基于 Prometheus 监控 Kubernetes 集群

目录

文档信息

文档作者	房佳亮
文档版本	Version1.0
文档版权	内部资料禁止传播
文档归类	Kubernetes 项目实战训练营
系统环境	CentOS-7.X-x86_64
作者邮箱	crushlinux@163.com
修订信息	2021-04-19
技术交流	

文档约定

[绿色背景] 知识重点
[红色背景] 错误警告
[黄色背景] 注意事项

执行命令

Prometheus 简介

Prometheus 是由 SoundCloud 开发的开源监控报警系统和时序列数据库(TSDB)；Prometheus 使用 Go 语言开发，是 Google BorgMon 监控系统的开源版本；2016 年由 Google 发起 Linux 基金会旗下的原生云基金会(Cloud Native Computing Foundation)，将 Prometheus 纳入其下第二大开源项目；Prometheus 和 Heapster(Heapster 是 K8S 的一个子项目，用于获取集群的性能数据)，相比功能更完善、更全面；Prometheus 性能也足够支撑上万台规模的集群 。官网地址：https://prometheus.io/

Prometheus 的基本原理是通过 HTTP 协议周期性抓取被监控组件的状态，任意组件只要提供对应的 HTTP 接口就可以接入监控。这样做非常适合做虚拟化环境监控系统，比如 VM、Docker、Kubernetes 等。输出被监控组件信息的 HTTP 接口被叫做 exporter 。目前互联网公司常用的组件大部分都有 exporter 可以直接使用，比如 Varnish、Haproxy、Nginx、MySQL、Linux 系统信息(包括磁盘、内存、CPU、网络等等)。目前官方最新版本为 2-23.0 版本。

Prometheus 优势

- 多维度数据模型。
- 灵活的查询语言。
- 不依赖分布式存储，单个服务器节点是自主的。
- 通过基于 HTTP 的 pull 方式采集时序数据。
- 可以通过中间网关进行时序列数据推送。
- 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，比如 Grafana 等。

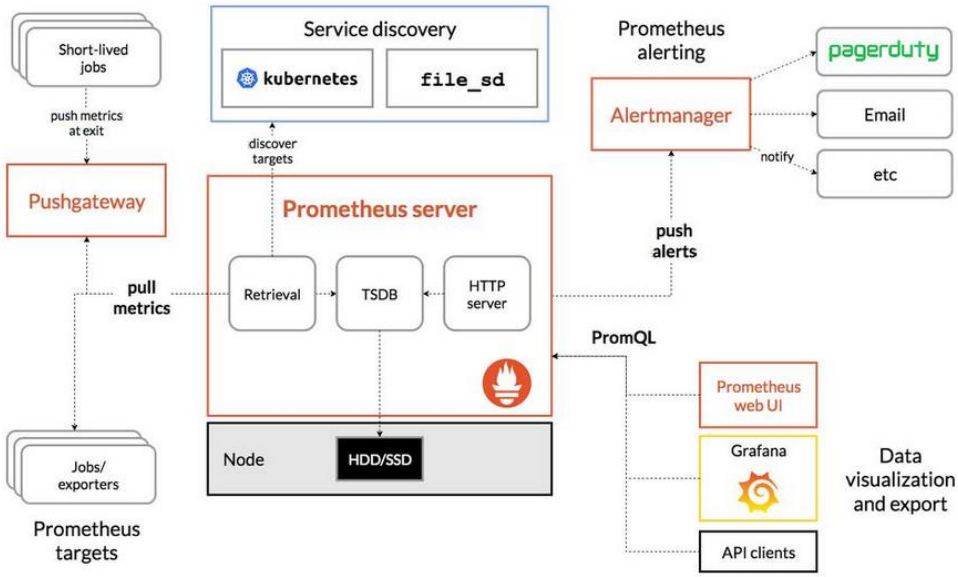
Prometheus 工作服务过程

1. Prometheus Daemon 负责定时去目标上抓取 metrics(指标)数据，每个抓取目标需要暴露一个 http 服务的接口给它定时抓取。Prometheus 支持通过配置文件、文本文件、Zookeeper、Consul、DNS SRV Lookup 等方式指定抓取目标。Prometheus 采用 PULL 的方式进行监控，即服务器可以直接通过目标 PULL 数据或者间接地通过中间网关来 Push 数据。
2. Prometheus 在本地存储抓取的所有数据，并通过一定规则进行清理和整理数据，并把得到的结果存储到新的时间序列中。
3. Prometheus 通过 PromQL 和其他 API 可视化地展示收集的数据。Prometheus 支持很多方式的图表可视化，例如 Grafana、自带的 Promdash 以及自身提供的模版引擎等等。Prometheus 还提供 HTTP API 的查询方式，自定义所需要的输出。
4. PushGateway 支持 Client 主动推送 metrics 到 PushGateway，而 Prometheus 只是定时去 Gateway 上抓取数据。
5. Alertmanager 是独立于 Prometheus 的一个组件，可以支持 Prometheus 的查询语句，提供十分灵活的报警方式。

Prometheus 核心组件

- Server 主要负责数据采集和存储，提供 PromQL 查询语言的支持
- Alertmanager 警告管理器，用来进行报警
- Push Gateway 主要是实现接收由 Client push 过来的指标数据，在指定的时间间隔，由主程序来抓取。
- node_exporter 用来监控服务器 CPU、内存、磁盘、I/O 等信息。

