# K8S之监控etcd集群（自带metrics接口）

# 云原生应用之监控etcd集群

## 一、监控etcd集群

### 1.1、查看接口信息

```
[root@k8s-master01 ~]# curl --cert /etc/etcd/ssl/etcd.pem --key
/etc/etcd/ssl/etcd-key.pem  https://192.168.1.201:2379/metrics -k
# 这样也行
curl -L http://localhost:2379/metrics
```

### 1.2、创建service和Endpoints

```
# 创建ep和svc代理外部的etcd服务，其他自带metrics接口的服务也是如此!
apiVersion: v1
kind: Endpoints
metadata:
  labels:
    app: etcd-k8s
  name: etcd-k8s
  namespace: kube-system
subsets:
- addresses:     # etcd节点对应的主机ip，有几台就写几台
  - ip: 192.168.1.110
  - ip: 192.168.1.111
  - ip: 192.168.1.112
  ports:
  - name: etcd-port
    port: 2379   # etcd端口
    protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: etcd-k8s
  name: etcd-k8s
  namespace: kube-system
spec:
  ports:
  - name: etcd-port # svc 代理pod的端口自定义name
    port: 2379
    protocol: TCP
    targetPort: 2379
  type: ClusterIP
```

### 1.3、测试是否代理成功

```
#再次curl，把IP换成svc的IP测试，输出相同内容即创建成功
[root@k8s-master01 ~]# kubectl get svc -n kube-system etcd-k8s
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
etcd-ep   ClusterIP   10.103.53.103   <none>        2379/TCP   8m54s

# 再次请求接口
[root@k8s-master01 ~]# curl --cert /etc/etcd/ssl/etcd.pem --key
/etc/etcd/ssl/etcd-key.pem  https://10.111.200.116:2379/metrics -k
```

## 1.4、创建secret

```
# 1、这里我们k8s-master01节点进行创建,ca为k8sca证书，剩下2个为etcd证书，这是我证书所在位置
  cert-file: '/etc/kubernetes/pki/etcd/etcd.pem'
  key-file: '/etc/kubernetes/pki/etcd/etcd-key.pem'
  trusted-ca-file: '/etc/kubernetes/pki/etcd/etcd-ca.pem'

# 2、接下来我们需要创建一个secret，让prometheus pod节点挂载
kubectl create secret generic etcd-ssl --from-
file=/etc/kubernetes/pki/etcd/etcd-ca.pem --from-
file=/etc/kubernetes/pki/etcd/etcd.pem --from-
file=/etc/kubernetes/pki/etcd/etcd-key.pem -n monitoring

# 3、创建完成后可以检查一下
[root@k8s-master01 prometheus-down]# kubectl describe secrets -n monitoring
etcd-ssl
Name:         etcd-ssl
Namespace:    monitoring
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
etcd-ca.pem:   1367 bytes
etcd-key.pem:  1679 bytes
etcd.pem:      1509 bytes
```

## 1.5、编辑prometheus，把证书挂载进去

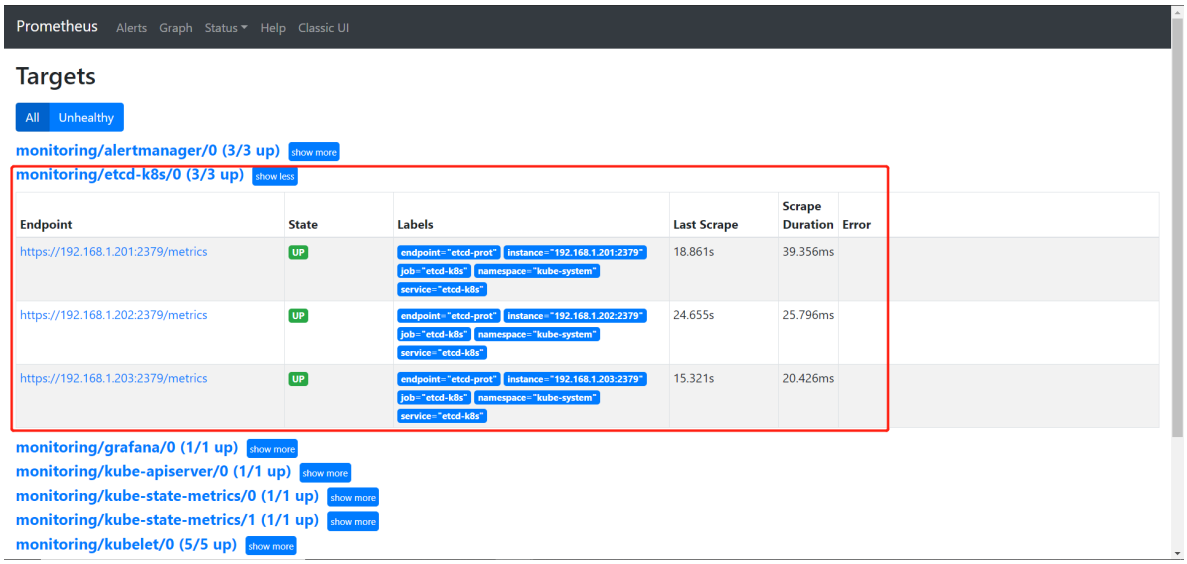
```
# 1、通过edit直接编辑prometheus
[root@k8s-master01 ~]# kubectl edit prometheus k8s -n monitoring
# 在replicas底下加上secret名称
replicas:2
secrets:
- etcd-ssl #添加secret名称

# 进入容器查看，就可以看到证书挂载进去了
[root@k8s-master01 prometheus-down]# kubectl exec -it -n monitoring prometheus-
k8s-0 /bin/sh

# 查看文件是否存在
/prometheus $ ls /etc/prometheus/secrets/etcd-ssl/
etcd-ca.pem   etcd-key.pem  etcd.pem
```

