K8S高可用集群安装

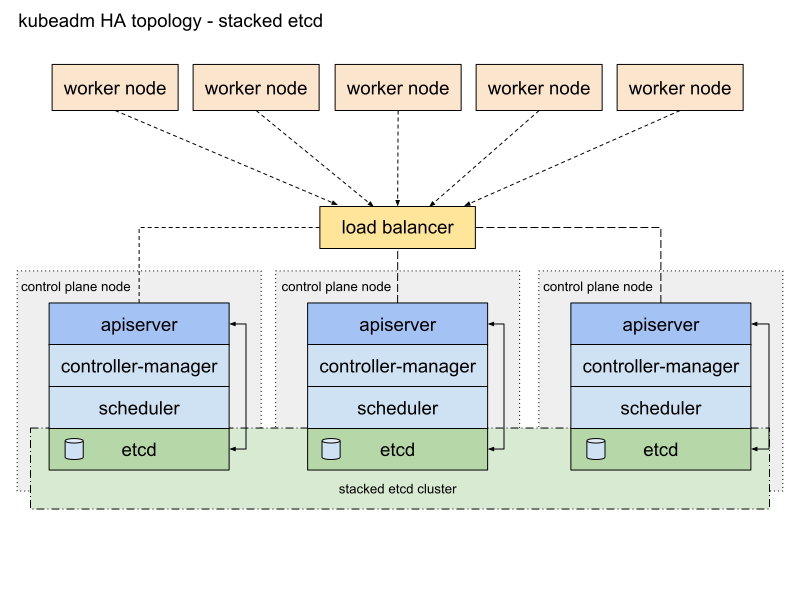
参考：

[官方教程](https://kubernetes.io/zh/docs/setup/production-environment/tools/kubeadm/)

<https://www.kuboard.cn/>

<https://www.kubernetes.org.cn/6632.html>

1. 架构拓扑图

[高可用选项参考](https://kubernetes.io/zh/docs/setup/production-environment/tools/kubeadm/ha-topology/)，本教程选用堆叠etcd的方式搭建高可用集群，拓扑图如下。

1. 部署环境
   1. 主机列表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 主机名 | Centos版本 | IP | Docker Version | Flannel Version | Keepalived Version | 主机配置 | 备注 |
| master01 | 7.6.1810 | 192.168.137.201 | 19.03.4 | v0.11.0-amd64 | v1.3.5 | 2C2G | control plane |
| master02 | 7.6.1810 | 192.168.137.202 | 19.03.4 | v0.11.0-amd64 | v1.3.5 | 2C2G | control plane |
| master03 | 7.6.1810 | 192.168.137.203 | 19.03.4 | v0.11.0-amd64 | v1.3.5 | 2C2G | control plane |
| worker01 | 7.6.1810 | 192.168.137.211 | 19.03.4 | / | / | 2C2G | worker nodes |
| worker02 | 7.6.1810 | 192.168.137.212 | 19.03.4 | / | / | 2C2G | worker nodes |
| worker03 | 7.6.1810 | 192.168.137.213 | 19.03.4 | / | / | 2C2G | worker nodes |
| VIP | 7.6.1810 | 192.168.137.200 | 19.03.4 | / | v1.3.5 | 2C2G | 在master上浮动 |
| client | 7.6.1810 | 192.168.137.210 | / | / | / | 2C2G | client |

* 1. K8s版本

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 主机名 | kubelet version | kubeadm version | kubectl version | 备注 |
| master01 | 1.17.3 | 1.17.3 | 1.17.3 | kubectl选装 |
| master02 | 1.17.3 | 1.17.3 | 1.17.3 | kubectl选装 |
| master03 | 1.17.3 | 1.17.3 | 1.17.3 | kubectl选装 |
| worker01 | 1.17.3 | 1.17.3 | 1.17.3 | kubectl选装 |
| worker02 | 1.17.3 | 1.17.3 | 1.17.3 | kubectl选装 |
| worker03 | 1.17.3 | 1.17.3 | 1.17.3 | kubectl选装 |
| client | / | / | 1.17.3 | client |

1. 准备工作
   1. 环境检查
      1. CentOS 7+，本文使用Centos7.6
      2. 每台机器 2 GB 或更多的 RAM (如果少于这个数字将会影响您应用的运行内存)
      3. 2 CPU 核或更多
      4. 集群中的所有机器的网络彼此均能相互连接(公网和内网都可以)
      5. 节点之中不可以有重复的主机名、MAC 地址或 product\_uuid。
      6. 禁用交换分区。为了保证 kubelet 正常工作，您 **必须** 禁用交换分区。
   2. 确保每个节点上 MAC 地址和 product\_uuid 的唯一性
      1. 您可以使用命令 ip link 或 ifconfig -a 来获取网络接口的 MAC 地址
      2. 可以使用 sudo cat /sys/class/dmi/id/product\_uuid 命令对 product\_uuid 校验

注：一般来讲，硬件设备会拥有唯一的地址，但是有些虚拟机的地址可能会重复。Kubernetes 使用这些值来唯一确定集群中的节点。 如果这些值在每个节点上不唯一，可能会导致安装失败。

* 1. 检查网络适配器

如果您有一个以上的网络适配器，同时您的 Kubernetes 组件通过默认路由不可达，我们建议您预先添加 IP 路由规则，这样 Kubernetes 集群就可以通过对应的适配器完成连接。

* 1. 确保 iptables 工具不使用 nftables 后端

在 Linux 中，nftables 当前可以作为内核 iptables 子系统的替代品。 iptables 工具可以充当兼容性层，其行为类似于 iptables 但实际上是在配置 nftables。 nftables 后端与当前的 kubeadm 软件包不兼容：它会导致重复防火墙规则并破坏 kube-proxy。

如果您系统的 iptables 工具使用 nftables 后端，则需要把 iptables 工具切换到“旧版”模式来避免这些问题。 默认情况下，至少在 Debian 10 (Buster)、Ubuntu 19.04、Fedora 29 和较新的发行版本中会出现这种问题。RHEL 8 不支持切换到旧版本模式，因此与当前的 kubeadm 软件包不兼容。

* 1. 修改所有节点hosts

cat >> /etc/hosts << EOF

192.168.137.201 master01

192.168.137.202 msater02

192.168.137.203 master03

192.168.137.211 worker01

192.168.137.212 worker02

192.168.137.213 worker03

EOF

* 1. 禁用swap
     1. 临时禁用

swapoff -a

* + 1. 永久禁用

若需要重启后也生效，在禁用swap后还需修改配置文件/etc/fstab，注释swap

sed -i.bak '/swap/s/^/#/' /etc/fstab

* 1. 关闭防火墙

firewall-cmd --state #查看防火墙状态

systemctl stop firewalld.service #停止firewall

systemctl disable firewalld.service #禁止firewall开机启动

* 1. 关闭SELinux

*# 将 SELinux 设置为 permissive 模式（相当于将其禁用）*

setenforce 0

sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

* 1. 内核参数修改

本文的k8s网络使用flannel，该网络需要设置内核参数bridge-nf-call-iptables=1，修改这个参数需要系统有br\_netfilter模块

* + 1. br\_netfilter模块加载

lsmod |grep br\_netfilter #查看模块是否存在

modprobe br\_netfilter #加载模块（临时生效）

永久生效：

[root@master01 ~]# cat > /etc/rc.sysinit << EOF

#!/bin/bash

for file in /etc/sysconfig/modules/\*.modules ; do

[ -x $file ] && $file

done

EOF

[root@master01 ~]# cat > /etc/sysconfig/modules/br\_netfilter.modules << EOF

modprobe br\_netfilter

EOF

[root@master01 ~]# chmod 755 /etc/sysconfig/modules/br\_netfilter.modules

* + 1. 修改内核参数

cat <<EOF > /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sysctl --system

* 1. 重启系统

reboot

1. 安装容器运行时

注：请跟踪 Kubernetes 发行说明中经过验证的 Docker 最新版本变化

参考：《Docker安装文档》

1. 配置Kubernetes源
   1. 新增Kubernetes源（使用阿里镜像源）

[root@master01 ~]# cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://mirrors.aliyun.com/kubernetes/yum/repos/kubernetes-el7-x86\_64/

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://mirrors.aliyun.com/kubernetes/yum/doc/yum-key.gpg https://mirrors.aliyun.com/kubernetes/yum/doc/rpm-package-key.gpg

EOF

* 1. 更新缓存

yum clean all

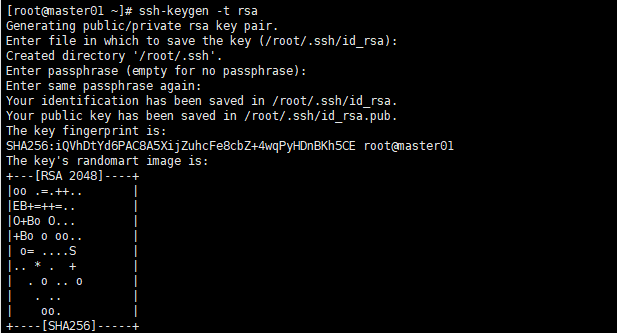
yum -y makecache

1. 配置免密登录

配置master01免密登录到master02和master03，此步骤只在master01上执行

* 1. 创建秘钥

ssh-keygen -t rsa



* 1. 将秘钥同步至master02、master03

ssh-copy-id -i /root/.ssh/id\_rsa.pub [root@192.168.137.202](mailto:root@192.168.137.202)

ssh-copy-id -i /root/.ssh/id\_rsa.pub [root@192.168.137.203](mailto:root@192.168.137.203)