Prometheus 实践架构图



Grafana 简介

Grafana 是一个可视化面板（Dashboard），有着非常漂亮的图表和布局展示，功能齐全的度量仪表盘和图形编辑器。支持 Graphite、zabbix、InfluxDB、Prometheus 和 OpenTSDB 作为数据源。

Grafana 特点

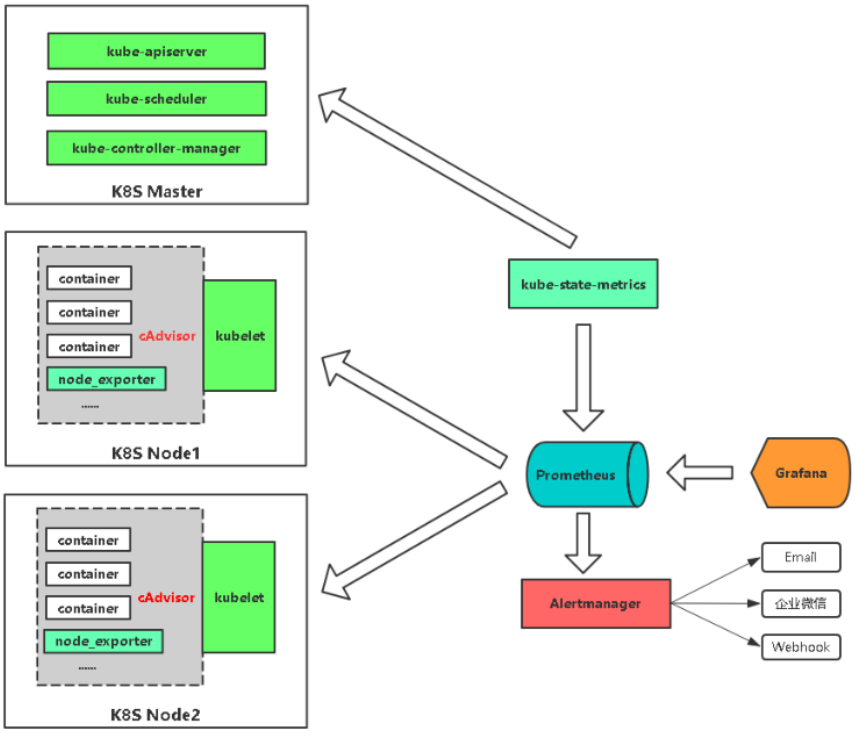
- Grafana 是一个可视化面板（Dashboard），有着非常漂亮的图表和布局展示，功能齐全的度量仪表盘和图形编辑器。支持 Graphite、zabbix、InfluxDB、Prometheus 和 OpenTSDB 作为数据源。
- Grafana 支持许多不同的时间序列数据（数据源）存储后端。每个数据源都有一个特定查询编辑器。官方支持以下数据源：Graphite、infloxdb、opensdb、prometheus、elasticsearch、cloudwatch。每个数据源的查询语言和功能明显不同。你可以将来自多个数据源的数据组合到一个仪表板上，但每个面板都要绑定到属于特定组织的特定数据源。
- Grafana 中的警报允许您将规则附加到仪表板面板上。保存仪表板时，Gravana 会将警报规则提取到单独的警报规则存储中，并安排它们进行评估。报警消息还能通过钉钉、邮箱等推送至移动端。但目前 grafana 只支持 graph 面板的报警。
- Grafana 使用来自不同数据源的丰富事件注释图表，将鼠标悬停在事件上会显示完整的事件元数据和标记；
- Grafana 使用 Ad-hoc 过滤器允许动态创建新的键/值过滤器，这些过滤器会自动应用于使用该数据源的所有查询

一、环境准备

操作系统	IP 地址	主机名	组件
CentOS7.5	192.168.200.111	k8s-master	kubeadm、kubelet、kubectl、docker-ce
CentOS7.5	192.168.200.112	k8s-node01	kubeadm、kubelet、kubectl、docker-ce
CentOS7.5	192.168.200.113	k8s-node02	kubeadm、kubelet、kubectl、docker-ce

注意：所有主机配置推荐 CPU：2C+ Memory：2G+

项目拓扑



1.1、主机初始化配置

所有主机配置禁用防火墙和 selinux

```
[root@localhost ~]# setenforce 0
[root@localhost ~]# iptables -F
[root@localhost ~]# systemctl stop firewalld
[root@localhost ~]# systemctl disable firewalld
[root@localhost ~]# systemctl stop NetworkManager
[root@localhost ~]# systemctl disable NetworkManager
[root@localhost ~]# sed -i '/^SELINUX=/s/enforcing/disabled/' /etc/selinux/config
```

配置主机名并绑定 hosts，不同主机名称不同

```
[root@localhost ~]# hostname k8s-master
[root@localhost ~]# bash
```

```
[root@k8s-master ~]# cat << EOF >> /etc/hosts
192.168.200.111 k8s-master
192.168.200.112 k8s-node01
192.168.200.113 k8s-node02
EOF

[root@k8s-master ~]# scp /etc/hosts 192.168.200.112:/etc/
[root@k8s-master ~]# scp /etc/hosts 192.168.200.113:/etc/

[root@localhost ~]# hostname k8s-node01
[root@localhost ~]# bash
[root@k8s-node01 ~]#

[root@localhost ~]# hostname k8s-node02
[root@localhost ~]# bash
[root@k8s-node02 ~]#
```

