Kubernetes 集群状态异常排错

本章介绍集群状态异常的排错方法，包括 Kubernetes 主要组件以及必备扩展（如 kube-dns）等，而有关网络的异常排错请参考网络异常排错方法。

概述

排查集群状态异常问题通常从 Node 和 Kubernetes 服务 的状态出发，定位出具体的异常服务，再进而寻找解决方法。集群状态异常可能的原因比较多，常见的有

- 虚拟机或物理机宕机
- 网络分区
- Kubernetes 服务未正常启动
- 数据丢失或持久化存储不可用（一般在公有云或私有云平台中）
- 操作失误（如配置错误）

按照不同的组件来说，具体的原因可能包括

- kube-apiserver 无法启动会导致
  - 集群不可访问
  - 已有的 Pod 和服务正常运行（依赖于 Kubernetes API 的除外）
- etcd 集群异常会导致
  - kube-apiserver 无法正常读写集群状态，进而导致 Kubernetes API 访问出错
  - kubelet 无法周期性更新状态
- kube-controller-manager/kube-scheduler 异常会导致
  - 复制控制器、节点控制器、云服务控制器等无法工作，从而导致 Deployment、Service 等无法工作，也无法注册新的 Node 到集群中来
  - 新创建的 Pod 无法调度（总是 Pending 状态）
- Node 本身宕机或者 Kubelet 无法启动会导致
  - Node 上面的 Pod 无法正常运行
  - 已在运行的 Pod 无法正常终止

为了维持集群的健康状态，推荐在部署集群时就考虑以下

- 在云平台上开启 VM 的自动重启功能
- 为 Etcd 配置多节点高可用集群，使用持久化存储（如 AWS EBS 等），定期备份数据
- 为控制平面配置高可用，比如多 kube-apiserver 负载均衡以及多节点运行 kube-controller-manager、kube-scheduler 以及 kube-dns 等
- 尽量使用复制控制器和 Service，而不是直接管理 Pod
- 跨地域的多 Kubernetes 集群

查看 Node 状态

一般来说，可以首先查看 Node 的状态，确认 Node 本身是不是 Ready 状态

```
kubectl get nodes
kubectl describe node <node-name>
```

如果是 NotReady 状态，则可以执行 `kubectl describe node <node-name>` 命令来查看当前 Node 的事件。这些事件通常都会有助于排查 Node 发生的问题。

SSH 登录 Node

在排查 Kubernetes 问题时，通常需要 SSH 登录到具体的 Node 上面查看 kubelet、docker、iptables 等的状态和日志。在使用云平台时，可以给相应的 VM 绑定一个公网 IP；而在物理机部署时，可以通过路由器上的端口映射来访问。但更简单的方法是使用 SSH Pod （不要忘记替换成你自己的 nodeName）：

```
# cat ssh.yaml
apiVersion: v1
kind: Service
metadata:
  name: ssh
spec:
  selector:
    app: ssh
  type: LoadBalancer
  ports:
  - protocol: TCP
    port: 22
    targetPort: 22
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: ssh
  labels:
    app: ssh
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ssh
  template:
    metadata:
      labels:
        app: ssh
    spec:
      containers:
      - name: alpine
        image: alpine
        ports:
        - containerPort: 22
        stdin: true
        tty: true
      hostNetwork: true
      nodeName: <node-name>

$ kubectl create -f ssh.yaml
$ kubectl get svc ssh
NAME      TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)         AGE
ssh       LoadBalancer   10.0.99.149   52.52.52.52   22:32008/TCP   5m
```

接着，就可以通过 ssh 服务的外网 IP 来登录 Node，如 `ssh user@52.52.52.52`。

在使用完后， 不要忘记删除 SSH 服务 `kubectl delete -f ssh.yaml`。

查看日志

一般来说，Kubernetes 的主要组件有两种部署方法

- 直接使用 systemd 等启动控制节点的各个服务
- 使用 Static Pod 来管理和启动控制节点的各个服务

使用 systemd 等管理控制节点服务时，查看日志必须要首先 SSH 登录到机器上，然后查看具体的日志文件。如

```
journalctl -l -u kube-apiserver
journalctl -l -u kube-controller-manager
journalctl -l -u kube-scheduler
journalctl -l -u kubelet
journalctl -l -u kube-proxy
```

