韩先超老师微信公众号，定期发布技术类文章



韩先超老师51cto博客地址：<https://blog.51cto.com/u_12974849>

# 13、配置alertmanager-发送报警到qq邮箱

报警：指prometheus将监测到的异常事件发送给alertmanager   
通知：alertmanager将报警信息发送到邮件、微信、钉钉等

#创建alertmanager配置文件

在k8s的控制节点创建alertmanager-cm.yaml文件，alertmanager-cm.yaml文件在课件，可上传到k8s的xianchaomaster1节点

通过kubectl apply 更新文件

kubectl apply -f alertmanager-cm.yaml

alertmanager-cm.yaml文件内容如下：

kind: ConfigMap

apiVersion: v1

metadata:

name: alertmanager

namespace: monitor-sa

data:

alertmanager.yml: |-

global:

resolve\_timeout: 1m

smtp\_smarthost: 'smtp.163.com:25'

smtp\_from: '15011572657@163.com'

smtp\_auth\_username: '15011572657'

smtp\_auth\_password: ' BGWHYUOSOOHWEUJM'

smtp\_require\_tls: false

route: #用于配置告警分发策略

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: 10s # 上下两组发送告警的间隔时间

repeat\_interval: 10m # 重复发送告警的时间，减少相同邮件的发送频率，默认是1h

receiver: default-receiver #定义谁来收告警

receivers:

- name: 'default-receiver'

email\_configs:

- to: '1980570647@qq.com'

send\_resolved: true

alertmanager配置文件解释说明：

smtp\_smarthost: 'smtp.163.com:25'

#163邮箱的SMTP服务器地址+端口

smtp\_from: '15011572657@163.com'

#这是指定从哪个邮箱发送报警

smtp\_auth\_username: '15011572657'

#这是发送邮箱的认证用户，不是邮箱名

smtp\_auth\_password: ' BGWHYUOSOOHWEUJM'

#这是发送邮箱的授权码而不是登录密码，你们需要用自己的，不要用我的，用我的你会发不出来报警

email\_configs:

- to: '1980570647@qq.com'

#to后面指定发送到哪个邮箱，我发送到我的qq邮箱，大家需要写自己的邮箱地址，不应该跟smtp\_from的邮箱名字重复

route: #用于设置告警的分发策略

group\_by: [alertname]

#alertmanager会根据group\_by配置将Alert分组

group\_wait: 10s

# 分组等待时间。也就是告警产生后等待10s，如果有同组告警一起发出

group\_interval: 10s # 上下两组发送告警的间隔时间

repeat\_interval: 10m # 重复发送告警的时间，减少相同邮件的发送频率，默认是1h

receiver: default-receiver #定义谁来收告警

Prometheus 一条告警的触发流程、等待时间

报警处理流程如下：  
1. Prometheus Server监控目标主机上暴露的http接口（这里假设接口A），通过Promethes配置的'scrape\_interval'定义的时间间隔，定期采集目标主机上监控数据。  
2. 当接口A不可用的时候，Server端会持续的尝试从接口中取数据，直到"scrape\_timeout"时间后停止尝试。这时候把接口的状态变为“DOWN”。  
3. Prometheus同时根据配置的"evaluation\_interval"的时间间隔，定期（默认1min）的对Alert Rule进行评估；当到达评估周期的时候，发现接口A为DOWN，即UP=0为真，激活Alert，进入“PENDING”状态，并记录当前active的时间；  
4. 当下一个alert rule的评估周期到来的时候，发现UP=0继续为真，然后判断警报Active的时间是否已经超出rule里的‘for’ 持续时间，如果未超出，则进入下一个评估周期；如果时间超出，则alert的状态变为“FIRING”；同时调用Alertmanager接口，发送相关报警数据。  
5. AlertManager收到报警数据后，会将警报信息进行分组，然后根据alertmanager配置的“group\_wait”时间先进行等待。等wait时间过后再发送报警信息。  
6. 属于同一个Alert Group的警报，在等待的过程中可能进入新的alert，如果之前的报警已经成功发出，那么间隔“group\_interval”的时间间隔后再重新发送报警信息。比如配置的是邮件报警，那么同属一个group的报警信息会汇总在一个邮件里进行发送。  
7. 如果Alert Group里的警报一直没发生变化并且已经成功发送，等待‘repeat\_interval’时间间隔之后再重复发送相同的报警邮件；如果之前的警报没有成功发送，则相当于触发第6条条件，则需要等待group\_interval时间间隔后重复发送。

同时最后至于警报信息具体发给谁，满足什么样的条件下指定警报接收人，设置不同报警发送频率，这里有alertmanager的route路由规则进行配置。

#创建prometheus和告警规则配置文件

在k8s的控制节点生成一个prometheus-alertmanager-cfg.yaml文件，prometheus-alertmanager-cfg.yaml文件在课件，上传到k8s的xianchaomaster1节点

通过kubectl apply 更新资源文件

[root@xianchaomaster1]# kubectl delete -f prometheus-cfg.yaml

[root@xianchaomaster1]# kubectl apply -f prometheus-alertmanager-cfg.yaml

注意：prometheus-alertmanager-cfg.yaml文件大家做实验需要修改，修改内容如下：

- job\_name: 'kubernetes-schedule'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.40.180:10251'] #scheduler组件所在节点的ip

- job\_name: 'kubernetes-controller-manager'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.40.180:10252']

- job\_name: 'kubernetes-kube-proxy'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.40.180:10249','192.168.40.181:10249']

#kube-proxy组件所在节点的ip

- job\_name: 'kubernetes-etcd'

scheme: https

tls\_config:

ca\_file: /var/run/secrets/kubernetes.io/k8s-certs/etcd/ca.crt

cert\_file: /var/run/secrets/kubernetes.io/k8s-certs/etcd/server.crt

key\_file: /var/run/secrets/kubernetes.io/k8s-certs/etcd/server.key

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.40.180:2379']

#etcd组件所在节点的ip

#安装prometheus和alertmanager

需要把alertmanager.tar.gz镜像包上传的k8s的各个工作节点，手动解压：

[root@xianchaonode1 ~]# docker load -i alertmanager.tar.gz

在k8s的控制节点生成一个prometheus-alertmanager-deploy.yaml文件，prometheus-alertmanager-deploy.yaml文件在课件，可上传到k8s的控制节点xianchaomaster1上：

注意：配置文件指定了nodeName: xianchaonode1，这个位置要写你自己环境的k8s的node节点名字

生成一个etcd-certs，这个在部署prometheus需要

[root@xianchaomaster1]# kubectl -n monitor-sa create secret generic etcd-certs --from-file=/etc/kubernetes/pki/etcd/server.key --from-file=/etc/kubernetes/pki/etcd/server.crt --from-file=/etc/kubernetes/pki/etcd/ca.crt