主机配置初始化

```
[root@k8s-master ~]# yum -y install vim wget net-tools lrzsz

[root@k8s-master ~]# swapoff -a
[root@k8s-master ~]# sed -i '/swap/s/^/#/' /etc/fstab

[root@k8s-node01 ~]# cat << EOF >> /etc/sysctl.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
[root@k8s-node01 ~]# modprobe br_netfilter
[root@k8s-node01 ~]# sysctl -p
```

1.2、部署 docker 环境

三台主机上分别部署 Docker 环境，因为 Kubernetes 对容器的编排需要 Docker 的支持。

```
[root@k8s-master ~]# wget -O /etc/yum.repos.d/CentOS-Base.repo
http://mirrors.aliyun.com/repo/Centos-7.repo
[root@k8s-master ~]# yum install -y yum-utils device-mapper-persistent-data lvm2
```

使用 YUM 方式安装 Docker 时，推荐使用阿里的 YUM 源。

```
[root@k8s-master ~]# yum-config-manager --add-repo https://mirrors.aliyun.com/docker-
ce/linux/centos/docker-ce.repo

[root@k8s-master ~]# yum clean all && yum makecache fast
```

```
[root@k8s-master ~]# yum -y install docker-ce
[root@k8s-master ~]# systemctl start docker
[root@k8s-master ~]# systemctl enable docker
```

镜像加速器（所有主机配置）

```
[root@k8s-master ~]# cat << END > /etc/docker/daemon.json
{
        "registry-mirrors":[ "https://nyakyfun.mirror.aliyuncs.com" ]
}
END
[root@k8s-master ~]# systemctl daemon-reload
[root@k8s-master ~]# systemctl restart docker
```

二、部署 kubernetes 集群

2.1、组件介绍

三个节点都需要安装下面三个组件

- kubeadm：安装工具，使所有的组件都会以容器的方式运行
- kubectl：客户端连接 K8S API 工具
- kubelet：运行在 node 节点，用来启动容器的工具

2.2、配置阿里云 yum 源

使用 YUM 方式安装 Kubernetes 时，推荐使用阿里的 YUM 源。

```
[root@k8s-master ~]# 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

[root@k8s-master ~]# ls /etc/yum.repos.d/
backup  Centos-7.repo  CentOS-Media.repo  CentOS-x86_64-kernel.repo  docker-ce.repo
kubernetes.repo
```

2.3、安装 kubelet kubeadm kubectl

所有主机配置

```
[root@k8s-master ~]# yum install -y kubelet kubeadm kubectl
[root@k8s-master ~]# systemctl enable kubelet
```

kubelet 刚安装完成后，通过 systemctl start kubelet 方式是无法启动的，需要加入节点或初始化为 master 后才可启动成功。

2.4、配置 init-config.yaml

Kubeadm 提供了很多配置项，Kubeadm 配置在 Kubernetes 集群中是存储在 ConfigMap 中的，也可将这些配置写入配置文件，方便管理复杂的配置项。Kubeadm 配内容是通过 kubeadm config 命令写入配置文件的。

在 master 节点安装，master 定于为 192.168.200.111，通过如下指令创建默认的 **init-config.yaml** 文件：

```
[root@k8s-master ~]# kubeadm config print init-defaults > init-config.yaml
```

init-config.yaml 配置

```
[root@k8s-master ~]# cat init-config.yaml
apiVersion: kubeadm.k8s.io/v1beta2
bootstrapTokens:
- groups:
  - system:bootstrappers:kubeadm:default-node-token
  token: abcdef.0123456789abcdef
  ttl: 24h0m0s
  usages:
  - signing
  - authentication
kind: InitConfiguration
localAPIEndpoint:
  advertiseAddress: 192.168.200.111        //master 节点 IP 地址
  bindPort: 6443
nodeRegistration:
  criSocket: /var/run/dockershim.sock
  name: k8s-master          //如果使用域名保证可以解析，或直接使用 IP 地址
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/master
---
apiServer:
  timeoutForControlPlane: 4m0s
```

```
apiVersion: kubeadm.k8s.io/v1beta2
certificatesDir: /etc/kubernetes/pki
clusterName: kubernetes
controllerManager: {}
dns:
   type: CoreDNS
etcd:
   local:
      dataDir: /var/lib/etcd          //etcd 容器挂载到本地的目录
imageRepository: registry.aliyuncs.com/google_containers      //修改为国内地址
kind: ClusterConfiguration
kubernetesVersion: v1.19.0
networking:
   dnsDomain: cluster.local
   serviceSubnet: 10.96.0.0/12
   podSubnet: 10.244.0.0/16       //新增加 Pod 网段
scheduler: {}
```

2.5、安装 master 节点

拉取所需镜像

```
[root@k8s-master ~]# kubeadm config images list --config init-config.yaml
W1113 13:29:17.607080     63523 configset.go:348] WARNING: kubeadm cannot validate
component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]
registry.aliyuncs.com/google_containers/kube-apiserver:v1.19.0
registry.aliyuncs.com/google_containers/kube-controller-manager:v1.19.0
registry.aliyuncs.com/google_containers/kube-scheduler:v1.19.0
registry.aliyuncs.com/google_containers/kube-proxy:v1.19.0
registry.aliyuncs.com/google_containers/pause:3.2
registry.aliyuncs.com/google_containers/etcd:3.4.13-0
registry.aliyuncs.com/google_containers/coredns:1.7.0
```

```
[root@k8s-master ~]# ls | while read line
do
docker load < $line
done
```

安装 matser 节点

```
[root@k8s-master ~]# kubeadm init --config=init-config.yaml   //初始化安装 K8S
```

