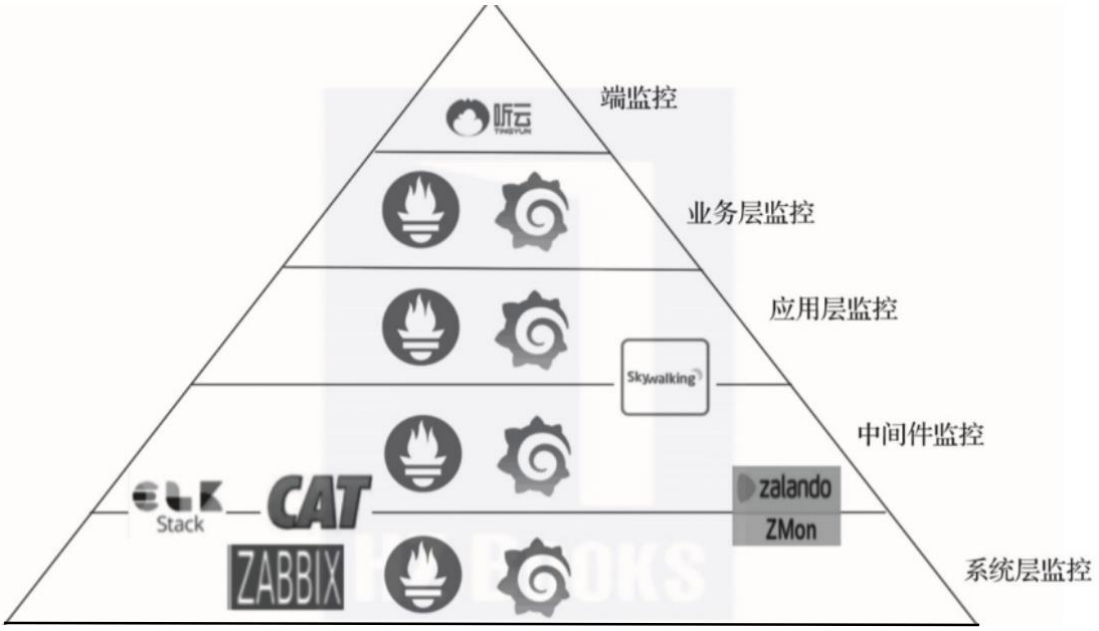
一：监控简介：

亚马逊副总裁、CTO 沃纳·沃格斯(Werner Voegls)说过："You build it, you run it, you monitor it."（你构建了它，你运行它，你就有责任监控它。）



1.1：监控概述：

系统层监控：

CPU、内存、网卡、磁盘利用率、带宽利用率、延迟、丢包率、交换机、路由器、防火墙等基础设施监控

web 监控：

打开速度、URL 打开状态码、API 接口可用性

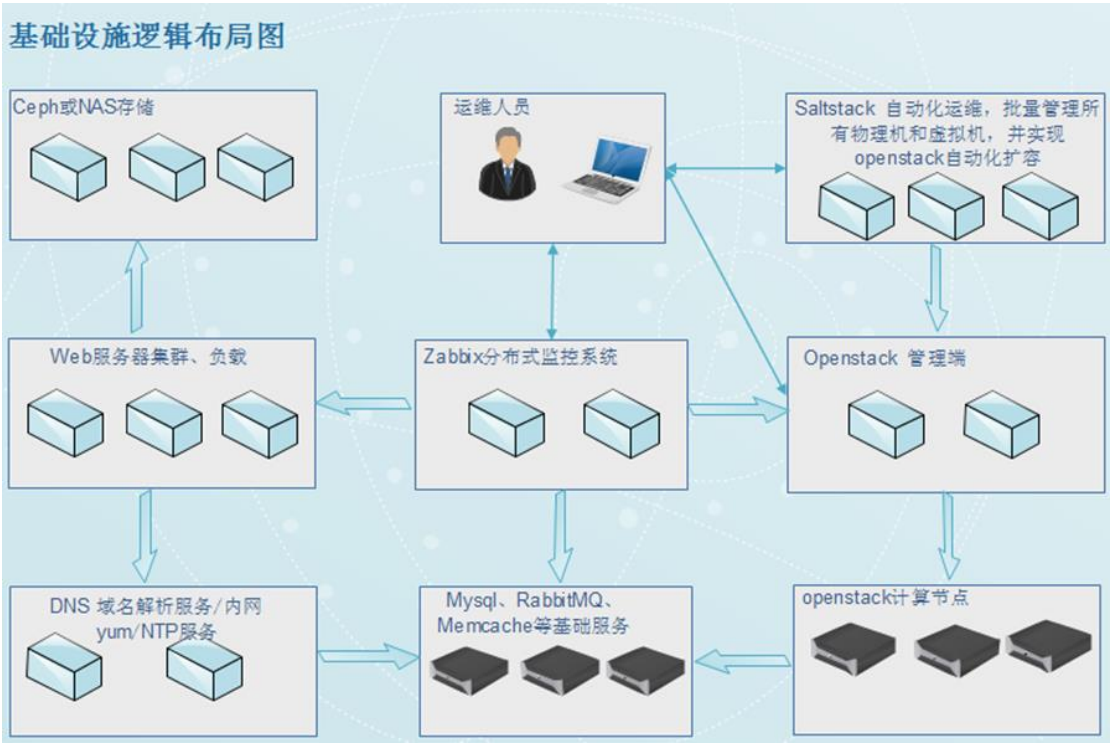
业务监控：

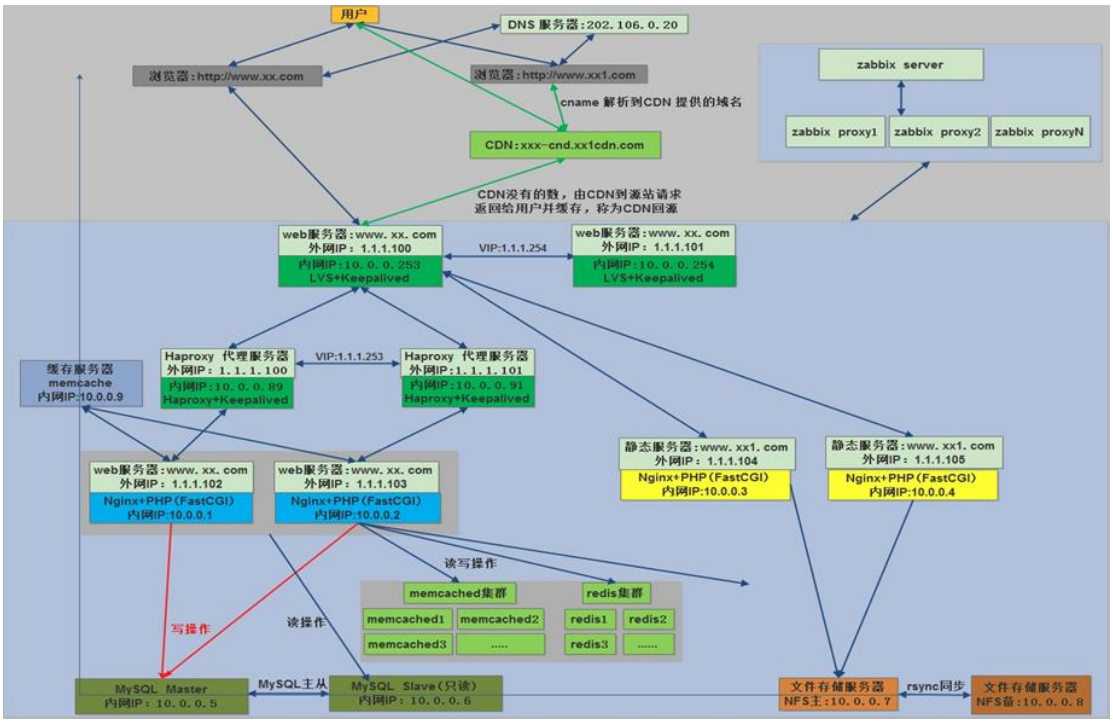
订单交易量、活跃用户量、支付量

中间件监控：

数据库、redis、kafka、MQ、API 等

1.2：监控规划：





1.3：监控方案：

开源监控软件：cacti、nagios、zabbix、smokeping、open-falcon、prometheus、商业监控等。

1.3.1：Cacti：

https://www.cacti.net/

https://github.com/Cacti/cacti

Cacti 是基于 LAMP 平台展现的网络流量监测及分析工具，通过 SNMP 技术或自定义脚本从目标设备/主机获取监控指标信息；其次进行数据存储，调用模板将数据存到数据库，使用 rrdtool 存储和更新数据，通过 rrdtool 绘制结果图形；最后进行数据展现，通过 Web 方式将监控结果呈现出来，常用于在数据中心监控网络设备。

1.3.2：Nagios：

https://www.nagios.org/

Nagios 用来监视系统和网络的开源应用软件，利用其众多的插件实现对本机和远端服务的监控，当被监控对象发生异常时，会及时向管理员告警，提供一批预设好的监控插件，用户可以之间调用，也可以自定义 Shell 脚本来监控服务，适合各企业的业务监控，可通过 Web 页面显示对象状态、日志、告警信息，分层告警机制及自定义监控相对薄弱。

1.3.3：SmokePing：

https://oss.oetiker.ch/smokeping/

http://blogs.studylinux.net/?p=794

Smokeping 是一款用于网络性能监测的开源监控软件，主要用于对 IDC 的网络状况，网络质量，稳定性等做检测，通过 rrdtool 制图方式，图形化地展示网络的时延情况，进而能够清楚的判断出网络的即时通信情况。

1.3.4：Open-falcon：

https://www.open-falcon.org/

https://github.com/XiaoMi/open-falcon

小米公司开源出来的监控软件 open-falcon(鹰隼)，监控能力和性能较强。

1.3.5：Nightingale 夜莺：

https://n9e.didiyun.com/

一款经过大规模生产环境验证的、分布式高性能的运维监控系统，由滴滴基于 open-falcon 二次开发后开源出来的分布式监控系统。

1.3.6：Zabbix：

https://www.zabbix.com/cn/

目前使用较多的开源监控软件，可横向扩展、自定义监控项、支持多种监控方式、可监控网络与服务等。

1.3.7：商业监控解决方案：

监控宝(https://www.jiankongbao.com/)
听云(https://www.tingyun.com/)

1.3.8：Prometheus：

Prometheus 是基于 go 语言开发的一套开源的监控、报警和时间序列数据库的组合，是由 SoundCloud 公司开发的开源监控系统,Prometheus 于 2016 年加入 CNCF（Cloud Native Computing Foundation,云原生计算基金会）, 2018 年 8 月 9 日 prometheus 成为 CNCF 基金会继 kubernetes 之后毕业的第二个项目，prometheus 在容器和微服务领域中得到了广泛的应用，其特点主要如下：

使用 key-value 的多维度格式保存数据

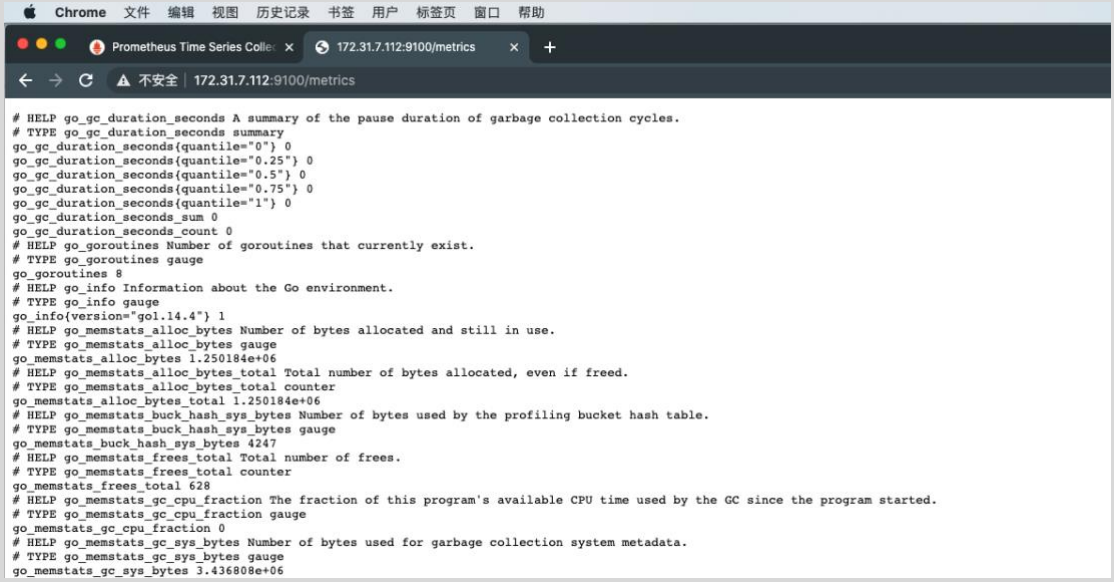
数据不使用 MySQL 这样的传统数据库，而是使用时序数据库，目前是使用的 TSDB，https://www.aliyun.com/product/hitsdb #阿里云 TSDB 简介

支持第三方 dashboard 实现更高的图形界面，如 grafana(Grafana 2.5.0 版本及以上)
功能组件化

不需要依赖存储，数据可以本地保存也可以远程保存

服务自动化发现

强大的数据查询语句功(PromQL,Prometheus Query Language)

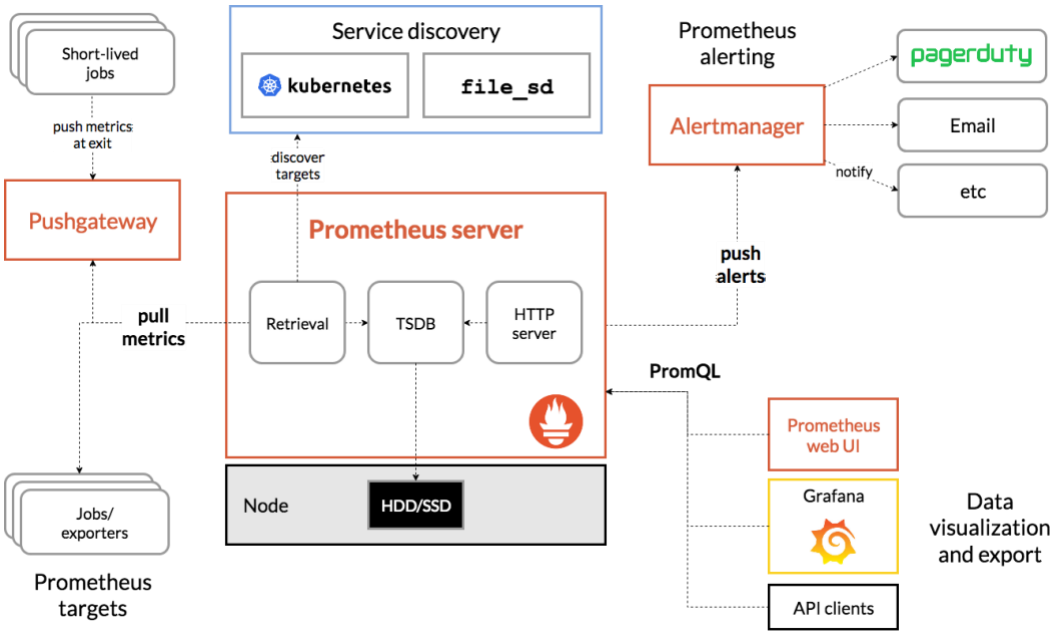


支持不同语言开发的客户端
官方和社区推出很多 exporter #https://prometheus.io/docs/instrumenting/exporters/

