Kubernetes 憑證到期導致整個叢集的通訊中斷

目錄

簡介

本文檔描述了客戶在安裝了超過365天的基於Kubernetes的系統時可能遇到的服務中斷問題。此外，它還將完成修復此情況所需的步驟，使基於Kubernetes的系統恢復運行。

問題

在預設安裝的Kubernetes群集的一年之後，客戶端證書將過期。您將無法訪問Cisco CloudCentre Suite(CCS)。 雖然它仍會顯示，但您將無法登入。如果導航到kubectl CLI，您將看到以下錯誤：「Unable to connect to the server:x509:證書已過期或尚未生效。"

您可以運行此bash指令碼以檢視其證書的到期日期：

```
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
```

您還可以找到Action Orchestrator的opensource工作流程，該工作流程每天監視此工作流程並提醒他們存在問題。

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

解決方案

必須通過Kubeadm跨群集重新頒發新證書，然後您需要將工作節點重新加入主節點。

1. 登入到主節點。

2. 通過**ip address** show**獲取其IP地址**。

```
[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. 通過**cd /etc/**kubernetes**導航到Kubernetes目錄。**

4. 通過**vi kubeadmCERT.yaml**建立名為**kubeadmCERT.yaml**的檔案。

5. 檔案應如下所示：

```
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. 備份您的舊證書和金鑰。這不是必需的，但建議這樣做。製作一個備份目錄並將這些檔案複製到其中。

```
#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. 如果跳過了步驟6。您可以通過rm命令(如**rm apiserver.crt**)刪除前面提到的檔案。

8. 導航回到**kubeadmCERT.yaml**檔案所在的位置。通過**kubeadm —config kubeadmCERT.yaml alpha phase certs apiserver**生成新的apiserver證書。

```
[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. 通過**kubeadm** —config kubeadmCERT.yaml alpha pha phase certs apiserver-kubelet-client生成新的apiserver kubelet cert。

10. 通過**kubeadm —config kubeadmCERT.yaml alpha phase certs front-proxy-client**生成新的front-proxy-client cert。

11. 在**/etc/kubernetes**資料夾中，備份**.conf檔案**。不需要，但建議使用。您應該有**kubelet.conf**、**controller-manager.conf**、**scheduler.conf**，可能還有**admin.conf**。如果您不想備份它們，可以將其刪除。

12. 通過**kubeadm** —**config kubeadmCERT.yaml alpha phase kubeconfig all**生成新的配置檔案。

```
[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. 將新的**admin.conf**檔案匯出到主機。

```
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. 通過**shutdown -r now**重新啟動主節點。

15. 備份主機後，檢查kubelet是否通過**system**status kubelet運行。

16. 通過kubectl get節**點驗證Kubernetes**。

```
[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. 對每個主節點重複步驟1到16。

18. 在一個主機上，通過**kubeadm token create —print-join-command**生成新的聯接令牌。複製該命令供以後使用。

```
[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. 通過kubectl get節點獲取**您員工的IP -o wide**。

20. 登入到**ssh -i /home/cloud-user/keys/gen3-ao-prod.key cloud-user@192.168.1.17**之類的工作程式並導航到root訪問許可權。

21. 通過**system**停止kubelet服務停止**kubelet服務**。

22. 移除舊組態檔，包括**ca.crt**、**kubelet.conf**和**bootstrap-kubelet.conf**。

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

23. 從步驟19獲取節點名稱。

24. 為輔助進程發出命令，以便重新加入群集。使用命令from 18.，但**在末尾新增node-name <**name of node**>**。

```
[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. 通過**kubectl get nodes**退出工作進程並檢查主節點上的狀態。它應該處於就緒狀態。

26. 為每個工作人員重複步驟20到25。

27. 最後**kubectl get節點**應顯示所有節點處於「就緒」狀態、重新聯機並加入群集。

```
[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
```