根据提示操作

kubectl 默认会在执行的用户家目录下面的.kube 目录下寻找 config 文件。这里是将在初始化时[kubeconfig]步骤生成的 admin.conf 拷贝到.kube/config

```
[root@k8s-master ~]# mkdir -p $HOME/.kube
```

```
[root@k8s-master ~]# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
[root@k8s-master ~]# chown $(id -u):$(id -g) $HOME/.kube/config
```

Kubeadm 通过初始化安装是不包括网络插件的，也就是说初始化之后是不具备相关网络功能的，比如 k8s-master 节点上查看节点信息都是“Not Ready”状态、Pod 的 CoreDNS 无法提供服务等。

2.6、安装 node 节点

根据 master 安装时的提示信息

```
[root@k8s-node01 ~]# kubeadm join 192.168.200.111:6443 --token abcdef.0123456789abcdef --discovery-token-ca-cert-hash sha256:2b029d829f70e7d7bf846ce43b493fe54325f8ccb1fda46f65e11b018d233cd4
```

```
[root@k8s-master ~]# kubectl get nodes
NAME          STATUS     ROLES    AGE     VERSION
k8s-master    NotReady   master   2m29s   v1.19.4
k8s-node01    NotReady   <none>   38s     v1.19.4
k8s-node02    NotReady   <none>   40s     v1.19.4
```

前面已经提到，在初始化 k8s-master 时并没有网络相关配置，所以无法跟 node 节点通信，因此状态都是“NotReady”。但是通过 kubeadm join 加入的 node 节点已经在 k8s-master 上可以看到。

2.7、安装 flannel

Master 节点 NotReady 的原因就是因为没有使用任何的网络插件，此时 Node 和 Master 的连接还不正常。目前最流行的 Kubernetes 网络插件有 Flannel、Calico、Canal、Weave 这里选择使用 flannel。

所有主机上传 flannel_v0.12.0-amd64.tar

```
[root@k8s-master ~]# docker load < flannel_v0.12.0-amd64.tar
```

master 上传 kube-flannel.yml

master 主机配置：

```
[root@k8s-master ~]# kubectl apply -f kube-flannel.yml
[root@k8s-master ~]# kubectl get nodes
NAME          STATUS   ROLES    AGE     VERSION
k8s-master    Ready    master   8m27s   v1.19.4
k8s-node01    Ready    <none>   6m36s   v1.19.4
k8s-node02    Ready    <none>   6m38s   v1.19.4

[root@k8s-master ~]# kubectl get pods -n kube-system
NAME                                  READY   STATUS    RESTARTS   AGE
```

```
coredns-6d56c8448f-4rvnr                 1/1     Running   0          10m
coredns-6d56c8448f-9mn9h                 1/1     Running   0          10m
etcd-k8s-master                          1/1     Running   0          10m
kube-apiserver-k8s-master                1/1     Running   0          10m
kube-controller-manager-k8s-master       1/1     Running   0          10m
kube-flannel-ds-amd64-2h859              1/1     Running   0          2m23s
kube-flannel-ds-amd64-6gqct              1/1     Running   0          2m23s
kube-flannel-ds-amd64-rvljf              1/1     Running   0          2m23s
kube-proxy-bxh2m                         1/1     Running   0          8m28s
kube-proxy-s7jxv                         1/1     Running   0          10m
kube-proxy-xdr5v                         1/1     Running   0          8m30s
kube-scheduler-k8s-master                1/1     Running   0          10m
```

已经是 ready 状态

2.8、部署测试应用

所有 node 主机导入测试镜像

```
[root@k8s-node01 ~]# docker load < nginx-1.19.tar
[root@k8s-node01 ~]# docker tag nginx nginx:1.19.6
```

在 Kubernetes 集群中创建一个 pod，验证是否正常运行。

```
[root@k8s-master ~]# mkdir demo
[root@k8s-master ~]# cd demo
[root@k8s-master demo]# vim nginx-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.19.6
```

```
        ports:
        - containerPort: 80
```

创建完 Deployment 的资源清单之后，使用 create 执行资源清单来创建容器。通过 get pods 可以查看到 Pod 容器资源已经自动创建完成。

```
[root@k8s-master demo]# kubectl create -f nginx-deployment.yaml
deployment.apps/nginx-deployment created

[root@k8s-master demo]# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-76ccf9dd9d-c4vjm   1/1     Running   0          10s
nginx-deployment-76ccf9dd9d-r7cqs   1/1     Running   0          10s
nginx-deployment-76ccf9dd9d-vfkhs   1/1     Running   0          10s

[root@k8s-master demo]# kubectl get pods -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP
NODE           NOMINATED NODE   READINESS GATES
nginx-deployment-76ccf9dd9d-c4vjm   1/1     Running   0          28s     10.244.2.4
k8s-node02     <none>           <none>
nginx-deployment-76ccf9dd9d-r7cqs   1/1     Running   0          28s     10.244.1.3
k8s-node01     <none>           <none>
nginx-deployment-76ccf9dd9d-vfkhs   1/1     Running   0          28s     10.244.2.3
k8s-node02     <none>           <none>
```

