# 监控失败排查步骤

- 1、确认Service Momitor是否创建成功
- 2、确认Service Momitor标签是否配置正确
- 3、确认Prometheus是否生成了相关配置
- 4、确认存在Service Monitor匹配的Service
- 5、确认通过SVC能够访问程序的Metrics接口
- 6、确认SVC的端口跟Scheme和Service Monitor一致

## 一、以kube-controller-manager为例子

- 二进制部署的k8s+kube-Prometheus部署的监控默认没监控上
- 这也是监控kube-controller-manager的步骤了，同理kube-scheduler改一下端口即可

### 1、确认Service Momitor是否创建成功

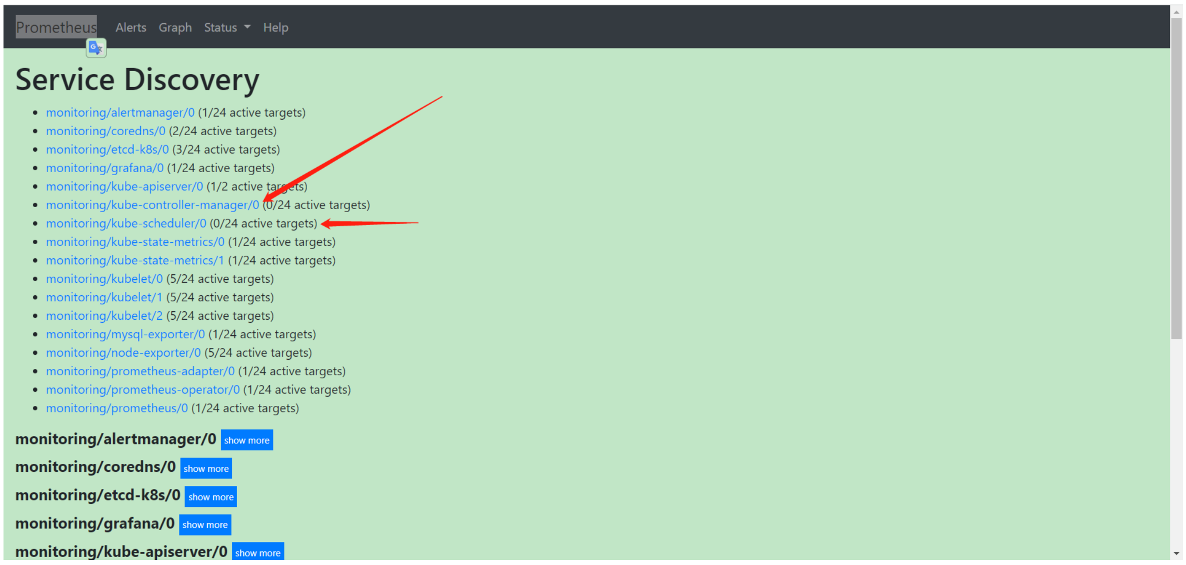
```
[root@k8s-master01 ~]# kubectl get servicemonitors -n monitoring | grep kube-controller-manager
kube-controller-manager   6d4h
```

### 2、确认Service Momitor标签是否配置正确

```
[root@k8s-master01 ~]# kubectl get servicemonitors -n monitoring kube-controller-manager  -oyaml | grep -A 3  ...
...
...
  selector:
    matchLabels:
      k8s-app: kube-controller-manager
...
```

### 3、确认Prometheus是否生成了相关配置

```
# 访问Prometheus的ui
http://192.168.1.110:30062/service-discovery
```



Service Discovery

- monitoring/alertmanager/0 (1/24 active targets)
- monitoring/coredns/0 (2/24 active targets)
- monitoring/etcd-k8s/0 (3/24 active targets)
- monitoring/grafana/0 (1/24 active targets)
- monitoring/kube-apiserver/0 (1/2 active targets)
- monitoring/kube-controller-manager/0 (0/24 active targets)
- monitoring/kube-scheduler/0 (0/24 active targets)
- monitoring/kube-state-metrics/0 (1/24 active targets)
- monitoring/kube-state-metrics/1 (1/24 active targets)
- monitoring/kubelet/0 (5/24 active targets)
- monitoring/kubelet/1 (5/24 active targets)
- monitoring/kubelet/2 (5/24 active targets)
- monitoring/mysql-exporter/0 (1/24 active targets)
- monitoring/node-exporter/0 (5/24 active targets)
- monitoring/prometheus-adapter/0 (1/24 active targets)
- monitoring/prometheus-operator/0 (1/24 active targets)
- monitoring/prometheus/0 (1/24 active targets)