二： prometheus 简介与安装：

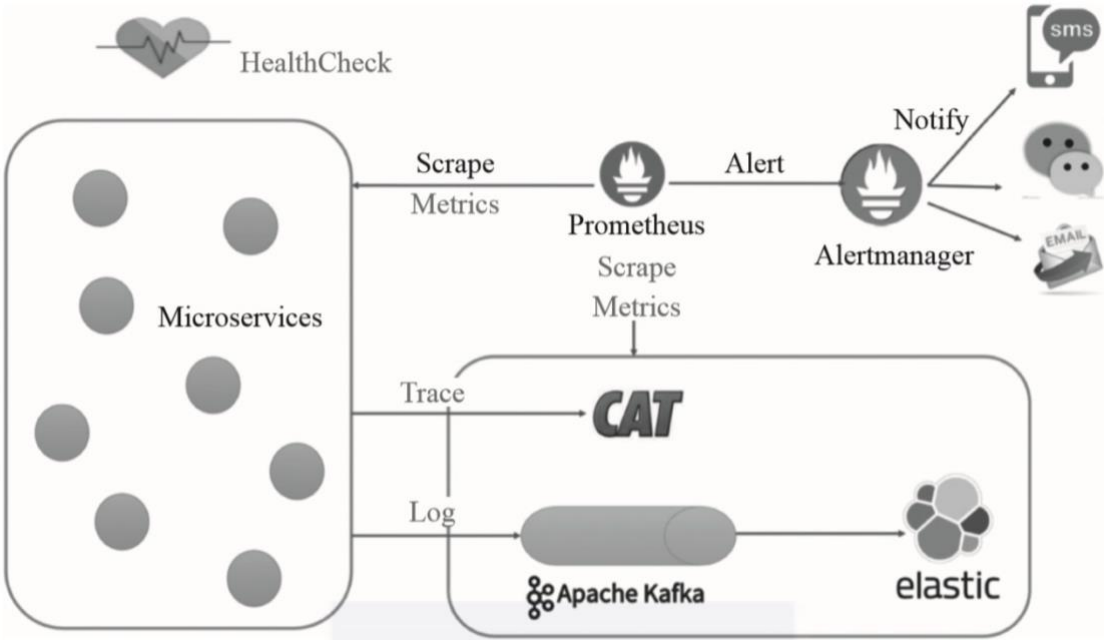
2.1：组件介绍:

prometheus server：主服务，接受外部 http 请求，收集、存储与查询数据等
prometheus targets: 静态收集的目标服务数据
service discovery：动态发现服务
prometheus alerting：报警通知
push gateway：数据收集代理服务器(类似于 zabbix proxy)
data visualization and export： 数据可视化与数据导出(访问客户端)



2.2：Prometheus server 安装：

prometheus 可以通过多种方式安装



2.2.1：通过容器启动：

https://prometheus.io/docs/prometheus/latest/installation/ #docker 镜像直接启动

```
# docker pull prom/prometheus:v2.31.1
# docker run -it    -d --restart=always -p 9090 prom/prometheus:v2.31.1
```

2.2.2：在线安装：

```
# apt search prometheus
# apt-cache    madison prometheus
prometheus | 2.15.2+ds-2 | http://mirrors.tuna.tsinghua.edu.cn/ubuntu focal/universe amd64
```

Packages

apt install prometheus

2.2.3：operator 部署：

https://github.com/coreos/kube-prometheus #operator 部署

git clone https://github.com/prometheus-operator/kube-prometheus.git

cd kube-prometheus/

kubectl apply -f manifests/setup

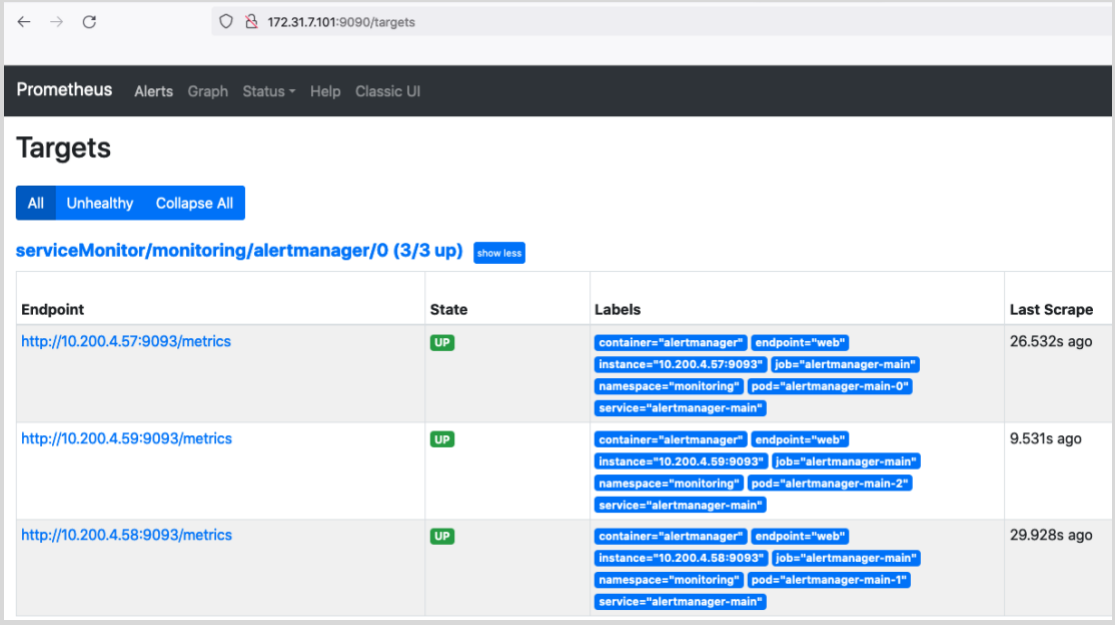
kubectl apply -f manifests/

```
root@k8s-master1:~/kube-prometheus# kubectl  get pod -n monitoring
NAME                                   READY   STATUS    RESTARTS   AGE
alertmanager-main-0                    2/2     Running   0          43m
alertmanager-main-1                    2/2     Running   0          43m
alertmanager-main-2                    2/2     Running   0          43m
blackbox-exporter-6798fb5bb4-jsltt     3/3     Running   0          43m
grafana-7476b4c65b-kwqns               1/1     Running   0          43m
kube-state-metrics-7fcc9c66b-tf7l2     3/3     Running   0          40m
node-exporter-42pjw                    2/2     Running   0          43m
node-exporter-6v95c                    2/2     Running   0          43m
node-exporter-ct2w9                    2/2     Running   0          43m
node-exporter-hn4lw                    2/2     Running   0          43m
node-exporter-htnnw                    2/2     Running   0          43m
node-exporter-n4clr                    2/2     Running   0          43m
prometheus-adapter-84db55f8d-fzx9n     1/1     Running   0          43m
prometheus-adapter-84db55f8d-z247s     1/1     Running   0          43m
prometheus-k8s-0                       2/2     Running   0          43m
prometheus-k8s-1                       2/2     Running   0          43m
prometheus-operator-75d9b475d9-pcp9l   2/2     Running   0          43m
root@k8s-master1:~/kube-prometheus#
```

2.2.3.1：验证 prometheus：

kubectl port-forward --help

kubectl --namespace monitoring port-forward --address 0.0.0.0 svc/prometheus-k8s 9090:9090

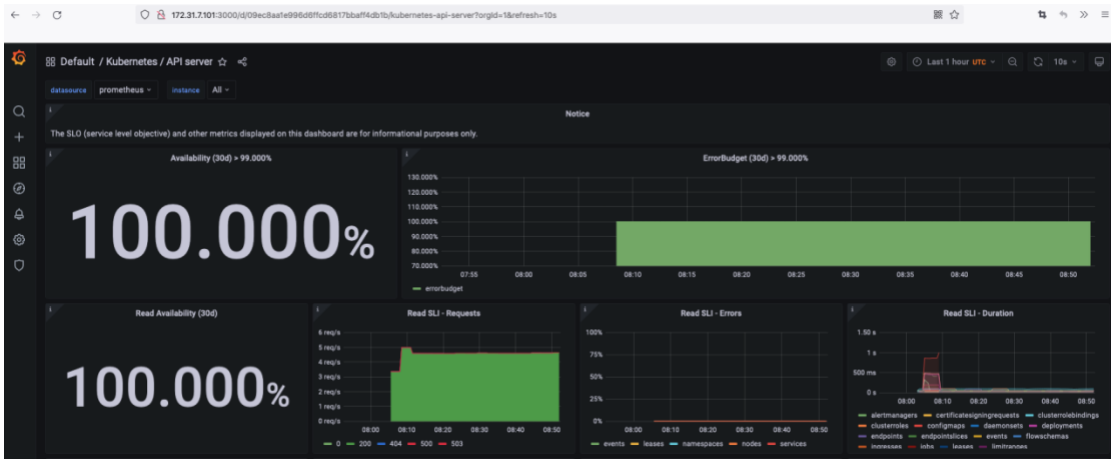


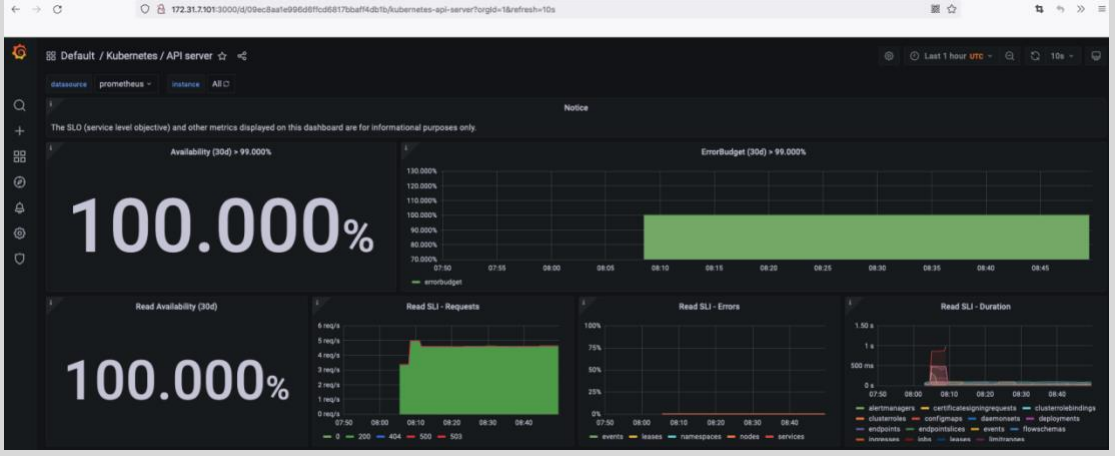
2.2.3.3：svc 暴露 prometheus：

```
# vim manifests/prometheus-service.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: prometheus
    app.kubernetes.io/name: prometheus
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 2.29.1
    prometheus: k8s
  name: prometheus-k8s
  namespace: monitoring
spec:
  type: NodePort
  ports:
  - name: web
    port: 9090
    nodePort: 39090
    targetPort: web
  selector:
    app: prometheus
    app.kubernetes.io/component: prometheus
    app.kubernetes.io/name: prometheus
    app.kubernetes.io/part-of: kube-prometheus
    prometheus: k8s
  sessionAffinity: ClientIP

# kubectl apply -f    manifests/prometheus-service.yaml
service/prometheus-k8s configured
```

2.2.3.3：验证 grafana：

```
# kubectl --namespace monitoring port-forward --address 0.0.0.0 svc/grafana 3000:3000
```





2.3.3.3：svc 暴露 grafana 服务：

```
# vim manifests/grafana-service.yaml
# cat manifests/grafana-service.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/component: grafana
    app.kubernetes.io/name: grafana
    app.kubernetes.io/part-of: kube-prometheus
    app.kubernetes.io/version: 8.1.1
  name: grafana
  namespace: monitoring
spec:
  type: NodePort
  ports:
  - name: http
    port: 3000
    targetPort: http
    nodePort: 33000
```

```
selector:
      app.kubernetes.io/component: grafana
      app.kubernetes.io/name: grafana
      app.kubernetes.io/part-of: kube-prometheus
```

2.2.4：官方二进制：

https://prometheus.io/download/

2.2.4.1：下载官方二进制：

```
# mkdir    /apps
# tar xvf prometheus-2.31.1.linux-amd64.tar.gz
# ln -sv /apps/prometheus-2.31.1.linux-amd64 /apps/prometheus
'/apps/prometheus' -> '/apps/prometheus-2.31.1.linux-amd64'
# cd /apps/prometheus
# ll
   prometheus* #prometheus 服务可执行程序
   prometheus.yml #配置文件
   promtool* #测试工具，用于检测配置 prometheus 配置文件、检测 metrics 数据等

# ./promtool check config prometheus.yml
Checking prometheus.yml
   SUCCESS: 0 rule files found
```

