OpenShift Container Platform 4.3

备份和恢复

备份和恢复 OpenShift Container Platform 集群

Last Updated: 2020-11-30

OpenShift Container Platform 4.3 备份和恢复

备份和恢复 OpenShift Container Platform 集群

法律通告

摘要

本文档提供了备份集群数据以及从各种灾难场景中恢复的步骤。

目录

第 1 章 备份 ETCD

etcd 是 OpenShift Container Platform 的以"键-值"形式进行的存储，它会保留所有资源对象的状态。

定期备份集群的 etcd 数据，并在 OpenShift Container Platform 环境以外的安全位置保存备份数据。不要在第一个证书轮转完成前（安装后的 24 小时内）进行 etcd 备份，否则备份将包含过期的证书。另外，建议您在非使用高峰时段对 etcd 进行备份，因为备份可能会影响到系统性能。

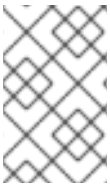
一旦您有了 etcd 备份，就可以对丢失的 master 主机进行恢复，并恢复到以前的集群状态。

您可以在任何已连接到 etcd 集群的 master 主机上执行 etcd 数据备份过程，这里提供了正确的证书。

1.1. 备份 ETCD 数据

按照以下步骤，通过创建 etcd 快照并备份静态 Kubernetes API 服务器资源来备份 etcd 数据。这个备份可以被保存，并在以后需要时使用它来恢复 etcd 数据。

您应该只保存单一 master 主机的备份。您不需要对集群中的每个 master 主机都进行备份。



注意

如果您在 OpenShift Container Platform 4.3.0 或 4.3.1 中执行 etcd 备份，则此流程会生成一个包含了 etcd 快照和静态 Kubernetes API 服务器资源的单个文件。恢复时，**etcd-snapshot-restore.sh** 脚本可以向后兼容来接受这个单一文件。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已检查是否启用了集群范围代理。

  提示

  您可以通过查看 **oc get proxy cluster -o yaml** 的输出来检查代理是否已启用。如果 **httpProxy**、**httpsProxy**和 **noProxy** 字段设置了值，则会启用代理。

流程

1. 为 master 节点启动一个 debug 会话：

   ```
   $ oc debug node/<node_name>
   ```

2. 将您的根目录改为主机：

   ```
   sh-4.2# chroot /host
   ```

3. 进入 **/home/core** 目录：

   ```
   sh-4.4# cd /home/core
   ```

4. 如果启用了集群范围的代理，请确定已导出了 **NO_PROXY**、**HTTP_PROXY**和 **HTTPS_PROXY** 环境变量。

5. 运行**etcd-snapshot-backup.sh**脚本，输入保存备份的位置。

```
sh-4.4# /usr/local/bin/etcd-snapshot-backup.sh /home/core/assets/backup
```

在这个示例中，在 master 主机上的 **/home/core/assets/backup/** 目录中创建了两个文件：

- **snapshot_<datetimestamp>.db**：这个文件是 etcd 快照。
- **static_kuberesources_<datetimestamp>.tar.gz**：此文件包含静态 Kubernetes API 服务器资源。如果启用了 etcd 加密，它也包含 etcd 快照的加密密钥。

注意

如果启用了 etcd 加密，建议出于安全考虑，将第二个文件与 etcd 快照分开保存。但是，需要这个文件才能从 etcd 快照中进行恢复。

请记住，etcd 仅对值进行加密，而不对键进行加密。这意味着资源类型、命名空间和对象名称是不加密的。

第 2 章 替换失败的 MASTER 主机

本文档描述了替换单个 etcd 成员的过程。此流程假设集群中还存在 etcd 仲裁。

注意

如果您丢失了大多数 master 主机，并导致 etcd 仲裁丢失，那么您必须遵循灾难恢复流程来恢复丢失的 master 主机，而不是这个过程。

如果 control plane 证书在被替换的成员中无效，则必须遵循从已过期 control plane 证书中恢复的步骤，而不是此过程。

替换单一 master 主机：

- 从 etcd 集群中删除成员。
- 如果 master 主机的 etcd 证书有效，请将 成员重新添加到 etcd 集群。
- 如果 master 主机没有 etcd 证书，或者它们已不再有效,请 生成 etcd 证书并将成员添加到 etcd 集群。

2.1. 从 ETCD 集群中删除失败的 MASTER 主机

按照以下步骤从 etcd 集群中删除失败的 master 主机。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 有到 一个活跃的 master 主机的 SSH 访问权限。