* 1. 免密登录测试

ssh 192.168.137.202

1. Keepalived安装及配置
   1. Keepalived安装

在所有control plane节点上安装：

yum -y install keepalived-1.3.5

* 1. 在master01上配置

vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs{

router\_id master01

}

vrrp\_instance VI\_1{

state MASTER

interface eth0

virtual\_router\_id 50

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.137.200

}

}

* 1. 在master02上配置

vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs{

router\_id master02

}

vrrp\_instance VI\_1{

state BACKUP

interface eth0

virtual\_router\_id 50

priority 90

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.137.200

}

}

* 1. 在master03上配置

vim /etc/keepalived/keepalived.conf

! Configuration File for keepalived

global\_defs{

router\_id master03

}

vrrp\_instance VI\_1{

state BACKUP

interface eth0

virtual\_router\_id 50

priority 80

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.137.200

}

}

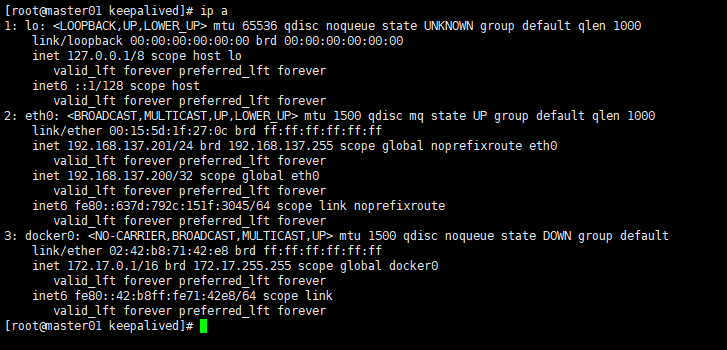
* 1. 在所有control plane启动Keepalived

service keepalived start #启动

systemctl enable keepalived #设置为开机自动启动

* 1. IP查看

ip a



1. K8s安装

注：所有control plane 和 worker nodes都执行本操作

* 1. 查看所有版本

yum list kubelet --showduplicates | sort -r

* 1. 安装kubelet、kubeadm、kubectl

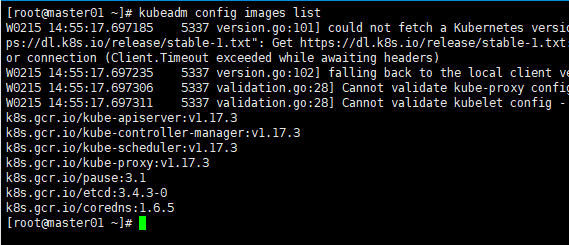
yum install -y kubelet-1.17.3 kubeadm-1.17.3 kubectl-1.17.3

* 1. 安装包说明
* kubelet运行在集群所有节点上，用于启动Pod和容器等对象的工具，是kube-scheduler调度的执行者
* kubeadm用于初始化集群启动集群的命令工具
* kubectl用于和集群通信的命令行，通过kubectl可以部署和管理应用，查看各种资源，创建、删除和更新各种组件
  1. 启动kubelet并设置开机启动

systemctl enable kubelet && systemctl start kubelet

* 1. 查看各组件镜像版本

kubeadm config images list



注：由于国内无法访问到k8s.gcr.io，所以需要科学上网或者先将这些镜像拖取到本地后再执行初始化操作。

* 1. 下载镜像

从阿里云下载提前构建好的镜像，拉取到本地后改回默认的镜像tag，通过执行以下脚本进行拖取镜像。

#!/bin/bash

url=registry.cn-shanghai.aliyuncs.com/ift520

version=v1.17.3

images=(`kubeadm config images list --kubernetes-version=$version|awk -F '/' '{print $2}'`)

for imagename in ${images[@]} ; do

docker pull $url/$imagename

docker tag $url/$imagename k8s.gcr.io/$imagename

docker rmi -f $url/$imagename

done

1. 执行初始化
   1. 在master01上执行以下命令进行初始化：

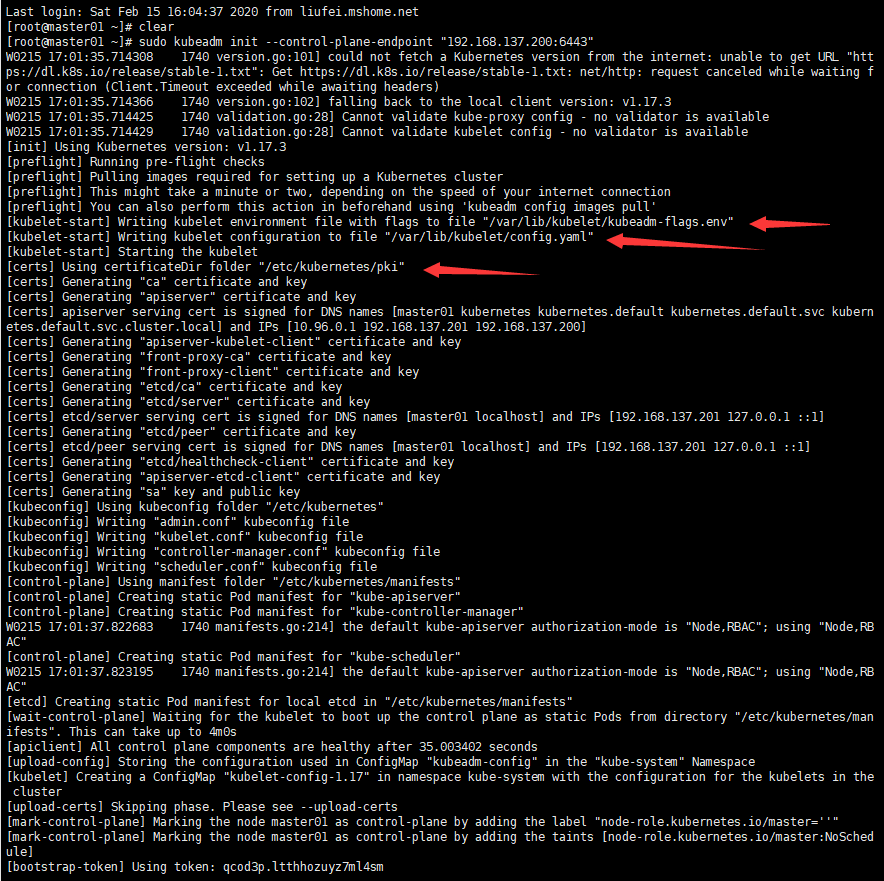
sudo kubeadm init --control-plane-endpoint "192.168.137.200:6443" --pod-network-cidr=10.244.0.0/16 --service-cidr=10.96.0.0/12 --upload-certs

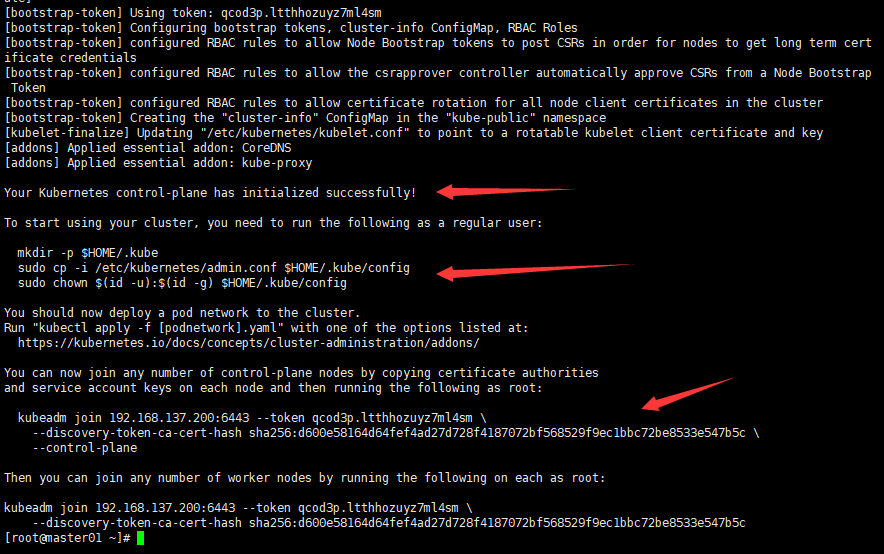
--control-plane-endpoint：所有control plane的负载均衡地址

--pod-network-cidr=10.244.0.0/16：Pod所在网段（如果不指定，在安装flannel时flannel对应的Pod启动不了）

--service-cidr=10.96.0.0/12：Service所在网段

--upload-certs：自动复制集群证书





kubelet环境文件位置：/var/lib/kubelet/kubeadm-flags.env

kubelet配置文件位置：/var/lib/kubelet/config.yaml

集群证书位置：/etc/kubernetes/pki

control plane节点加入集群的命令：

kubeadm join 192.168.137.200:6443 --token 67rse2.a7jkmq534pohhf3i --discovery-token-ca-cert-hash sha256:dda348a0853f4397db728742bab36f91be94eed7a22c6ac2254f3fff174ef23c --control-plane --certificate-key 7697920d6dd0da89e94dc3435c50b5b9064bce724b0fd40d5baaa22fcd92fd4c

worker nodes加入集群的命令：

kubeadm join 192.168.137.200:6443 --token 67rse2.a7jkmq534pohhf3i --discovery-token-ca-cert-hash sha256:dda348a0853f4397db728742bab36f91be94eed7a22c6ac2254f3fff174ef23c

