El vencimiento del certificado de Kubernetes causa el alto de la comunicación en todo el clúster

Contenido

[Introducción](#)
[Problema](#)
[Solución](#)

Introducción

Este documento describe un posible problema de interrupción que los clientes podrían enfrentar cuando tienen un sistema basado en Kubernetes que se ha instalado durante más de 365 días. Además, sigue los pasos necesarios para solucionar la situación y poner en marcha el sistema basado en Kubernetes.

Problema

Después de un año de instalación predeterminada del clúster de Kubernetes, los certificados de cliente caducan. No podrá acceder a Cisco CloudCenter Suite (CCS). Aunque todavía aparecerá, no podrá iniciar sesión. Si navega a la CLI kubectl, verá este error, "No se puede conectar al servidor: x509: el certificado ha caducado o aún no es válido."

Puede ejecutar esta secuencia de comandos bash para ver la fecha de vencimiento de sus certificados:

```
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
```

También puede encontrar un flujo de trabajo de código abierto para Action Orchestrator que supervisará este proceso diariamente y los alertará de problemas.

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

Solución

Los nuevos certificados se deben volver a emitir a través de Kubeadm a través del clúster y luego se debe volver a unir los nodos de los trabajadores a los maestros.

1. Inicie sesión en un nodo maestro.

2. Obtener su dirección IP **vía ip address show**.

```
[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. Navegue hasta el directorio de Kubernetes a través de **cd /etc/kubernetes**.

4. Cree un archivo llamado **kubeadmCERT.yaml** vía **vi kubeadmCERT.yaml**.

5. El archivo debe tener el siguiente aspecto:

```
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. Respalde sus certificados y claves anteriores. Esto no es necesario, pero se recomienda. Haga un directorio de copia de seguridad y cópielo.

```
#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. Si ha omitido el paso 6., sólo puede eliminar los archivos antes mencionados a través del comando rm como **rm apiserver.crt**.

8. Vuelva a la ubicación del archivo **kubeadmCERT.yaml**. Genere una nueva certificación apiserver a través de **kubeadm —config kubeadmCERT.yaml alfa 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. Genere nuevos certificados kubelet de apiserver a través de **kubeadm —config kubeadmCERT.yaml de los certificados de fase alfa apiserver-kubelet-client**.

10. Genere nuevos certificados front-proxy-client a través de **kubeadm —config kubeadmCERT.yaml de los certificados de fase alfa front-proxy-client**.

11. En la carpeta **/etc/kubernetes**, haga una copia de seguridad de los **archivos.conf**. No es necesario pero se recomienda. Debe tener **kubelet.conf**, **controller-manager.conf**, **scheduler.conf** y posiblemente **admin.conf**. Puede eliminarlos si no desea realizar una copia de seguridad.

12. Genere nuevos archivos de configuración a través de **kubeadm —config kubeadmCERT.yaml alfa fase 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. Exporte su nuevo archivo **admin.conf** a su 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. Reinicie el nodo maestro a través de **shutdown -r now**.

15. Una vez que el maestro es respaldado, verifique si kubelet se está ejecutando a través de **systemctl status kubelet**.

16. Verifique a Kubernetes a través de **los nodos get kubectl**.

```
[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. Repita los pasos 1. a 16. para cada nodo maestro.

18. En un maestro, genere un nuevo token de unión a través de **kubeadm token create —print-Join-command**. Copie ese comando para su uso posterior.

```
[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. Consiga las IP de sus trabajadores a través de **kubectl get nodos -o wide**.

20. Inicie sesión en un trabajador como **ssh -i /home/cloud-user/keys/gen3-ao-prod.key cloud-user@192.168.1.17** y navegue hasta el acceso raíz.

21. Detenga el servicio kubelet a través de **systemctl stop kubelet**.

22. Elimine los archivos de configuración antiguos, entre los que se incluyen **ca.crt**, **kubelet.conf** y **bootstrap-kubelet.conf**.

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

23. Agarre el nombre del nodo del Paso 19.

24. Ejecute el comando para el trabajador para volver a unirse al clúster. Utilice el comando 18., pero agregue **—node-name <nombre del nodo>** al final.

```
[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. Salga del trabajador y verifique el estado en un maestro a través de **los nodos get kubectl**. Debe estar en estado Preparado.

26. Repita los pasos 20. a 25. para cada trabajador.

27. Los últimos **nodos get kubectl** deben mostrar que todos los nodos están en estado "Preparado", nuevamente en línea y unidos al clúster.

```
[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
```