流程

1. 查看与 etcd 关联的 Pod 列表。
   在一个已连接到集群的终端中，运行以下命令：

```
$ oc get pods -n openshift-etcd
NAME                                                READY   STATUS    RESTARTS   AGE
etcd-member-ip-10-0-128-73.us-east-2.compute.internal   2/2     Running   0          15h
etcd-member-ip-10-0-147-172.us-east-2.compute.internal  2/2     Running   7          122m
etcd-member-ip-10-0-171-108.us-east-2.compute.internal  2/2     Running   0          15h
```

2. 访问活跃的 master 主机。
3. 运行 **etcd-member-remove.sh** 脚本，并传递要删除的 etcd 成员名称：

```
[core@ip-10-0-128-73 ~]$ sudo -E /usr/local/bin/etcd-member-remove.sh etcd-member-ip-10-0-147-172.us-east-2.compute.internal
Downloading etcdctl binary..
etcdctl version: 3.3.10
API version: 3.3
etcd client certs already backed up and available ./assets/backup/
```

```
Member 23e4736df4451b32 removed from cluster 6e25bab1bb556673
etcd member etcd-member-ip-10-0-147-172.us-east-2.compute.internal with
23e4736df4451b32 successfully removed..
```

4. 验证 etcd 成员已从集群中成功移除：

   a. 连接到正在运行的 etcd 容器：

   ```
   [core@ip-10-0-128-73 ~] id=$(sudo crictl ps --name etcd-member | awk 'FNR==2{ print
   $1}') && sudo crictl exec -it $id /bin/sh
   ```

   b. 在 etcd 容器中，导出连接到 etcd 所需的变量：

   ```
   sh-4.3# export ETCDCTL_API=3 ETCDCTL_CACERT=/etc/ssl/etcd/ca.crt
   ETCDCTL_CERT=$(find /etc/ssl/ -name *peer*crt) ETCDCTL_KEY=$(find /etc/ssl/ -
   name *peer*key)
   ```

   c. 在etcd容器中，执行**etcdctl member list**并确定删除的成员已不再输出列表中。

   ```
   sh-4.3#  etcdctl member list -w table

   +------------------+---------+-----------------------------------------+-------------------------------
   ---------------------------------+--------------------------+
   |        ID        | STATUS  |                NAME                     |                        PEER ADDRS
   |      CLIENT ADDRS        |
   +------------------+---------+-----------------------------------------+-------------------------------
   ---------------------------------+--------------------------+
   | 29e461db6be4eaaa | started | etcd-member-ip-10-0-128-73.us-east-2.compute.internal
   | https://etcd-2.clustername.devcluster.openshift.com:2380 | https://10.0.128.73:2379 |
   |  cbe982c74cbb42f | started |  etcd-member-ip-10-0-171-108.us-east-2.compute.internal
   | https://etcd-1.clustername.devcluster.openshift.com:2380 |   https://10.0.171.108:2379 |
   +------------------+---------+-----------------------------------------+-------------------------------
   ---------------------------------+--------------------------+
   ```

2.2. 将成员重新添加到集群

从 etcd 集群中删除成员后，请使用以下流程之一将成员添加到集群中：

- 如果 master 主机的 etcd 证书有效，请将 成员重新添加到 etcd 集群。
- 如果 master 主机没有 etcd 证书，或者它们已不再有效,请 生成 etcd 证书并将成员添加到 etcd 集群。

2.2.1. 将 master 主机重新添加到 etcd 集群

按照以下步骤将 master 主机添加到 etcd 集群。此流程假设您之前从集群中删除了 master 主机，并且其 etcd 的依赖部分（如 TLS 证书和 DNS）有效。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 可以 SSH 到 master 主机来添加到 etcd 集群。

- 有已存在的活跃 etcd 成员的 IP 地址。

流程

1. 访问添加到 etcd 集群的 master 主机。

重要

您必须在添加到 etcd 集群的 master 主机上运行这个步骤。

2. 运行 **etcd-member-add.sh** 脚本并传递两个参数：
   - 现有 etcd 成员的 IP 地址
   - 要添加的 etcd 成员的名称

```
[core@ip-10-0-147-172 ~]$ sudo -E /usr/local/bin/etcd-member-add.sh \
10.0.128.73 \ 1
etcd-member-ip-10-0-147-172.us-east-2.compute.internal 2

Downloading etcdctl binary..
etcdctl version: 3.3.10
API version: 3.3
etcd-member.yaml found in ./assets/backup/
etcd.conf backup upready exists ./assets/backup/etcd.conf
Stopping etcd..
Waiting for etcd-member to stop
etcd data-dir backup found ./assets/backup/etcd..
Updating etcd membership..
Removing etcd data_dir /var/lib/etcd..

ETCD_NAME="etcd-member-ip-10-0-147-172.us-east-2.compute.internal"
ETCD_INITIAL_CLUSTER="etcd-member-ip-10-0-147-172.us-east-
2.compute.internal=https://etcd-1.clustername.devcluster.openshift.com:2380,etcd-member-
ip-10-0-171-108.us-east-2.compute.internal=https://etcd-
2.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-128-73.us-east-
2.compute.internal=https://etcd-0.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="https://etcd-
1.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_CLUSTER_STATE="existing"'
Member  1e42c7070decd39 added to cluster 6e25bab1bb556673
Starting etcd..
```