* 1. 初始化完成后依次按照提示执行以下命令

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

* 1. 安装Flannel

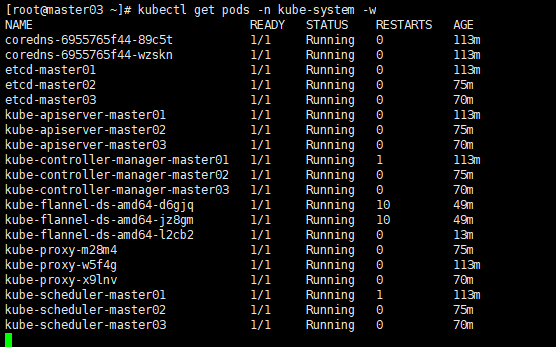
kubectl apply -f <https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml>

注：有可能会出现网络错误，如果出现网络错误请先下载该文件然后再执行

* 1. 验证是否成功：

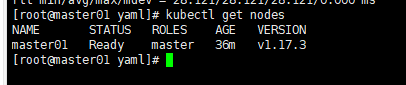
安装完成后执行以下命令查看是否安装成功，如下图所示，所有coredns和flannel处于Running状态表示Flannel已经安装成功：

kubectl get pods -n kube-system -w



查看集群状态：

kubectl get nodes



**警告：必须要满足以上两个状态才能执行下一步。**

1. 将master02和master03加入集群中
   1. 加入集群

分别在master02和master03上执行以下命令将节点加入集群：

kubeadm join 192.168.137.200:6443 --token 67rse2.a7jkmq534pohhf3i --discovery-token-ca-cert-hash sha256:dda348a0853f4397db728742bab36f91be94eed7a22c6ac2254f3fff174ef23c --control-plane --certificate-key 7697920d6dd0da89e94dc3435c50b5b9064bce724b0fd40d5baaa22fcd92fd4c

注：如果忘记certificate-key可使用一下命令在master01上重新生成：

kubeadm init phase upload-certs --upload-certs

加入完成后分别执行：

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

* 1. 查看集群状态

kubectl get nodes

在任意一台control plane节点上执行以上命令会发现集群节点的STATUS处于，Ready状态则已经安装完成。

1. 加入Worker节点

在所有worker节点上执行步骤9中的worker nodes加入集群的命令，若忘记该命令或执行后一直处于Running pre-flight checks，可使用以下命令在control plane节点上执行重新生成Token：

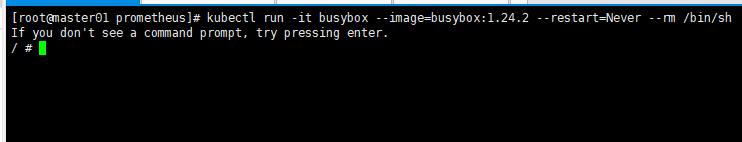
kubeadm token create --print-join-command

注：若一直处于Running pre-flight checks则表示Token已过期

执行完成后在control plane节点查看集群状态发现刚加入的worker节点处于NotReady状态，原因是正在下载Flannel的镜像，下载完成后会变为Ready状态。

1. 验证集群DNS是否正常工作
   1. 执行以下命令

kubectl run -it busybox --image=busybox:1.24.2 --restart=Never --rm /bin/sh

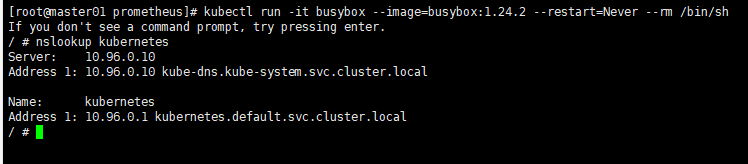


* 1. 再执行以下命令

nslookup kubernetes

若出现下图所示结果则表示DNS正常。

注：busybox版本不可大于1.28.4，否则解析不了DNS。



1. 安装Dashboard
   1. 如果是安装新版本请先执行一下命令删除：

kubectl delete ns kubernetes-dashboard

* 1. 下载yaml资源清单文件

wget <https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-rc5/aio/deploy/recommended.yaml>

* 1. 修改yaml文件

手动将yaml文件中的dashboard对应的service的ports类型改为NodePort并增加nodeProt:30001或执行以下命令修改：

sed -i '/targetPort: 8443/a\ \ \ \ \ \ nodePort: 30001\n\ \ type: NodePort' recommended.yaml

* 1. 创建管理员账号

cat >> recommended.yaml << EOF

---

# ------------------- dashboard-admin ------------------- #

apiVersion: v1

kind: ServiceAccount

metadata:

name: dashboard-admin

namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: dashboard-admin

subjects:

- kind: ServiceAccount

name: dashboard-admin

namespace: kubernetes-dashboard

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

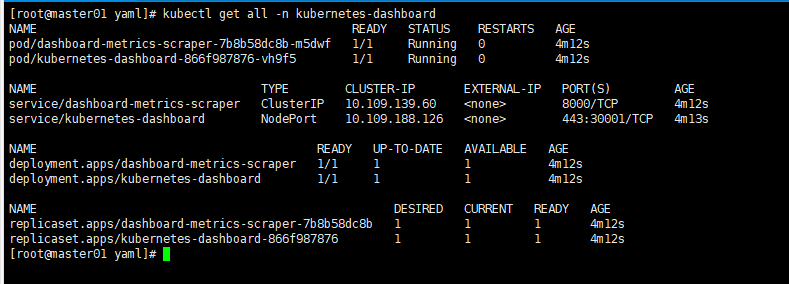
* 1. 安装命令：

kubeclt apply -f [recommended.yaml](https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0-rc5/aio/deploy/recommended.yaml)

以上命令也许会存在网络问题导致该地址访问失败，请先下载该文件再执行，Dashboard对应的版本请查看[Github](https://github.com/kubernetes/dashboard/releases)