monitoring/alertmanager/0 show more

monitoring/coredns/0 show more

monitoring/etcd-k8s/0 show more

monitoring/grafana/0 show more

monitoring/kube-apiserver/0 show more

## 4、确认存在Service Monitor匹配的Service

```
[root@k8s-master01 ~]# kubectl get servicemonitors -n monitoring kube-
controller-manager  -oyaml | grep -A 3  ...
...
...
  namespaceSelector:                            # 名称空间看namespaceSelector
    matchNames:
    - kube-system
  selector:
    matchLabels:
      k8s-app: kube-controller-manager          # 匹配的是有这个标签svc,名称空间看
namespaceSelector
...

# 发现没有所以需要创建
[root@k8s-master01 ~]# kubectl get svc -n kube-system  -l  k8s-app=kube-
controller-manager
No resources found in kube-system namespace.

# 创建ep+svc
[root@k8s-master01 kube-controller-manager-监控]# cat kube-controller-
manager.yaml
apiVersion: v1
kind: Endpoints
metadata:
  labels:
    k8s-app: kube-controller-manager
  name: kube-controller-manage-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: 10252
      protocol: TCP
```

```
---
apiVersion: v1
kind: Service
metadata:
  labels:
    k8s-app: kube-controller-manager
  name: kube-controller-manage-monitor
  namespace: kube-system
spec:
  ports:
  - name: http-metrics
    port: 10252
    protocol: TCP
    targetPort: 10252
  sessionAffinity: None
  type: ClusterIP
```

## 5、确认通过kube-controller-manager SVC能够访问程序的Metrics接口

```
# 先确定kube-controller-manager服务是ok的
curl -k 127.0.0.1:10252/metrics

# 注意如果你的kube-controller-manager是监听在127.0.0.1上需要改成0.0.0.0
[root@k8s-master01 ~]# ss -ntlp | grep 10252
LISTEN    0   16384  127.0.0.1:10252      *:*     users:(("kube-
controller",pid=63861,fd=7))
# 更改方法【 --address=0.0.0.0】
sed -i  's/address=127.0.0.1/address=0.0.0.0/' /usr/lib/systemd/system/kube-
controller-manager.service
systemctl daemon-reload && systemctl restart kube-controller-manager



# 查看svc
[root@k8s-master01 kube-controller-manager-监控]# kubectl get svc -n kube-system
kube-controller-manage-monitor
NAME                               TYPE        CLUSTER-IP     EXTERNAL-IP
PORT(S)     AGE
kube-controller-manage-monitor   ClusterIP   10.108.144.3   <none>
 10252/TCP   7m15s

# 再确认通过SVC能够访问程序的Metrics接口
curl -k 10.108.144.3:10252/metrics
```

## 6、确认kube-controller-manager SVC的端口跟Scheme和Service Monitor一致

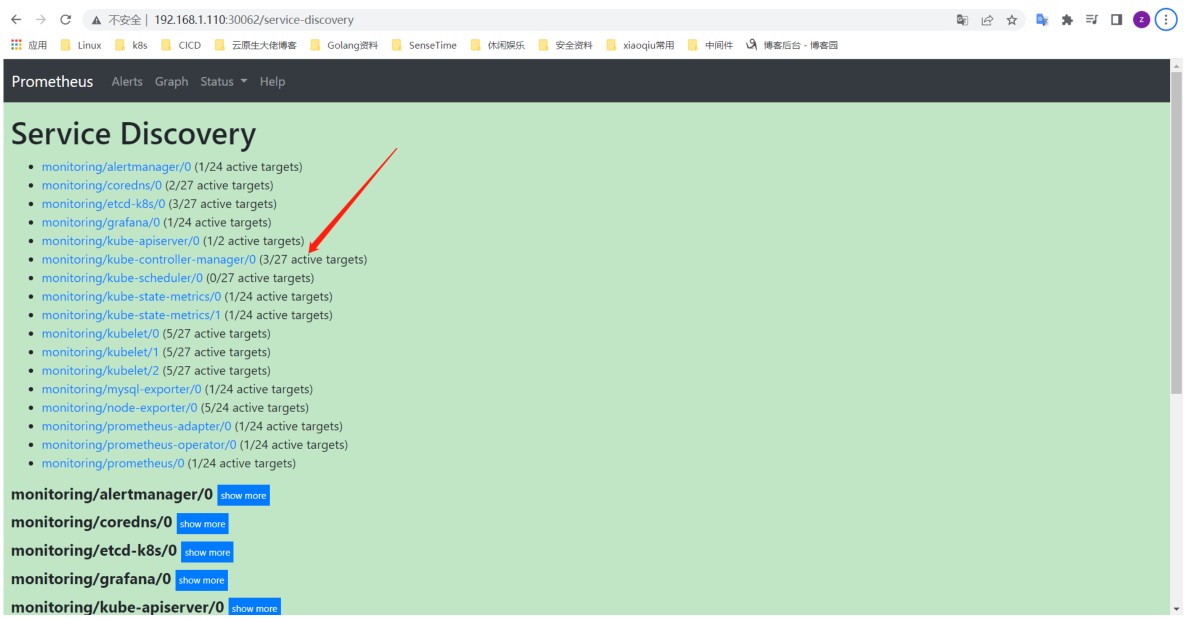
```
[root@k8s-master01 ~]# kubectl get svc -n kube-system kube-controller-manage-
monitor -oyaml | grep  -A 3 ports
--
  ports:
  - name: http-metrics
    port: 10252
    protocol: TCP

# 此处不一样，所以要么跟etcd一样把证书挂载进去，要么修改kube-controller-manager的servicemonitors的port name，还有协议scheme一致
[root@k8s-master01 ~]# kubectl get servicemonitors -n monitoring  kube-controller-manager -oyaml | grep port -A 1
    port: https-metrics
    scheme: https

# 直接edit
[root@k8s-master01 ~]# kubectl edit servicemonitors -n monitoring  kube-controller-manager
[root@k8s-master01 ~]# kubectl get servicemonitors -n monitoring  kube-controller-manager -oyaml | grep port -A 2
    port: http-metrics
    scheme: http
```