通过kubectl apply更新资源清单yaml文件

[root@xianchaomaster1# kubectl delete -f prometheus-deploy.yaml

[root@xianchaomaster1]# kubectl apply -f prometheus-alertmanager-deploy.yaml

#查看prometheus是否部署成功

[root@xianchaomaster1]# kubectl get pods -n monitor-sa | grep prometheus

#显示如下，说明创建成功：

prometheus-server-6bfc4755f6-cn487 2/2 Running 0 16s

#部署alertmanager的service，方便在浏览器访问

在k8s的控制节点生成一个alertmanager-svc.yaml文件，alertmanager-svc.yaml文件在课件里，可上传到k8s的控制节点xianchaomaster1：

通过kubectl apply更新yaml文件

[root@xianchaomaster1]# kubectl apply -f alertmanager-svc.yaml

alertmanager-svc.yaml文件内容如下:

---

apiVersion: v1

kind: Service

metadata:

labels:

name: prometheus

kubernetes.io/cluster-service: 'true'

name: alertmanager

namespace: monitor-sa

spec:

ports:

- name: alertmanager

nodePort: 30066

port: 9093

protocol: TCP

targetPort: 9093

selector:

app: prometheus

sessionAffinity: None

type: NodePort

#查看service在物理机映射的端口

kubectl get svc -n monitor-sa

显示如下：

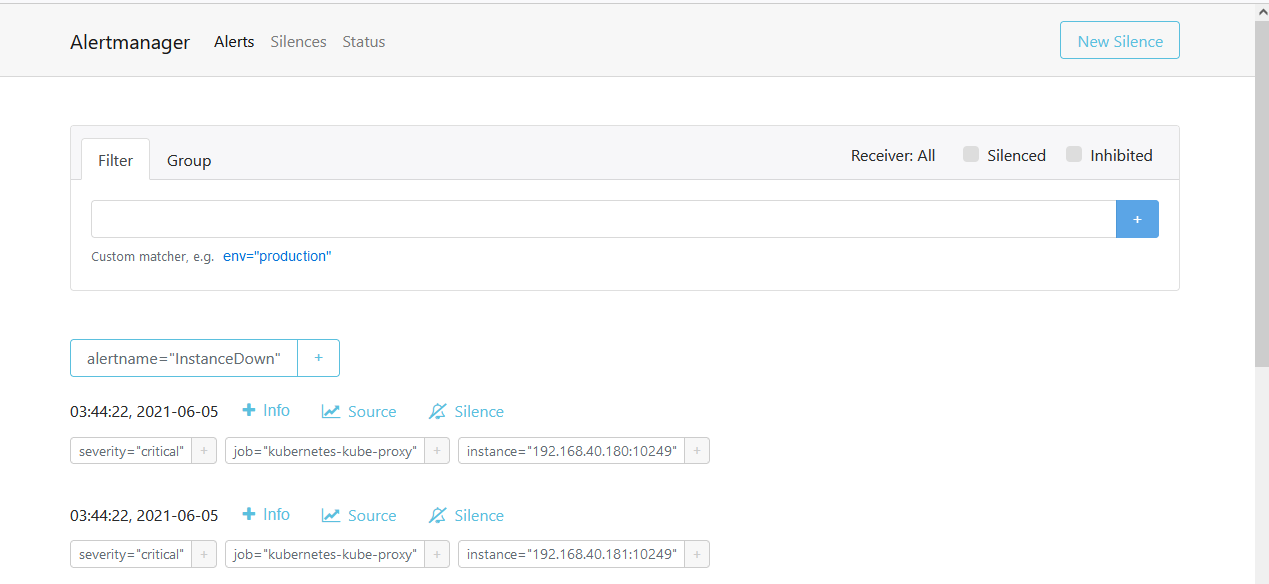
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

alertmanager NodePort 10.101.253.221 <none> 9093:30066/TCP 20s

prometheus NodePort 10.103.243.87 <none> 9090:32732/TCP 96m

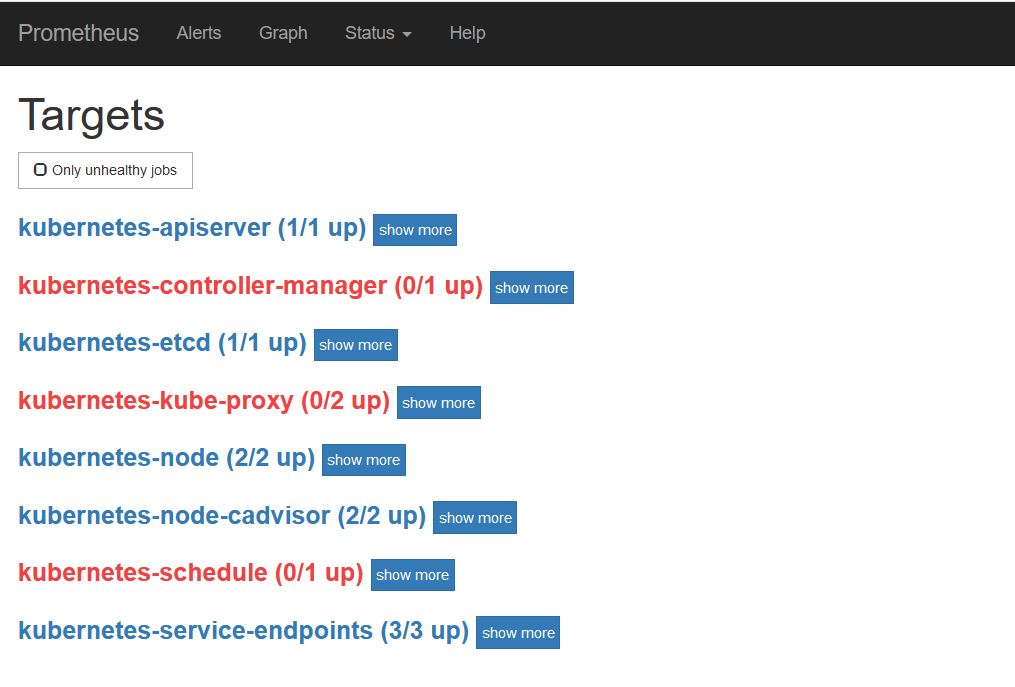
注意：上面可以看到prometheus的service在物理机映射的端口是32732，alertmanager的service在物理机映射的端口是30066

