

Arresto della comunicazione a livello di cluster a causa di un certificato Kubernetes scaduto

Sommario

[Introduzione](#)

[Problema](#)

[Soluzione](#)

Introduzione

Questo documento descrive un possibile problema di interruzione che i clienti potrebbero dover affrontare quando hanno un sistema basato su Kubernetes che è stato installato per più di 365 giorni. Inoltre, esamina i passaggi necessari per risolvere la situazione e ripristinare l'operatività del sistema di Kubernetes.

Problema

Dopo un anno di installazione predefinita del cluster Kubernetes, i certificati client scadono. Non sarà possibile accedere a Cisco CloudCenter Suite (CCS). Anche se apparirà ancora aperto, non sarà possibile accedere. Se si passa alla CLI di kubectl, viene visualizzato questo errore, "Unable to connect to the server: x509: certificato scaduto o non ancora valido."

È possibile eseguire questo script bash per visualizzare la data di scadenza dei relativi certificati:

```
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
```

È inoltre possibile trovare un flusso di lavoro open source per Action Orchestrator che esegua il monitoraggio giornaliero e li avvisa in caso di problemi.

https://github.com/cisco-cx-workflows/cx-ao-shared-workflows/tree/master/CCSCheckKubernetesExpiration_definition_workflow_01E01VIRWZDE24mWlsHrqCGB9xUix0f9ZxG

Soluzione

È necessario rimettere i nuovi certificati tramite Kubeadm nel cluster e quindi aggiungere nuovamente i nodi di lavoro ai master.

1. Accedere a un nodo master.
2. Ottenere il relativo indirizzo IP tramite **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. Passare alla directory Kubernetes tramite `cd /etc/kubernetes`.

4. Creare un file denominato `kubeadmCERT.yaml` tramite `vi kubeadmCERT.yaml`.

5. Il file dovrebbe avere il seguente aspetto:

```
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. Eseguire il backup dei certificati e delle chiavi precedenti. Questa operazione non è obbligatoria ma consigliata. Creare una directory di backup e copiarvi i file.

```
#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. Se il passaggio 6 è stato ignorato, è possibile eliminare i file indicati in precedenza tramite il comando `rm`, ad esempio `rm apiserver.crt`.

8. Tornare alla posizione del file **kubeadmCERT.yaml**. Generare un nuovo certificato apiserver tramite **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. Genera nuovo certificato kubelet apiserver tramite **kubeadm —config kubeadmCERT.yaml certificati di fase alfa apiserver-kubelet-client**.
10. Genera nuovo certificato client proxy anteriore tramite **kubeadm —config kubeadmCERT.yaml certificati fase alfa front-proxy-client**.
11. Nella cartella **/etc/kubernetes**, eseguire il backup dei file **.conf**. Non obbligatorio, ma consigliato. È necessario disporre di **kubelet.conf**, **controller-manager.conf**, **scheduler.conf** e possibilmente **admin.conf**. È possibile eliminarli se non si desidera eseguirne il backup.
12. Generare nuovi file di configurazione tramite **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. Esportare il nuovo file **admin.conf** nell'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. Riavviare il nodo master tramite **shutdown -r ora**.
15. Una volta eseguito il backup del dispositivo master, controllare se kubelet è in esecuzione tramite **systemctl status kubelet**.
16. Verificare Kubernetes tramite i nodi **get di 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. Ripetere i passaggi da 1 a 16 per ogni nodo principale.

18. Su un master, generare un nuovo token di join tramite **kubeadm token create --print-join-command**. Copiare il comando per utilizzarlo successivamente.

```

[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 mlynvj.f4n3et3poki88ry4
--discovery-token-ca-cert-hash
sha256:4d0c569985c1d460ef74dc01c85740285e4af2c2369ff833eed1ba86e1167575

```

19. Ottieni gli IP dei tuoi dipendenti tramite **kubectl ottieni nodi -o larghe**.

20. Accedere a un worker come **ssh -i /home/cloud-user/keys/gen3-ao-prod.key cloud-user@192.168.1.17** e selezionare root access.

21. Arrestare il servizio kubelet tramite **systemctl stop kubelet**.

22. Rimuovere i file di configurazione precedenti, inclusi **ca.crt**, **kubelet.conf** e **bootstrap-kubelet.conf**.

```

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

```

23. Recuperare il nome del nodo dal passaggio 19.

24. Eseguire il comando per il ruolo di lavoro per rientrare nel cluster. Utilizzare il comando da 18., ma aggiungere **--node-name <nome del nodo>** alla fine.

```

[root@cx-ccs-prod-worker-d7f34f25-f524-4f90-9037-7286202ed13a1 kubernetes]# kubeadm join
192.168.1.14:6443 --token mlynvj.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. Uscire dal worker e controllare lo stato su un master tramite i nodi get di kubectl. Deve essere nello stato Pronto.
26. Ripetere i passaggi da 20 a 25 per ogni lavoratore.
27. Gli ultimi nodi di recupero kubectl devono indicare che tutti i nodi sono nello stato "Pronto", sono di nuovo online e sono stati aggiunti al cluster.

```
[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
```