Das Ablauf eines kuberneten Zertifikats verursacht die Unterbrechung der clusterweiten Kommunikation.

Inhalt

Einführung

Dieses Dokument beschreibt ein mögliches Ausfälle, mit dem Kunden konfrontiert sein könnten, wenn sie ein kubernetes System besitzen, das seit mehr als 365 Tagen installiert ist. Darüber hinaus durchläuft er die Schritte, die zur Behebung der Situation und zur Sicherung und Inbetriebnahme des kuberneten Systems erforderlich sind.

Problem

Nach einem Jahr nach der vordefinierten installierten Kubernet-Cluster laufen die Client-Zertifikate ab. Sie können nicht auf die Cisco CloudCenter Suite (CCS) zugreifen. Obwohl es noch angezeigt wird, können Sie sich nicht anmelden. Wenn Sie zur kubectl-CLI navigieren, sehen Sie die folgende Fehlermeldung: "Es konnte keine Verbindung zum Server hergestellt werden: x509: das Zertifikat ist abgelaufen oder noch nicht gültig."

Sie können dieses Bash-Skript ausführen, um das Ablaufdatum der Zertifikate anzuzeigen:

```
for crt in /etc/kubernetes/pki/*.crt; do
     printf '%s: %s\n' \
     "$(date --date="$(openssl x509 -enddate -noout -in "$crt"|cut -d= -f 2)" --iso-8601)" \
     "$crt"
done | sort
```

Sie können auch einen Open-Source-Workflow für Action Orchestrator finden, der diesen täglichen Vorgang überwacht und ihn auf Probleme aufmerksam macht.

https://github.com/cisco-cx-workflows/cx-ao-shared-workflows/tree/master/CCSCheckKubernetesExpiration__definition_workflow_01E01VIRWZDE24mWIsHrqCGB9xUix0f9ZxG

Lösung

Neue Zertifikate müssen über Kubeadm im gesamten Cluster neu ausgegeben werden und dann müssen Sie die Arbeiterknoten wieder den Kapitänen anschließen.

1. Melden Sie sich bei einem Master-Knoten an.

2. Die IP-Adresse wird **über ip address show angezeigt**.

```
[root@cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a3 kubernetes]# ip address show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 8920 qdisc pfifo_fast state UP group default
qlen 1000
link/ether fa:16:3e:19:63:a2 brd ff:ff:ff:ff:ff:ff
inet 192.168.1.20/24 brd 192.168.1.255 scope global dynamic eth0
valid_lft 37806sec preferred_lft 37806sec
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group
default
link/ether 02:42:d0:29:ce:5e brd ff:ff:ff:ff:ff:ff
inet 172.17.0.1/16 scope global docker0
valid_lft forever preferred_lft forever
13: tunl0@NONE: <NOARP,UP,LOWER_UP> mtu 1430 qdisc noqueue state UNKNOWN group default qlen
1000
link/ipip 0.0.0.0 brd 0.0.0.0
inet 172.16.176.128/32 brd 172.16.176.128 scope global tunl0
valid_lft forever preferred_lft forever
14: cali65453a0219d@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1430 qdisc noqueue state UP
group default
link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netnsid 4
```

3. Navigieren Sie über **cd /etc/kuberettes** zum kuberneten Verzeichnis.

4. Erstellen Sie eine Datei mit dem Namen **kubeadmCERT.yaml** über **vi kubeadmCERT.yaml**.

5. Die Datei sollte wie folgt aussehen:

```
apiVersion: kubeadm.k8s.io/v1alpha1
kind: MasterConfiguration
api:
  advertiseAddress: <IP ADDRESS FROM STEP 2>
kubernetesVersion: v1.11.6
#NOTE: If the customer is running a load balancer VM then you must add these lines after...
#apiServerCertSANs:
#- <load balancer IP>
```

