tmp/prometheus.yml
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    #主键的端口

# 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'
```

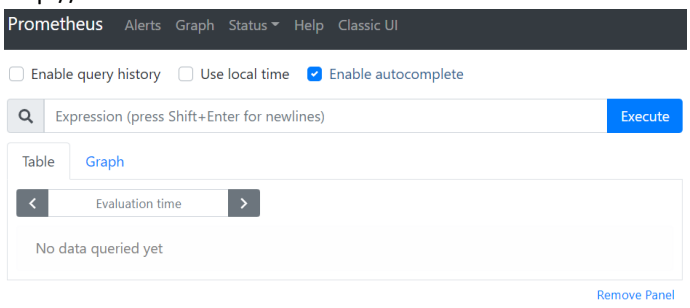
```
  # scheme defaults to 'http'.

  static_configs:
  - targets: ['localhost:9090']       #访问的端口
- job_name: 'docker'
  static_configs:
   - targets: ['192.168.200.14:8080']  #注意：本机的 IP 地址 cAdvisor 的 8080 端口
```

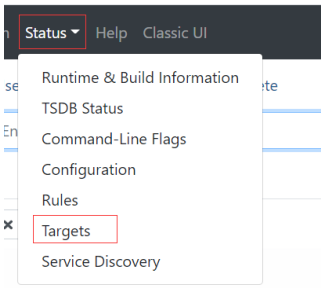
3）运行容器

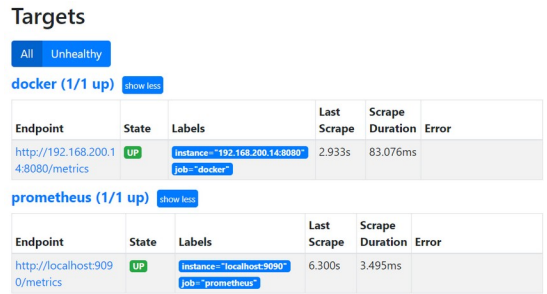
```
[root@server04 ~]#
docker run -d \
--name=prometheus  -p 9090:9090  \   #名字和端口
-v /tmp/prometheus.yml:/etc/prometheus/prometheus.yml \ #将/tmp/prometheus.yml
映射到/etc/prometheus/prometheus.yml 文件下
-v /etc/localtime:/etc/localtime \      #定义时间
prom/prometheus                 #镜像名
6ffc053bbce40a385d38d07f6517ecb8b57b443fe44
```

4）访问 prometheus 页面

http://192.168.200.14:9090

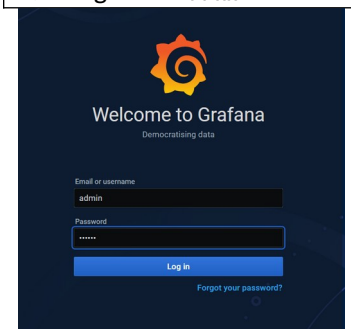






5）配置 grafana

```
[root@server04 ~]# docker run -d --name grafana --net monitor -p 3000:3000 grafana/grafana
#启动 grafana 容器
```



添加过程基本与上述一致

注意：prometheus 有现成的模板，可以直接调用使用