

Kubernetes证书过期导致集群范围通信中断

目录

[简介](#)

[问题](#)

[解决方案](#)

简介

本文档介绍客户在安装基于Kubernetes的系统超过365天时可能面临的故障问题。此外，它还完成了修复情况和使基于Kubernetes的系统恢复和运行所需的步骤。

问题

在默认安装Kubernetes群集一年后，客户端证书将过期。您将无法访问Cisco CloudCentre套件 (CCS)。虽然它仍将显示为up，但您无法登录。如果导航到kubectl CLI，您将看到以下错误：“Unable to connect to the server:x509:证书已过期或尚无效。”

您可以运行此bash脚本查看其证书的到期日期：

```
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
```

您还可以找到Action Orchestrator的开源工作流程，该工作流程将每天监控此工作流，并提醒他们注意问题。

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

解决方案

必须通过群集中的Kubeadm重新颁发新证书，然后您需要将工作节点重新加入主节点。

1. 登录到主节点。
2. 通过ip address show获取其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. 通过cd /etc/kubernetes导航到Kubernetes目录。

4. 通过vi kubeadmCERT.yaml创建名为kubeadmCERT.yaml的文件。

5. 文件应如下所示：

```

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. 备份旧证书和密钥。这不是必需的，但建议。创建备份目录并将这些文件复制到备份目录。

```

#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. 如果跳过步骤6.，则只需通过rm命令（如rm apiserver.crt）删除前面提到的文件。

8. 导航回您的kubeadmCERT.yaml文件所在的位置。通过kubeadm --config kubeadmCERT.yaml alpha phase certs apiserver生成新apiserver证书。

```

[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. 通过kubeadm `—config kubeadmCERT.yaml alpha phase certs apiserver-kubelet-client`生成新apiserver kubelet证书。
10. 通过kubeadm `—config kubeadmCERT.yaml alpha phase certs front-proxy-client`生成新的front-proxy-client证书。
11. 在/etc/kubernetes文件夹中，备份.conf文件。不是必需的，但建议。您应该有kubelet.conf、controller-manager.conf、scheduler.conf，可能还有admin.conf。如果不想备份，可以删除它们。
12. 通过kubeadm `—config kubeadmCERT.yaml alpha phase kubeconfig all`生成新配置文件。

```
[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. 将新的admin.conf文件导出到主机。

```
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. 立即通过关闭重新启动主节点。
15. 备份主设备后，检查kubelet是否通过systemctl status kubelet运行。
16. 通过kubectl获取节点验证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. 对每个主节点重复步骤1到16。。

18. 在一个主设备上，通过kubeadm token create —print-join-command生成新的加入令牌。复制该命令以备以后使用。

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

19. 通过kubectl获取节点获取工作人员的IP -o wide。

20. 登录到ssh -i /home/cloud-user/keys/gen3-ao-prod.key cloud-user@192.168.1.17等工作人员，然后导航到根访问。

21. 通过systemctl停止kubelet，停止kubelet服务。

22. 删除旧的配置文件，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. 从步骤19获取节点的名称。

24. 为工作人员发出命令以重新加入集群。从18.开始使用命令，但是将 — node-name <name of node>添加到末尾。

```
[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. 退出工作人员，并通过kubectl get节点检查主设备上的状态。它应处于就绪状态。

26. 对每个员工重复步骤20.到25.。

27. 最后一个kubectl获取节点应显示所有节点都处于“就绪”状态，重新联机并加入集群。

```
[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
```