1. 一个活跃 etcd 成员的 IP 地址。这*不是*您要添加的成员的 IP 地址。
2. 要添加的 etcd 成员的名称

3. 验证 etcd 成员是否已成功添加到 etcd 集群：
   a. 连接到正在运行的 etcd 容器：

```
[core@ip-10-0-147-172 ~] id=$(sudo crictl ps --name etcd-member | awk 'FNR==2{ print
$1}') && sudo crictl exec -it $id /bin/sh
```

b. 在 etcd 容器中，导出连接到 etcd 所需的变量：

```
sh-4.3# export ETCDCTL_API=3 ETCDCTL_CACERT=/etc/ssl/etcd/ca.crt
ETCDCTL_CERT=$(find /etc/ssl/ -name *peer*crt) ETCDCTL_KEY=$(find /etc/ssl/ -
name *peer*key)
```

c. 在 etcd 容器中，执行 **etcdctl member list** 并确定列出了新成员。

```
sh-4.3#  etcdctl member list -w table

+------------------+---------+-----------------------------------------+-------------------------------
---------------------------------+--------------------------+
|        ID        | STATUS  |                NAME                     |                        PEER ADDRS
|      CLIENT ADDRS        |
+------------------+---------+-----------------------------------------+-------------------------------
---------------------------------+--------------------------+
| 29e461db6be4eaaa | started | etcd-member-ip-10-0-128-73.us-east-2.compute.internal
| https://etcd-2.clustername.devcluster.openshift.com:2380 | https://10.0.128.73:2379 |
|  cbe982c74cbb42f | started | etcd-member-ip-10-0-147-172.us-east-2.compute.internal |
https://etcd-0.clustername.devcluster.openshift.com:2380 | https://10.0.147.172:2379 |
| a752f80bcb0da3e8 | started |   etcd-member-ip-10-0-171-108.us-east-
2.compute.internal | https://etcd-1.clustername.devcluster.openshift.com:2380 |
https://10.0.171.108:2379 |
+------------------+---------+-----------------------------------------+-------------------------------
---------------------------------+--------------------------+
```

请注意，新成员最多可能需要10分钟才能启动。

d. 在 etcd 容器中，执行 **etcdctl endpoint health** 并确定新成员处于健康状态。

```
sh-4.3# etcdctl endpoint health --cluster
https://10.0.128.73:2379 is healthy: successfully committed proposal: took = 4.5576ms
https://10.0.147.172:2379 is healthy: successfully committed proposal: took = 5.1521ms
https://10.0.171.108:2379 is healthy: successfully committed proposal: took = 4.2631ms
```

4. 验证新成员是否处于与 etcd 关联的 Pod 列表中，其状态是否是 **Running**。
在一个已连接到集群的终端中，运行以下命令：

```
$ oc get pods -n openshift-etcd
NAME                                                READY   STATUS    RESTARTS   AGE
etcd-member-ip-10-0-128-73.us-east-2.compute.internal    2/2     Running   0          15h
etcd-member-ip-10-0-147-172.us-east-2.compute.internal   2/2     Running   7          122m
etcd-member-ip-10-0-171-108.us-east-2.compute.internal   2/2     Running   0          15h
```

2.2.2. 生成 etcd 证书并将成员添加到集群中

如果节点是新节点，或者节点上的 etcd 证书已不再有效，则必须先生成 etcd 证书，然后才能将成员添加到 etcd 集群。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 可以 SSH 到新的 master 主机来添加到 etcd 集群。

- 可以 SSH 到一个健康的 master 主机。
- 有一个健康的 master 主机的 IP 地址。

流程

1. 在一个健康的 master 节点上设置临时 etcd 证书 signer 服务。
   a. 访问一个健康的 master 主机节点，使用以下命令以 **cluster-admin** 用户身份登录到集群。

```
[core@ip-10-0-143-125 ~]$ sudo oc login https://localhost:6443
Authentication required for https://localhost:6443 (openshift)
Username: kubeadmin
Password:
Login successful.
```

   b. 获取**kube-etcd-signer-server**镜像的pull规范。

```
[core@ip-10-0-143-125 ~]$ export KUBE_ETCD_SIGNER_SERVER=$(sudo oc adm
release info --image-for kube-etcd-signer-server --registry-
config=/var/lib/kubelet/config.json)
```

   c. 运行**tokenize-signer.sh**脚本。
   确保在**sudo**中使用**-E**以便将环境变量正确传递给脚本。

```
[core@ip-10-0-143-125 ~]$ sudo -E /usr/local/bin/tokenize-signer.sh ip-10-0-143-125 (1)
Populating template /usr/local/share/openshift-recovery/template/kube-etcd-cert-
signer.yaml.template
Populating template ./assets/tmp/kube-etcd-cert-signer.yaml.stage1
Tokenized template now ready: ./assets/manifests/kube-etcd-cert-signer.yaml
```