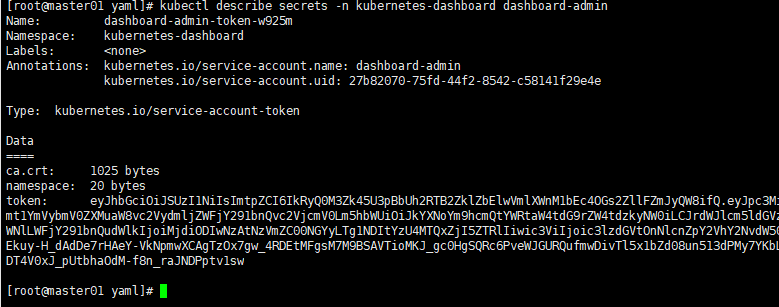
* 1. 状态查看

kubectl get all -n kubernetes-dashboard



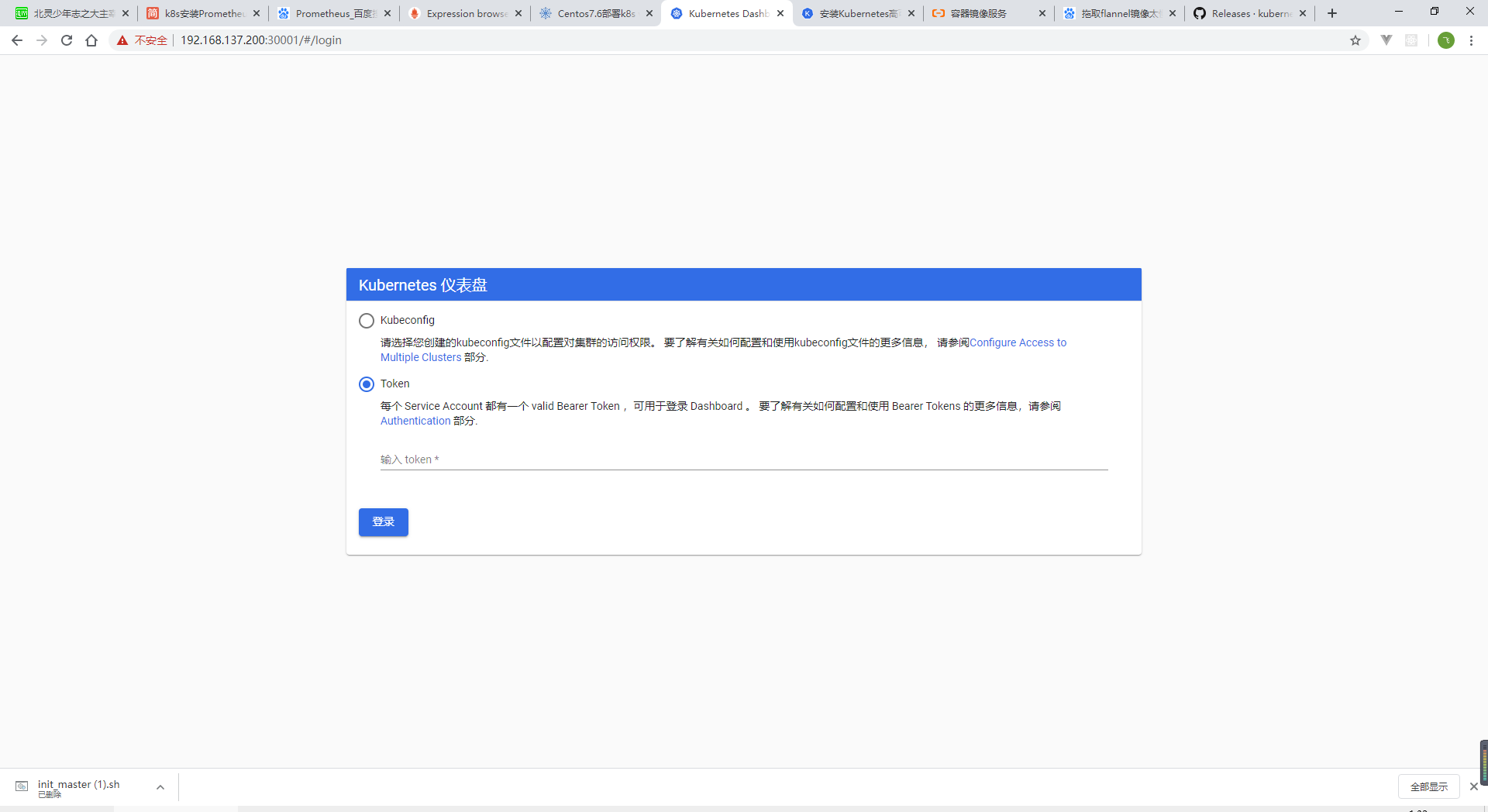
* 1. 令牌查看

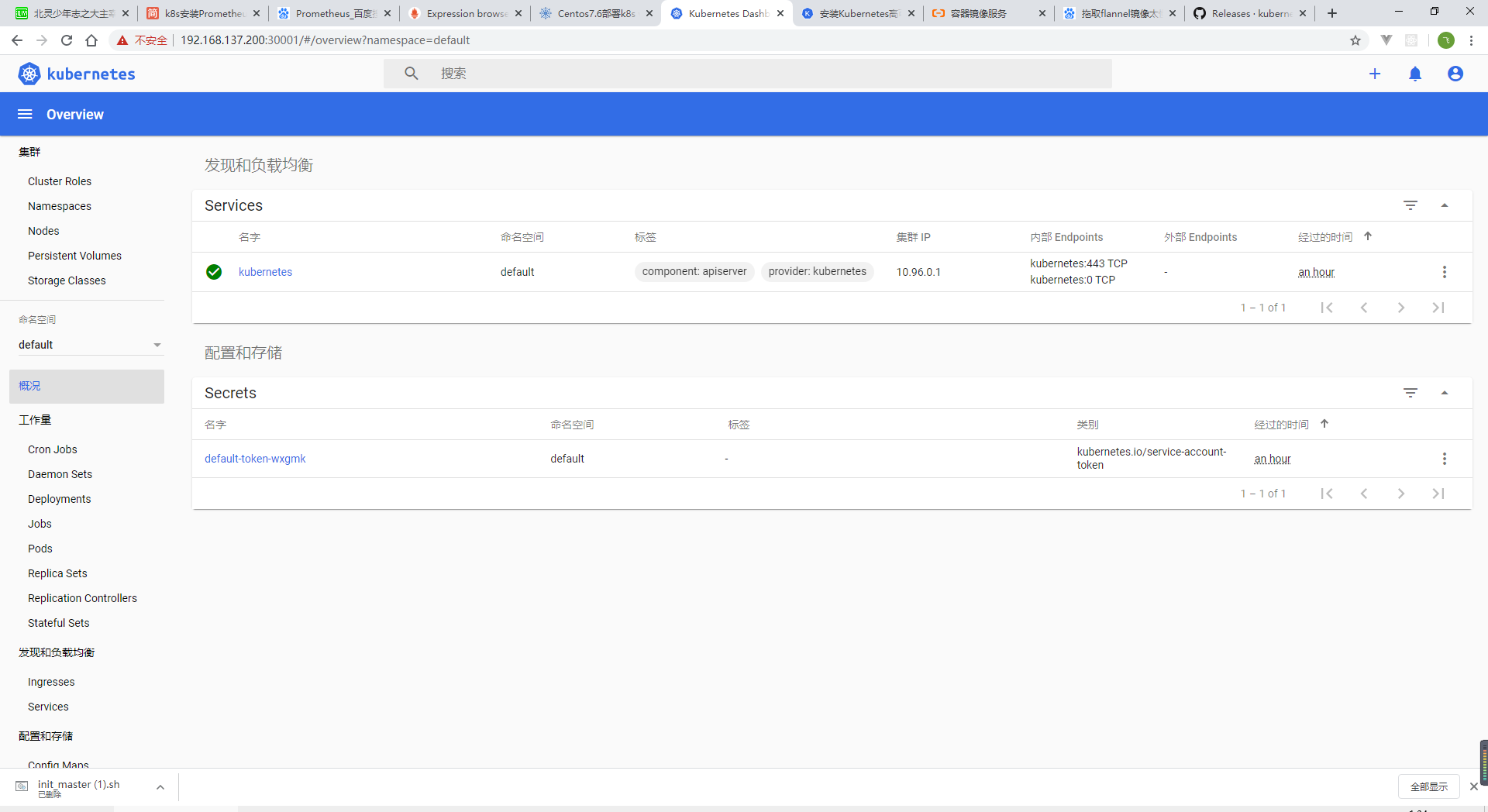
kubectl describe secrets -n kubernetes-dashboard dashboard-admin



* 1. 访问Dashboard

浏览器打开：<https://192.168.137.200:30001>，选择Token方式访问，输入上一步获取到的Token输入即可进入Dashboard：





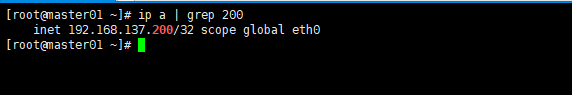
* 1. 使用Ingress暴露Dashboard

如果需要使用Ingress暴露Dashboard则参照并且不执行步骤13.3：

<https://blog.csdn.net/java_mt/article/details/104395748>

1. 高可用测试
   1. 通过以下命令查看kube-apiserver所在节点

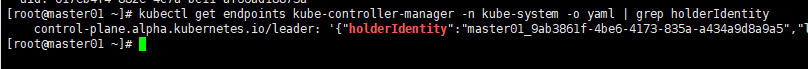
ip a | grep 200



通过以上命令发现VIP在master01。

* 1. 查看kube-controller-manager、kube-scheduler所在节点

kubectl get endpoints kube-controller-manager -n kube-system -o yaml | grep holderIdentity



kubectl get endpoints kube-scheduler -n kube-system -o yaml | grep holderIdentity



通过以上步骤发现这是三个服务均在master01上。

* 1. 将master01关机，模拟宕机

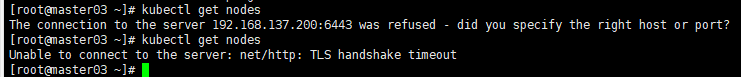
init 0

再次使用13.1和 13.2的命令查看发现VIP漂移到了master02，kube-controller-manager和kube-scheduler也发生了迁移，并且此时Dashboard依旧可以正常访问。

* 1. 关闭master02

此时VIP漂移到了master03，但是通过以下命令不能获取到集群状态此时集群已经不能正常提供服务，etcd已崩溃。

kubectl get nodes



1. 安装nginx-ingress-controller

注：如果需要使用nginx-ingress-controller暴露服务则安装。

参考：<https://kubernetes.github.io/ingress-nginx/deploy/>

步骤一： <https://kubernetes.github.io/ingress-nginx/deploy/#prerequisite-generic-deployment-command>

步骤二：<https://kubernetes.github.io/ingress-nginx/deploy/#bare-metal>

安装之前由于网络原因请先下载步骤一中文件中的Docker镜像到worker节点。

[Ingerss-controller](https://kubernetes.io/zh/docs/concepts/services-networking/ingress-controllers/)和[Ingress](https://kubernetes.io/zh/docs/concepts/services-networking/ingress/)必须配合使用