<http://192.168.40.180:30066/#/alerts>



访问prometheus的web界面

点击status->targets，可看到如下



从上面可以发现kubernetes-controller-manager和kubernetes-schedule都显示连接不上对应的端口

可按如下方法处理：

vim /etc/kubernetes/manifests/kube-scheduler.yaml  
修改如下内容：

把--bind-address=127.0.0.1变成--bind-address=192.168.40.180

把httpGet:字段下的hosts由127.0.0.1变成192.168.40.180

把—port=0删除

#注意：192.168.40.180是k8s的控制节点xianchaomaster1的ip

vim /etc/kubernetes/manifests/kube-controller-manager.yaml

把--bind-address=127.0.0.1变成--bind-address=192.168.40.130

把httpGet:字段下的hosts由127.0.0.1变成192.168.40.180

把—port=0删除

修改之后在k8s各个节点执行

systemctl restart kubelet

kubectl get cs

显示如下:

NAME STATUS MESSAGE ERROR

controller-manager Healthy ok

scheduler Healthy ok

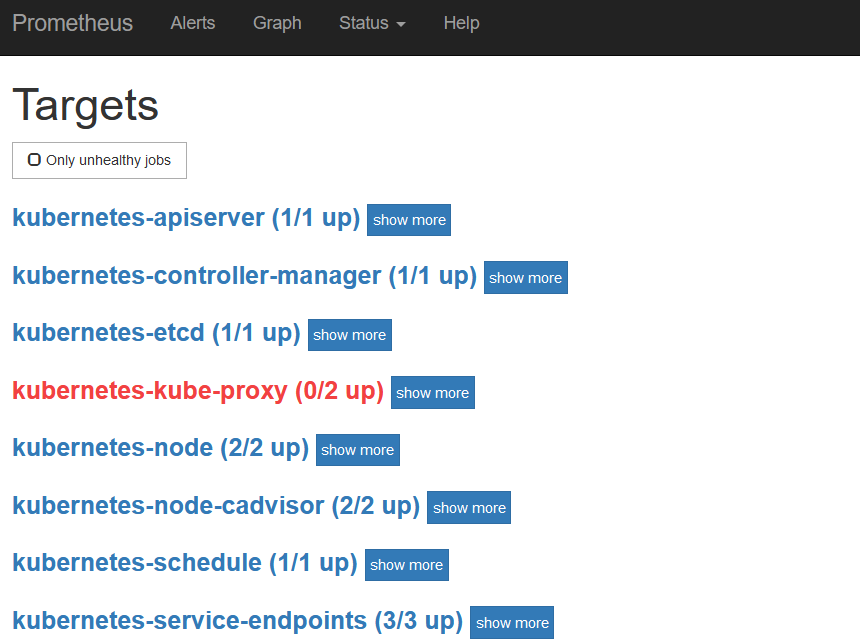
etcd-0 Healthy {"health":"true"}

ss -antulp | grep :10251

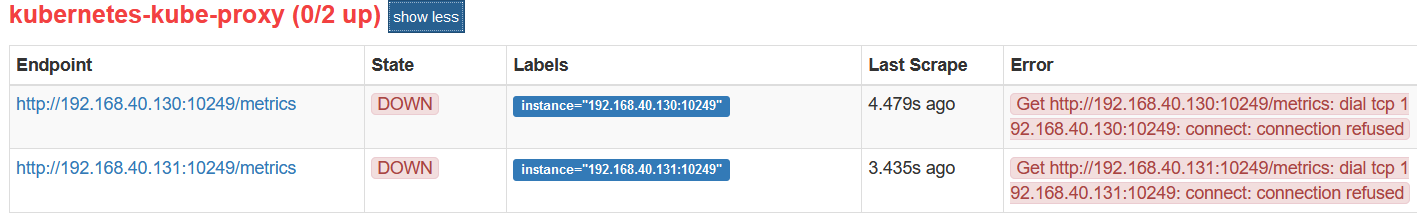
ss -antulp | grep :10252

可以看到相应的端口已经被物理机监听了

点击status->targets，可看到如下



kubernetes-kube-proxy显示如下：



是因为kube-proxy默认端口10249是监听在127.0.0.1上的，需要改成监听到物理节点上，按如下方法修改，线上建议在安装k8s的时候就做修改，这样风险小一些：

kubectl edit configmap kube-proxy -n kube-system

把metricsBindAddress这段修改成metricsBindAddress: 0.0.0.0:10249

然后重新启动kube-proxy这个pod

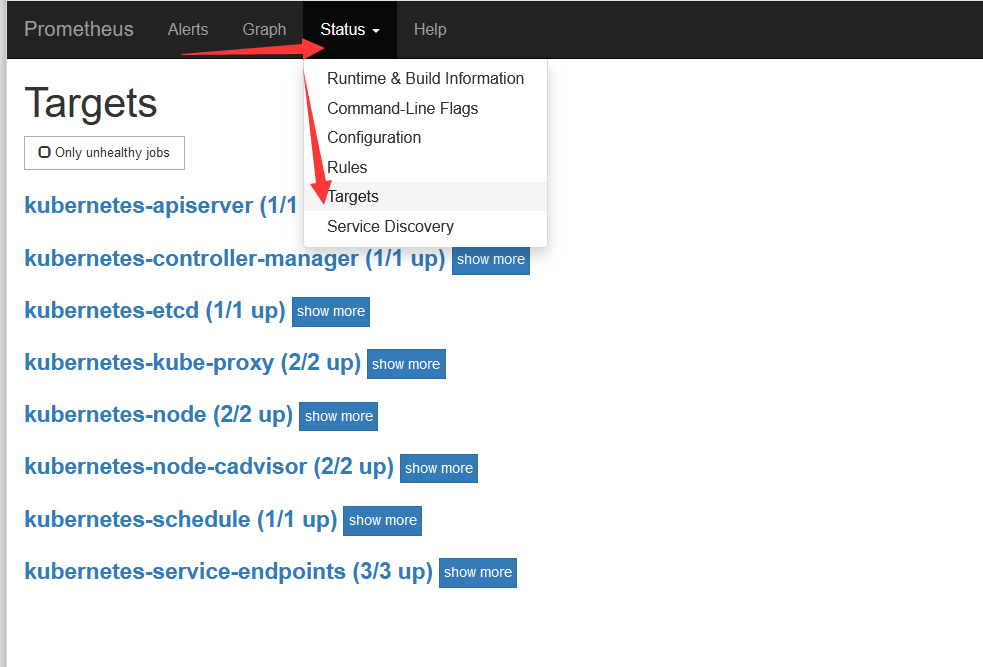
[root@xianchaomaster1]# kubectl get pods -n kube-system | grep kube-proxy |awk '{print $1}' | xargs kubectl delete pods -n kube-system