   (1) 部署 signer 的健康 master 主机的名称

   d. 使用生成的文件创建 signer Pod。

```
[core@ip-10-0-143-125 ~]$ sudo oc create -f assets/manifests/kube-etcd-cert-
signer.yaml
pod/etcd-signer created
```

   e. 确定 signer 在监听 master 节点。

```
[core@ip-10-0-143-125 ~]$ ss -ltn | grep 9943
LISTEN   0         128                    *:9943                 *:*
```

2. 将新的 master 主机添加到 etcd 集群。
   a. 访问要添加到集群中的新 master 主机，使用以下命令以 **cluster-admin** 用户身份登录到集群。

```
[core@ip-10-0-156-255 ~]$ sudo oc login https://localhost:6443
Authentication required for https://localhost:6443 (openshift)
Username: kubeadmin
```

```
Password:
Login successful.
```

b. 导出**etcd-member-recover.sh**脚本所需的两个环境变量。

```
[core@ip-10-0-156-255 ~]$ export SETUP_ETCD_ENVIRONMENT=$(sudo oc adm
release info --image-for machine-config-operator --registry-
config=/var/lib/kubelet/config.json)
```

```
[core@ip-10-0-156-255 ~]$ export KUBE_CLIENT_AGENT=$(sudo oc adm release info
--image-for kube-client-agent --registry-config=/var/lib/kubelet/config.json)
```

c. 运行**etcd-member-recover.sh**脚本。
确保在**sudo**中使用**-E**以便将环境变量正确传递给脚本。

```
[core@ip-10-0-156-255 ~]$ sudo -E /usr/local/bin/etcd-member-recover.sh 10.0.143.125
etcd-member-ip-10-0-156-255.ec2.internal (1)
Downloading etcdctl binary..
etcdctl version: 3.3.10
API version: 3.3
etcd-member.yaml found in ./assets/backup/
etcd.conf backup upready exists ./assets/backup/etcd.conf
Trying to backup etcd client certs..
etcd client certs already backed up and available ./assets/backup/
Stopping etcd..
Waiting for etcd-member to stop
etcd data-dir backup found ./assets/backup/etcd..
etcd TLS certificate backups found in ./assets/backup..
Removing etcd certs..
Populating template /usr/local/share/openshift-recovery/template/etcd-generate-
certs.yaml.template
Populating template ./assets/tmp/etcd-generate-certs.stage1
Populating template ./assets/tmp/etcd-generate-certs.stage2
Starting etcd client cert recovery agent..
Waiting for certs to generate..
Waiting for certs to generate..
Waiting for certs to generate..
Waiting for certs to generate..
Stopping cert recover..
Waiting for generate-certs to stop
Patching etcd-member manifest..
Updating etcd membership..
Member 249a4b9a790b3719 added to cluster 807ae3bffc8d69ca

ETCD_NAME="etcd-member-ip-10-0-156-255.ec2.internal"
ETCD_INITIAL_CLUSTER="etcd-member-ip-10-0-143-125.ec2.internal=https://etcd-
0.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-156-
255.ec2.internal=https://etcd-1.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="https://etcd-
1.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_CLUSTER_STATE="existing"
Starting etcd..
```

(1) 指定 signer 服务器运行的健康 master 的 IP 地址和新成员的 etcd 名称。

d. 验证新 master 主机已被添加到etcd成员列表中。

i. 访问健康 master，连接到正在运行的 etcd 容器。

```
[core@ip-10-0-143-125 ~] id=$(sudo crictl ps --name etcd-member | awk 'FNR==2{
print $1}') && sudo crictl exec -it $id /bin/sh
```

ii. 在 etcd 容器中，导出连接到 etcd 所需的变量。

```
sh-4.3# export ETCDCTL_API=3 ETCDCTL_CACERT=/etc/ssl/etcd/ca.crt
ETCDCTL_CERT=$(find /etc/ssl/ -name *peer*crt) ETCDCTL_KEY=$(find /etc/ssl/ -
name *peer*key)
```