## 1.6、创建ServiceMonitor

```
[root@k8s-master01 ~]# cat etcd-servicemonitor.yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: etcd-k8s
  namespace: monitoring
  labels:
    app: etcd-k8s
spec:
  jobLabel: app
  endpoints:
    - interval: 30s
      port: etcd-port  # 这个port对应 Service.spec.ports.name
      scheme: https
      # 证书位置
      tlsConfig:
        caFile: /etc/prometheus/secrets/etcd-ssl/etcd-ca.pem #证书路径（在
prometheus pod里路径)
        certFile: /etc/prometheus/secrets/etcd-ssl/etcd.pem
        keyFile: /etc/prometheus/secrets/etcd-ssl/etcd-key.pem
        insecureSkipVerify: true  # 关闭证书校验
  selector:
    matchLabels:
      app: etcd-k8s  # 跟scv的lables保持一致
  namespaceSelector:
    matchNames:
    - kube-system    # 跟svc所在namespace保持一致
# 匹配Kube-system这个命名空间下面具有app=etcd-k8s这个label标签的Serve，job label用于检
索job任务名称的标签。由于证书serverName和etcd中签发的证书可能不匹配，所以添加了
insecureSkipVerify=true将不再对服务端的证书进行校验
```

## 1.7、页面查看三个etcd节点都获取到数据

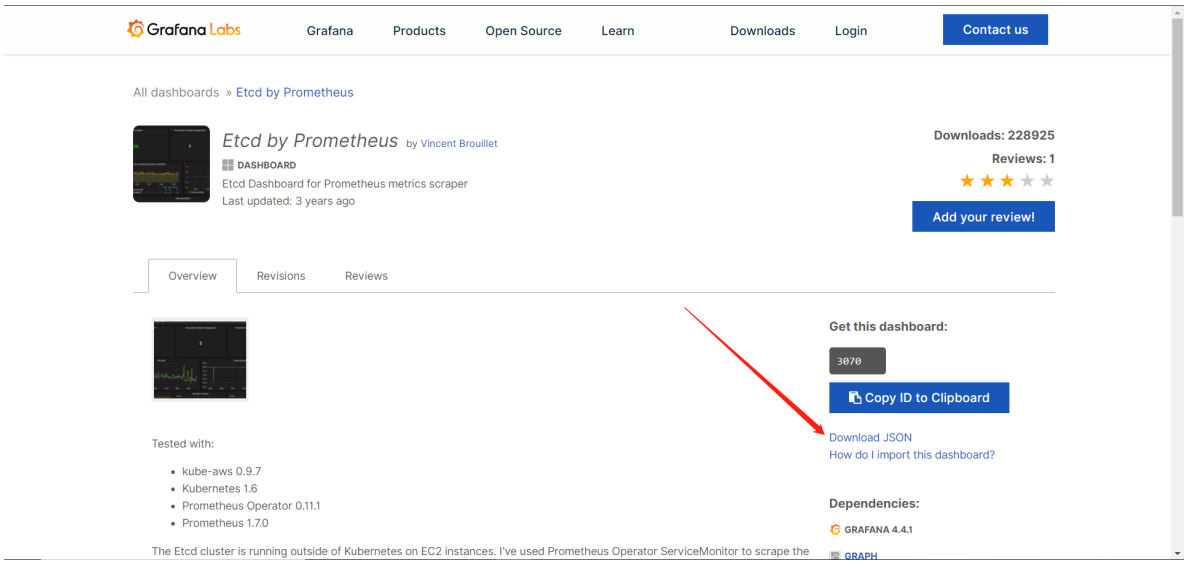
**此处数据获取有点慢，需要等待一下**



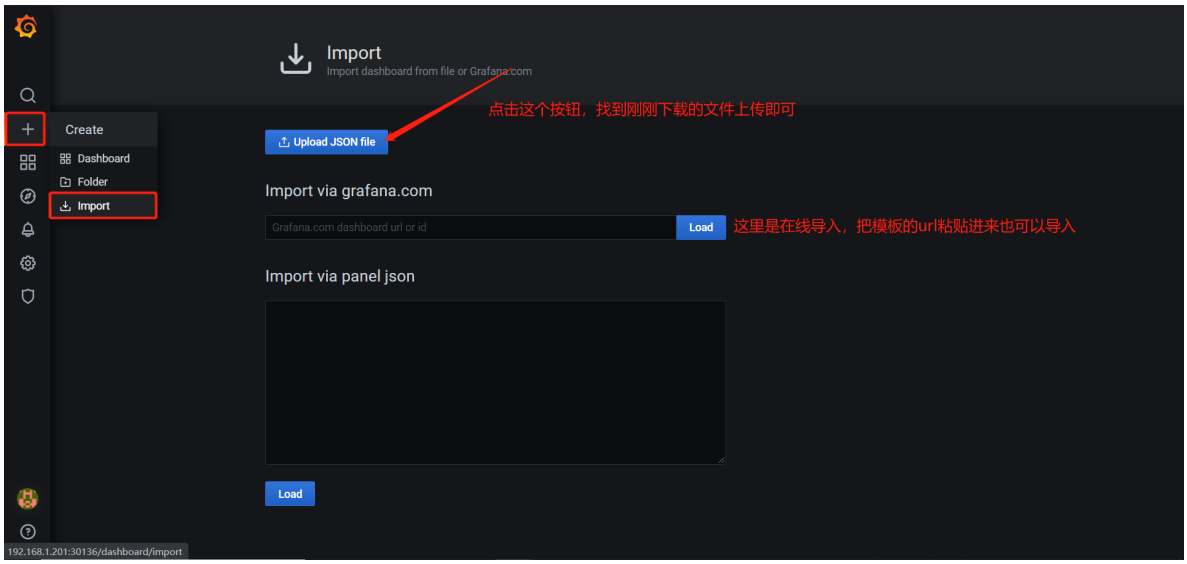

## 1.8、grafana模板导入

**数据采集完成后，接下来可以在grafana中导入dashboard**

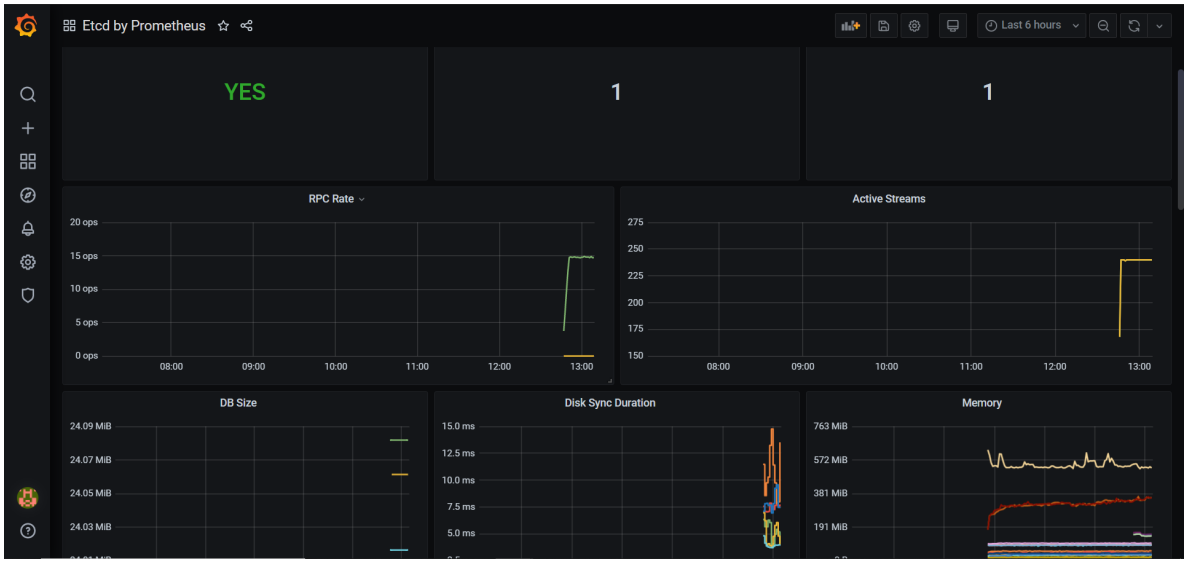
```
# 打开官网来的如下图所示，点击下载JSO文件
grafana官网：https://grafana.com/grafana/dashboards/3070
中文版ETCD集群插件：https://grafana.com/grafana/dashboards/9733
```




## 点击HOME->导入模板




## 导入后页面展示


Etcd by Prometheus
Last 6 hours
YES
1
1
RPC Rate
20 ops
15 ops
10 ops
5 ops
0 ops
08:00
09:00
10:00
11:00
12:00
13:00
Active Streams
275
250
225
200
175
150
08:00
09:00
10:00
11:00
12:00
13:00
DB Size
24.09 MiB
24.07 MiB
24.05 MiB
24.03 MiB
Disk Sync Duration
15.0 ms
12.5 ms
10.0 ms
7.5 ms
5.0 ms
Memory
763 MiB
572 MiB
381 MiB
191 MiB