或者直接查看日志文件

- /var/log/kube-apiserver.log
- /var/log/kube-scheduler.log
- /var/log/kube-controller-manager.log
- /var/log/kubelet.log
- /var/log/kube-proxy.log

而对于使用 Static Pod 部署集群控制平面服务的场景，可以参考下面这些查看日志的方法。

kube-apiserver 日志

```
PODNAME=$(kubectl -n kube-system get pod -l component=kube-apiserver -o
jsonpath='{.items[0].metadata.name}')
kubectl -n kube-system logs $PODNAME --tail 100
```

kube-controller-manager 日志

```
PODNAME=$(kubectl -n kube-system get pod -l component=kube-controller-manager -o
jsonpath='{.items[0].metadata.name}')
kubectl -n kube-system logs $PODNAME --tail 100
```

kube-scheduler 日志

```
PODNAME=$(kubectl -n kube-system get pod -l component=kube-scheduler -o
jsonpath='{.items[0].metadata.name}')
kubectl -n kube-system logs $PODNAME --tail 100
```

kube-dns 日志

```
PODNAME=$(kubectl -n kube-system get pod -l k8s-app=kube-dns -o
jsonpath='{.items[0].metadata.name}')
kubectl -n kube-system logs $PODNAME -c kubedns
```

Kubelet 日志

查看 Kubelet 日志需要首先 SSH 登录到 Node 上。

```
journalctl -l -u kubelet
```

Kube-proxy 日志

Kube-proxy 通常以 DaemonSet 的方式部署

```
$ kubectl -n kube-system get pod -l component=kube-proxy
NAME               READY     STATUS    RESTARTS   AGE
kube-proxy-42zpn   1/1       Running   0          1d
kube-proxy-7gd4p   1/1       Running   0          3d
kube-proxy-87dbs   1/1       Running   0          4d
$ kubectl -n kube-system logs kube-proxy-42zpn
```

Kube-dns/Dashboard CrashLoopBackOff

由于 Dashboard 依赖于 kube-dns，所以这个问题一般是由于 kube-dns 无法正常启动导致的。查看 kube-dns 的日志

```
$ kubectl logs --namespace=kube-system $(kubectl get pods --namespace=kube-system -l k8s-app=kube-
dns -o name) -c kubedns
$ kubectl logs --namespace=kube-system $(kubectl get pods --namespace=kube-system -l k8s-app=kube-
dns -o name) -c dnsmasq
$ kubectl logs --namespace=kube-system $(kubectl get pods --namespace=kube-system -l k8s-app=kube-
dns -o name) -c sidecar
```

可以发现如下的错误日志

```
Waiting for services and endpoints to be initialized from apiserver...
skydns: failure to forward request "read udp 10.240.0.18:47848->168.63.129.16:53: i/o timeout"
Timeout waiting for initialization
```

这说明 kube-dns pod 无法转发 DNS 请求到上游 DNS 服务器。解决方法为

- 如果使用的 Docker 版本大于 1.12，则在每个 Node 上面运行 `iptables -P FORWARD ACCEPT`
- 等待一段时间，如果还未恢复，则检查 Node 网络是否正确配置，比如是否可以正常请求上游DNS服务器、是否有安全组禁止了 DNS 请求等