iii. 在etcd容器中，执行**etcdctl member list**并确定列出了新成员。

```
sh-4.3#  etcdctl member list -w table

+------------------+---------+----------------------------------------+---------------------------
---------------------------------+--------------------------+
|        ID        | STATUS  |                NAME                    |                        PEER
ADDRS                        |      CLIENT ADDRS        |
+------------------+---------+----------------------------------------+---------------------------
---------------------------------+--------------------------+
|  cbe982c74cbb42f | started |  etcd-member-ip-10-0-156-255.ec2.internal |
https://etcd-0.clustername.devcluster.openshift.com:2380 |
https://10.0.156.255:2379 |
| 249a4b9a790b3719 | started | etcd-member-ip-10-0-143-125.ec2.internal |
https://etcd-1.clustername.devcluster.openshift.com:2380 | https://10.0.143.125:2379
|
+------------------+---------+----------------------------------------+---------------------------
---------------------------------+--------------------------+
```

请注意，新成员最多可能需要 20 分钟才能启动。

3. 添加新成员后，请删除 signer Pod，因为不再需要它。
在一个已连接到集群的终端中，运行以下命令：

```
$ oc delete pod -n openshift-config etcd-signer
```

第 3 章 灾难恢复

3.1. 灾难恢复

灾难恢复文档为管理员提供了如何从 OpenShift Container Platform 集群可能出现的几个灾难情形中恢复的信息。作为管理员，您可能需要遵循以下一个或多个步骤将集群恢复为工作状态。

恢复丢失的 master 主机

在已丢失了大多数 master 主机并导致 etcd quorum 丢失，且集群离线的情况下，可以使用这个解决方案。只要您执行了 etcd 备份并至少有一个处于健康状态的 master 主机时，可以按照这个步骤恢复集群。
如果适用，可能还需要从过期的 control plane 证书中恢复 。

注意

如果大多数 master 仍可用，且仍有 etcd 仲裁，请按照以下步骤替换单个失败的 master 主机。

恢复到一个以前的集群状态

如果您希望将集群恢复到一个以前的状态时（例如，管理员错误地删除了一些关键信息），则可以使用这个解决方案。只要您执行了 etcd 备份，就可以按照这个步骤将集群恢复到之前的状态。
如果适用，可能还需要从过期的 control plane 证书中恢复 。

从 control plane 证书已过期的情况下恢复

如果 control plane 证书已经过期，则可以使用这个解决方案。例如：在第一次证书轮转前（在安装后 24 小时内）关闭了集群，您的证书将不会被轮转，且会过期。可以按照以下步骤从已过期的 control plane 证书中恢复。

3.2. 恢复丢失的 MASTER 主机

本文档描述了恢复已完全丢失的 master 主机的过程。这包括大多数 master 主机已丢失并导致etcd仲裁（quorum）丢失，并且集群已下线的情况。这个步骤假设至少有一个健康的 master 主机。

总体来说，这个流程将：

1. 在一个仍存在的 master 主机上恢复etcd仲裁。
2. 创建新的 master 主机。
3. 修正DNS和负载均衡器条目。
4. 将etcd扩展到集群中的所有成员。

如果大多数 master 都已丢失，则需要 etcd 备份来在剩余的 master 主机上恢复 etcd 仲裁。

注意

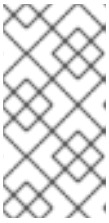
如果大多数 master 仍可用，且仍有 etcd 仲裁，请按照以下步骤替换单个失败的 master 主机。

3.2.1. 恢复丢失的 master 主机

当因为丢失了大多数 master 主机并导致 etcd 仲裁丢失时，按照以下步骤进行恢复。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 可以 SSH 到仍存在的 master 主机。
- 包含从同一备份中获取的 etcd 快照和静态 Kubernetes API 服务器资源的备份目录。该目录中的文件名必须采用以下格式: **snapshot_<datetimestamp>.db 和 static_kuberesources_<datetimestamp>.tar.gz** 。



注意

如果 etcd 备份是对 OpenShift Container Platform 4.3.0 或 4.3.1 进行的，那么它会是一个单一的文件，其中包含 etcd 快照和静态 Kubernetes API 服务器资源。**etcd-snapshot-restore.sh** 可以向后兼容来接受这个单一的文件，它的格式需要是 **snapshot_db_kuberesources_<datetimestamp>.tar.gz**。

