Initial Troubleshooting Guide

Regular maintenance for your Kubernetes cluster and applications are the key to ensuring there are no unexpected complications when running your containers, but when problems do arise, it can be tough to find the best methods to find the root of the problem. This guide will outline common troubleshooting commands and tools that customers can use to investigate problems on a cluster. See the Upstream Standardized Glossary for more information about the terminology that follows in this guide.

Cluster and Node Troubleshooting

A Kubernetes cluster is typically built from a number of nodes of different usage types: an etcd store, the control plane, and the workers. These node types typically lead to two architectural designs:

- **Stacked deployment:** an etcd and control plane are co-located on the same node
- **External etcd**: a separate etcd and control plane cluster

Regardless of architectural choice, the etcd cluster is never actually managed by Kubernetes and therefore can only be viewed, managed, and diagnosed using its own tooling. The following paragraphs will explain what to look for to ensure your cluster is running properly.

ETCD

To diagnose and health-check an etcd cluster, ssh to one of the nodes where the etcd containers are running and issue the following command, ensuring you replace the `ETCD_TAG` (the image version tag) and `HOST0` (the etcd host IP) variables:

```
docker run --rm -it --net host  \
-v /etc/kubernetes:/etc/kubernetes \
quay.io/coreos/etcd:${ETCD_TAG} etcdctl \
--cert-file /etc/kubernetes/pki/etcd/peer.crt \
--key-file /etc/kubernetes/pki/etcd/peer.key \
--ca-file /etc/kubernetes/pki/etcd/ca.crt \
--endpoints https://${HOST0}:2379 cluster-health
```

If all is well, the cluster should reply with `cluster is healthy`. Further debugging can be done by finding the running container ID with the command `docker ps -a` and viewing the logs of the etcd container with `docker logs -f <ID>`. Contact us for the commands to diagnose and health-check your etcd cluster if you are using one of our other supported container runtimes.

Control Plane

The control plane nodes are dependent on the etcd cluster for state information. In the event that the control plane is unresponsive, ensure the health of the etcd nodes.

Most connectivity issues are caused by a misconfiguration of kubectl. There are simple steps you can use to ensure proper configuration:

1. Ensure that `~/.kube` contains the configuration.
2. Make sure you're running as the correct user. If you've swapped user with `su` or `sudo,` you may be looking in the wrong place.
3. Verify that the correct context is being used (for the correct cluster) with the command `kubectl config get-contexts`

All Nodes

Underlying the operation of the nodes are two items: the container runtime (such as Docker or containerd), and the kubelet agent. For simplicity's sake, we will assume you're using Docker.

If the **kubelet** agent isn't running, then the Kubernetes platform won't be able to speak and manage the node as part of the Kubernetes cluster. In the event that the kubelet is stopped, it can be restarted with the command: `systemctl restart kubelet`. In the event that the kubelet fails to start, we recommend debugging the issue by using the `journalctl -xeu kubelet` or `/var/log/` command to pull relevant service logs. Typically, the messages or syslog log files will report the reasons why the kubelet has failed to start.

If the **container runtime** isn't running, then Kubernetes won't be able to run a number of its components such as kube-proxy and the CNI plugin (e.g. Calico, Weave, etc.). It also won't be able to schedule any workloads to run as part of the Kubernetes cluster. The container runtime can be restarted with `systemctl start docker`. In the event that the engine fails to start, we recommend debugging the issue by inspecting the logs located in `/var/log/`. Typically the messages or syslog log files will report the reasons why the engine has failed to start.

If the engine is up and running, then another system failure may be stopping the node from starting containers. Ensure that the underlying filesystem isn't full or that nothing is wrong with any of the required containers by finding them with a `docker ps -a` and viewing their logs with `docker logs -f <container_ID>`.

Kubernetes Troubleshooting

Management of the Kubernetes cluster is typically done through the `kubectl` command line interface (from one of the masters). In order to ensure that all nodes of the cluster are up and healthy, use the following command:

```
kubectl get nodes
NAME           STATUS   ROLES     AGE          VERSION
manager01   Ready       master    147d    v1.10.3
manager02   Ready       master    147d    v1.10.3
manager03   Ready       master    147d    v1.10.3
worker01        Ready   <none>    147d    v1.10.3
```

In the event that a node is unavailable, quickly diagnose the node with the command

`kubectl describe node <node name>`

and consult the Cluster/Node management section above.

Application Troubleshooting

Once an application has been deployed on a healthy cluster or node, it can be exposed as a Service for either internal consumption or for external access. However, in some situations, this access may fail. The following section contains tips to debug an application that is running on Kubernetes and has failed to provide access.

For this scenario, imagine a deployment of:

- 5 pods exposing HTTP on port 80
- A Service exposing a Service name that uses the 5 Pods as endpoints

First, ensure that everything is up and running. This can be achieved by querying the Kubernetes cluster for the expected resources:

Service Check:

```
kubectl get/describe svc <svc name>
```

Pod Check:

```
kubectl get pods
```

If the Service is defined, check that it has been configured correctly by attempting to ping its Service name from a container in the cluster.

```
$ ping <svc_name>
```

If this name resolves to an IP address, then it means that the Service has been both created and registered by the internal Kubernetes DNS.

We will be adding information to this guide as needed. If you have any troubleshooting tips you'd like us to feature, let us know by opening a ticket on our Support Portal.

Quick Links

What is VMware CRE?

How to open a ticket

Kubernetes Learning Resources