创建 Service 资源清单

在创建的 nginx-service 资源清单中，定义名称为 nginx-service 的 Service、标签选择器为 app: nginx、type 为 NodePort 指明外部流量可以访问内部容器。在 ports 中定义暴露的端口库号列表，对外暴露访问的端口是 80，容器内部的端口也是 80。

```
[root@k8s-master demo]# vim nginx-service.yaml
kind: Service
apiVersion: v1
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  type: NodePort
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80

[root@k8s-master1 demo]# kubectl create -f nginx-service.yaml
service/nginx-service created
```

```
[root@k8s-master demo]# kubectl get svc
NAME            TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
kubernetes      ClusterIP   10.96.0.1       <none>        443/TCP        11m
nginx-service   NodePort    10.96.160.246   <none>        80:32188/TCP   8s
```

通过浏览器访问 nginx：http://192.168.200.111:32188

3、部署 Prometheus 监控平台

3.1、准备 Prometheus 相关 YAML 文件

在 master 节点/opt 目录下新建 pgmonitor 目录

```
[root@k8s-master ~]# mkdir /opt/pgmonitor
[root@k8s-master ~]# cd /opt/pgmonitor
```

将下载 yaml 包上传至/opt/pgmonitor 目录并解压

```
[root@k8s-master ~]# unzip k8s-prometheus-grafana-master.zip
```

3.2、部署 prometheus

部署守护进程

```
[root@k8s-master pgmonitor]# cd k8s-prometheus-grafana-master/
[root@k8s-master k8s-prometheus-grafana-master]# kubectl create -f node-exporter.yaml
daemonset.apps/node-exporter created
service/node-exporter created
```

部署其他 yaml 文件

进入/opt/pgmonitor/k8s-prometheus-grafana-master/prometheus 目录

```
[root@k8s-master k8s-prometheus-grafana-master]# cd prometheus
```

部署 rbac

```
[root@k8s-master prometheus]# kubectl create -f rbac-setup.yaml
clusterrole.rbac.authorization.k8s.io/prometheus created
```

```
serviceaccount/prometheus created
clusterrolebinding.rbac.authorization.k8s.io/prometheus created
```

部署 configmap.yaml

```
[root@k8s-master prometheus]# kubectl create -f configmap.yaml
configmap/prometheus-config created
```

部署 prometheus.deploy.yml

```
[root@k8s-master prometheus]# kubectl create -f prometheus.deploy.yml
deployment.apps/prometheus created
```

部署 prometheus.svc.yml

```
[root@k8s-master prometheus]# kubectl create -f prometheus.svc.yml
service/prometheus created
```

查看 prometheus 状态

```
[root@k8s-master prometheus]# kubectl get pods -n kube-system
NAME                                    READY   STATUS    RESTARTS   AGE
coredns-7f89b7bc75-khfv4                1/1     Running   0          14m
coredns-7f89b7bc75-w7j6w                1/1     Running   0          14m
etcd-k8s-master                         1/1     Running   0          14m
kube-apiserver-k8s-master               1/1     Running   0          14m
kube-controller-manager-k8s-master      1/1     Running   0          14m
kube-flannel-ds-amd64-c6hgm             1/1     Running   0          13m
kube-flannel-ds-amd64-kf8nh             1/1     Running   0          13m
kube-flannel-ds-amd64-n7f5k             1/1     Running   0          13m
kube-proxy-cg4p6                        1/1     Running   0          13m
kube-proxy-mjtrc                        1/1     Running   0          14m
kube-proxy-n5q5f                        1/1     Running   0          13m
kube-scheduler-k8s-master               1/1     Running   0          14m
node-exporter-k6h88                     1/1     Running   0          81s
node-exporter-zkqm7                     1/1     Running   0          81s
prometheus-68546b8d9-p9kzn              1/1     Running   0          55s
```

4、部署 Grafana 服务

4.1、部署 Grafana 相关 yaml 文件

进入/opt/pgmonitor/k8s-prometheus-grafana-master/grafana 目录

```
[root@k8s-master prometheus]# cd ../grafana/
```

部署 grafana-deploy.yaml

```
[root@k8s-master grafana]# kubectl create -f grafana-deploy.yaml
deployment.apps/grafana-core created
```

部署 grafana-svc.yaml

```
[root@k8s-master grafana]# kubectl create -f grafana-svc.yaml
service/grafana created
```

部署 grafana-ing.yaml

```
[root@k8s-master grafana]# kubectl create -f grafana-ing.yaml
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in
v1.22+; use networking.k8s.io/v1
Ingressingress.extensions/grafana created
```

查看 Grafana 状态

```
[root@k8s-master grafana]# kubectl get pods -n kube-system
NAME                                   READY   STATUS    RESTARTS   AGE
coredns-7f89b7bc75-khfv4               1/1     Running   0          15m
coredns-7f89b7bc75-w7j6w               1/1     Running   0          15m
etcd-k8s-master                        1/1     Running   0          16m
grafana-core-6d6fb7566-8ft9g           1/1     Running   0          76s
kube-apiserver-k8s-master              1/1     Running   0          16m
kube-controller-manager-k8s-master     1/1     Running   0          16m
kube-flannel-ds-amd64-c6hgm            1/1     Running   0          14m
kube-flannel-ds-amd64-kf8nh            1/1     Running   0          14m
kube-flannel-ds-amd64-n7f5k            1/1     Running   0          14m
kube-proxy-cg4p6                       1/1     Running   0          15m
kube-proxy-mjtrc                       1/1     Running   0          15m
kube-proxy-n5q5f                       1/1     Running   0          15m
kube-scheduler-k8s-master              1/1     Running   0          16m
node-exporter-k6h88                    1/1     Running   0          3m2s
node-exporter-zkqm7                    1/1     Running   0          3m2s
prometheus-68546b8d9-p9kzn             1/1     Running   0          2m36s
```