流程

1. 在仍存在的 master 主机上恢复etcd仲裁。
   a. 将 etcd 备份目录复制到仍然存在的 master 主机上。
   此流程假设您将 **backup** 目录（其中包含 etcd 快照和静态 Kubernetes API 服务器资源）复制到 master 主机的 **/home/core/** 目录中。
   b. 访问仍存在的 master 主机。
   c. 将**INITIAL_CLUSTER**变量设置为以**<name>=<url>**格式代表的成员列表。此变量将会被传递给恢复脚本。在此假定此时只有一个成员。

```
[core@ip-10-0-143-125 ~]$ export INITIAL_CLUSTER="etcd-member-ip-10-0-143-125.ec2.internal=https://etcd-0.clustername.devcluster.openshift.com:2380"
```

   d. 如果启用了集群范围的代理，请确定已导出了 **NO_PROXY**、**HTTP_PROXY**和 **HTTPS_PROXY** 环境变量。

提示

您可以通过查看 **oc get proxy cluster -o yaml** 的输出来检查代理是否已启用。如果 **httpProxy**、**httpsProxy**和 **noProxy** 字段设置了值，则会启用代理。

   e. 运行**etcd-snapshot-restore.sh**脚本。
   将两个参数传递给**etcd-snapshot-restore.sh**脚本：etcd 备份目录的路径，成员列表（由**INITIAL_CLUSTER**变量定义）。

   确保在**sudo**中使用**-E**以便将环境变量正确传递给脚本。

```
[core@ip-10-0-143-125 ~]$ sudo -E /usr/local/bin/etcd-snapshot-restore.sh /home/core/backup $INITIAL_CLUSTER
Creating asset directory ./assets
Downloading etcdctl binary..
etcdctl version: 3.3.10
```

```
API version: 3.3
Backing up /etc/kubernetes/manifests/etcd-member.yaml to ./assets/backup/
Stopping all static pods..
..stopping kube-scheduler-pod.yaml
..stopping kube-controller-manager-pod.yaml
..stopping kube-apiserver-pod.yaml
..stopping etcd-member.yaml
Stopping etcd..
Waiting for etcd-member to stop
Stopping kubelet..
Stopping all containers..
bd44e4bc942276eb1a6d4b48ecd9f5fe95570f54aa9c6b16939fa2d9b679e1ea
d88defb9da5ae623592b81619e3690faeb4fa645440e71c029812cb960ff586f
3920ced20723064a379739c4a586f909497a7b6705a5b3cf367d9b930f23a5f1
d470f7a2d962c90f3a21bcc021970bde96bc8908f317ec70f1c21720b322c25c
Backing up etcd data-dir..
Removing etcd data-dir /var/lib/etcd
Restoring etcd member etcd-member-ip-10-0-143-125.ec2.internal from snapshot..
2019-05-15 19:03:34.647589 I | pkg/netutil: resolving etcd-
0.clustername.devcluster.openshift.com:2380 to 10.0.143.125:2380
2019-05-15 19:03:34.883545 I | mvcc: restore compact to 361491
2019-05-15 19:03:34.915679 I | etcdserver/membership: added member
cbe982c74cbb42f [https://etcd-0.clustername.devcluster.openshift.com:2380] to cluster
807ae3bffc8d69ca
Starting static pods..
..starting kube-scheduler-pod.yaml
..starting kube-controller-manager-pod.yaml
..starting kube-apiserver-pod.yaml
..starting etcd-member.yaml
Starting kubelet..
```

一旦**etcd-snapshot-restore.sh**脚本完成，您的集群将会有带有一个etcd成员的集群，API服务将开始重新启动。这可能需要15分钟。

在一个已连接到集群的终端中，运行以下命令验证它是否已准备就绪：

```
$ oc get nodes -l node-role.kubernetes.io/master
NAME                                         STATUS   ROLES    AGE   VERSION
ip-10-0-143-125.us-east-2.compute.internal   Ready    master   46m   v1.16.2
```

注意

确保所有被替换的旧etcd成员都已被关闭。否则，它们可能会尝试连接到新集群，并在日志中报告以下错误：

```
2019-05-20 15:33:17.648445 E | rafthttp: request cluster ID mismatch (got
9f5f9f05e4d43b7f want 807ae3bffc8d69ca)
```

2. 创建新的 master 主机。
   如果集群启用了 Machine API 且可以正常运行，则当 OpenShift **machine-api** Operator 被恢复后，它将会创建新的 master。如果未启用**machine-api** Operator，则必须使用最初创建它们时使用的相同方法创建新的 master。

您还需要批准这些新 master 主机的证书签名请求（CSR）。为添加到集群中的每个系统生成两个待处理的CSR。

a. 在一个已连接到集群的终端中，运行以下命令验证来批准 CSR：

i. 获取当前CSR列表。

```
$ oc get csr
```

ii. 查看CSR的详细信息以验证其是否有效。

```
$ oc describe csr <csr_name> 1
```

1 **<csr_name>**是当前CSR列表中CSR的名称。

iii. 批准每个有效的CSR。

```
$ oc adm certificate approve <csr_name>
```

确保为添加到集群的每个主服务器批准待处理的客户端和服务器CSR。

b. 在一个已连接到集群的终端中，运行以下命令验证 master 已准备就绪：

```
$ oc get nodes -l node-role.kubernetes.io/master
NAME                                         STATUS   ROLES    AGE   VERSION
ip-10-0-143-125.us-east-2.compute.internal   Ready    master   50m   v1.16.2
ip-10-0-156-255.us-east-2.compute.internal   Ready    master   92s   v1.16.2
ip-10-0-162-178.us-east-2.compute.internal   Ready    master   70s   v1.16.2
```