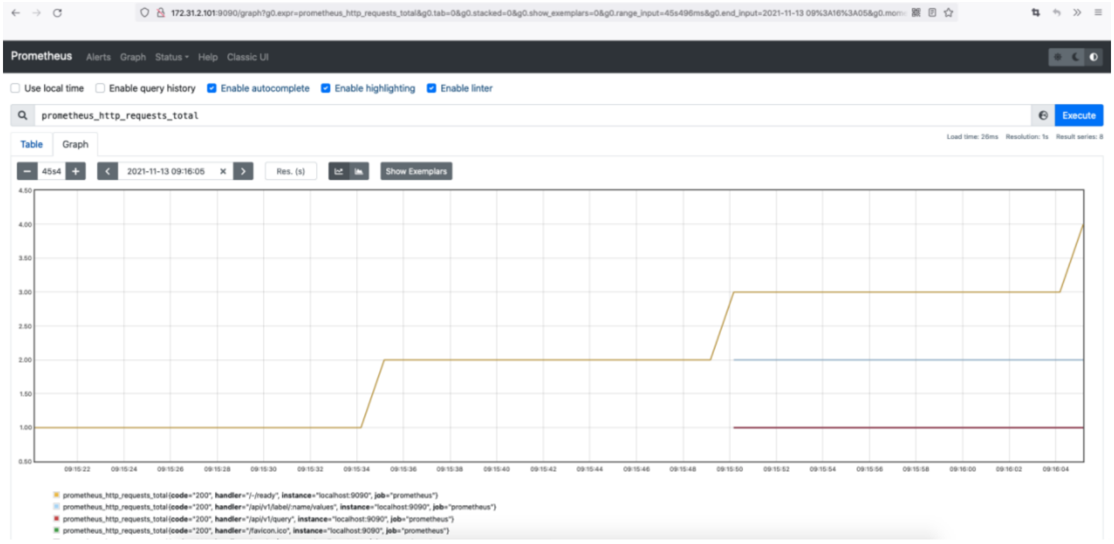
2.2.4.2：创建 service 文件：

```
# vim /etc/systemd/system/prometheus.service
[Unit]
Description=Prometheus Server
Documentation=https://prometheus.io/docs/introduction/overview/
After=network.target

[Service]
Restart=on-failure
WorkingDirectory=/apps/prometheus/
ExecStart=/apps/prometheus/prometheus      --config.file=/apps/prometheus/prometheus.yml

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

2.2.4.3：启动 prometheus 服务：

```
# systemctl    daemon-reload &&    systemctl    restart    prometheus &&    systemctl    enable
prometheus
```

2.2.4.4：验证 prometheu 界面：



2.3：node_exporter 安装：

收集并导出 node 节点的指标数据。

2. 3. 1：下载官方二进制：

https://prometheus.io/download/

```
# wget https://github.com/prometheus/node_exporter/releases/download/v1.2.2/node_exporter-1.2.2.linux-amd64.tar.gz

# tar xvf node_exporter-1.2.2.linux-amd64.tar.gz
# ln -sv /apps/blackbox_exporter-0.19.0.linux-amd64 /apps/blackbox_exporter
'/apps/blackbox_exporter' -> '/apps/blackbox_exporter-0.19.0.linux-amd64'

# ll /apps/node_exporter/
  node_exporter #可执行程序
```

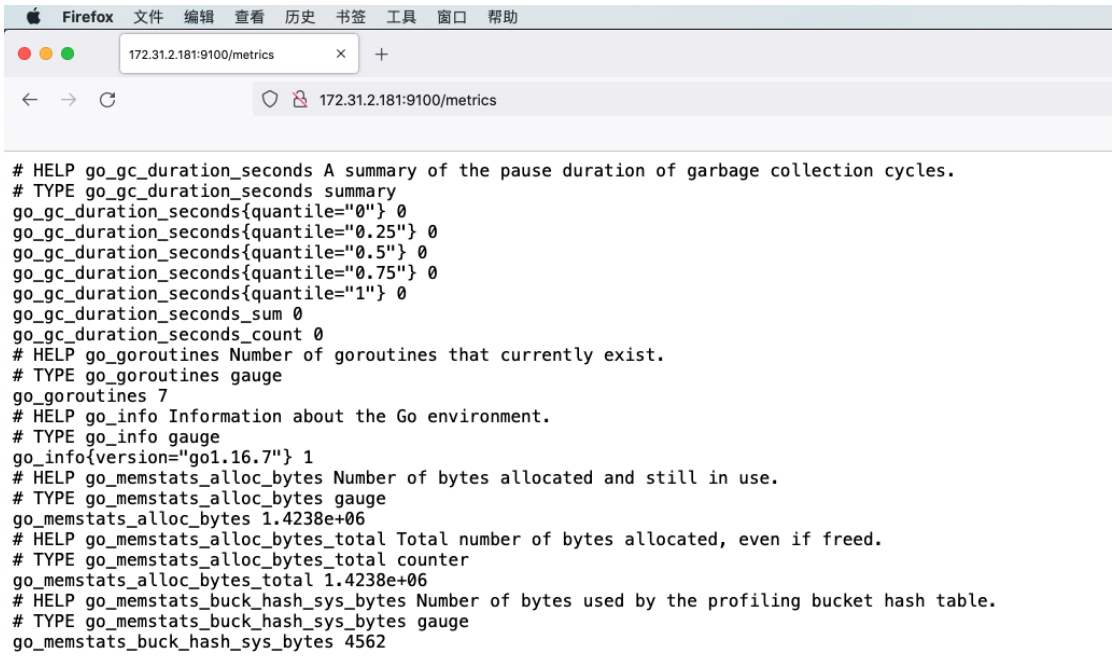
2. 3. 1：创建 service 文件：

```
# vim    /etc/systemd/system/node-exporter.service
[Unit]
Description=Prometheus Node Exporter
After=network.target

[Service]
ExecStart=/apps/node_exporter/node_exporter

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

2.3.3：启动 exporter 服务：

```
# systemctl daemon-reload && systemctl restart node-exporter && systemctl enable node-exporter.service
```

2.3.4：验证 node exporter web 界面：



2.4：Prometheus 采集 node 指标数据：

配置 Prometheus 通过 node exporter 采集 node 节点的监控指标数据。

2.4.1：prometheus 配置文件：

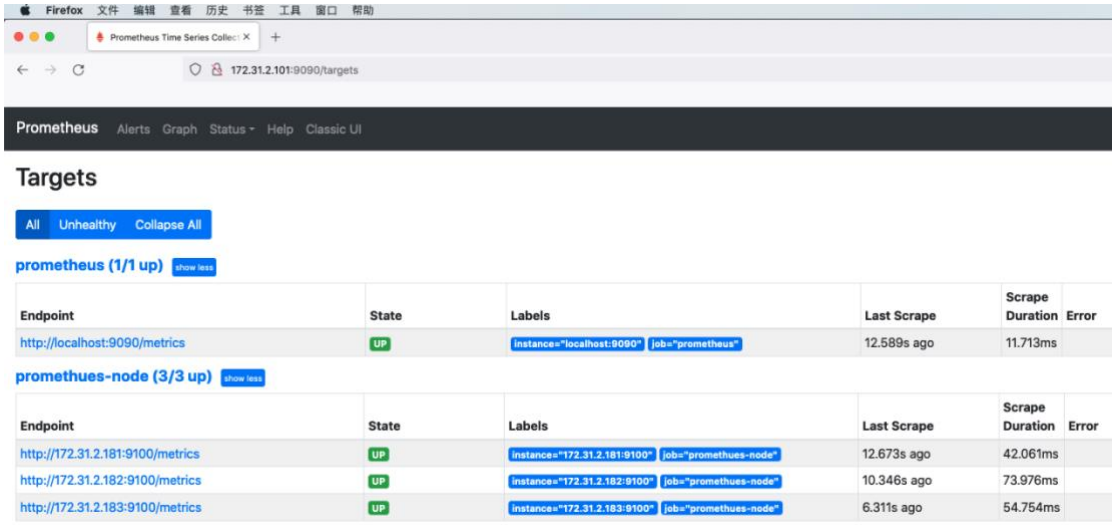
```
# vim /apps/prometheus/prometheus.yml

# my global config
global: #
  scrape_interval:  15s # Set the scrape interval to every 15 seconds. Default is every 1 minute. #数据收集间隔时间，如果不配置默认为一分钟
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute. #规则扫描间隔时间，如果不配置默认为一分钟
  # scrape_timeout is set to the global default (10s). #超时时间
```

```
 # Alertmanager configuration
alerting: #报警通知配置
   alertmanagers:
   - static_configs:
       - targets:
          # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files: #规则配置
   # - "first_rules.yml"
   # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs: #数据采集目标配置
   # 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']
   - job_name: 'promethues-node'
       static_configs:
          - targets: ['172.31.2.181:9100','172.31.2.182:9100','172.31.2.183:9100']
```

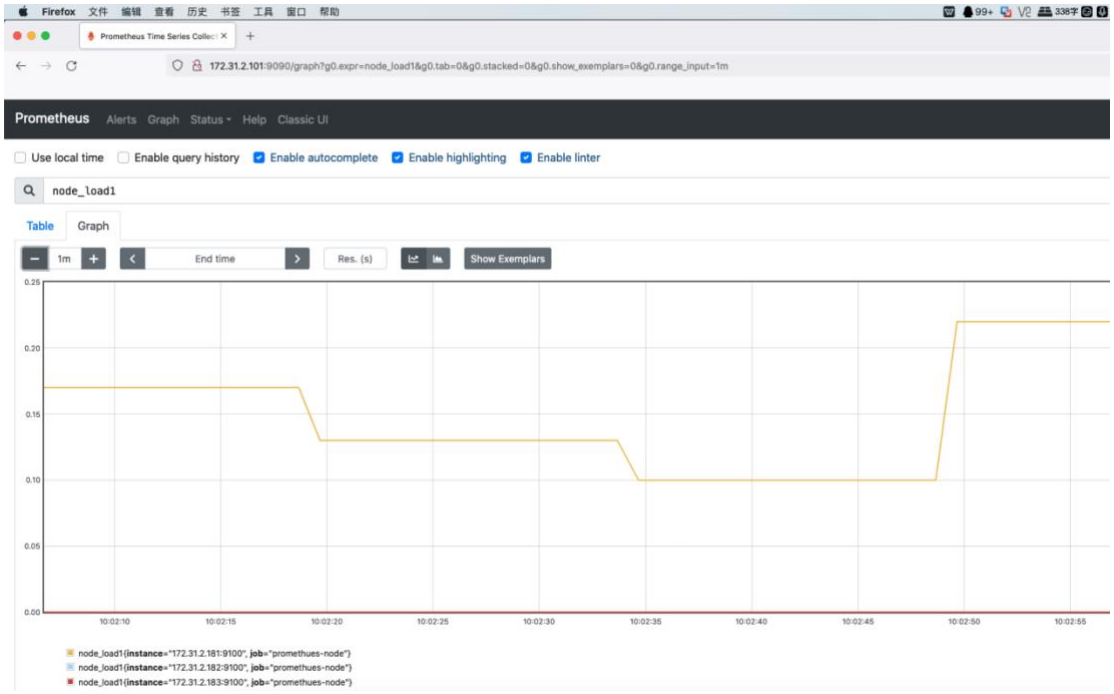
2.4.2：重启 prometheus 服务：

```
# systemctl    restart prometheus.service
```

2.4.3：prometheus 验证 node 节点状态：



2.4.4：prometheus 验证 node 数据：



2.5：blackbox exporter：

https://prometheus.io/download/#blackbox_exporter

blackbox_exporter 是 Prometheus 官方提供的一个 exporter，可以通过 HTTP, HTTPS, DNS, TCP 和 ICMP 对被监控节点进行监控和数据采集。

HTTP/HTPPS：URL/API 可用性检测
TCP：端口监听检测
ICMP：主机存活检测
DNS：域名解析

2.5.1：部署 blackbox exporter：

```
#                                                                                                    wget
https://github.com/prometheus/blackbox_exporter/releases/download/v0.19.0/blackbox_exporter-0.19.0.linux-amd64.tar.gz
# ln -sv /apps/node_exporter-1.2.2.linux-amd64 /apps/blackbox_exporter
```

2.5.2：便捷配置文件：

2.5.3：创建 blackbox exporter 启动文件：

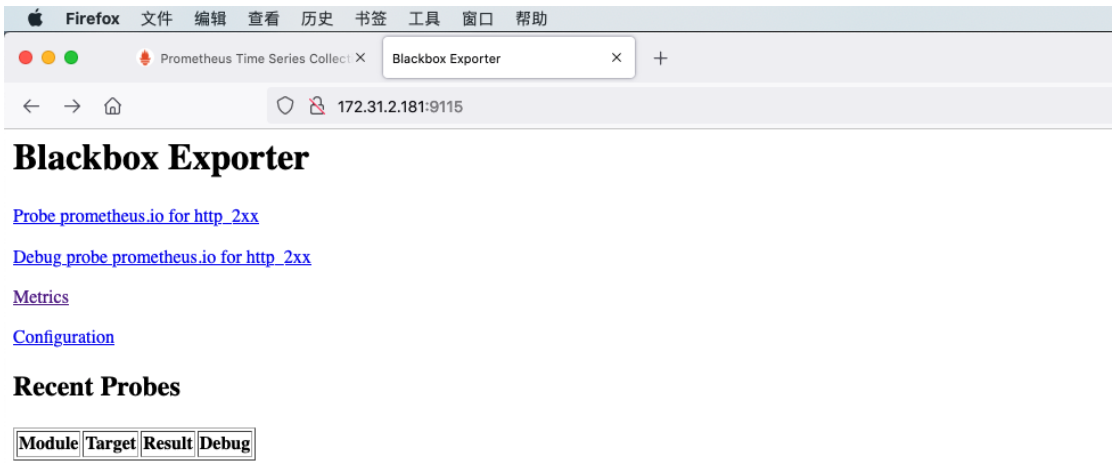
```
# vim /etc/systemd/system/blackbox-exporter.service
[Unit]
Description=Prometheus Blackbox Exporter
After=network.target
```

```
[Service]
Type=simple
User=root
Group=root
ExecStart=/apps/blackbox_exporter/blackbox_exporter \
      --config.file=/apps/blackbox_exporter/blackbox.yml \
      --web.listen-address=:9115
Restart=on-failure

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

2.5.4：启动服务：