## 7、页面查看是否监控成功

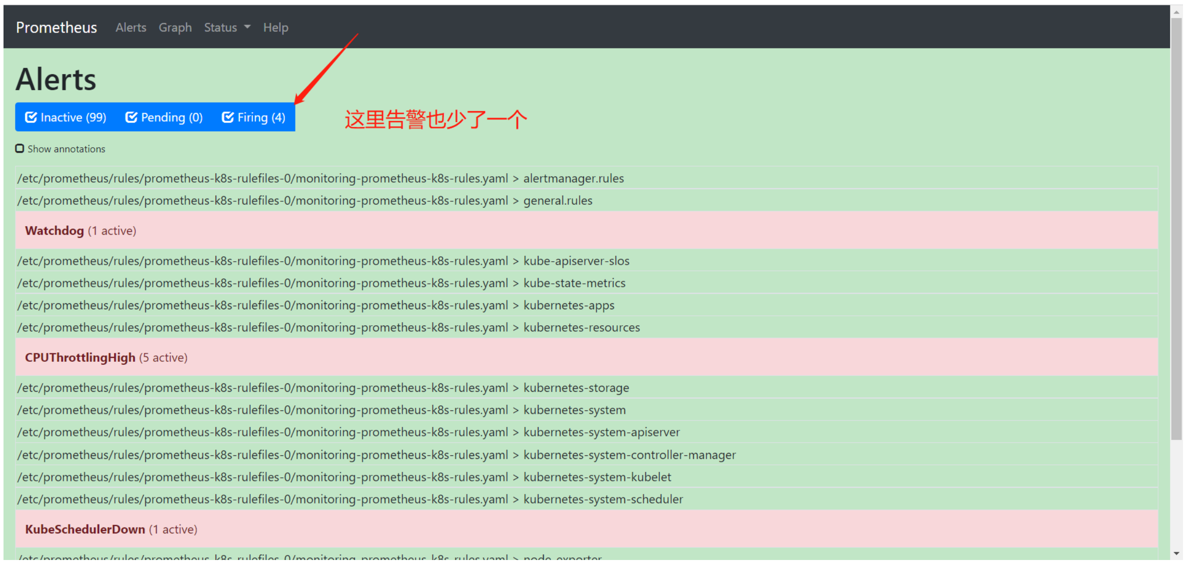
- 看到以下数据说明监控是成功的



Service Discovery

- monitoring/alertmanager/0 (1/24 active targets)
- monitoring/coredns/0 (2/27 active targets)
- monitoring/etcd-k8s/0 (3/27 active targets)
- monitoring/grafana/0 (1/24 active targets)
- monitoring/kube-apiserver/0 (1/2 active targets)
- monitoring/kube-controller-manager/0 (3/27 active targets)
- monitoring/kube-scheduler/0 (0/27 active targets)
- monitoring/kube-state-metrics/0 (1/24 active targets)
- monitoring/kube-state-metrics/1 (1/24 active targets)
- monitoring/kubelet/0 (5/27 active targets)
- monitoring/kubelet/1 (5/27 active targets)
- monitoring/kubelet/2 (5/27 active targets)
- monitoring/mysql-exporter/0 (1/24 active targets)
- monitoring/node-exporter/0 (5/24 active targets)
- monitoring/prometheus-adapter/0 (1/24 active targets)
- monitoring/prometheus-operator/0 (1/24 active targets)
- monitoring/prometheus/0 (1/24 active targets)