如果错误日志中不是转发 DNS 请求超时，而是访问 kube-apiserver 超时，比如

```
E0122 06:56:04.774977       1 reflector.go:199] k8s.io/dns/vendor/k8s.io/client-go/tools/cache/reflector.go:94: Failed to list *v1.Endpoints: Get https://10.0.0.1:443/api/v1/endpoints?resourceVersion=0: dial tcp 10.0.0.1:443: i/o timeout
I0122 06:56:04.775358       1 dns.go:174] Waiting for services and endpoints to be initialized from apiserver...
E0122 06:56:04.775574       1 reflector.go:199] k8s.io/dns/vendor/k8s.io/client-go/tools/cache/reflector.go:94: Failed to list *v1.Service: Get https://10.0.0.1:443/api/v1/services?resourceVersion=0: dial tcp 10.0.0.1:443: i/o timeout
I0122 06:56:05.275295       1 dns.go:174] Waiting for services and endpoints to be initialized from apiserver...
I0122 06:56:05.775182       1 dns.go:174] Waiting for services and endpoints to be initialized from apiserver...
I0122 06:56:06.275288       1 dns.go:174] Waiting for services and endpoints to be initialized from apiserver...
```

这说明 Pod 网络（一般是多主机之间）访问异常，包括 Pod->Node、Node->Pod 以及 Node-Node 等之间的往来通信异常。可能的原因比较多，具体的排错方法可以参考网络异常排错指南。

Kubelet: failed to initialize top level QOS containers

重启 kubelet 时报错 `Failed to start ContainerManager failed to initialise top level QOS containers`（参考 #43856），临时解决方法是：

1. 在 docker.service 配置中增加 `--exec-opt native.cgroupdriver=systemd` 选项。
2. 重启主机

该问题已于2017年4月27日修复（v1.7.0+， #44940）。更新集群到新版本即可解决这个问题。

Kubelet 一直报 FailedNodeAllocatableEnforcement 事件

当 NodeAllocatable 特性未开启时（即 kubelet 设置了 `--cgroups-per-qos=false`），查看 node 的事件会发现每分钟都会有 `Failed to update Node Allocatable Limits` 的警告信息：

```
$ kubectl describe node node1
Events:
  Type     Reason                             Age                 From                                                Message
  ----     ------                             ----                ----                                                -------
  Warning  FailedNodeAllocatableEnforcement  2m (x1001 over 16h)  kubelet, aks-agentpool-22604214-0  Failed to update Node Allocatable Limits "": failed to set supported cgroup subsystems for cgroup : Failed to set config for supported subsystems : failed to write 7285047296 to memory.limit_in_bytes: write /var/lib/docker/overlay2/5650a1aadf9c758946073fefa1558446ab582148ddd3ee7e7cb9d269fab20f72/merged/sys/f: invalid argument
```

如果 NodeAllocatable 特性确实不需要，那么该警告事件可以忽略。但根据 Kubernetes 文档 Reserve Compute Resources for System Daemons，最好开启该特性：

> Kubernetes nodes can be scheduled to `Capacity`. Pods can consume all the available capacity on a node by default. This is an issue because nodes typically run quite a few system daemons that power the OS and Kubernetes itself. Unless resources are set aside for these system daemons, pods and system daemons compete for resources and lead to resource starvation issues on the node.
>
> The `kubelet` exposes a feature named `Node Allocatable` that helps to reserve compute resources for system daemons. Kubernetes recommends cluster administrators to configure `Node Allocatable` based on their workload density on each node.
>
> Node Capacity
> ---

```
| kube-reserved |
|-----------------------|
| system-reserved |
|-----------------------|
| eviction-threshold |
|-----------------------|
| |
| allocatable |
| (available for pods) |
| |
| |
--------------------------
```

开启方法为：

```
kubelet --cgroups-per-qos=true --enforce-node-allocatable=pods ...
```