[root@xianchaomaster1]# ss -antulp |grep :10249

可显示如下

tcp LISTEN 0 128 [::]:10249 [::]:\*

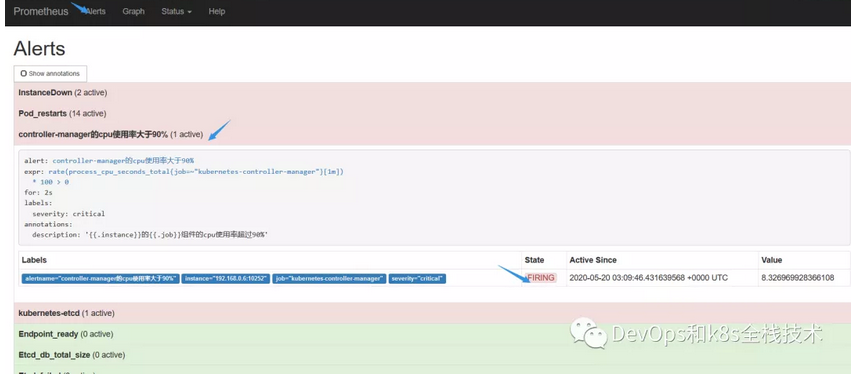
点击status->targets，可看到如下



点击Alerts，可看到如下



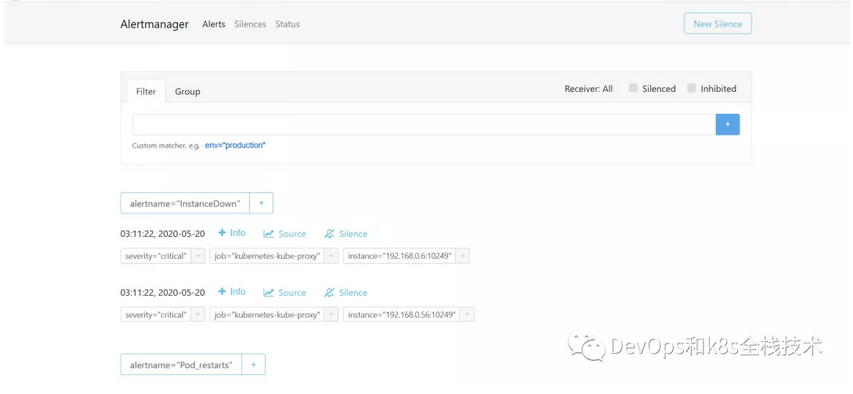
把controller-manager的cpu使用率大于90%展开，可看到如下



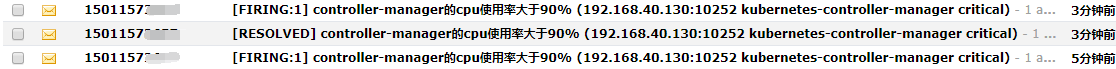
FIRING表示prometheus已经将告警发给alertmanager，在Alertmanager 中可以看到有一个 alert。

登录到alertmanager web界面

浏览器输入192.168.40.180:30066，显示如下



这样我在我的qq邮箱，1980570647@qq.com就可以收到报警了，如下



扩展：暴力更新配置文件

修改prometheus任何一个配置文件之后，可通过kubectl apply使配置生效，执行顺序如下：

kubectl delete -f alertmanager-cm.yaml

kubectl apply -f alertmanager-cm.yaml

kubectl delete -f prometheus-alertmanager-cfg.yaml

kubectl apply  -f prometheus-alertmanager-cfg.yaml

kubectl delete -f prometheus-alertmanager-deploy.yaml  
kubectl apply  -f prometheus-alertmanager-deploy.yaml

# 14、配置alertmanager-发送报警到钉钉

打开电脑版钉钉创建机器人  
1.创建钉钉机器人

打开电脑版钉钉，创建一个群，创建自定义机器人，按如下步骤创建

<https://ding-doc.dingtalk.com/doc#/serverapi2/qf2nxq>

https://developers.dingtalk.com/document/app/custom-robot-access

我创建的机器人如下：

群设置-->智能群助手-->添加机器人-->自定义-->添加

机器人名称：test

接收群组：钉钉报警测试

安全设置：

自定义关键词：cluster1

上面配置好之后点击完成即可，这样就会创建一个test的报警机器人，创建机器人成功之后怎么查看webhook，按如下：

点击智能群助手，可以看到刚才创建的test这个机器人，点击test，就会进入到test机器人的设置界面

出现如下内容：

机器人名称：test

接受群组：钉钉报警测试

消息推送：开启

webhook：

<https://oapi.dingtalk.com/robot/send?access_token=8a53475677339a11cec453c608543c3d85ea73b330ea70c4b2de96a0839cbb90>

安全设置：

自定义关键词：cluster1

2.安装钉钉的webhook插件，在k8s的控制节点xianchaomaster1操作  
tar zxvf prometheus-webhook-dingtalk-0.3.0.linux-amd64.tar.gz

prometheus-webhook-dingtalk-0.3.0.linux-amd64.tar.gz压缩包所在的百度网盘地址如下：

链接：https://pan.baidu.com/s/1\_HtVZsItq2KsYvOlkIP9DQ

提取码：d59o

cd prometheus-webhook-dingtalk-0.3.0.linux-amd64

启动钉钉报警插件

nohup ./prometheus-webhook-dingtalk --web.listen-address="0.0.0.0:8060" --ding.profile="cluster1=<https://oapi.dingtalk.com/robot/send?access_token=8a53475677339a11cec453c608543c3d85ea73b330ea70c4b2de96a0839cbb90>" &

对原来的alertmanager-cm.yaml文件做备份  
cp alertmanager-cm.yaml alertmanager-cm.yaml.bak

重新生成一个新的alertmanager-cm.yaml文件

cat >alertmanager-cm.yaml <<EOF

kind: ConfigMap

apiVersion: v1

metadata:

name: alertmanager

namespace: monitor-sa

data:

alertmanager.yml: |-

global:

resolve\_timeout: 1m

smtp\_smarthost: 'smtp.163.com:25'

smtp\_from: '15011572657@163.com'

smtp\_auth\_username: '1501157\*\*\*\*'

smtp\_auth\_password: ‘BGWHYUOSOOHWEUJM'

smtp\_require\_tls: false

route:

group\_by: [alertname]

group\_wait: 10s

group\_interval: 10s

repeat\_interval: 10m

receiver: cluster1

receivers:

- name: cluster1

webhook\_configs:

- url: 'http://192.168.40.180:8060/dingtalk/cluster1/send'

send\_resolved: true

EOF

修改prometheus任何一个配置文件之后，可通过kubectl apply使配置生效，执行顺序如下：

kubectl delete -f alertmanager-cm.yaml

kubectl apply  -f alertmanager-cm.yaml

kubectl delete -f prometheus-alertmanager-cfg.yaml

kubectl apply  -f prometheus-alertmanager-cfg.yaml

kubectl delete -f prometheus-alertmanager-deploy.yaml  
kubectl apply  -f prometheus-alertmanager-deploy.yaml