```
# systemctl daemon-reload && systemctl restart blackbox_exporter && systemctl enable blackbox_exporter
```

2.5.5：验证 web 界面：



2.5.6：blackbox exporter 实现 URL 监控：

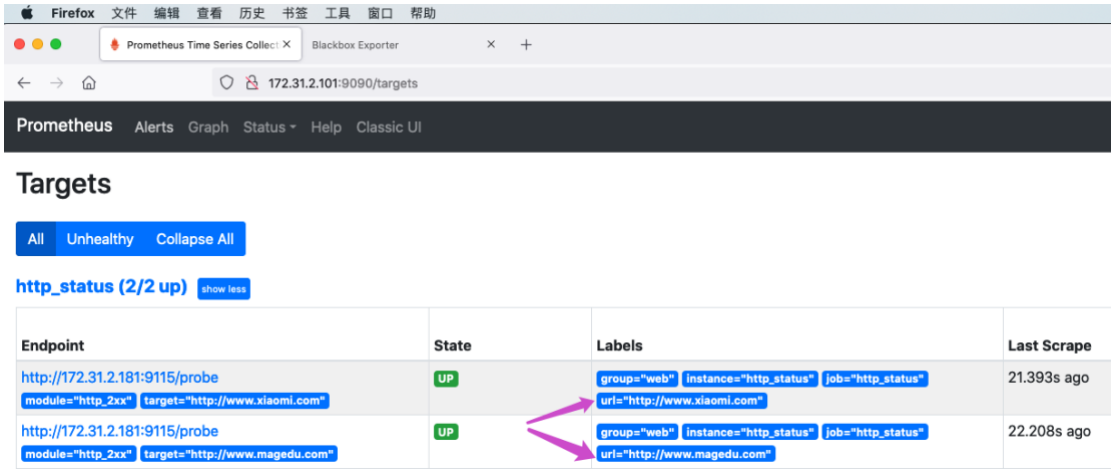
prometheus 调用 blackbox exporter 实现对 URL/ICMP 的监控。

2.5.6.1：URL 监控配置：

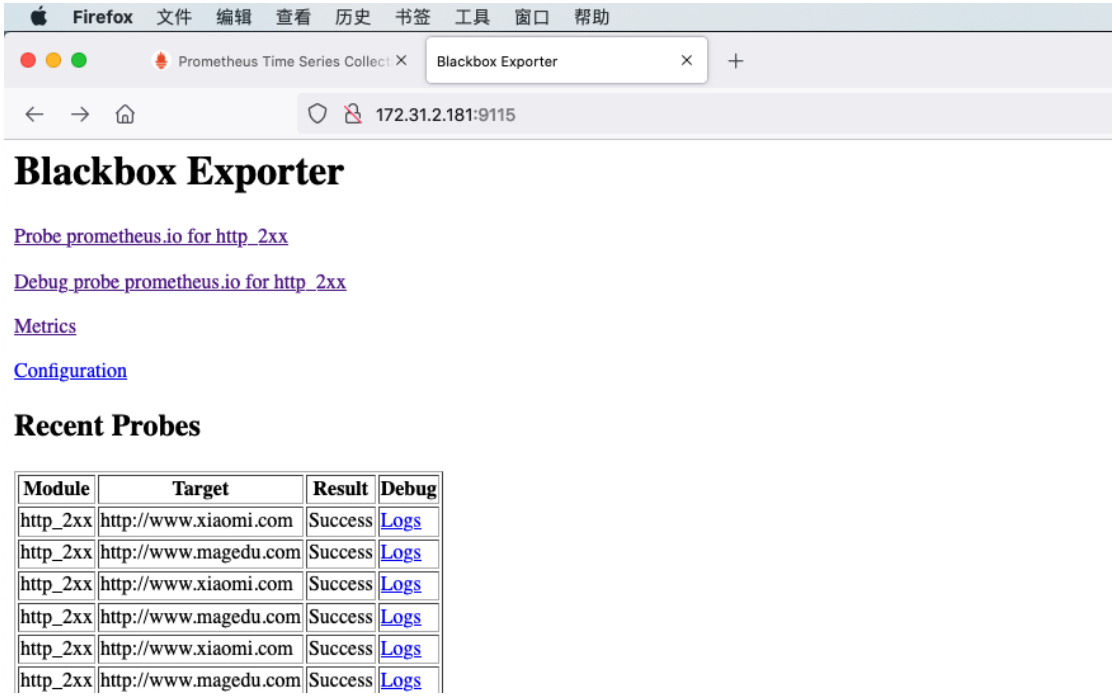
```
# vim /apps/prometheus/prometheus.yml
#  网站监控
  - job_name: 'http_status'
    metrics_path: /probe
    params:
      module: [http_2xx]
    static_configs:
      - targets: ['http://www.xiaomi.com', 'http://www.magedu.com']
        labels:
```

```
instance: http_status
            group: web
      relabel_configs:
        - source_labels: [__address__] #relabel 通过将__address__(当前目标地址)写入
__param_target 标签来创建一个 label。
          target_label: __param_target #监控目标 www.xiaomi.com,作为__address__的 value
        - source_labels: [__param_target] #监控目标
          target_label: url #将监控目标与 url 创建一个 label
        - target_label: __address__
          replacement: 172.31.2.181:9115

# /apps/prometheus/promtool    check config /apps/prometheus/prometheus.yml
Checking /apps/prometheus/prometheus.yml
   SUCCESS: 0 rule files found

# systemctl    restart    prometheus.service
```

2.5.6.2：prometheus 验证数据：



2.5.6.3：blackbox exporter 界面验证数据:



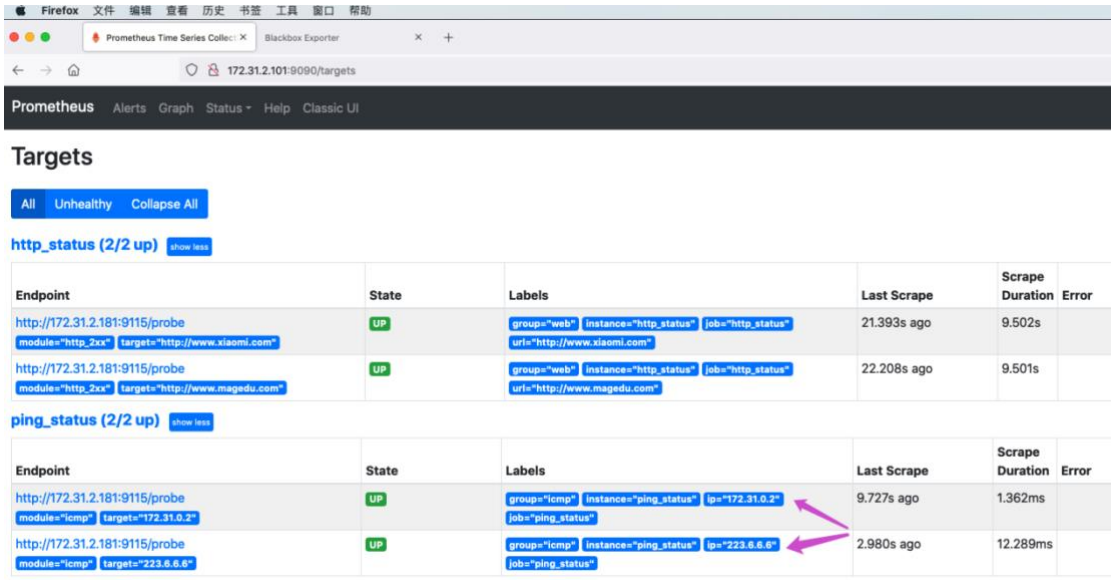
2.5.7：blackbox exporter 实现 ICMP 监控:

2.5.7.1：ICMP 监控配置:

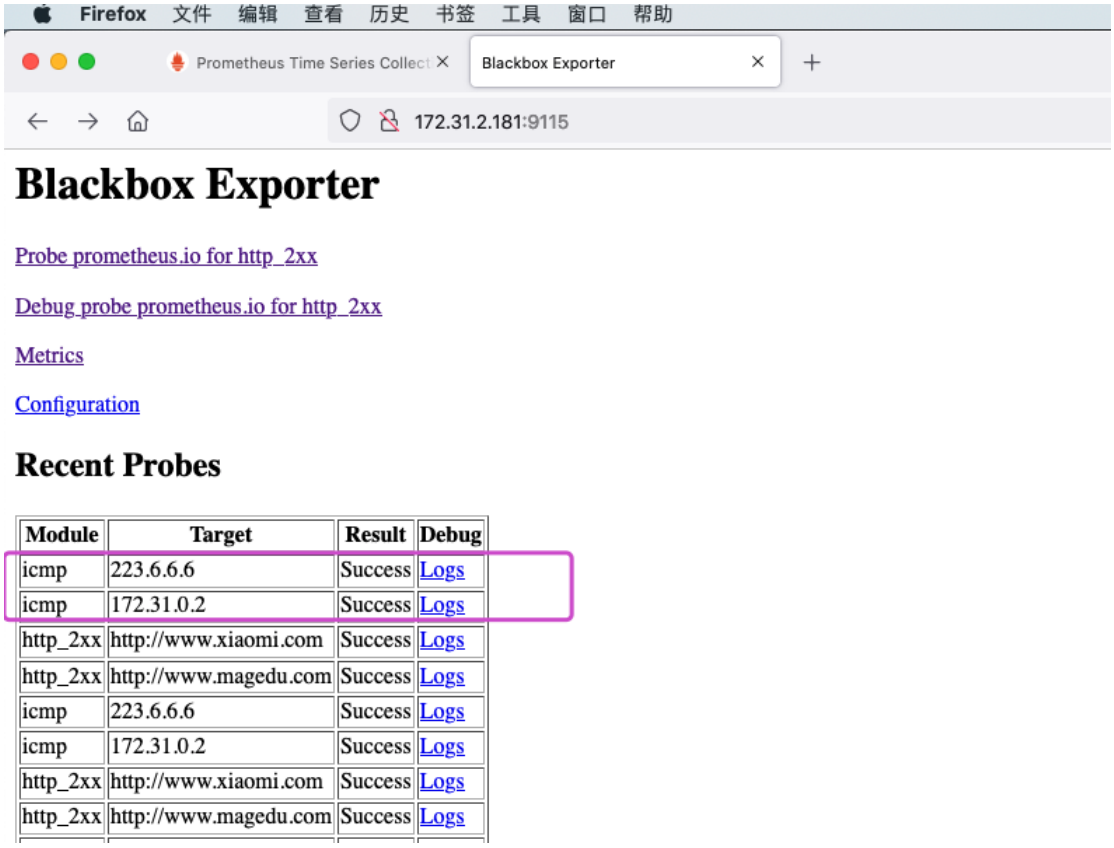
```
# vim /apps/prometheus/prometheus.yml
# icmp 检测
  - job_name: 'ping_status'
    metrics_path: /probe
    params:
      module: [icmp]
    static_configs:
      - targets: ['172.31.0.2',"223.6.6.6"]
        labels:
          instance: 'ping_status'
          group: 'icmp'
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: ip #将 ip 与__param_target 创建一个 label
      - target_label: __address__
        replacement: 172.31.2.181:9115

# /apps/prometheus/promtool  check config /apps/prometheus/prometheus.yml
Checking /apps/prometheus/prometheus.yml
  SUCCESS: 0 rule files found
```

```
# systemctl    restart    prometheus.service
```

2.5.7.2：prometheus 验证数据：



2.5.7.3：blackbox exporter 验证数据：

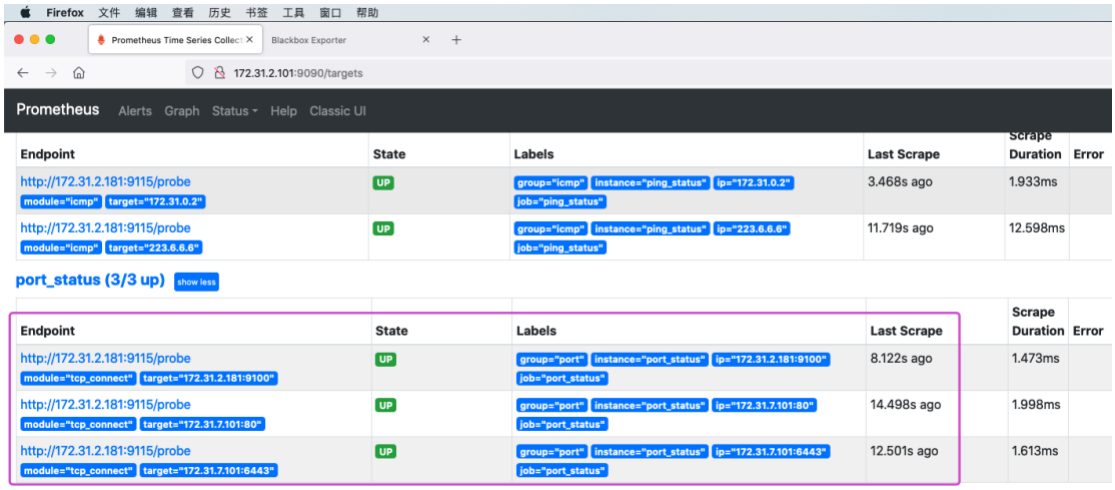


2.5.8：blackbox exporter 实现端口监控：

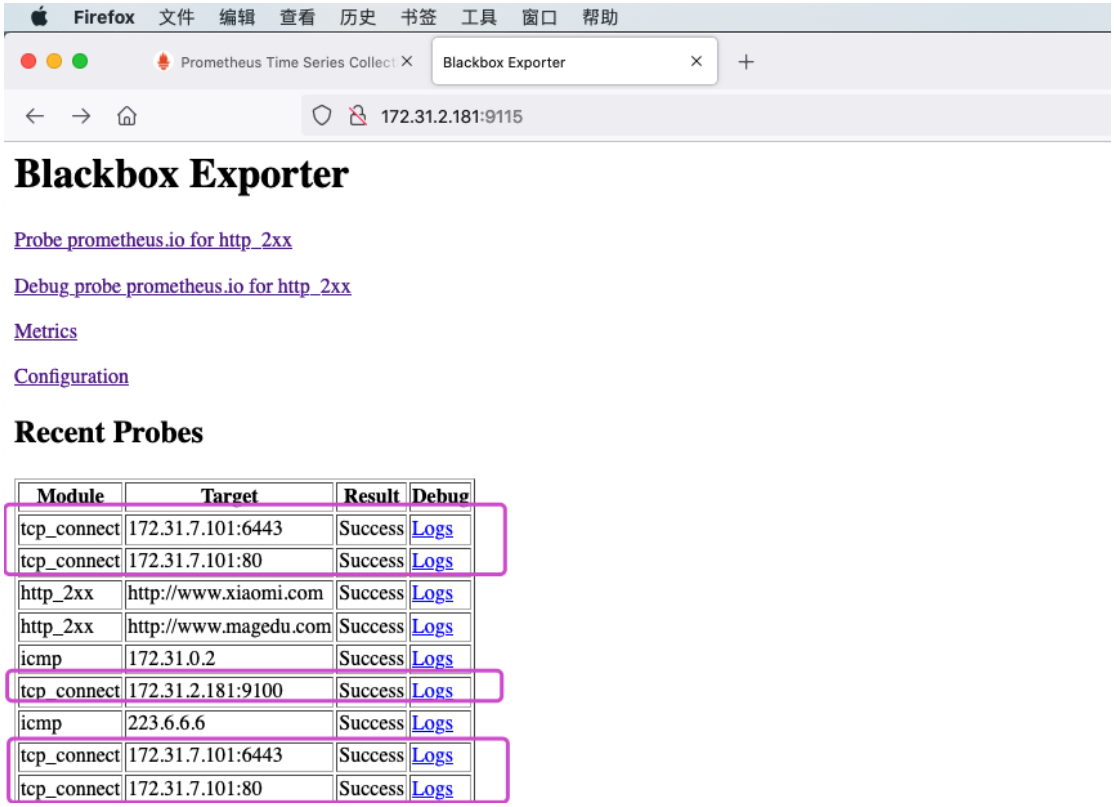
2.5.8.1：端口监控配置：