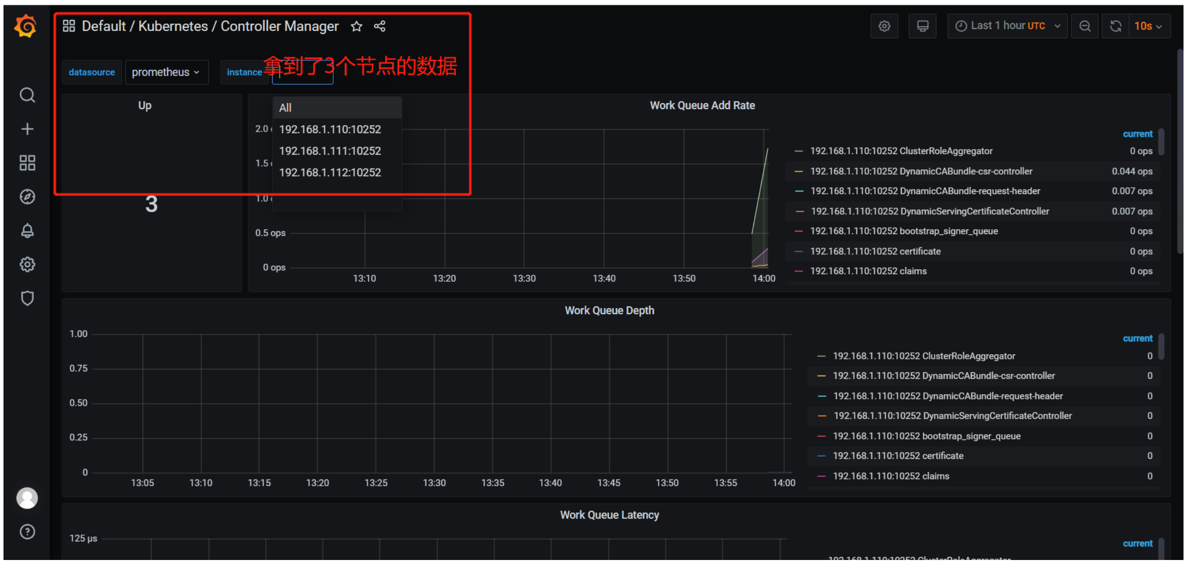
monitoring/alertmanager/0 show more
monitoring/coredns/0 show more
monitoring/etcd-k8s/0 show more
monitoring/grafana/0 show more
monitoring/kube-apiserver/0 show more



Alerts

这里告警也少了一个

/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > alertmanager.rules
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > general.rules
Watchdog (1 active)
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kube-apiserver-slos
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kube-state-metrics
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kubernetes-apps
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kubernetes-resources
CPUThrottlingHigh (5 active)
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kubernetes-storage
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kubernetes-system
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kubernetes-system-apiserver
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kubernetes-system-controller-manager
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kubernetes-system-kubelet
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > kubernetes-system-scheduler
KubeSchedulerDown (1 active)
/etc/prometheus/rules/prometheus-k8s-rulefiles-0/monitoring-prometheus-k8s-rules.yaml > node-exporter



Default / Kubernetes / Controller Manager

Last 1 hour UTC    10s

datasource prometheus    instance

拿到了3个节点的数据

All
192.168.1.110:10252
192.168.1.111:10252
192.168.1.112:10252

Up: 3

Work Queue Add Rate

| | current |
|---|---|
| 192.168.1.110:10252 ClusterRoleAggregator | 0 ops |
| 192.168.1.110:10252 DynamicCABundle-csr-controller | 0.044 ops |
| 192.168.1.110:10252 DynamicCABundle-request-header | 0.007 ops |
| 192.168.1.110:10252 DynamicServingCertificateController | 0.007 ops |
| 192.168.1.110:10252 bootstrap_signer_queue | 0 ops |
| 192.168.1.110:10252 certificate | 0 ops |
| 192.168.1.110:10252 claims | 0 ops |

Work Queue Depth

| | current |
|---|---|
| 192.168.1.110:10252 ClusterRoleAggregator | 0 |
| 192.168.1.110:10252 DynamicCABundle-csr-controller | 0 |
| 192.168.1.110:10252 DynamicCABundle-request-header | 0 |
| 192.168.1.110:10252 DynamicServingCertificateController | 0 |
| 192.168.1.110:10252 bootstrap_signer_queue | 0 |
| 192.168.1.110:10252 certificate | 0 |
| 192.168.1.110:10252 claims | 0 |

Work Queue Latency

current