1. 安装Prometheus

参照：<https://www.jianshu.com/p/ac8853927528>

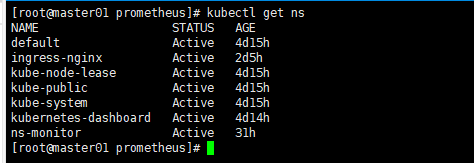
注：若需使用Ingress暴露服务请自行修改和增加对应资源。

本节所使用所有yaml文件均在：https://github.com/ift520/document.git

* 1. 创建名称空间

kubectl apply -f namespace.yaml

验证是否正常运行：

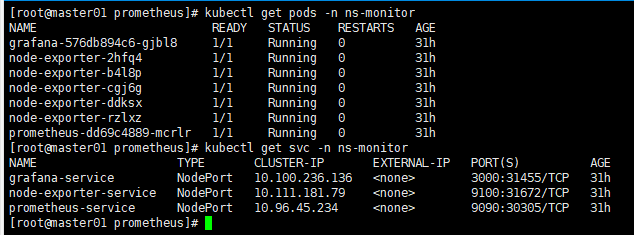


* 1. 安装node-export

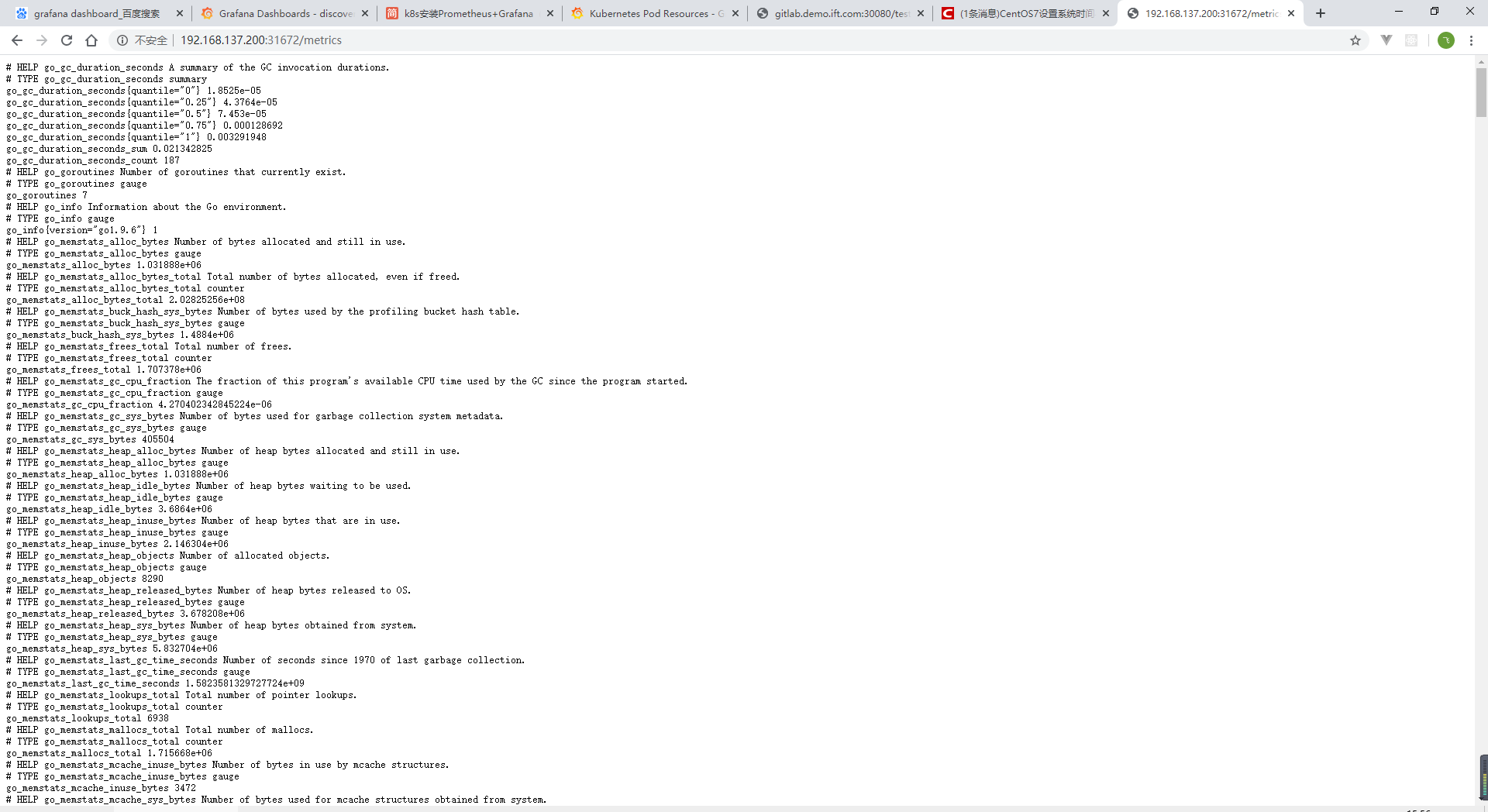
在kubernetest集群中部署node-exporter，Node-exporter用于采集kubernetes集群中各个节点的物理指标，比如：Memory、CPU等。可以直接在每个物理节点是直接安装，这里我们使用DaemonSet部署到每个节点上，使用 hostNetwork: true 和 hostPID: true 使其获得Node的物理指标信息，配置tolerations使其在master节点也启动一个pod。

kubectl apply -f node-export.yaml

验证是否正常运行：



浏览器访问：http://主机ip:31672/metrics

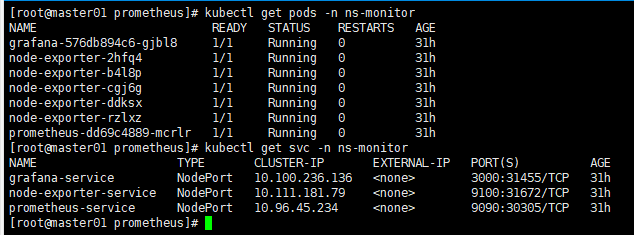


* 1. 部署Prometheus

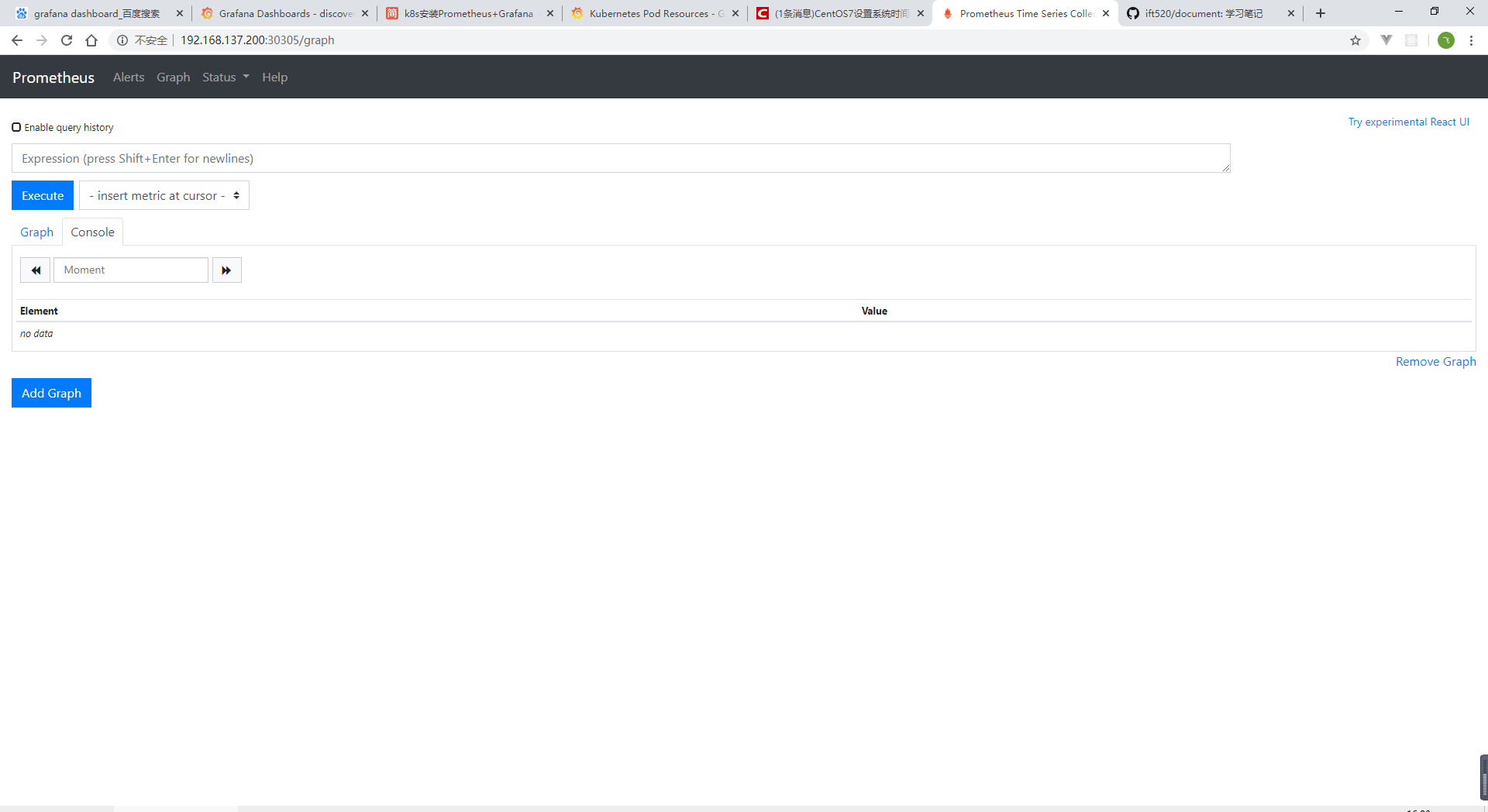
kubectl apply -f prometheus.yaml

注：本次部署因无网络存储设备（如：nfs），所以将PV修改为了本地并且固定Pod在worker01上运行，若有网络存储设备请使用网络存储设备并删除文件中的nodeSelector属性。