```
# vim /apps/prometheus/prometheus.yml
#  端口监控
  - job_name: 'port_status'
    metrics_path: /probe
    params:
      module: [tcp_connect]
    static_configs:
      - targets: ['172.31.2.181:9100', '172.31.7.101:80','172.31.7.101:6443']
        labels:
          instance: 'port_status'
          group: 'port'
    relabel_configs:
      - source_labels: [__address__]
        target_label: __param_target
      - source_labels: [__param_target]
        target_label: ip
      - target_label: __address__
        replacement: 172.31.7.181:9115

# /apps/prometheus/promtool    check config /apps/prometheus/prometheus.yml
Checking /apps/prometheus/prometheus.yml
  SUCCESS: 0 rule files found
```

2.5.8.2：prometheus 验证数据：



2.5.8.3：blackbox exporter 界面验证数据：



三： PromQL 简介：

Prometheus 提供一个函数式的表达式语言 PromQL (Prometheus Query Language)，可以使用户实时地查找和聚合时间序列数据，表达式计算结果可以在图表中展示，也可以在 Prometheus 表达式浏览器中以表格形式展示，或者作为数据源，以 HTTP API 的方式提供给外部系统使用.

3.1：PromQL 基本查询：

```
node_memory_MemTotal_bytes #查询 node 节点总内存大小
node_memory_MemFree_bytes #查询 node 节点剩余可用内存
node_memory_MemTotal_bytes{instance="172.31.7.111:9100"} #查询指定节点的总内存
node_memory_MemFree_bytes{instance="172.31.7.111:9100"} #查询指定节点的可用内存

node_disk_io_time_seconds_total{device="sda"} #查询指定磁盘的每秒磁盘 io
node_filesystem_free_bytes{device="/dev/sda1",fstype="xfs",mountpoint="/"} #查看指定磁盘的磁盘剩余空间

# HELP node_load1 1m load average. #CPU 负载
# TYPE node_load1 gauge
node_load1 0.1
```

```
# HELP node_load15 15m load average.
# TYPE node_load15 gauge
node_load15 0.17
# HELP node_load5 5m load average.
# TYPE node_load5 gauge
node_load5 0.13
```

3.2：PromQL 数据类型：

瞬时向量(instant vector): 是一组时间序列，每个时间序列包含单个数据样本，比如node_memory_MemTotal_bytes 查询当前剩余内存就是一个瞬时向量，该表达式的返回值中只会包含该时间序列中的最新的一个样本值，而相应的这样的表达式称之为瞬时向量表达式。

范围向量(range vector):是指在任何一个时间范围内，抓取的所有度量指标数据.比如最近一天的网卡流量趋势图。

标量(scalar)：是一个浮点数类型的数据值，使用 node_load1 获取到时一个瞬时向量，但是可用使用内置函数 scalar()将瞬时向量转换为标量。

字符串(string):字符串类型的数据，目前使用较少

3.3：PromQL 匹配器：

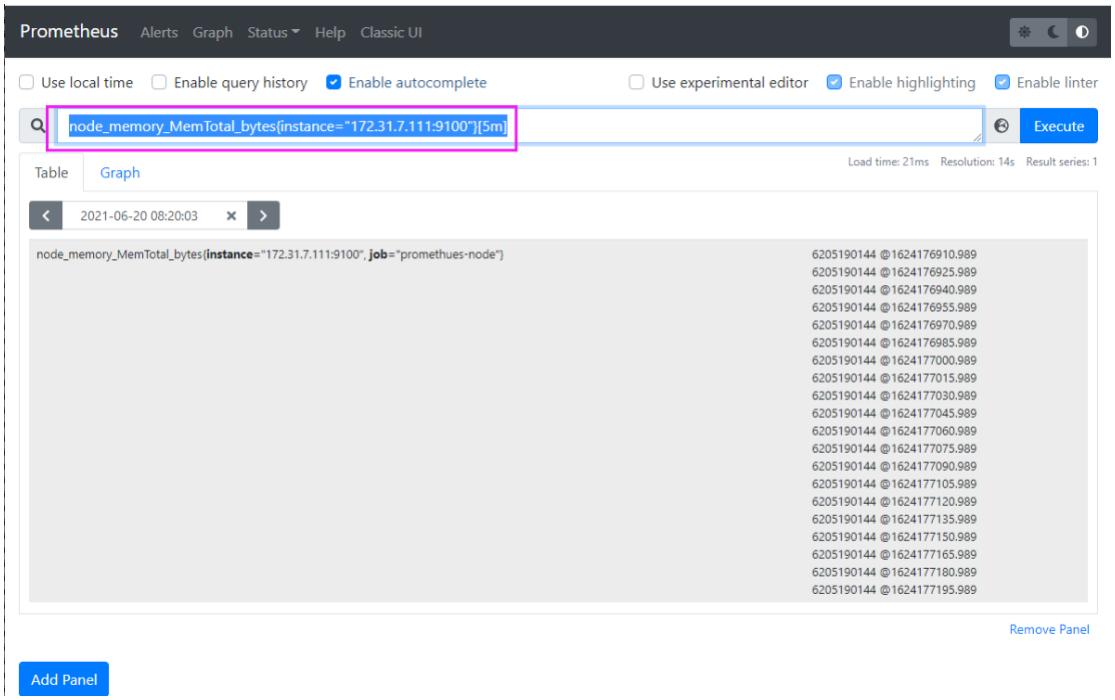
= :选择与提供的字符串完全相同的标签。
!= :选择与提供的字符串不相同的标签。
=~ :选择正则表达式与提供的字符串（或子字符串）相匹配的标签。
!~ :选择正则表达式与提供的字符串（或子字符串）不匹配的标签。

```
#查询格式<metric name>{<label name>=<label value>, ...}
node_load1{instance="172.31.7.111:9100"}
node_load1{job="promethues-node"}

node_load1{job="promethues-node",instance="172.31.7.111:9100"}
node_load1{job="promethues-node",instance!="172.31.7.111:9100"}
```

3.3：PromQL 时间范围：

```
s - 秒
m - 分钟
h - 小时
d - 天
w - 周
y - 年

node_memory_MemTotal_bytes{} # 瞬时向量表达式，选择当前最新的数据
node_memory_MemTotal_bytes{}[5m] # 区间向量表达式，选择以当前时间为基准，5 分钟内的数据
```

node_memory_MemTotal_bytes{instance="172.31.7.111:9100"}[5m]



3.4：PromQL 运算符：

\+ 加法

\- 减法

* 乘法

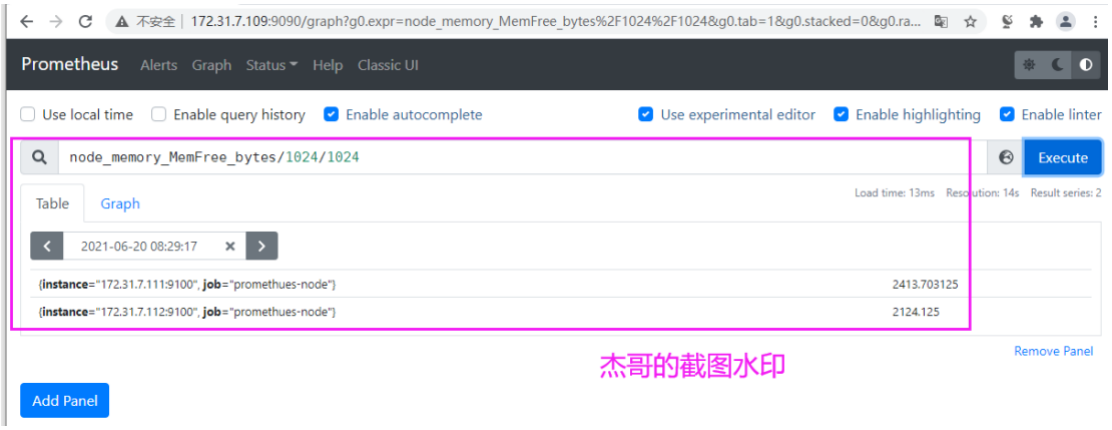
/ 除法

% 模

^ 幂等

node_memory_MemFree_bytes/1024/1024 #将内存进行单位转换

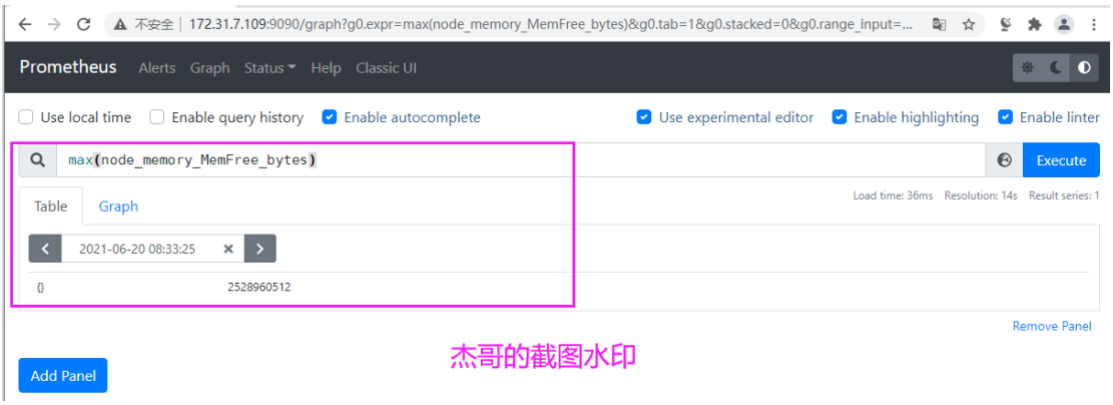
node_disk_read_bytes_total{device="sda"} + node_disk_written_bytes_total{device="sda"} #计算磁盘每秒读写数据量



3.5：PromQL 聚合运算：

sum (求和)

```
min (最小值)
max (最大值)
avg (平均值)
stddev (标准差)
stdvar (标准差异)
count (计数)
count_values (对 value 进行计数)
bottomk (样本值最小的 k 个元素)
topk (样本值最大的 k 个元素)
quantile (分布统计)

max(node_memory_MemFree_bytes) #某个指标数据的最大值
sum(http_requests_total) #计算 http_requests_total 最近的请求总量
```



四： grafana：

https://grafana.com/docs/ #官方安装文档

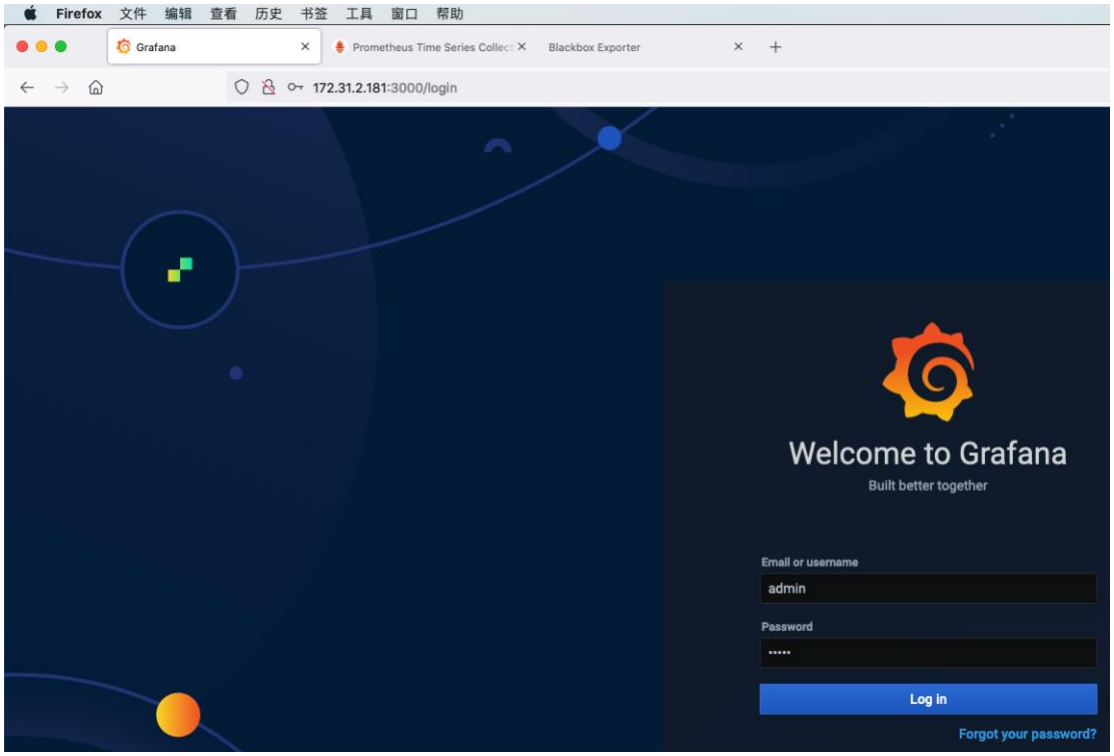
grafana 是一个开源的可视化工具，可以调用 prometheus、mysql 等数据源进行更绚丽的前端可视化。

4.1：安装并启动 grafana：

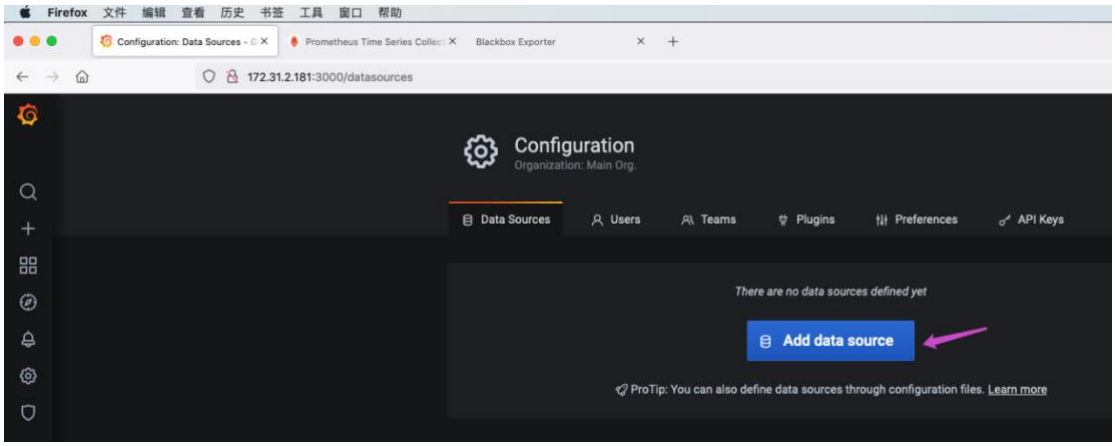
```
# sudo apt-get install -y adduser libfontconfig1
# dpkg -i grafana-enterprise_7.5.11_amd64.deb