6. Sichern Sie Ihre alten Zertifikate und Schlüssel. Dies ist nicht erforderlich, wird jedoch empfohlen. Erstellen Sie ein Sicherungsverzeichnis, und kopieren Sie diese Dateien darauf.

```
#Files
#apiserver.crt
#apiserver.key
#apiserver-kubelet-client.crt
#apiserver-kubelet-client.key
#front-proxy-client.crt
#front-proxy-client.key

#ie
cd /etc/kubernetes/pki
mkdir backup
mv apiserver.key backup/apiserver.key.bak
```

7. Wenn Sie Schritt 6 übersprungen haben, können Sie die oben genannten Dateien einfach mit dem Befehl rm löschen, wie **rm apiserver.crt**.

8. Navigieren Sie zurück zu der Datei **kubeadmCERT.yaml**. Erstellen Sie ein neues apiserver cert via **kubeadm —config kubeadmCERT.yaml alpha phase certs apiserver**.

```
[root@cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a3 kubernetes]# kubeadm --
config kubeadmCERT.yaml alpha phase certs apiserver
[certificates] Generated apiserver certificate and key.
[certificates] apiserver serving cert is signed for DNS names [cx-ccs-prod-master-d7f34f25-
f524-4f90-9037-7286202ed13a3 kubernetes kubernetes.default kubernetes.default.svc
kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.1.20]
```

9. Neue apiserver kubelet cert via **kubeadm —config kubeadmCERT.yaml alpha phase certs apiserver-kubelet-client**.

10. Generieren Sie neue Front-Proxy-Client cert via **kubeadm —config kubeadmCERT.yaml alpha Phase certs front-proxy-client**.

11. Sichern Sie im Ordner **/etc/kuberetted** die **.conf-Dateien**. Nicht erforderlich, aber empfohlen. Sie sollten über **kubelet.conf**, **controller-manager.conf**, **Scheduler.conf** und möglicherweise **admin.conf** verfügen. Sie können sie löschen, wenn Sie sie nicht sichern möchten.

12. Erstellen Sie neue Konfigurationsdateien über **kubeadm —config kubeadmCERT.yaml alpha Phase kubeconfig all**.

```
[root@cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a3 kubernetes]# kubeadm --
config kubeadmCERT.yaml alpha phase kubeconfig all
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.conf"
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"
```

13. Exportieren Sie die neue **admin.conf-**Datei auf Ihren Host.

```
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
chown $(id -u):$(id -g) $HOME/.kube/config
chmod 777 $HOME/.kube/config
export KUBECONFIG=.kube/config
```

14. Starten Sie den Master-Knoten **jetzt** über **shutdown -r neu**.

15. Nachdem der Master gesichert wurde, überprüfen Sie, ob kubelet über **systemctl status kubelet** ausgeführt wird.

16. Überprüfen Sie Kubernetes über **kubectl get-Knoten**.

```
[root@cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a3 ~]# kubectl get nodes
NAME                                                  STATUS     ROLES     AGE
VERSION
cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a1   Ready      master    1y
v1.11.6
```

```
cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a2   Ready      master    1y
v1.11.6
cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a3   Ready      master    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a1   NotReady   <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a2   NotReady   <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a3   NotReady   <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a4   NotReady   <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a5   NotReady   <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a6   NotReady   <none>    1y
v1.11.6
```

17. Wiederholen Sie die Schritte 1. auf 16. für jeden Master-Knoten.

18. Generieren Sie auf einem Master ein neues Join-Token über **kubeadm token create —print-join-command**. Kopieren Sie diesen Befehl zur späteren Verwendung.

```
[root@cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a1 k8s-mgmt]# kubeadm token
create
--print-join-command kubeadm join 192.168.1.14:6443 --token m1ynvj.f4n3et3poki88ry4
--discovery-token-ca-cert-hash
sha256:4d0c569985c1d460ef74dc01c85740285e4af2c2369ff833eed1ba86e1167575
```

19. Holen Sie sich die IPs Ihrer Mitarbeiter über **kubectl get-Knoten - oder breit**.

20. Melden Sie sich bei einem Mitarbeiter an, z. B. **ssh -i /home/cloud-user/keys/gen3-ao-prod.key cloud-user@192.168.1.17**, und navigieren Sie zum Stammzugriff.