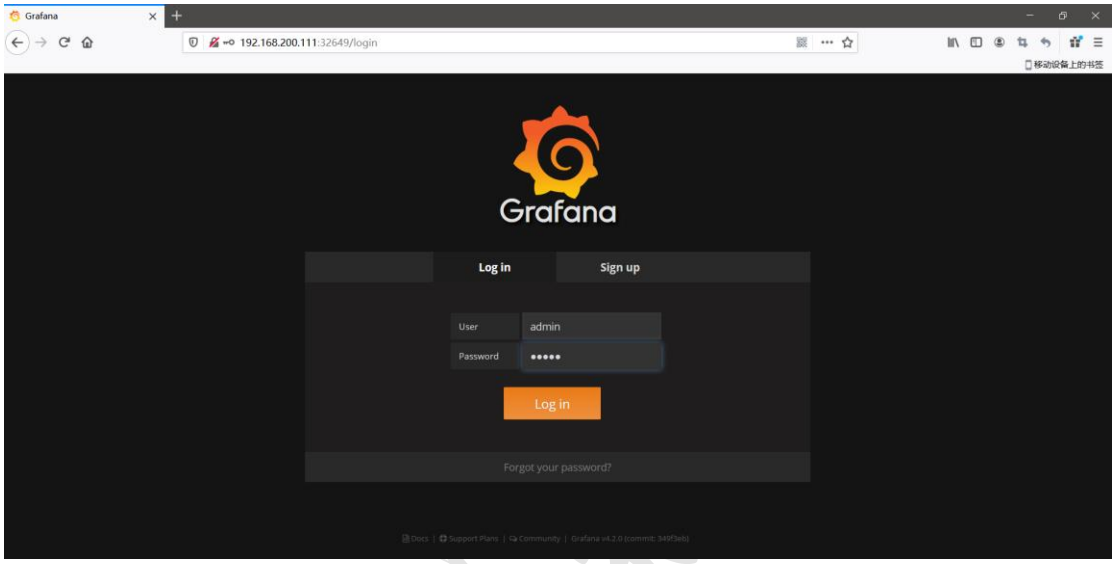
4.2、配置 Grafana 数据源

查看 grafana 的端口

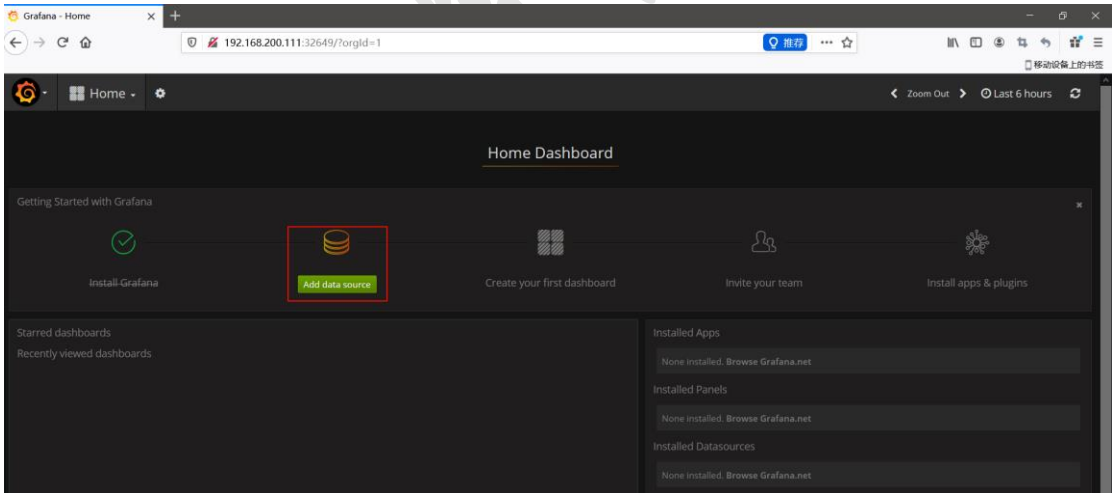
```
[root@k8s-master grafana]# kubectl get svc -n kube-system
NAME                   TYPE           CLUSTER-IP          EXTERNAL-IP     PORT(S)
AGE
grafana                NodePort       10.106.188.105      <none>          3000:32649/TCP
106s
kube-dns               ClusterIP      10.96.0.10          <none>          53/UDP,53/TCP,9153/TCP
```