3. 更正DNS条目。

a. 在AWS控制台中，查看私有DNS区域中的etcd-0，etcd-1和etcd-2 Route 53记录，如有必要，将值更新为相应的新私有IP地址。详情请参阅AWS文档中的编辑记录部分 。
在已连接到集群的终端中运行以下命令来获取实例的私有IP地址。

```
$ oc get node ip-10-0-143-125.us-east-2.compute.internal -o
jsonpath='{.status.addresses[?(@.type=="InternalIP")].address}{"\n"}'
10.0.143.125
```

4. 更新负载均衡器条目。
如果您使用由集群管理的负载均衡器，相关条目会被自动更新。如果没有使用由集群管理的负载均衡器，则需要手工为负载均衡器更新 master 主机的信息。

如果您的负载均衡功能由AWS管理，请参阅AWS文档中的按IP地址注册或取消注册目标部分。

5. 将etcd扩展到集群中的所有成员。

a. 在您已恢复etcd的master上设置临时etcd证书 signer 服务。

i. 访问原始 master，使用以下命令以 **cluster-admin**用户身份登录到您的集群。

```
[core@ip-10-0-143-125 ~]$ sudo oc login https://localhost:6443
Authentication required for https://localhost:6443 (openshift)
Username: kubeadmin
```

```
Password:
Login successful.
```

ii. 获取**kube-etcd-signer-server**镜像的pull规范。

```
[core@ip-10-0-143-125 ~]$ export KUBE_ETCD_SIGNER_SERVER=$(sudo oc
adm release info --image-for kube-etcd-signer-server --registry-
config=/var/lib/kubelet/config.json)
```

iii. 运行**tokenize-signer.sh**脚本。
确保在**sudo**中使用**-E**以便将环境变量正确传递给脚本。

```
[core@ip-10-0-143-125 ~]$ sudo -E /usr/local/bin/tokenize-signer.sh ip-10-0-143-125
(1)
Populating template /usr/local/share/openshift-recovery/template/kube-etcd-cert-
signer.yaml.template
Populating template ./assets/tmp/kube-etcd-cert-signer.yaml.stage1
Tokenized template now ready: ./assets/manifests/kube-etcd-cert-signer.yaml
```

(1) 刚恢复的原始 master 的主机名，signer 需要在这个主机上部署。

iv. 使用生成的文件创建 signer Pod。

```
[core@ip-10-0-143-125 ~]$ sudo oc create -f assets/manifests/kube-etcd-cert-
signer.yaml
pod/etcd-signer created
```

v. 确定 signer 在监听 master 节点。

```
[core@ip-10-0-143-125 ~]$ ss -ltn | grep 9943
LISTEN   0         128                    *:9943                 *:*
```

b. 将新的 master 主机添加到 etcd 集群。

i. 访问新 master 主机中的一个，使用以下命令以 **cluster-admin**用户身份登录集群。

```
[core@ip-10-0-156-255 ~]$ sudo oc login https://localhost:6443
Authentication required for https://localhost:6443 (openshift)
Username: kubeadmin
Password:
Login successful.
```

ii. 导出**etcd-member-recover.sh**脚本所需的两个环境变量。

```
[core@ip-10-0-156-255 ~]$ export SETUP_ETCD_ENVIRONMENT=$(sudo oc adm
release info --image-for machine-config-operator --registry-
config=/var/lib/kubelet/config.json)
```

```
[core@ip-10-0-156-255 ~]$ export KUBE_CLIENT_AGENT=$(sudo oc adm release
info --image-for kube-client-agent --registry-config=/var/lib/kubelet/config.json)
```