# 15、配置alertmanager-发送报警到微信

1注册企业微信

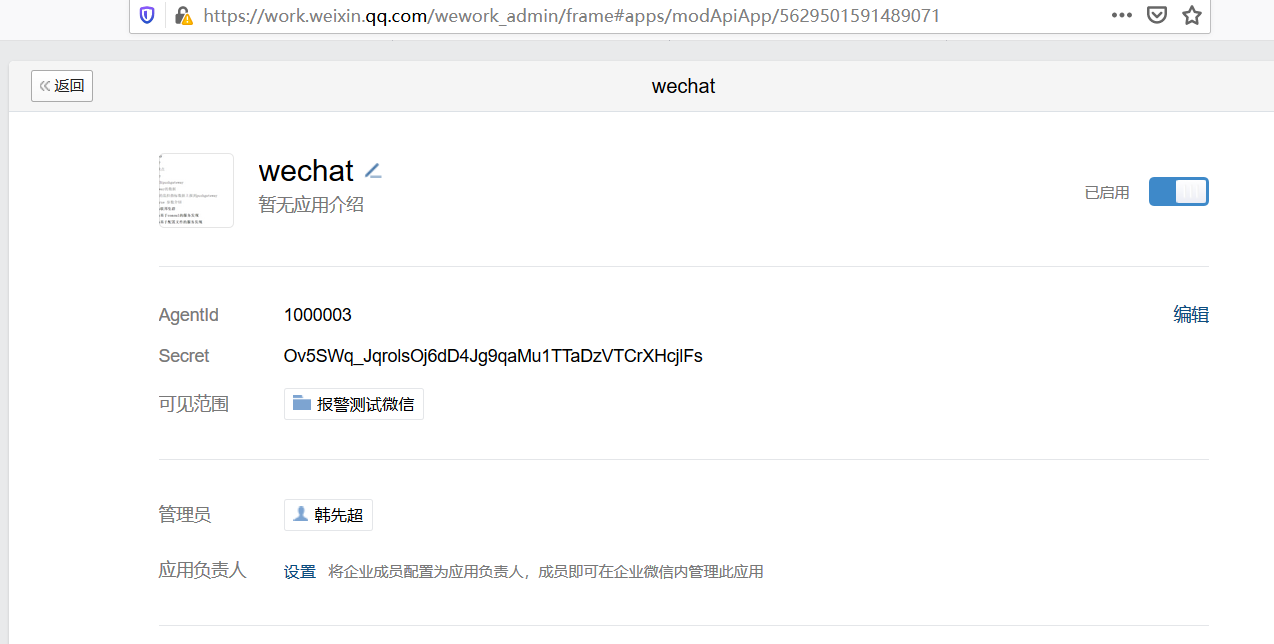
登陆网址：

<https://work.weixin.qq.com/>

找到应用管理，创建应用

应用名字wechat

创建成功之后显示如下：



AgentId：1000003

Secret：Ov5SWq\_JqrolsOj6dD4Jg9qaMu1TTaDzVTCrXHcjlFs

2.修改alertmanager-cm.yaml

global:

smtp\_smarthost: 'smtp.163.com:25'

smtp\_from: '15011572657@163.com'

smtp\_auth\_username: '15011572657'

smtp\_auth\_password: 'BGWHYUOSOOHWEUJM'

smtp\_require\_tls: false

route:

group\_by: [alertname]

group\_wait: 10s

group\_interval: 10s

repeat\_interval: 3m

receiver: "prometheus"

receivers:

- name: 'prometheus'

wechat\_configs:

- corp\_id: wwa82df90a693abb15

to\_user: '@all'

agent\_id: 1000003

api\_secret: Ov5SWq\_JqrolsOj6dD4Jg9qaMu1TTaDzVTCrXHcjlFs

参数说明：

secret: 企业微信("企业应用"-->"自定应用"[Prometheus]--> "Secret")

wechat是本人自创建应用名称  
corp\_id: 企业信息("我的企业"--->"CorpID"[在底部])  
agent\_id: 企业微信("企业应用"-->"自定应用"[Prometheus]--> "AgentId")

wechat是自创建应用名称 #在这创建的应用名字是wechat，那么在配置route时，receiver也应该是Prometheus

to\_user: '@all' :发送报警到所有人

3.配置自定义告警模板

cat template\_wechat.tmpl

{{ define "wechat.default.message" }}

{{ range .Alerts }}

========start==========

告警程序：node\_exporter

告警名称：{{ .Labels.alertname }}

故障主机: {{ .Labels.instance }}

告警主题: {{ .Annotations.summary }}

告警信息: {{ .Annotations.description }}

========end==========

{{ end }}

{{ end }}

# 16、Prometheus PromQL语法

PromQL（Prometheus Query Language）是 Prometheus 自己开发的表达式语言，语言表现力很丰富，内置函数也很多。使用它可以对时序数据进行筛选和聚合。

## 16.1 数据类型

PromQL 表达式计算出来的值有以下几种类型：

瞬时向量 (Instant vector): 一组时序，每个时序只有一个采样值

区间向量 (Range vector): 一组时序，每个时序包含一段时间内的多个采样值

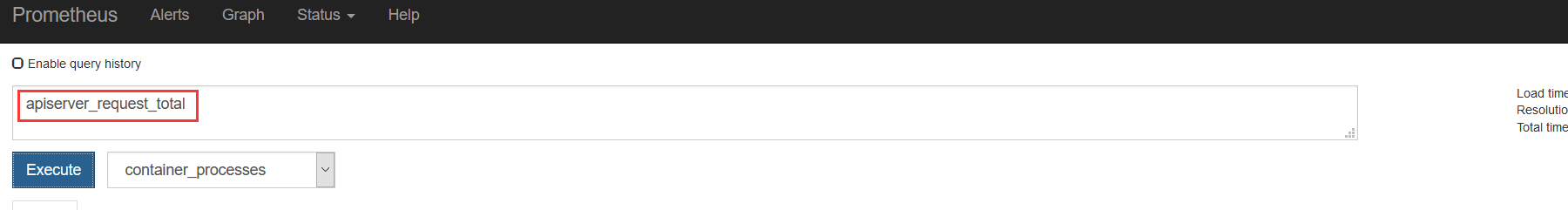
标量数据 (Scalar): 一个浮点数

字符串 (String): 一个字符串，暂时未用

### 16.1.1 瞬时向量选择器

瞬时向量选择器用来选择一组时序在某个采样点的采样值。  
最简单的情况就是指定一个度量指标，选择出所有属于该度量指标的时序的当前采样值。比如下面的表达式：

apiserver\_request\_total



可以通过在后面添加用大括号包围起来的一组标签键值对来对时序进行过滤。比如下面的表达式筛选出了 job 为 kubernetes-apiservers，并且 resource为 pod的时序：

