# 监控kube-scheduler 【kube-controller-manager 改下端口即可】

## 一、kube-scheduler接入k8s中

- 因为是二进制部署,所以kube-scheduler不是部署在k8s上，但是kube-scheduler属于原生产物，自带/metres接口
- 将kube-scheduler接入k8s中的方案是创建ep、svc

### 1.1、查看kube-scheduler使用的端口

```
[root@k8s-master01 ~]# ss -ntlp | grep kube-scheduler
LISTEN     0      16384  127.0.0.1:10251                    *:*
  users:(("kube-scheduler",pid=32418,fd=7))
```

### 1.2、修改kube-scheduler端口监听的IP

- 如果是高可用的，三个节点都需要修改

```
# 修改监听路径
[root@k8s-master01 ~]# sed -i  's/address=127.0.0.1/address=0.0.0.0/'
/usr/lib/systemd/system/kube-scheduler.service

# 重启
[root@k8s-master01 ~]# systemctl daemon-reload && systemctl restart  kube-
scheduler
```

### 1.3、创建ServiceMonitor

- 我们都知道Prometheus需要通过ServiceMonitor形式去监控，但是kube-Prometheus部署已经帮我们准备好

```
[root@k8s-master01 ~]# kubectl get servicemonitors -n monitoring kube-scheduler
NAME             AGE
kube-scheduler   7d6h
```

### 1.4、创建endpoint、service

- Service Momitor标签得匹配上应用的标签
- 所以我们自定义的SVC的labels、ns、协议、端口是跟kube-Prometheus部署ServiceMonitor得保持一致
- labels、ns保持一致是关联起来，协议、端口一致是为了获取到数据

```
[root@k8s-master01 ~]# kubectl get servicemonitors -n monitoring kube-scheduler
-oyaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  creationTimestamp: "2022-09-12T08:24:18Z"
  generation: 1
  labels:
```

```
    k8s-app: kube-scheduler
  name: kube-scheduler
  namespace: monitoring
spec:
  endpoints:
  - bearerTokenFile: /var/run/secrets/kubernetes.io/serviceaccount/token
    interval: 30s
    port: https-metrics          # 注意点
    scheme: https                # 注意点
    tlsConfig:
      insecureSkipVerify: true
  jobLabel: k8s-app              # 注意点
  namespaceSelector:
    matchNames:
    - kube-system                # 注意点
  selector:
    matchLabels:
      k8s-app: kube-scheduler    # 注意点
```

- 创建endpoint、service

```
[root@k8s-master01 kube-controller-manager-监控]# cat kube-scheduler.yaml
apiVersion: v1
kind: Endpoints
metadata:
  labels:
    k8s-app: kube-scheduler
  name: kube-scheduler-monitor
  namespace: kube-system
subsets:
  - addresses:
    - ip: 192.168.1.110  # 改成master01宿主机的ip
    - ip: 192.168.1.111  # 改成master02宿主机的ip
    - ip: 192.168.1.112  # 改成master03宿主机的ip
    ports:
    - name: http-metrics
      port: 10251
      protocol: TCP
---
apiVersion: v1
kind: Service
metadata:
  labels:
    k8s-app: kube-scheduler
  name: kube-scheduler-monitor
  namespace: kube-system
spec:
  ports:
  - name: http-metrics
    port: 10251
    protocol: TCP
    targetPort: 10251
  sessionAffinity: None
  type: ClusterIP
```

### 1.5、协议保持一致

- 上面创建的svc是https、但是ServiceMonitor是https，所以此处有2种方案

**方案一：ServiceMonitor改http 【本案例使用此种方案】**

```
[root@k8s-master01 ~]# kubectl edit servicemonitors -n monitoring kube-scheduler
...
...
    # 更改了2个地方
    port: http-metrics
    scheme: http
...
...
```

**方案二：改https【参考监控etcd的】**

```
## 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
```

```
## 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
```

**1.7、确认通过kube-scheduler SVC能够访问程序的Metrics接口**

```
[root@k8s-master01 ~]# kubectl get svc -n kube-system kube-scheduler-monitor
NAME                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kube-scheduler-monitor     ClusterIP   10.107.197.207   <none>        10251/TCP
35m


# 确认通过kube-scheduler  SVC能够访问程序的Metrics接口  【如果curl 3次有不能访问就是kube-scheduler监听的IP忘改】
[root@k8s-master01 ~]# curl 10.107.197.207:10251/metrics
```

## 1.8、页面查看是否监控成功

- 看到以下数据说明监控是成功