Kube-proxy: error looking for path of conntrack

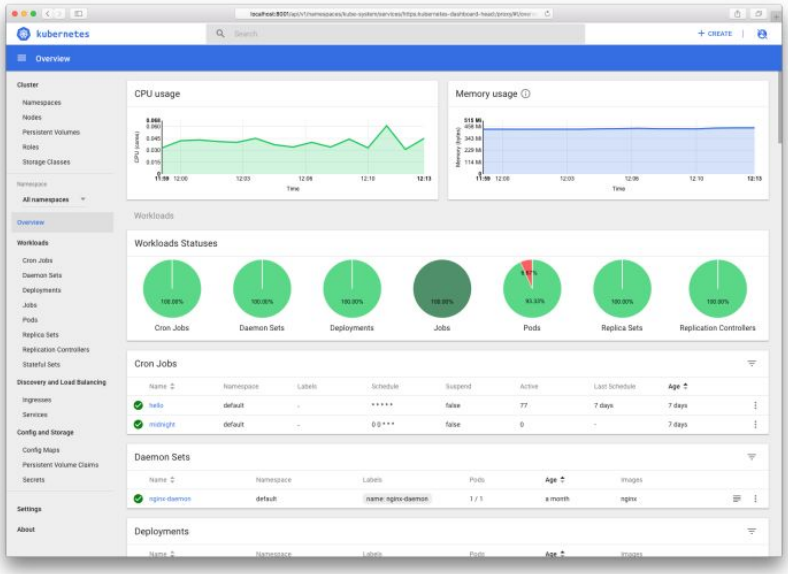
kube-proxy 报错，并且 service 的 DNS 解析异常

```
kube-proxy[2241]: E0502 15:55:13.889842    2241 conntrack.go:42] conntrack returned error: error looking for path of conntrack: exec: "conntrack": executable file not found in $PATH
```

解决方式是安装 `conntrack-tools` 包后重启 kube-proxy 即可。

Dashboard 中无资源使用图表

正常情况下，Dashboard 首页应该会显示资源使用情况的图表，如



如果没有这些图表，则需要首先检查 Heapster 是否正在运行（因为Dashboard 需要访问 Heapster 来查询资源使用情况）：

```
kubectl -n kube-system get pods -l k8s-app=heapster
NAME                        READY     STATUS    RESTARTS   AGE
heapster-86b59f68f6-h4vt6   2/2       Running   0          5d
```

如果查询结果为空，说明 Heapster 还未部署，可以参考 https://github.com/kubernetes/heapster 来部署。

但如果 Heapster 处于正常状态，那么需要查看 dashboard 的日志，确认是否还有其他问题

```
$ kubectl -n kube-system get pods -l k8s-app=kubernetes-dashboard
NAME                                  READY     STATUS    RESTARTS   AGE
kubernetes-dashboard-665b4f7df-dsjpn   1/1       Running   0          5d

$ kubectl -n kube-system logs kubernetes-dashboard-665b4f7df-dsjpn
```

HPA 不自动扩展 Pod

查看 HPA 的事件，发现

```
$ kubectl describe hpa php-apache
Name:                                                  php-apache
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Wed, 27 Dec 2017 14:36:38 +0800
Reference:                                             Deployment/php-apache
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  <unknown> / 50%
Min replicas:                                          1
Max replicas:                                          10
Conditions:
  Type           Status  Reason                   Message
  ----           ------  ------                   -------
  AbleToScale    True    SucceededGetScale        the HPA controller was able to get the target's current scale
  ScalingActive  False   FailedGetResourceMetric  the HPA was unable to compute the replica count: unable to get metrics for resource cpu: unable to fetch metrics from API: the server could not find the requested resource (get pods.metrics.k8s.io)
Events:
  Type     Reason                   Age                 From                       Message
  ----     ------                   ----                ----                       -------
  Warning  FailedGetResourceMetric  3m (x2231 over 18h)  horizontal-pod-autoscaler  unable to get metrics for resource cpu: unable to fetch metrics from API: the server could not find the requested resource (get pods.metrics.k8s.io)
```

这说明 metrics-server 未部署，可以参考 这里 部署。

参考文档

- Troubleshoot Clusters
- SSH into Azure Container Service (AKS) cluster nodes
- Kubernetes dashboard FAQ

编辑于 2018-03-20

Kubernetes

Docker

容器

下一篇Kubernetes 云平台异常排错