apiserver\_request\_total{job="kubernetes-apiserver",resource="pods"}

匹配标签值时可以是等于，也可以使用正则表达式。总共有下面几种匹配操作符：

=：完全相等

!=： 不相等

=~： 正则表达式匹配

!~： 正则表达式不匹配

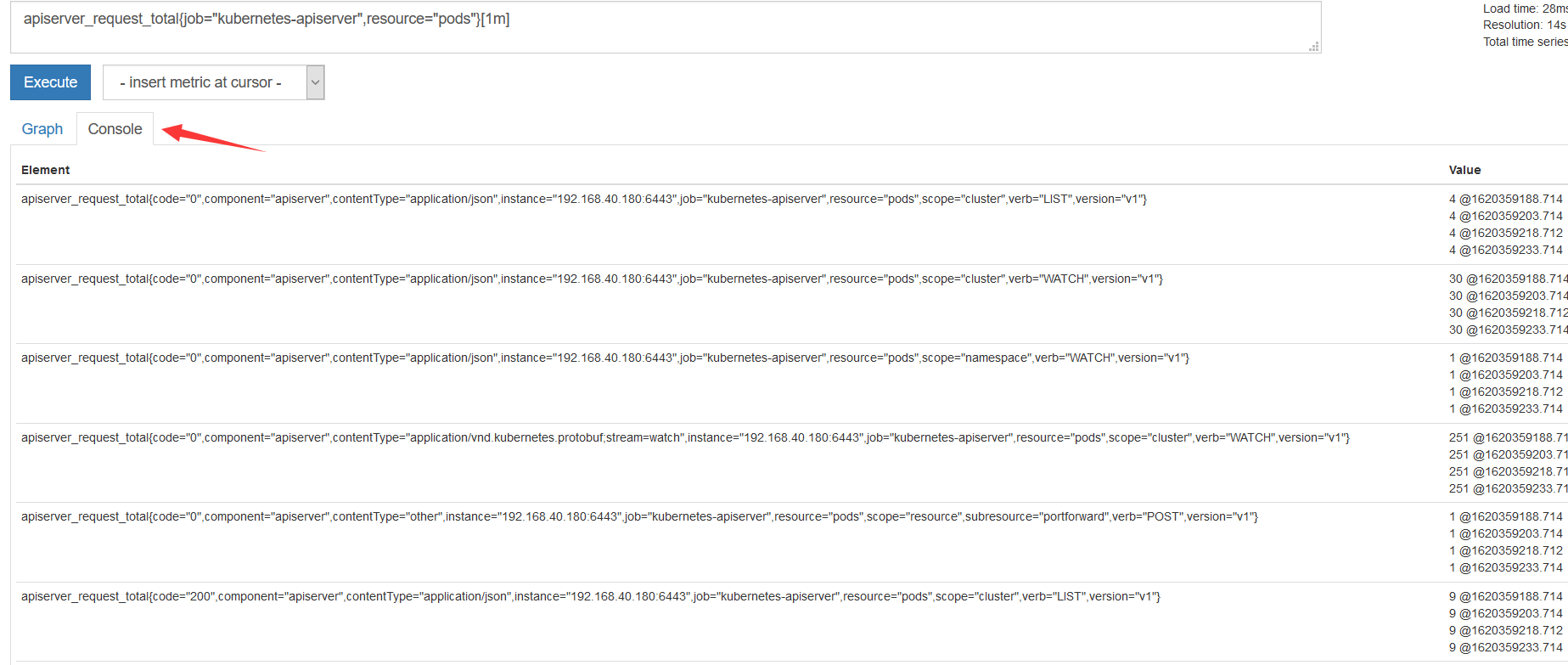
下面的表达式筛选出了container是kube-scheduler或kube-proxy或kube-apiserver的时序数据

container\_processes{container=~"kube-scheduler|kube-proxy|kube-apiserver"}

### 16.1.2 区间向量选择器

区间向量选择器类似于瞬时向量选择器，不同的是它选择的是过去一段时间的采样值。可以通过在瞬时向量选择器后面添加包含在 [] 里的时长来得到区间向量选择器。比如下面的表达式选出了所有度量指标为apiserver\_request\_total且resource是pod的时序在过去1 分钟的采样值。

apiserver\_request\_total{job="kubernetes-apiserver",resource="pods"}[1m]



这个不支持Graph，需要选择Console，才会看到采集的数据

说明：时长的单位可以是下面几种之一：

s：seconds

m：minutes

h：hours

d：days

w：weeks

y：years

### 16.1.3 偏移向量选择器

前面介绍的选择器默认都是以当前时间为基准时间，偏移修饰器用来调整基准时间，使其往前偏移一段时间。偏移修饰器紧跟在选择器后面，使用 offset 来指定要偏移的量。比如下面的表达式选择度量名称为apiserver\_request\_total的所有时序在 5 分钟前的采样值。

apiserver\_request\_total{job="kubernetes-apiserver",resource="pods"} offset 5m

下面的表达式选择apiserver\_request\_total 度量指标在 1 周前的这个时间点过去 5 分钟的采样值。

apiserver\_request\_total{job="kubernetes-apiserver",resource="pods"} [5m] offset 1w

### 16.1.4 聚合操作符

PromQL 的聚合操作符用来将向量里的元素聚合得更少。总共有下面这些聚合操作符：

sum：求和

min：最小值

max：最大值

avg：平均值

stddev：标准差

stdvar：方差

count：元素个数

count\_values：等于某值的元素个数

bottomk：最小的 k 个元素

topk：最大的 k 个元素

quantile：分位数

如：

计算xianchaomaster1节点所有容器总计内存

sum(container\_memory\_usage\_bytes{instance=~"xianchaomaster1"})/1024/1024/1024

计算xianchaomaster1节点最近1m所有容器cpu使用率

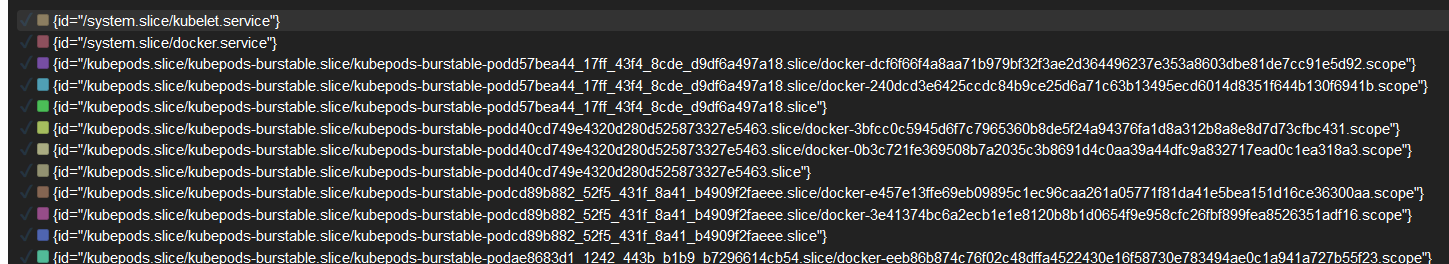
sum (rate (container\_cpu\_usage\_seconds\_total{instance=~"xianchaomaster1"}[1m])) / sum (machine\_cpu\_cores{ instance =~"xianchaomaster1"}) \* 100