配置文件：
# vim /etc/grafana/grafana.ini
[server]
# Protocol (http, https, socket)
```

```
protocol = http

# The ip address to bind to, empty will bind to all interfaces
http_addr = 0.0.0.0

# The http port    to use
http_port = 3000

# systemctl    restart grafana-server
# systemctl    enable grafana-server
```

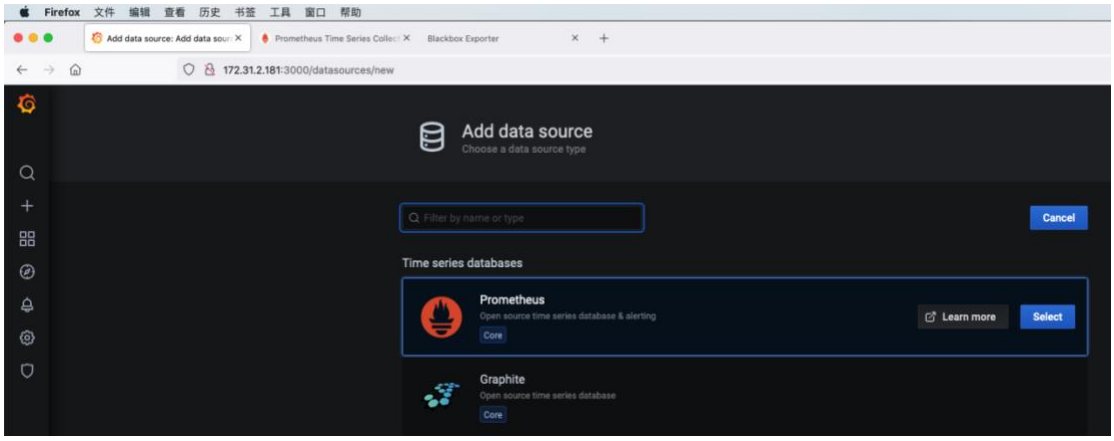
4.2：登录 web 界面：

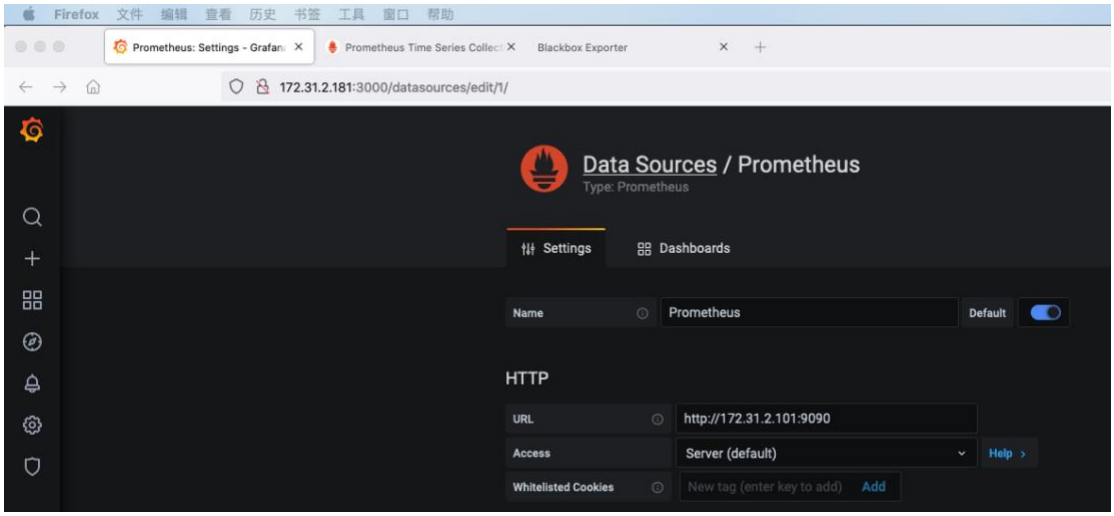


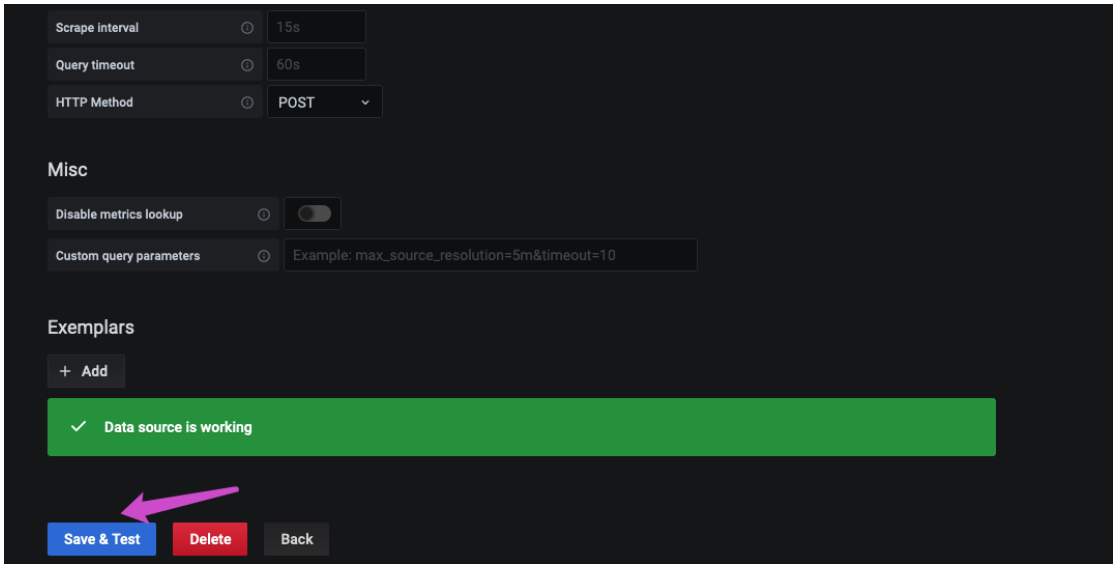
添加数据源：



选择 prometheus：





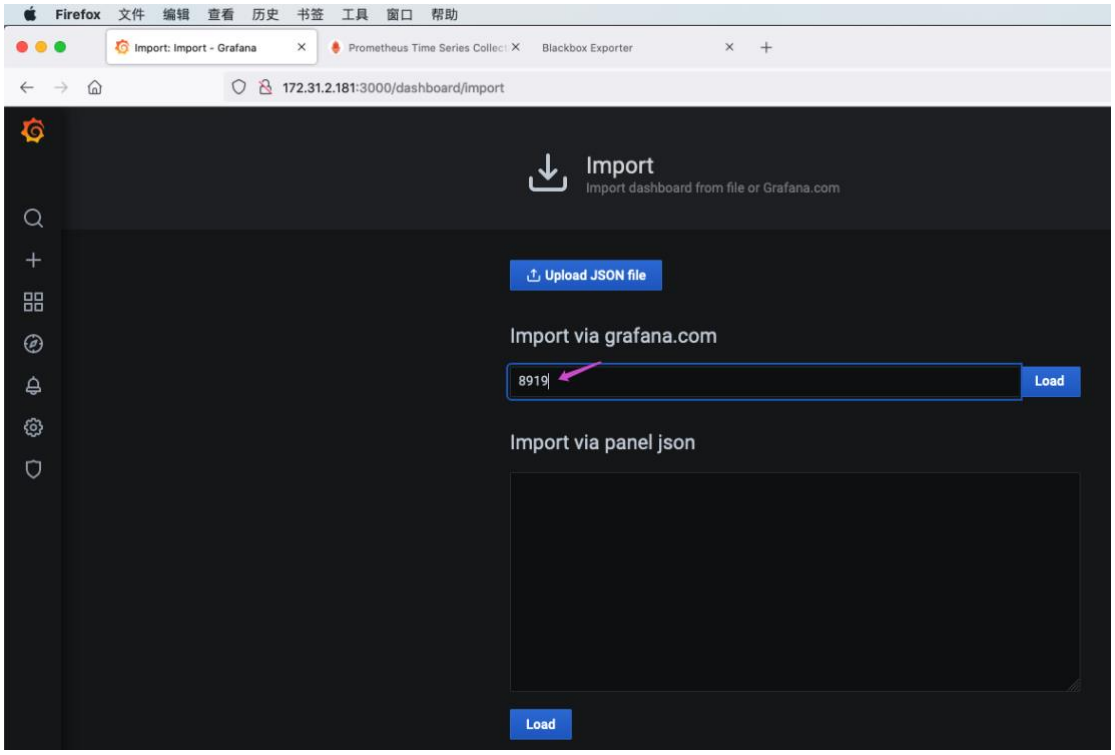


4.3：导入 node exporter 模板：

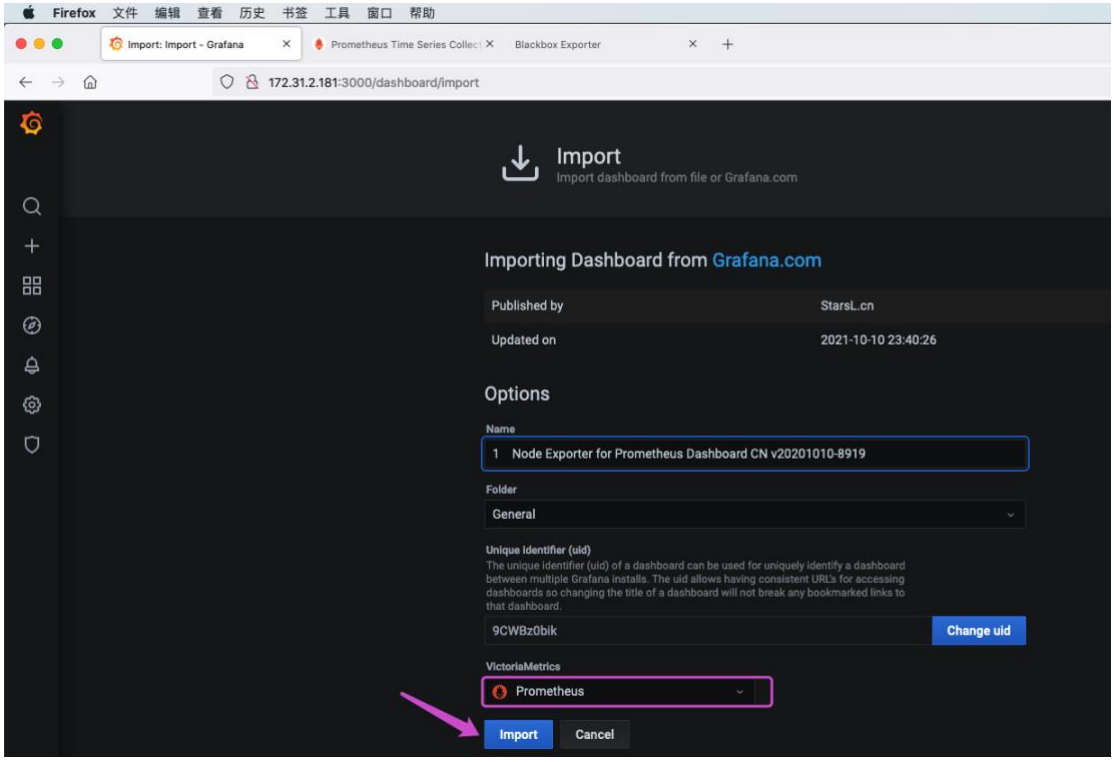
在 grafana 官网搜索并在 grafana 导入模板

4.3.1：搜索模板-8919：

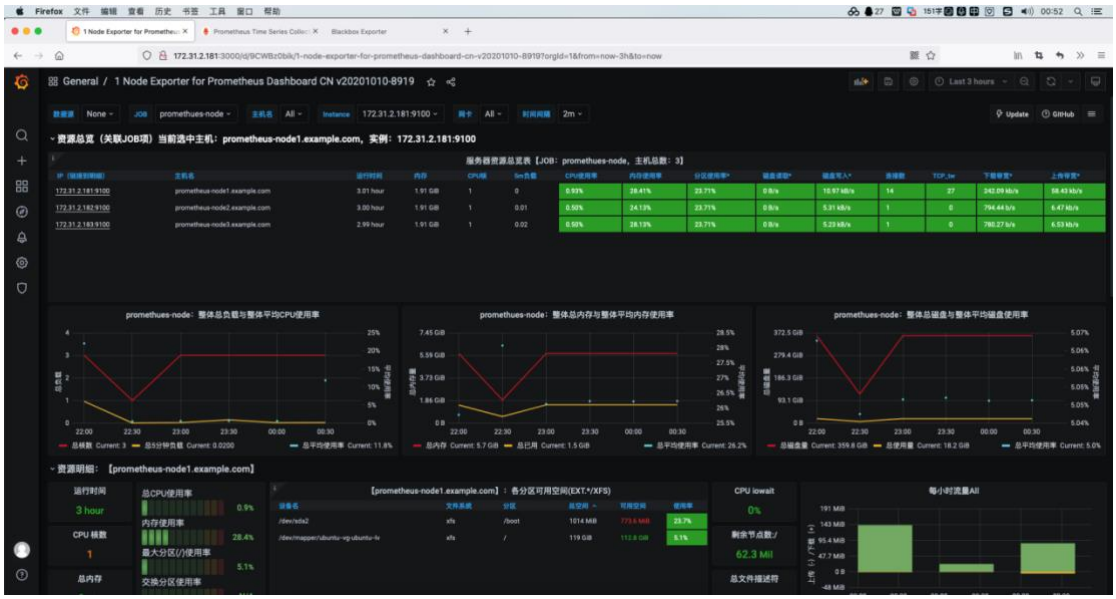
4.3.2：通过模板 ID 导入：



4.3.3：选择数据源并导入：

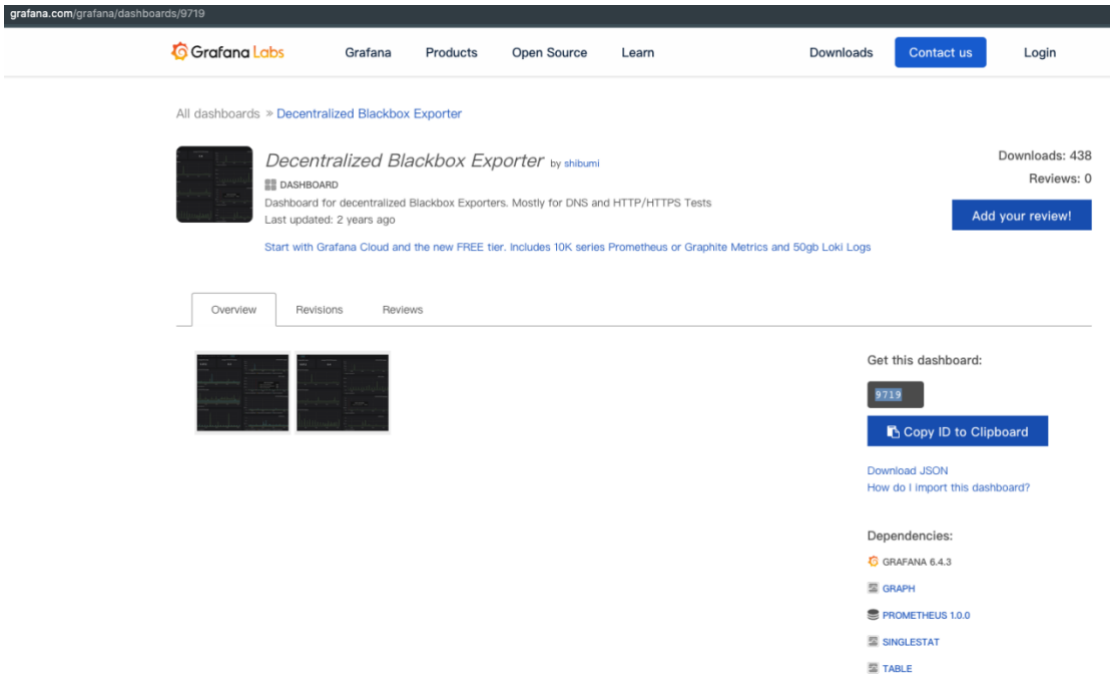


4.3.4：验证 dashboard：

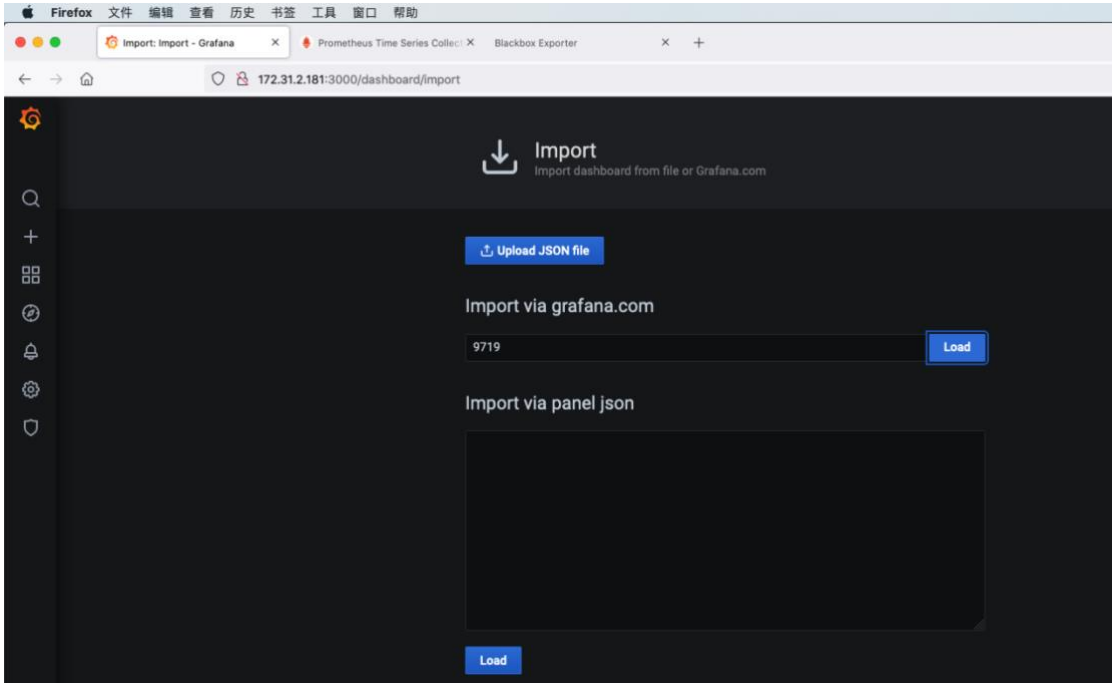