验证是否正常运行：



浏览器访问：http://主机ip:30305/graph



1. 安装Grafana

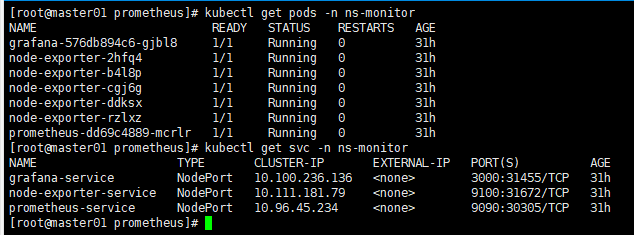
注：若需使用Ingress暴露服务请自行修改和增加对应资源。

* 1. 部署

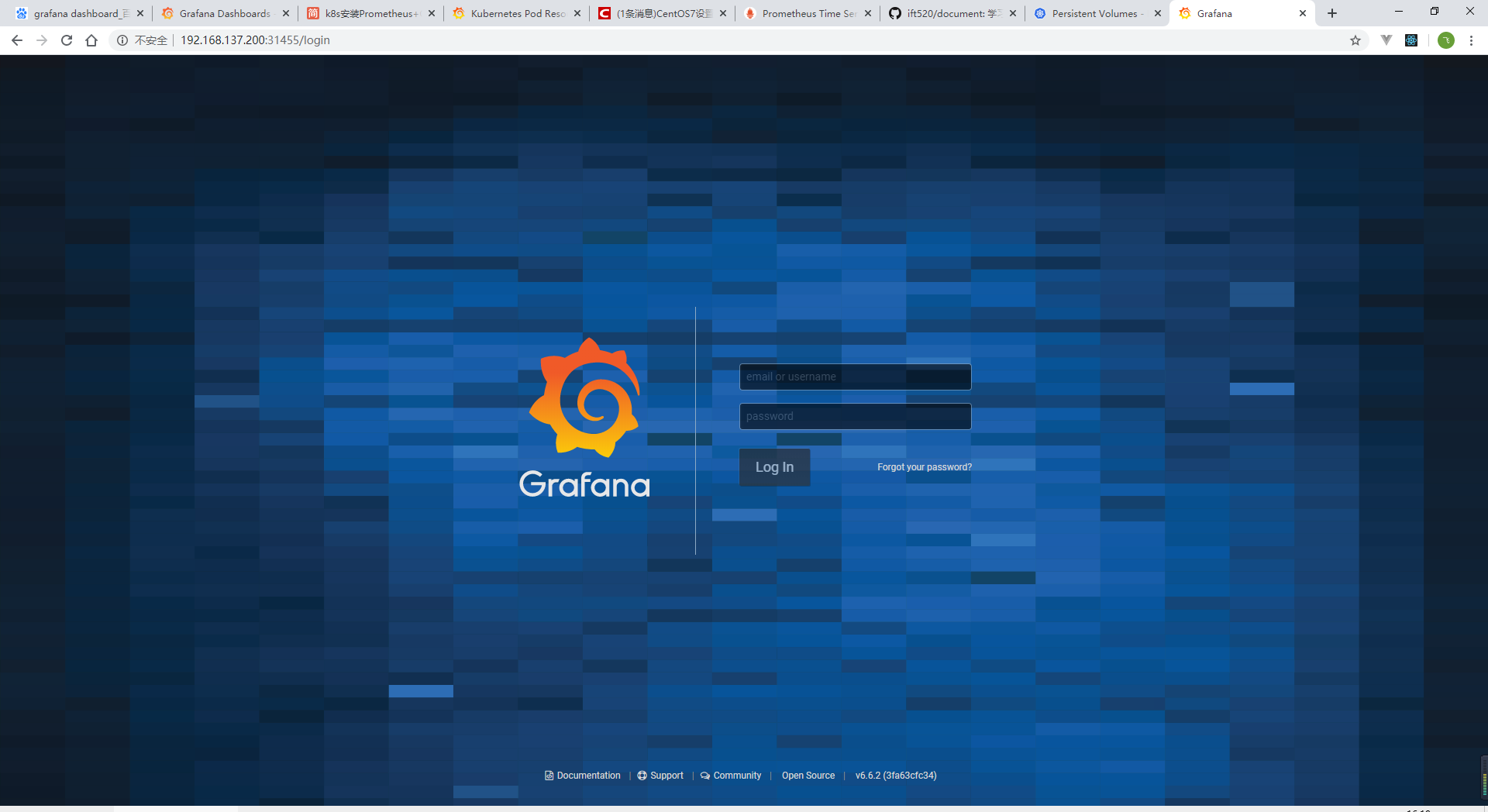
kubectl apply -f grafana.yaml

注：本次部署因无网络存储设备（如：nfs），所以将PV修改为了本地并且固定Pod在worker01上运行，若有网络存储设备请使用网络存储设备并删除文件中的nodeSelector属性。