21. Beenden Sie den Dienst kubelet über **systemctl stop kubelet**.

22. Entfernen Sie die alten Konfigurationsdateien, darunter **ca.crt**, **kubelet.conf** und **bootstrap-kubelet.conf**.

```
rm /etc/kubernetes/pki/ca.crt
rm /etc/kubernetes/kubelet.conf
rm /etc/kubernetes/bootstrap-kubelet.conf
```

23. Nennen Sie den Knoten aus Schritt 19.

24. Geben Sie den Befehl für den Mitarbeiter aus, um wieder dem Cluster beizutreten. Verwenden Sie den Befehl von 18., aber fügen Sie **—node-name <name des Knotens>** zum Ende hinzu.

```
[root@cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a1 kubernetes]# kubeadm join
192.168.1.14:6443 --token m1ynvj.f4n3et3poki88ry4 --discovery-token-ca-cert-hash
sha256:4d0c569985c1d460ef74dc01c85740285e4af2c2369ff833eed1ba86e1167575 --node-name cx-
ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a1
[preflight] running pre-flight checks
```

```
        [WARNING RequiredIPVSKernelModulesAvailable]: the IPVS proxier will not be used,
because the following required kernel modules are not loaded: [ip_vs_rr ip_vs_wrr
ip_vs_sh] or no builtin kernel ipvs support: map[ip_vs:{} ip_vs_rr:{} ip_vs_wrr:{}
ip_vs_sh:{} nf_conntrack_ipv4:{}]
you can solve this problem with following methods:
 1. Run 'modprobe -- ' to load missing kernel modules;
2. Provide the missing builtin kernel ipvs support

I0226 17:59:52.644282   19170 kernel_validator.go:81] Validating kernel version
I0226 17:59:52.644421   19170 kernel_validator.go:96] Validating kernel config
[discovery] Trying to connect to API Server "192.168.1.14:6443"
[discovery] Created cluster-info discovery client, requesting info from
"https://192.168.1.14:6443"
[discovery] Requesting info from "https://192.168.1.14:6443" again to validate TLS against
the pinned public key
[discovery] Cluster info signature and contents are valid and TLS certificate validates
against pinned roots, will use API Server "192.168.1.14:6443"
[discovery] Successfully established connection with API Server "192.168.1.14:6443"
[kubelet] Downloading configuration for the kubelet from the "kubelet-config-1.11"
ConfigMap in the kube-system namespace
[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-
flags.env"
[preflight] Activating the kubelet service
[tlsbootstrap] Waiting for the kubelet to perform the TLS Bootstrap...
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node
API object "cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a1" as an annotation

This node has joined the cluster:
* Certificate signing request was sent to master and a response
  was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the master to see this node join the cluster.
```

25. Beenden Sie den Mitarbeiter und überprüfen Sie den Status auf einem Master über **kubectl get-Knoten**. Er sollte den Status "Bereit" aufweisen.

26. Wiederholen Sie die Schritte 20. auf 25. für jeden Mitarbeiter.

27. Die letzten **kubectl get-Knoten** sollten anzeigen, dass alle Knoten den Status "Ready" haben, wieder online sind und mit dem Cluster verbunden sind.

```
[root@cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a1 ~]# kubectl get nodes
NAME                                                  STATUS    ROLES     AGE
VERSION
cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a1   Ready     master    1y
v1.11.6
cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a2   Ready     master    1y
v1.11.6
cx-ccs-prod-master-d7f34f25-f524-4f90-9037-7286202ed13a3   Ready     master    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a1   Ready     <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a2   Ready     <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a3   Ready     <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a4   Ready     <none>    1y
```

```
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a5   Ready     <none>    1y
v1.11.6
cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a6   Ready     <none>    1y
v1.11.6
```