4.4：导入 blackbox exporter 模板：

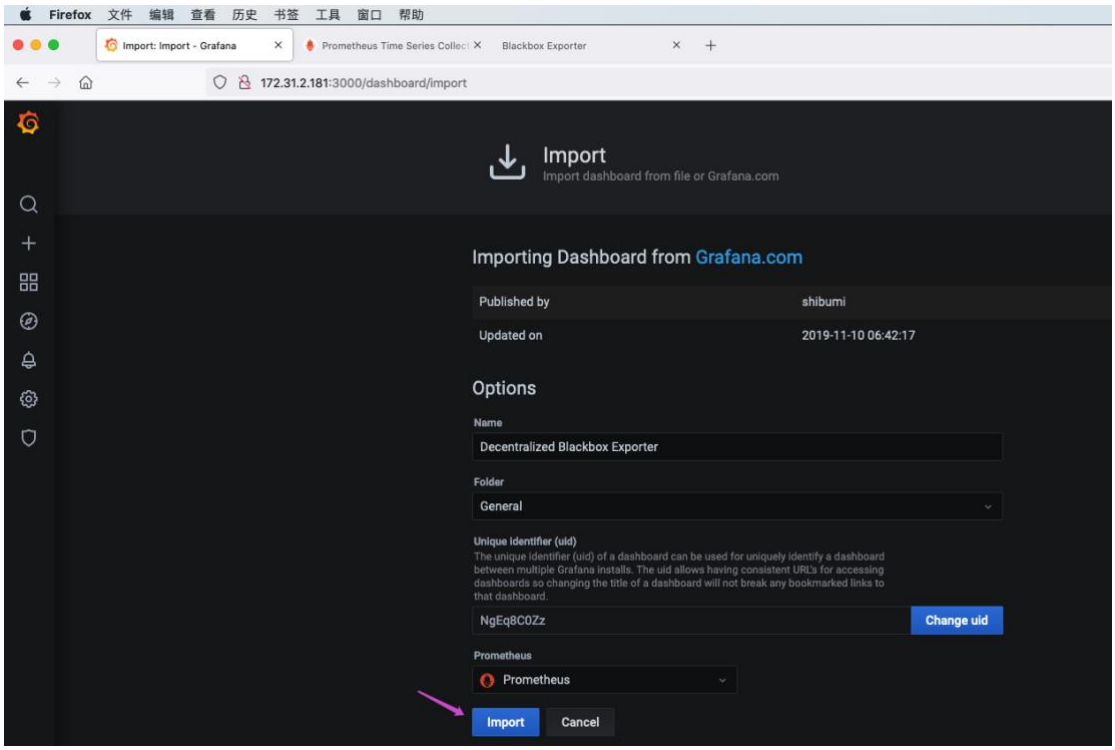
4.4.1：搜索模板-9719：



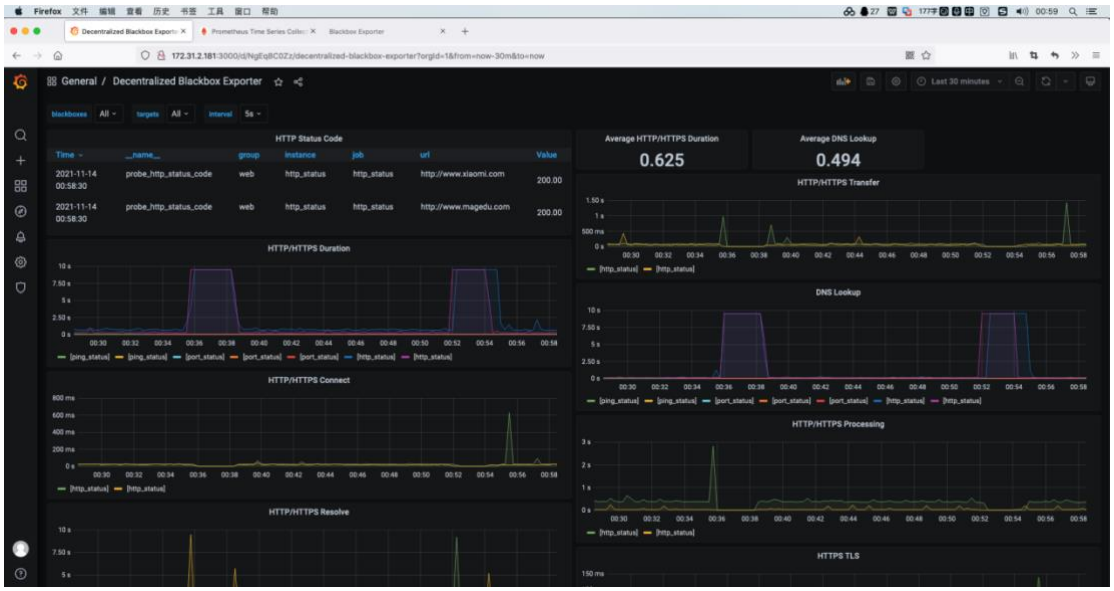
4.4.2：通过模板 ID 导入：



4.4.3：选择数据源并导入：



4.3.4：验证 dashboard：



五：监控 pod：

cadvisor 由谷歌开源，cadvisor 不仅可以搜集一台机器上所有运行的容器信息，还提供基础查询界面和 http 接口，方便其他组件如 Prometheus 进行数据抓取，cAdvisor 可以对节点机器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。

k8s 1.12 之前 cadvisor 集成在 node 节点的上 kubelet 服务中，从 1.12 版本开始分离为两个组件，因此需要在 node 节点单独部署 cadvisor。

https://github.com/google/cadvisor

5.1：cadvisor 镜像准备：

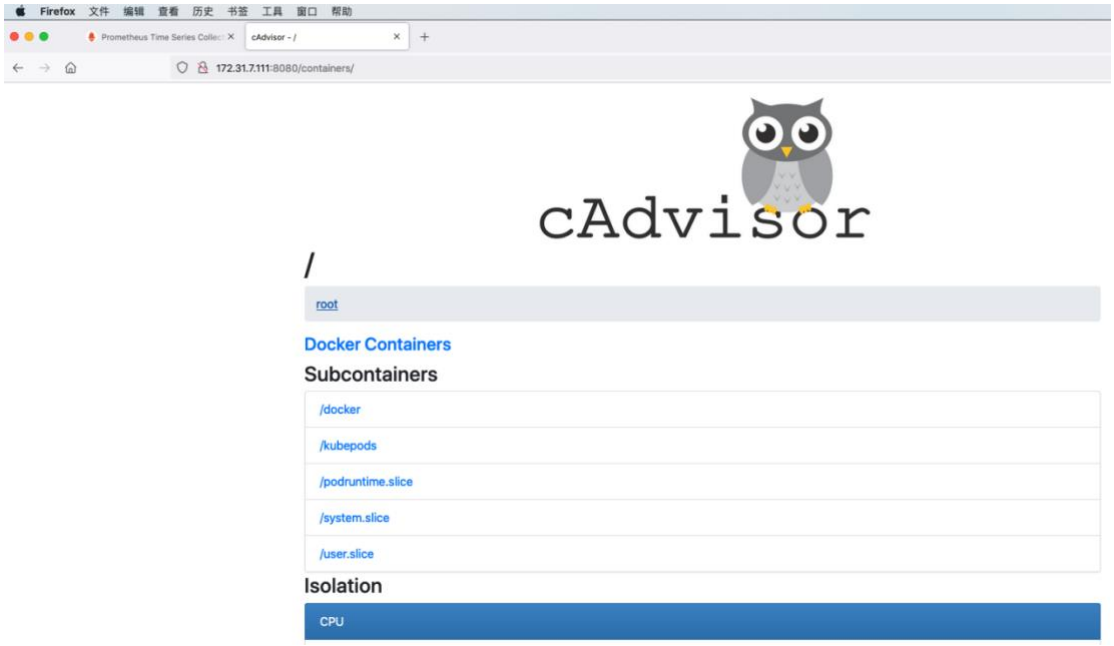
```
# docker load -i cadvisor-v0.38.7.tar.gz
# docker tag gcr.io/cadvisor/cadvisor:v0.38.7 harbor.magedu.net/baseimages/cadvisor:v0.38.7
# docker push harbor.magedu.net/baseimages/cadvisor:v0.38.7
```

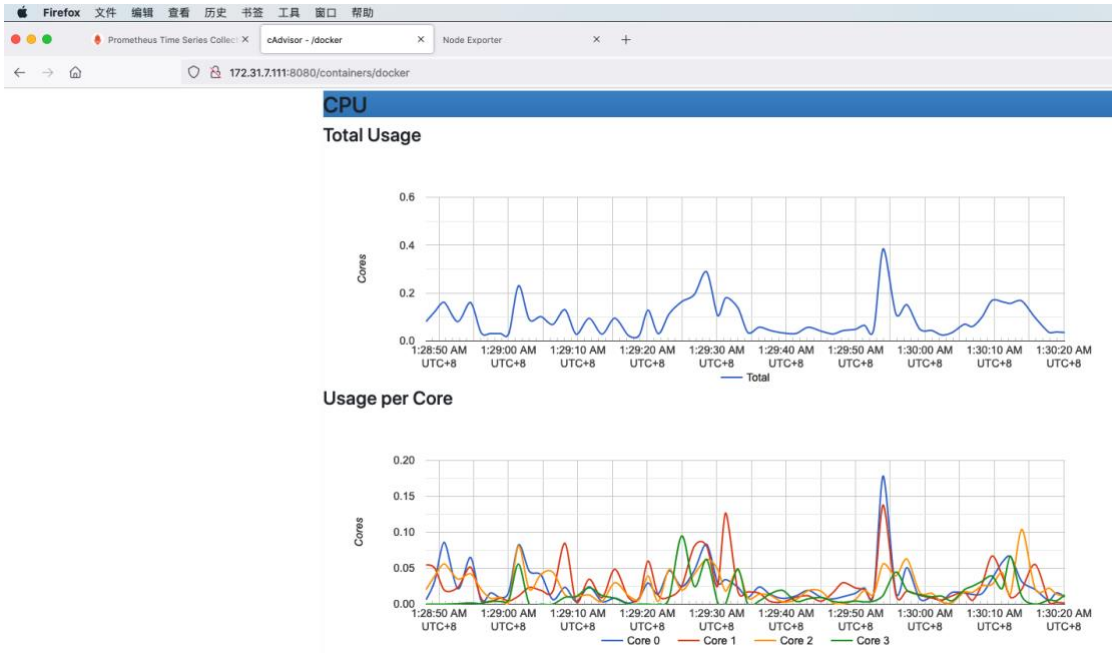
5.2：启动 cadvisor 容器：