计算最近1m所有容器cpu使用率

sum (rate (container\_cpu\_usage\_seconds\_total{id!="/"}[1m])) by (id)

#把id会打印出来

结果如下：



### 16.1.5 函数

Prometheus 内置了一些函数来辅助计算，下面介绍一些典型的。

abs()：绝对值

sqrt()：平方根

exp()：指数计算

ln()：自然对数

ceil()：向上取整

floor()：向下取整

round()：四舍五入取整

delta()：计算区间向量里每一个时序第一个和最后一个的差值

sort()：排序

# 17、Prometheus监控扩展

1、promethues采集tomcat监控数据

笔记：

https://note.youdao.com/ynoteshare/index.html?id=0ddfc17eaf7bac94ad4497d7f5356213&type=note

2、promethues采集redis监控数据

笔记：

https://note.youdao.com/ynoteshare/index.html?id=b9f87092ce8859cd583967677ea332df&type=note

3、Prometheus监控mysql

[root@xianchaomaster1 prometheus]# yum install mysql -y

[root@xianchaomaster1 prometheus]# yum install mariadb -y

tar -xvf mysqld\_exporter-0.10.0.linux-amd64.tar.gz

cd mysqld\_exporter-0.10.0.linux-amd64

cp -ar mysqld\_exporter /usr/local/bin/

chmod +x /usr/local/bin/mysqld\_exporter

2.登陆mysql为mysql\_exporter创建账号并授权

# 创建数据库用户。

mysql> CREATE USER 'mysql\_exporter'@'localhost' IDENTIFIED BY 'Abcdef123!.';

# 对mysql\_exporter用户授权

mysql> GRANT PROCESS, REPLICATION CLIENT, SELECT ON \*.\* TO 'mysql\_exporter'@'localhost';

exit 退出mysql

3.创建mysql配置文件、运行时可免密码连接数据库：

cd mysqld\_exporter-0.10.0.linux-amd64

cat my.cnf

[client]

user=mysql\_exporter

password=Abcdef123!.

4.启动mysql\_exporter客户端

nohup ./mysqld\_exporter --config.my-cnf=./my.cnf &

mysqld\_exporter的监听端口是9104

5.修改prometheus-alertmanager-cfg.yaml文件，添加如下

- job\_name: 'mysql'

static\_configs:

- targets: ['192.168.40.180:9104']

kubectl apply -f prometheus-alertmanager-cfg.yaml

kubectl delete -f prometheus-alertmanager-deploy.yaml

kubectl apply -f prometheus-alertmanager-deploy.yaml

grafana导入mysql监控图表

mysql-overview\_rev5.json

4、Prometheus监控Nginx

笔记：

https://note.youdao.com/ynoteshare/index.html?id=bea7b4b8f9a78db1679e1ac2ab747da5&type=note

5、prometheus监控mongodb

笔记：

https://note.youdao.com/ynoteshare/index.html?id=39b54acb1fbc0199f966115ce9523bb6&type=note

# 18、Pushgateway

Pushgateway简介：

Pushgateway是prometheus的一个组件，prometheus server默认是通过exporter主动获取数据（默认采取pull拉取数据），pushgateway则是通过被动方式推送数据到prometheus server，用户可以写一些自定义的监控脚本把需要监控的数据发送给pushgateway， 然后pushgateway再把数据发送给Prometheus server

Pushgateway优点：

Prometheus 默认采用定时pull 模式拉取targets数据，但是如果不在一个子网或者防火墙，prometheus就拉取不到targets数据，所以可以采用各个target往pushgateway上push数据，然后prometheus去pushgateway上定时pull数据

在监控业务数据的时候，需要将不同数据汇总, 汇总之后的数据可以由pushgateway统一收集，然后由 Prometheus 统一拉取。

pushgateway缺点：

Prometheus拉取状态只针对 pushgateway, 不能对每个节点都有效；

Pushgateway出现问题，整个采集到的数据都会出现问题

监控下线，prometheus还会拉取到旧的监控数据，需要手动清理 pushgateway不要的数据。

安装pushgateway，在k8s-node节点（192.168.40.181）操作：

在k8s-node节点操作

[root@xianchaonode1 ~]# docker load -i pushgateway.tar.gz

[root@xianchaonode1 ~]# docker run -d --name pushgateway -p 9091:9091 prom/pushgateway

在浏览器访问192.168.40.181:9091出现如下ui界面



修改prometheus-alertmanager-cfg.yaml文件，在k8s-master节点操作

添加如下job

- job\_name: 'pushgateway'

scrape\_interval: 5s

static\_configs:

- targets: ['192.168.40.181:9091']