验证是否正常运行：

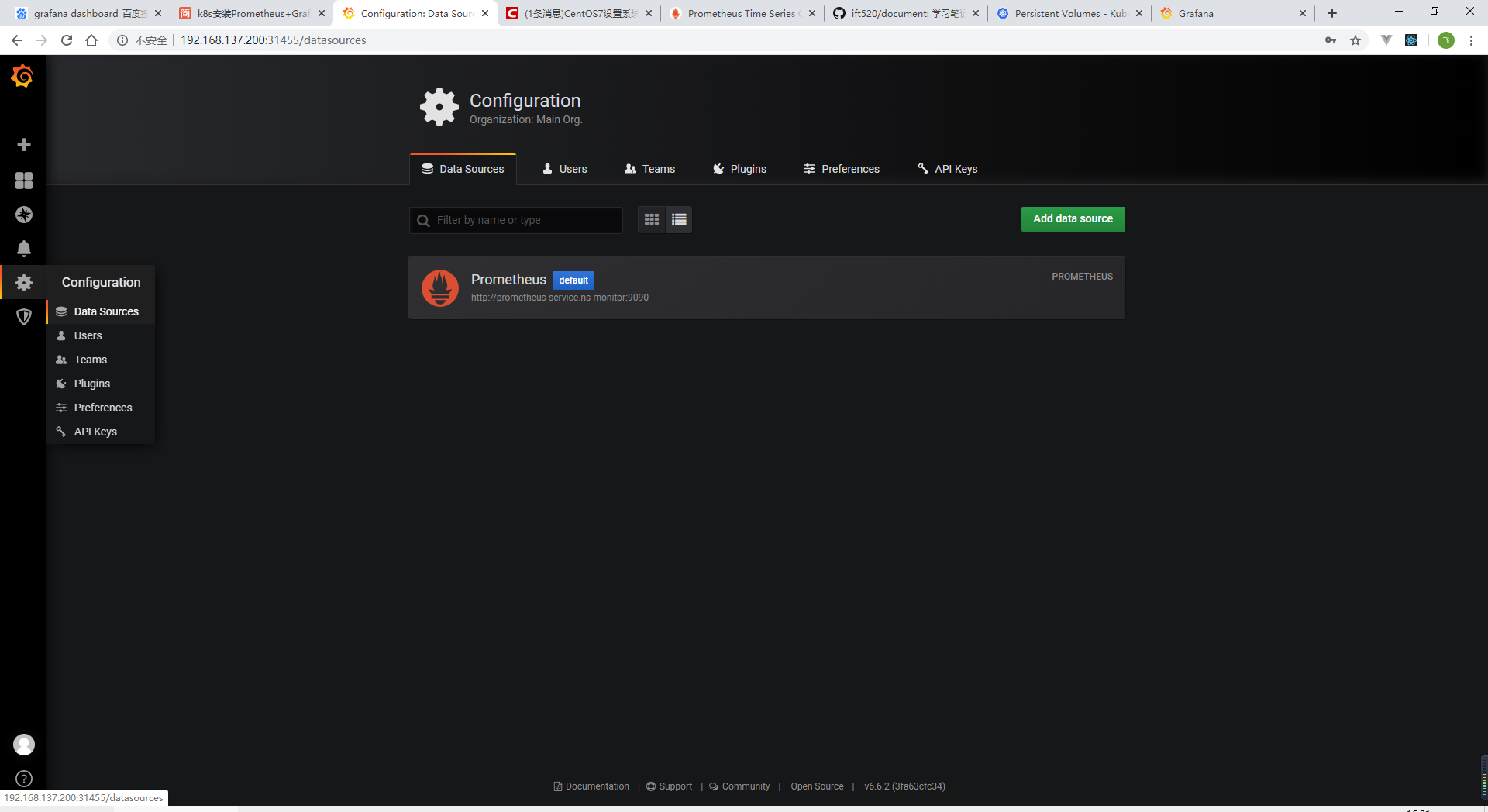


浏览器访问：<http://主机IP:31455/login>；默认用户名/密码：admin/admin

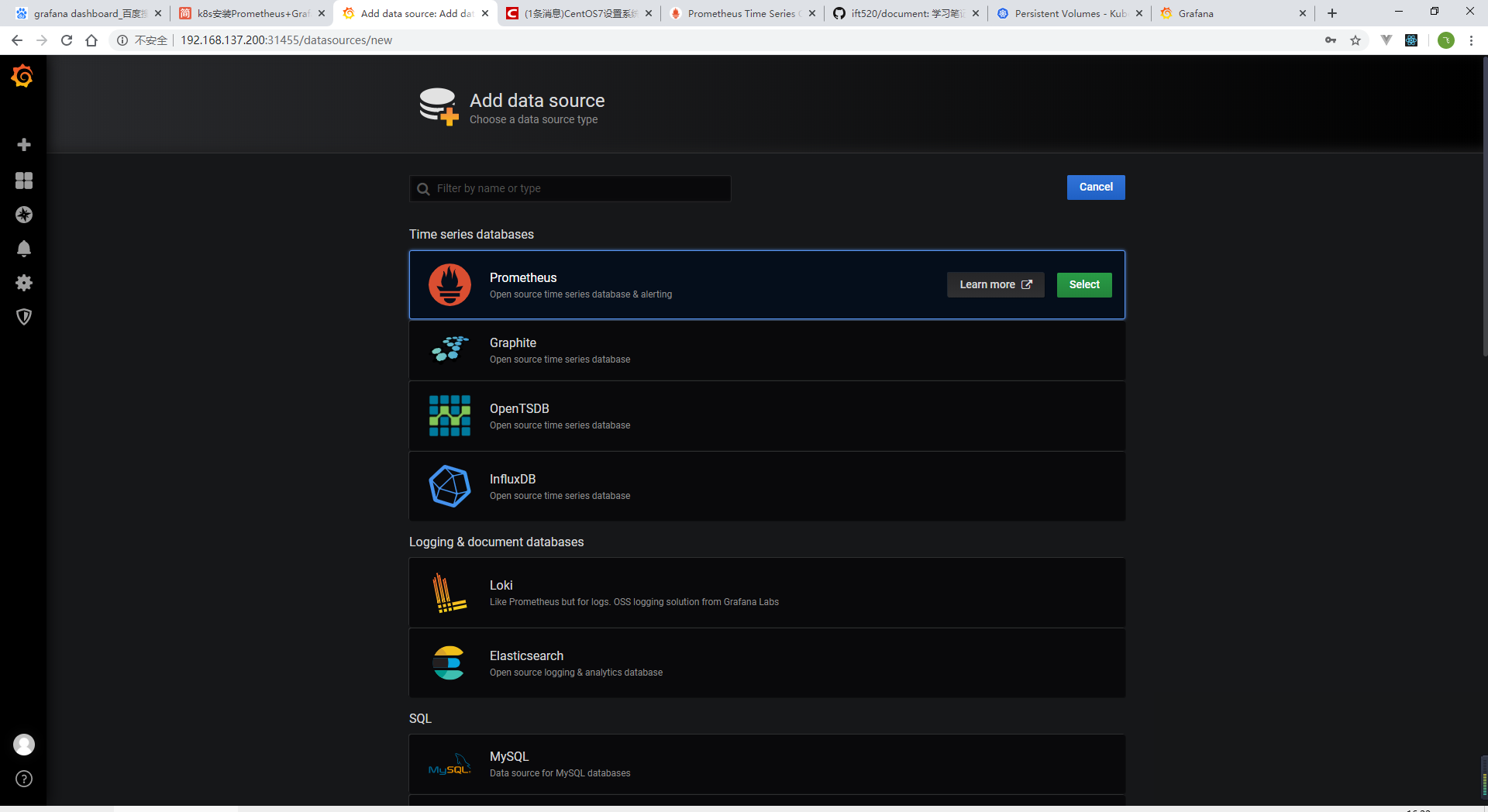


* 1. 配置数据源

点击下图页面中的Add data source

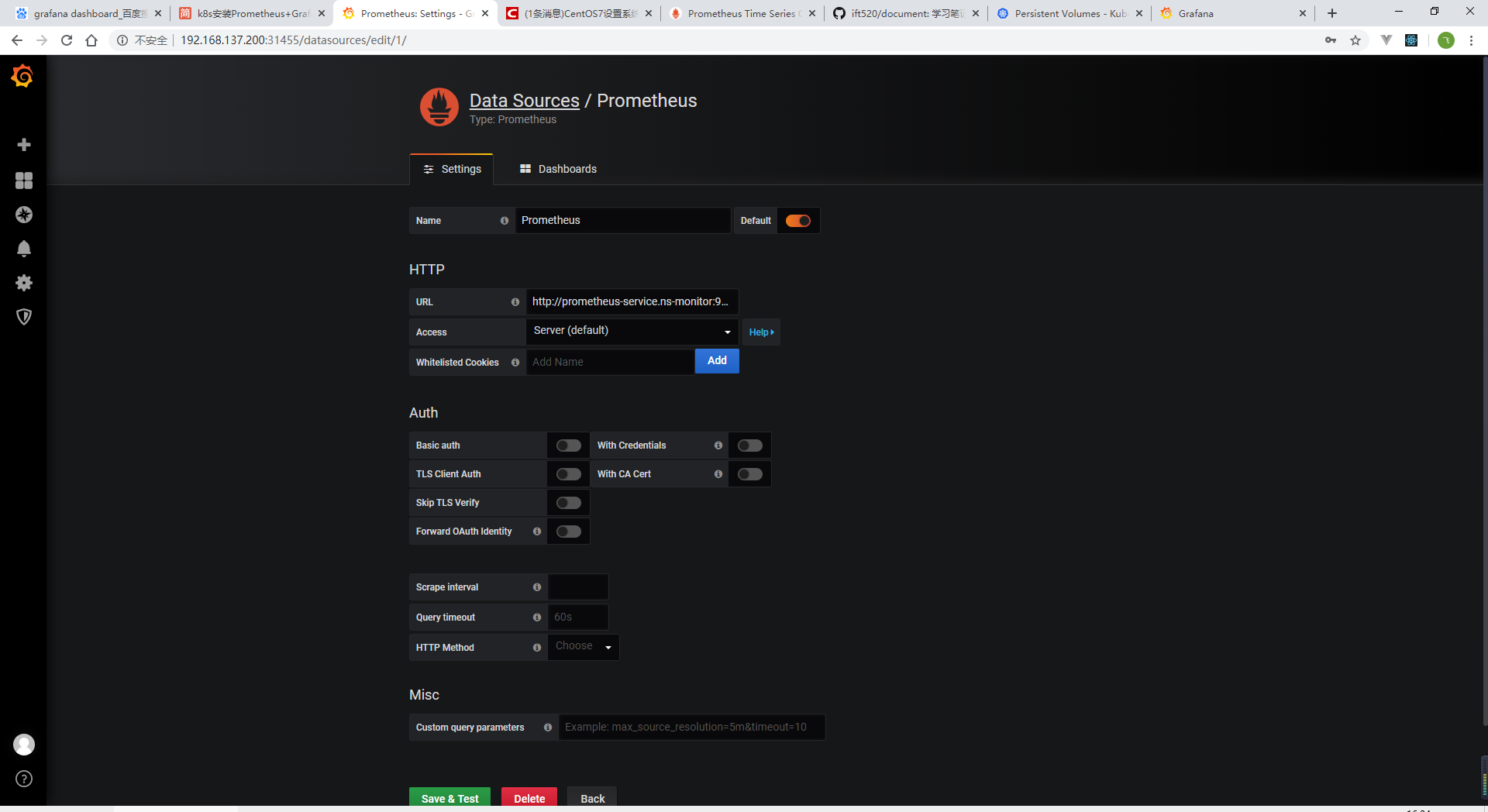


选择下图中的Prometheus：

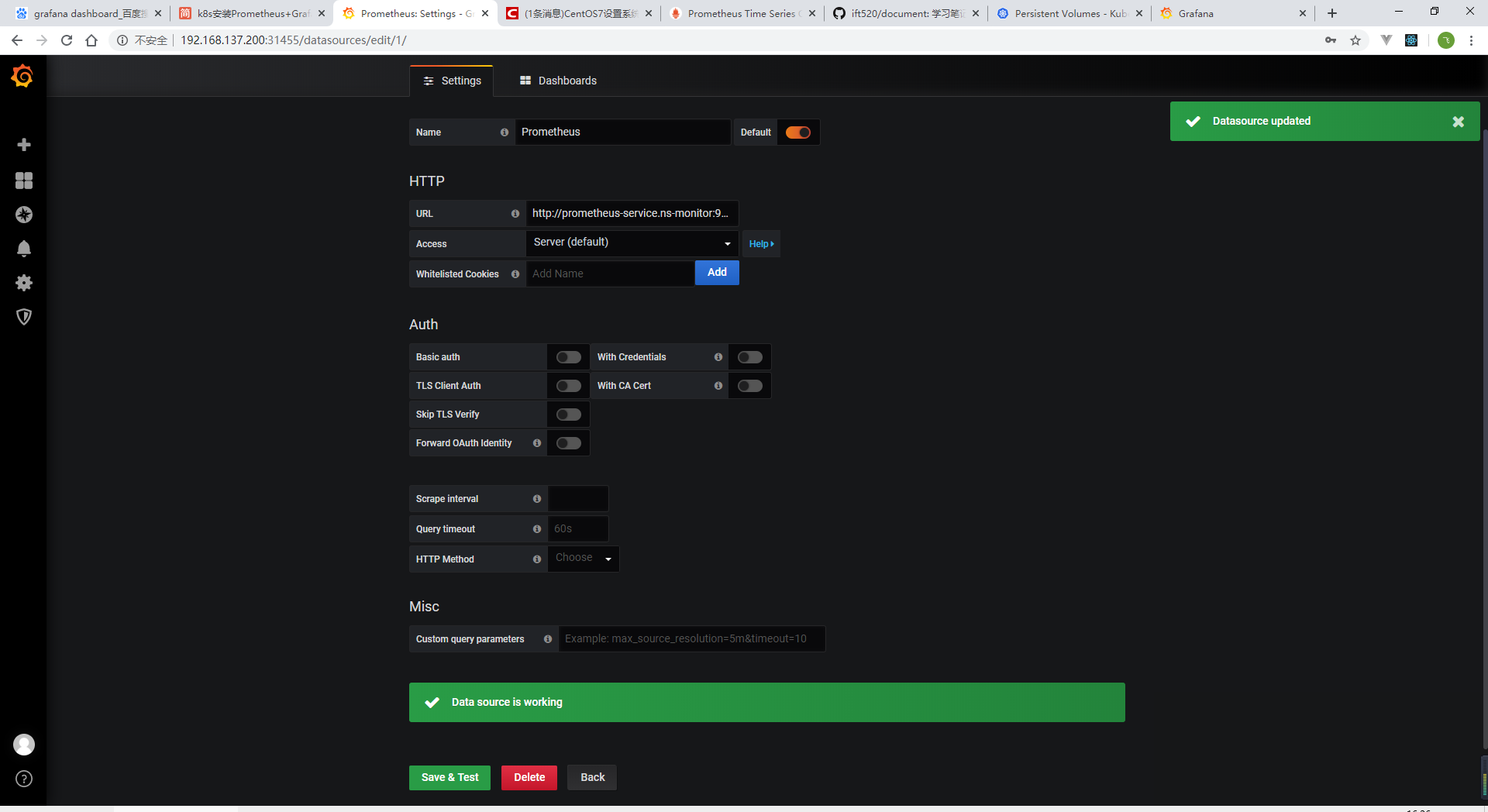


填写属性如下图：

HTTP下的url：<http://prometheus-service.ns-monitor:9090>，也可使用一下地址：http://节点IP:nodePort