```
docker run -it -d    \
   --restart=always \
   --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 \
   --privileged \
   --device=/dev/kmsg \
   harbor.magedu.net/baseimages/cadvisor:v0.38.7
```

5.3：验证 cadvisor web 界面：

访问 node 节点的 cadvisor 监听端口：http://172.31.7.110:8080/





5.4：prometheus 采集 cadvisor 数据：

```
# vim /usr/local/prometheus/prometheus.yml
  - job_name: 'prometheus-containers'
    static_configs:
      - targets: ["172.31.7.111:8080","172.31.7.112:8080","172.31.7.113:8080"]

# systemctl  restart  prometheus.service
```

5.5：prometheus 验证数据：

5.6：grafana 添加 pod 监控模板：

395 893 容器模板 ID

395 模板：

893 模板：

修改模板变量

修改模板 key 的名称

六： prometheus 报警设置：

6.1： prometheus 报警设置：

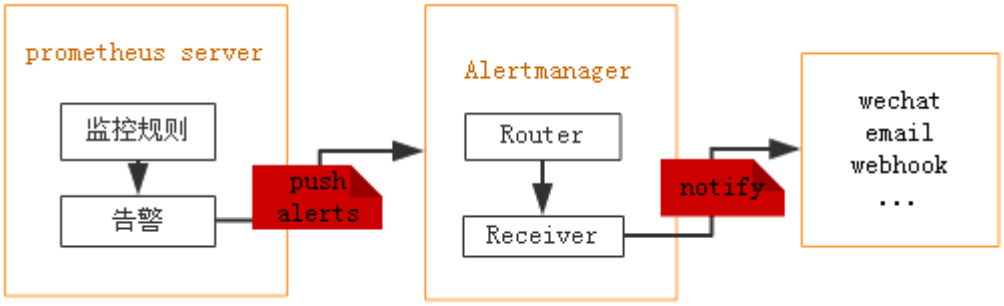
prometheus--->触发阈值--->超出持续时间--->alertmanager--->分组|抑制|静默--->媒体类型--->邮件|钉钉|微信等。

分组(group)：将类似性质的警报合并为单个通知，比如网络通知、主机通知、服务通知。

静默(silences)：是一种简单的特定时间静音的机制，例如：服务器要升级维护可以先设置这个时间段告警静默。

抑制(inhibition)：当警报发出后，停止重复发送由此警报引发的其他警报即合并一个故障引起的多个报警事件，可以消除冗余告警


02:05
15
519
开始时间:
结束时间:
报警描述: 同类型报警还有1条已省略
发消息...



6.2：下载并报警组件 alertmanager：

```
# cd /apps/
#                                                                                    wget
https://github.com/prometheus/alertmanager/releases/download/v0.23.0/alertmanager-0.23.0.
linux-amd64.tar.gz
# tar xvf alertmanager-0.23.0.linux-amd64.tar.gz

# ln -sv /apps/alertmanager-0.23.0.linux-amd64 /apps/alertmanager
'/apps/alertmanager' -> '/apps/alertmanager-0.23.0.linux-amd64'

# vim    /etc/systemd/system/alertmanager.service
[Unit]
Description=Prometheus Server
Documentation=https://prometheus.io/docs/introduction/overview/
After=network.target

[Service]
Restart=on-failure
WorkingDirectory=/apps/alertmanager
ExecStart=/apps/alertmanager/alertmanager

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

6.3：配置 alertmanager：

https://prometheus.io/docs/alerting/configuration/ #官方配置文档

```
global:
  smtp_from:        #发件人邮箱地址
  smtp_smarthost: #邮箱 smtp 地址。
  smtp_auth_username:   #发件人的登陆用户名，默认和发件人地址一致。
  smtp_auth_password:   #发件人的登陆密码，有时候是授权码。
  smtp_require_tls:      #是否需要 tls 协议。默认是 true。

  wechart_api_url:       #企业微信 API  地址。
```

```
  wechart_api_secret：  #企业微信 API secret
  wechat_api_corp_id:   #企业微信 corp id 信息。

  resolve_timeout:   #在指定时间内没有产生新的事件就发送恢复通知
```

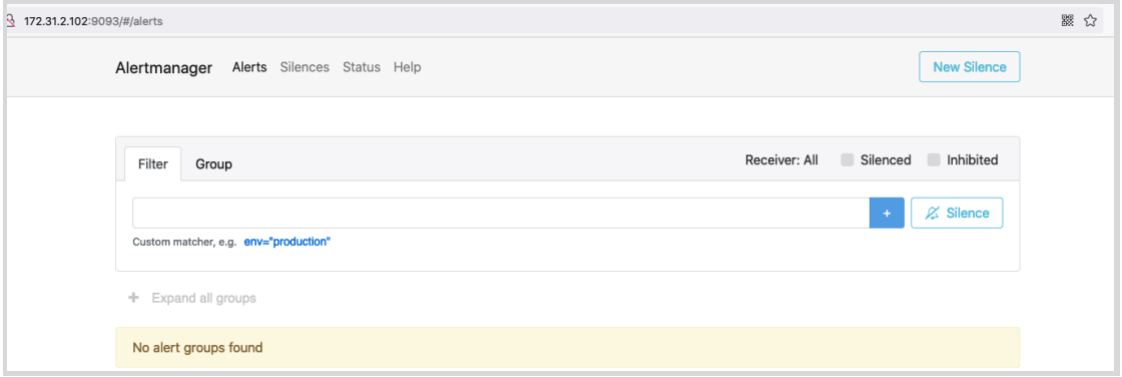
6.3.1：配置示例：

```
# pwd
/apps/alertmanager

# cat alertmanager.yml
global:
   resolve_timeout: 5m #在指定时间内没有产生新的事件就发送恢复通知
   smtp_smarthost: 'smtp.qq.com:465'
   smtp_from: '2973707860@qq.com'
   smtp_auth_username: '2973707860@qq.com'
   smtp_auth_password: 'fzhkygrgrphwddje'
   smtp_hello: '@qq.com'
   smtp_require_tls: false



route: #route 用来设置报警的分发策略
   group_by: ['alertname']   #采用哪个标签来作为分组依据
   group_wait: 10s #组告警等待时间。也就是告警产生后等待 10s，如果有同组告警一起发出
   group_interval: 2s #两组告警的间隔时间
   repeat_interval: 2m #重复告警的间隔时间，减少相同邮件的发送频率
   receiver: 'web.hook' #设置接收人
receivers:
- name: 'web.hook'
   #webhook_configs:
   #- url: 'http://127.0.0.1:5001/'
   email_configs:
      - to: '2973707860@qq.com'

inhibit_rules: #抑制的规则
   - source_match: #源匹配级别，当匹配成功发出通知，但是其他的通知将被抑制
         severity: 'critical' #
      target_match:
         severity: 'warning'
      equal: ['alertname', 'dev', 'instance']
```

6.3.2：启动并验证：

```
# systemctl    enable alertmanager && # systemctl    restart    alertmanager
# lsof    -i:9093
COMMAND       PID USER      FD      TYPE DEVICE SIZE/OFF NODE NAME
```

```
alertmana 9203 root       8u     IPv6     61784          0t0     TCP *:9093 (LISTEN)
```



6.3.3：prometheus 报警配置：

```
# cd /apps/prometheus
# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          - 172.31.2.102:9093
# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  - "/apps/prometheus/rule.yml"
```

6.3.4：创建报警规则文件：

```
# vim /apps/prometheus/rule.yml
groups:
  - name: alertmanager_pod.rules
    rules:
    - alert: Pod_all_cpu_usage
      expr: (sum by(name)(rate(container_cpu_usage_seconds_total{image!=""}[5m]))*100) > 1
      for: 2m
      labels:
        severity: critical
        service: pods
      annotations:
        description: 容器 {{ $labels.name }} CPU 资源利用率大于 10% , (current value is {{ $value }})
        summary: Dev CPU 负载告警

    - alert: Pod_all_memory_usage
      #expr: sort_desc(avg by(name)(irate(container_memory_usage_bytes{name!=""}[5m]))*100) > 10   #内存大于 10%
```

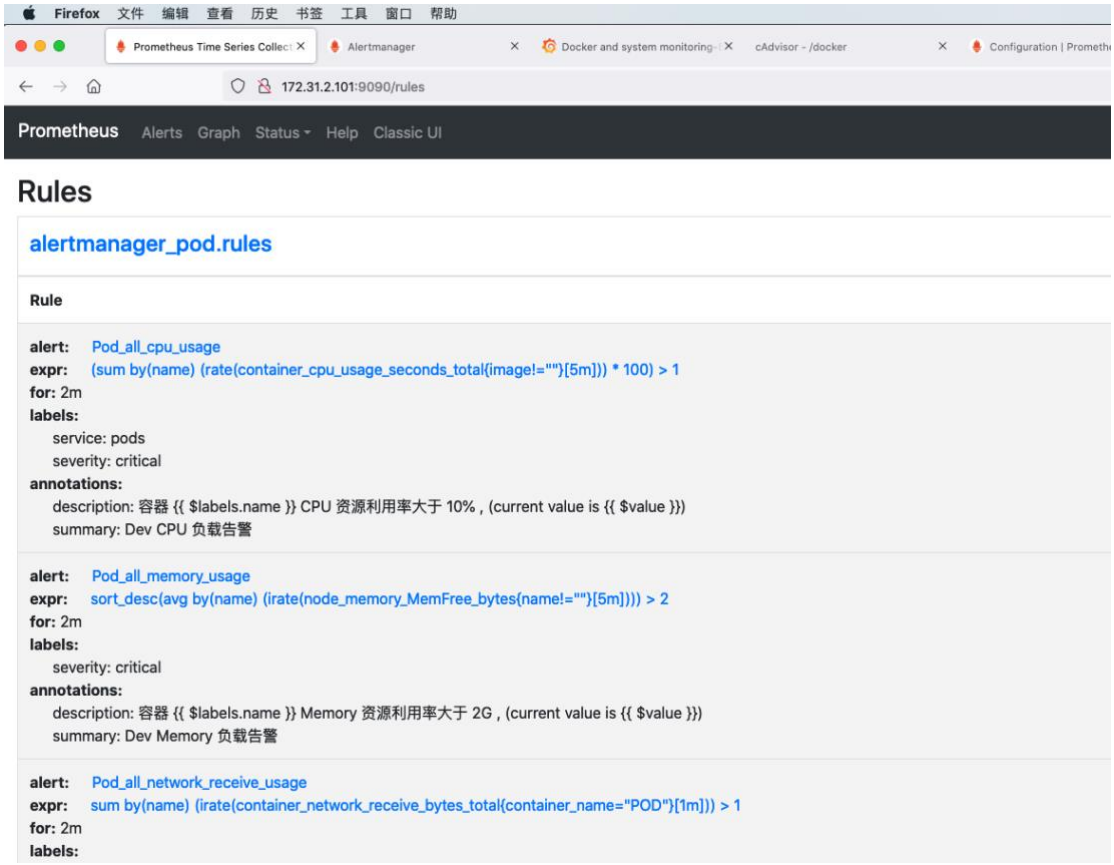
```
expr: sort_desc(avg by(name)(irate(node_memory_MemFree_bytes {name!=""}[5m]))) >
2     #内存大于 2G
          for: 2m
          labels:
             severity: critical
          annotations:
             description: 容器 {{ $labels.name }} Memory 资源利用率大于 2G , (current value is
{{ $value }})
             summary: Dev Memory 负载告警

       - alert: Pod_all_network_receive_usage
          expr:                                                sum                                                by
(name)(irate(container_network_receive_bytes_total{container_name="POD"}[1m])) > 1
          for: 2m
          labels:
             severity: critical
          annotations:
             description: 容器 {{ $labels.name }} network_receive 资源利用率大于 50M ,
(current value is {{ $value }})

       - alert: pod 内存可用大小
          expr: node_memory_MemFree_bytes > 1 #故意写错的
          for: 2m
          labels:
             severity: critical
          annotations:
             description: 容器可用内存小于 100k
```

6.3.5：规则验证：

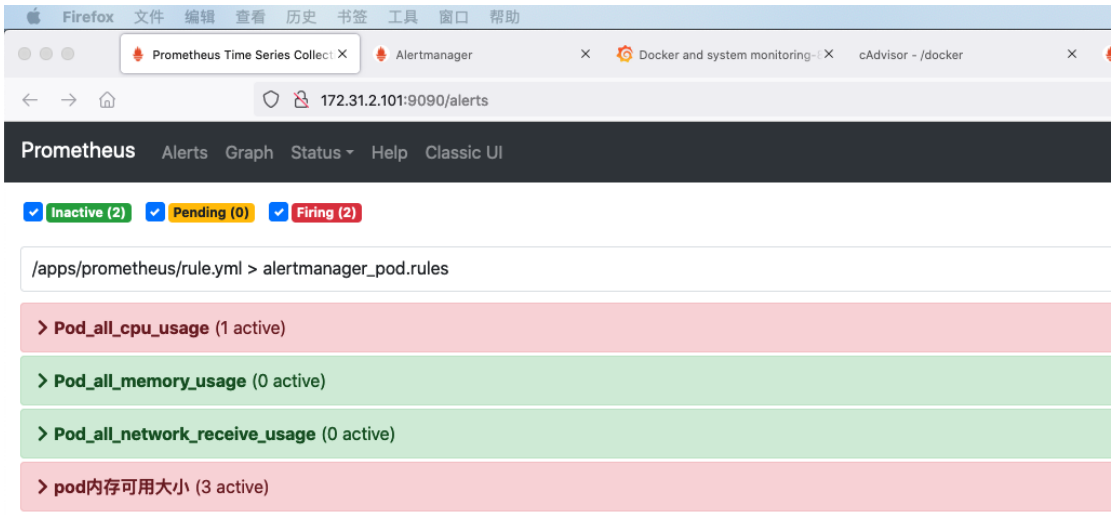
```
# /apps/prometheus/promtool    check rules    /apps/prometheus/rule.yml
Checking /apps/prometheus/rule.yml
   SUCCESS: 4 rules found
```

6.3.6：重启 prometheus 并验证规则：

```
systemctl    restart prometheus
```



6.3.7：已发送告警邮件：



6.3.8：验证告警邮件：

Labels
alertname = pod内存可用大小
instance = 172.31.2.181:9100
job = promethues-node
severity = critical
Annotations
description = 容器可用内存小于100k
Source

Labels
alertname = pod内存可用大小
instance = 172.31.2.182:9100
job = promethues-node
severity = critical
Annotations
description = 容器可用内存小于100k
Source

Labels
alertname = pod内存可用大小
instance = 172.31.2.183:9100
job = promethues-node
severity = critical
Annotations
description = 容器可用内存小于100k
Source