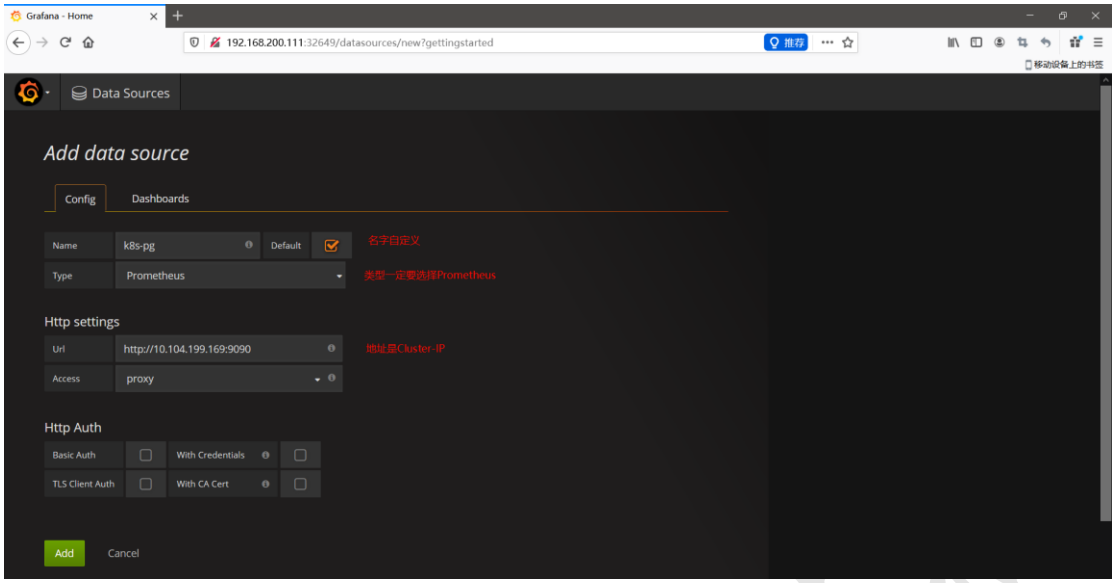
```
16m
node-exporter   NodePort   10.101.150.79   <none>   9100:31672/TCP
3m36s
prometheus      NodePort   10.104.199.169  <none>   9090:30003/TCP
3m4s
```

通过浏览器访问 grafana，http://[masterIP]:[grafana 端口]
例如：http://192.168.200.111:32649，默认的用户名和密码：admin/admin