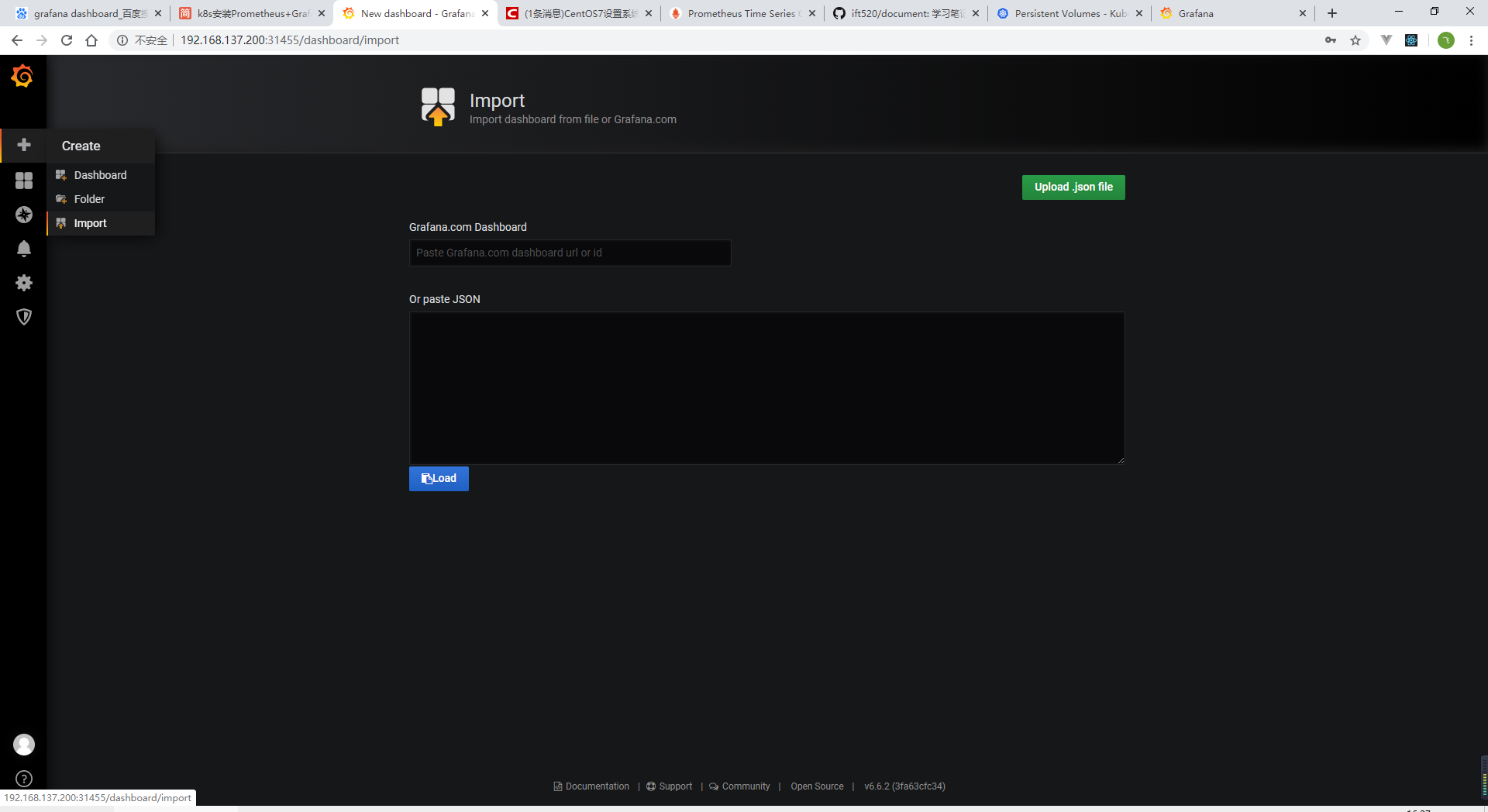


然后点击Save & Test：

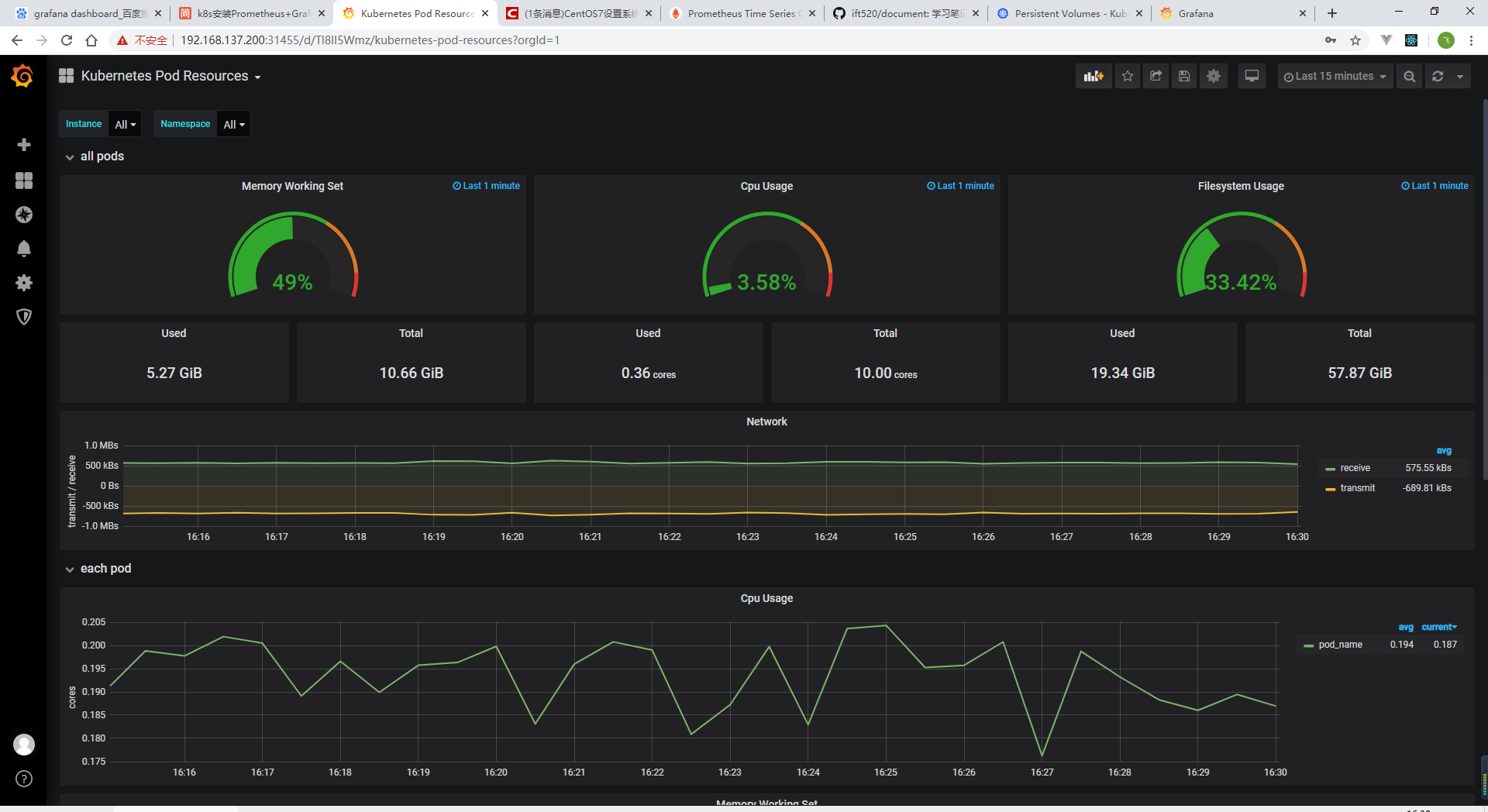


* 1. 导入Dashboard

选择如下菜单进入页面：



将文件Kubernetes-Pod-Resources.json的文件内容拷贝粘贴到Or past JSON下的文本域中，然后点击Load，回到主页选择刚才添加的Dashboard



注：如果此页面无数据展示也许是服务器时间不一致导致的。