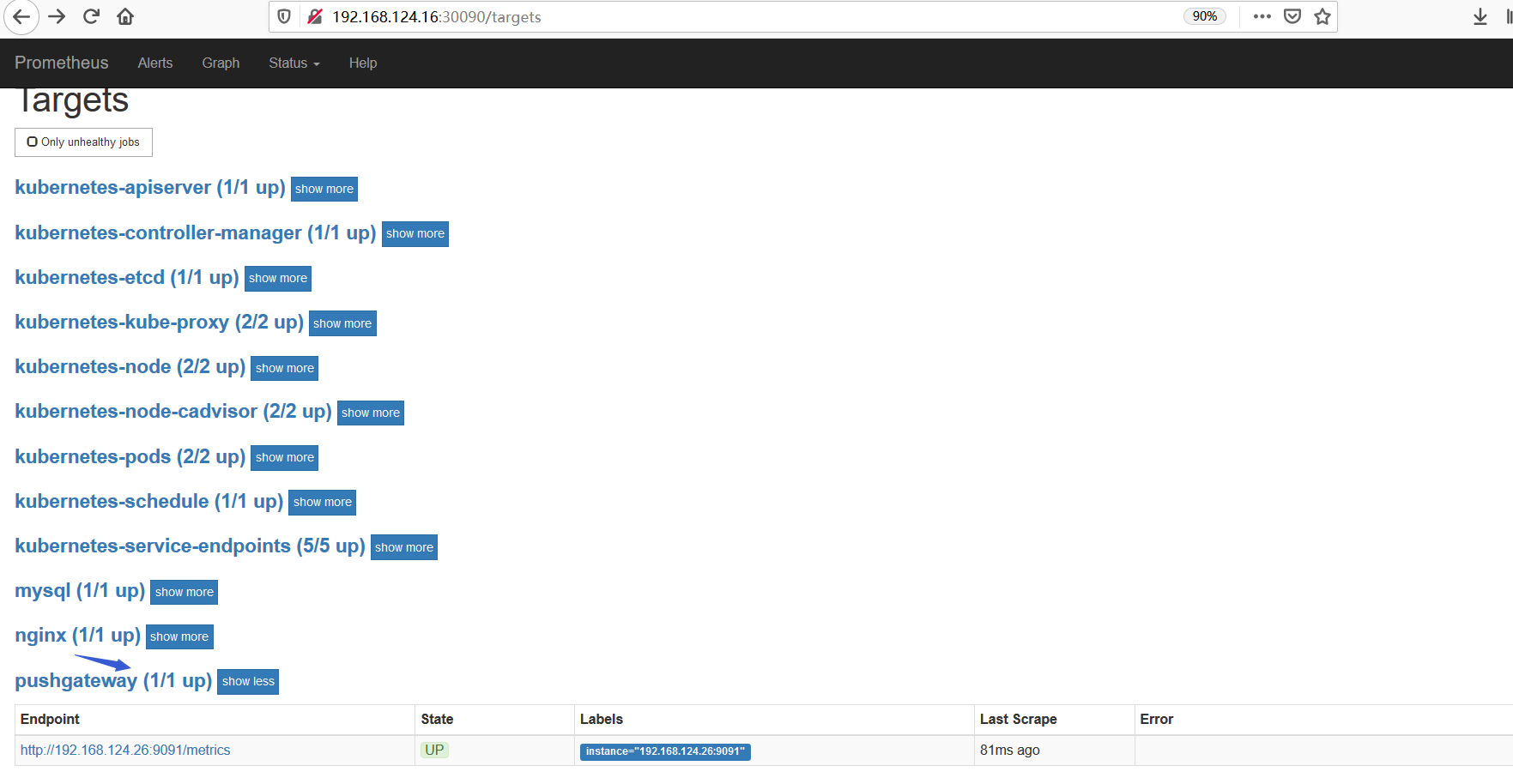
honor\_labels: true

kubectl apply -f prometheus-alertmanager-cfg.yaml

kubectl delete -f prometheus-alertmanager-deploy.yaml

kubectl apply -f prometheus-alertmanager-deploy.yaml

在prometheus的targets列表可以看到pushgateway



推送指定的数据格式到pushgateway

向 {job="test\_job"} 添加单条数据：

echo " metric 3.6" | curl --data-binary @- <http://192.168.40.181:9091/metrics/job/test_job>

 注：--data-binary 表示发送二进制数据，注意：它是使用POST方式发送的！

添加复杂数据

cat <<EOF | curl --data-binary @- http://192.168.40.181:9091/metrics/job/test\_job/instance/test\_instance

#TYPE node\_memory\_usage gauge

node\_memory\_usage 36

# TYPE memory\_total gauge

node\_memory\_total 36000

EOF

删除某个组下某个实例的所有数据

curl -X DELETE http://192.168.40.181:9091/metrics/job/test\_job/instance/test\_instance

删除某个组下的所有数据：

curl -X DELETE http://192.168.40.181:9091/metrics/job/test\_job

把数据上报到pushgateway

在被监控服务所在的机器配置数据上报,想要把192.168.40.181这个机器的内存数据上报到pushgateway，下面步骤需要在192.168.40.181操作

cat push.sh

node\_memory\_usages=$(free -m | grep Mem | awk '{print $3/$2\*100}')

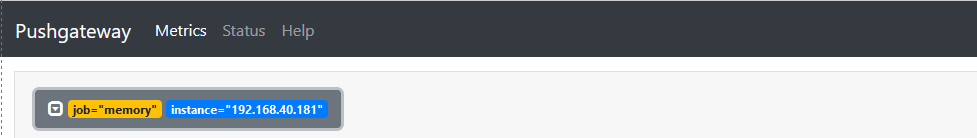
job\_name="memory"  
instance\_name="192.168.40.181"

cat <<EOF | curl --data-binary @- http://192.168.40.181:9091/metrics/job/$job\_name/instance/$instance\_name  
#TYPE node\_memory\_usages  gauge

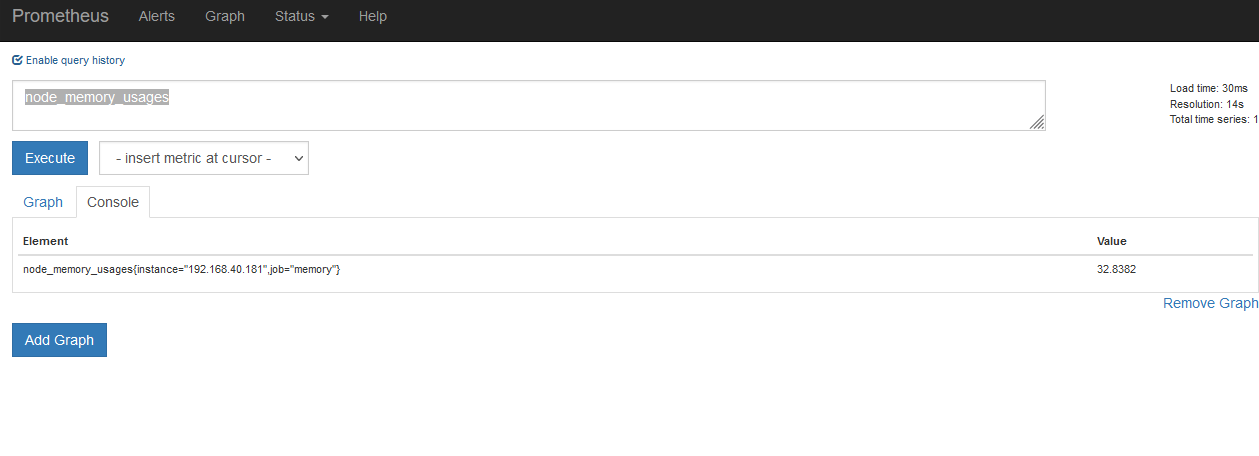
node\_memory\_usages $node\_memory\_usages  
EOF

sh push.sh

打开pushgateway web ui界面，可看到如下：



打开prometheus ui界面，可看到如下node\_memory\_usages的metrics指标



设置计划任务，定时上报数据

chmod +x push.sh

crontab -e

\*/1 \* \* \* \* /usr/bin/bash /root/push.sh

注意：从上面配置可以看到，我们上传到pushgateway中的数据有job也有instance，而prometheus配置pushgateway这个job\_name中也有job和instance，这个job和instance是指pushgateway实例本身，添加 honor\_labels: true 参数， 可以避免promethues的targets列表中的job\_name是pushgateway的 job 、instance 和上报到pushgateway数据的job和instance冲突。