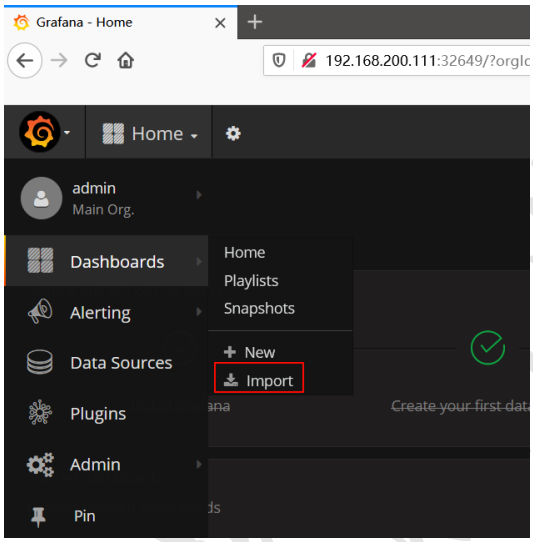


设置 DataSource

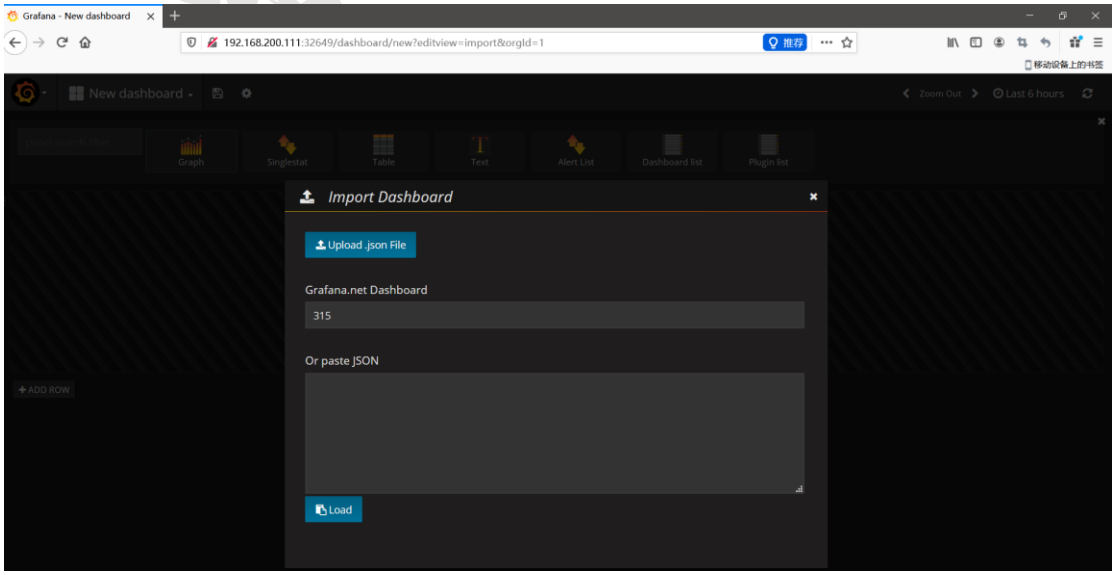




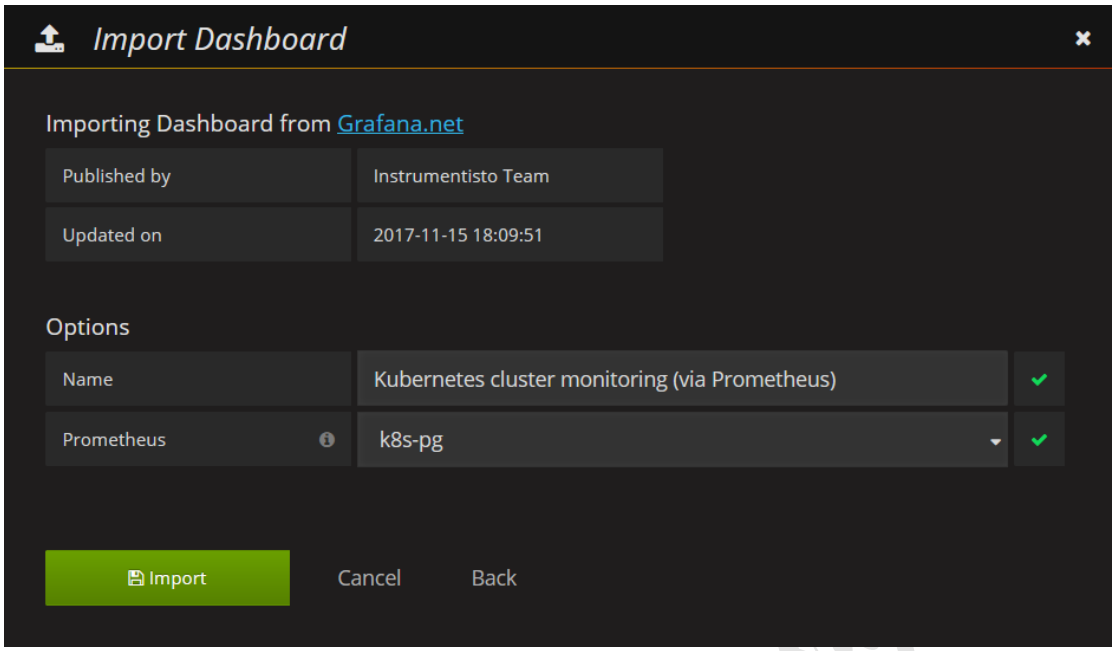
设置显示数据的模版

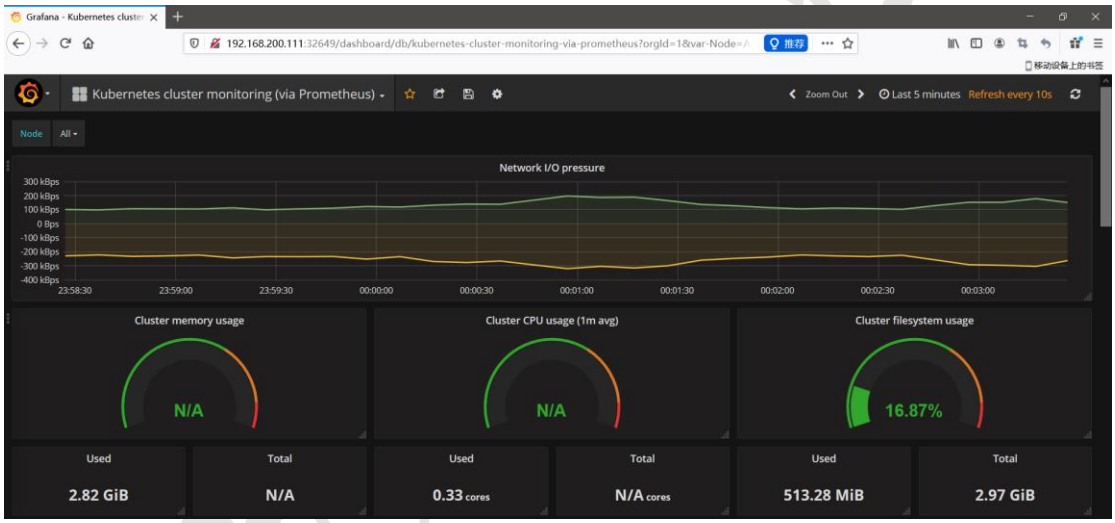


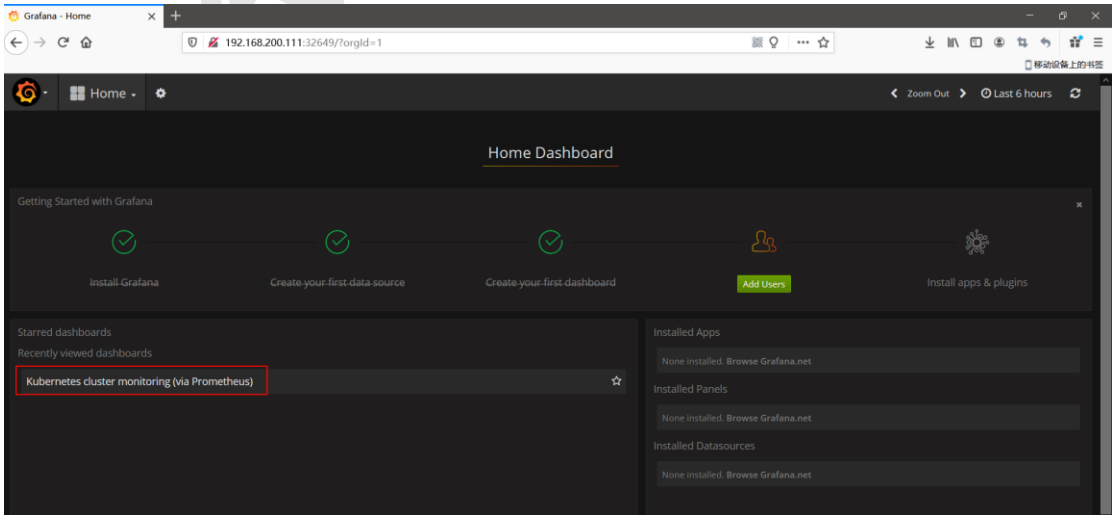
输入 315 并移除光标，等一会儿即可进入下一个页面



选择自己定义的数据源名称







至此已经利用 Prometheus+Granfana 监控了 Kubernetes 平台。