iii. 运行**etcd-member-recover.sh**脚本。

确保在**sudo**中使用**-E**以便将环境变量正确传递给脚本。

```
[core@ip-10-0-156-255 ~]$ sudo -E /usr/local/bin/etcd-member-recover.sh
10.0.143.125 etcd-member-ip-10-0-156-255.ec2.internal (1)
Downloading etcdctl binary..
etcdctl version: 3.3.10
API version: 3.3
etcd-member.yaml found in ./assets/backup/
etcd.conf backup upready exists ./assets/backup/etcd.conf
Trying to backup etcd client certs..
etcd client certs already backed up and available ./assets/backup/
Stopping etcd..
Waiting for etcd-member to stop
etcd data-dir backup found ./assets/backup/etcd..
etcd TLS certificate backups found in ./assets/backup..
Removing etcd certs..
Populating template /usr/local/share/openshift-recovery/template/etcd-generate-
certs.yaml.template
Populating template ./assets/tmp/etcd-generate-certs.stage1
Populating template ./assets/tmp/etcd-generate-certs.stage2
Starting etcd client cert recovery agent..
Waiting for certs to generate..
Waiting for certs to generate..
Waiting for certs to generate..
Waiting for certs to generate..
Stopping cert recover..
Waiting for generate-certs to stop
Patching etcd-member manifest..
Updating etcd membership..
Member 249a4b9a790b3719 added to cluster 807ae3bffc8d69ca

ETCD_NAME="etcd-member-ip-10-0-156-255.ec2.internal"
ETCD_INITIAL_CLUSTER="etcd-member-ip-10-0-143-125.ec2.internal=https://etcd-
0.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-156-
255.ec2.internal=https://etcd-1.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_ADVERTISE_PEER_URLS="https://etcd-
1.clustername.devcluster.openshift.com:2380"
ETCD_INITIAL_CLUSTER_STATE="existing"
Starting etcd..
```

(1) 指定 signer 服务器运行的原始 master 的 IP 地址和新成员的 etcd 名称。

iv. 验证新 master 主机已被添加到etcd成员列表中。

A. 访问原始 master，连接到正在运行的etcd容器。

```
[core@ip-10-0-143-125 ~] id=$(sudo crictl ps --name etcd-member | awk
'FNR==2{ print $1}') && sudo crictl exec -it $id /bin/sh
```

B. 在 etcd 容器中，导出连接到 etcd 所需的变量。

```
sh-4.3# export ETCDCTL_API=3 ETCDCTL_CACERT=/etc/ssl/etcd/ca.crt
ETCDCTL_CERT=$(find /etc/ssl/ -name *peer*crt) ETCDCTL_KEY=$(find
/etc/ssl/ -name *peer*key)
```

C. 在etcd容器中，执行**etcdctl member list**并确定列出了新成员。

```
sh-4.3#  etcdctl member list -w table

+------------------+---------+-----------------------------------------+----------------------
----------------------------------------+--------------------------+
|        ID        | STATUS  |                  NAME                   |                      PEER
ADDRS                       |      CLIENT ADDRS       |
+------------------+---------+-----------------------------------------+----------------------
----------------------------------------+--------------------------+
|  cbe982c74cbb42f | started |  etcd-member-ip-10-0-156-255.ec2.internal |
https://etcd-0.clustername.devcluster.openshift.com:2380 |
https://10.0.156.255:2379 |
| 249a4b9a790b3719 | started | etcd-member-ip-10-0-143-125.ec2.internal |
https://etcd-1.clustername.devcluster.openshift.com:2380 |
https://10.0.143.125:2379 |
+------------------+---------+-----------------------------------------+----------------------
----------------------------------------+--------------------------+
```

请注意，新成员最多可能需要 20 分钟才能启动。

v. 重复这些步骤以添加其他新的 master 主机，直到完成所有的 etcd 成员。

c. 当所有成员都被恢复后，请删除 signer pod，因为不再需要它。
在一个已连接到集群的终端中，运行以下命令：

```
$ oc delete pod -n openshift-config etcd-signer
```

请注意，在完成这个过程后，可能需要几分钟才能恢复所有服务。例如，在重启 OAuth 服务器 Pod 前，使用 **oc login** 进行身份验证可能无法立即正常工作。

3.3. 恢复到一个以前的集群状态

为了将集群还原到以前的状态，您必须已通过创建快照备份了 etcd 数据 。您将需要使用此快照来还原集群状态。

3.3.1. 恢复到一个以前的集群状态

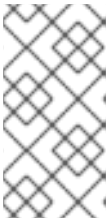
您可以使用已保存的 etcd 备份恢复到先前的集群状态。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 具有对 master 主机的 SSH 访问权限。
- 包含从同一备份中获取的 etcd 快照和静态 Kubernetes API 服务器资源的备份目录。该目录中的文件名必须采用以下格式: **snapshot_<datetimestamp>.db 和 static_kuberesources_<datetimestamp>.tar.gz** 。

注意

您必须在集群中的所有 master 主机上使用相同的 etcd 备份目录。



注意

如果 etcd 备份是对 OpenShift Container Platform 4.3.0 或 4.3.1 进行的，那么它会是一个单一的文件，其中包含 etcd 快照和静态 Kubernetes API 服务器资源。**etcd-snapshot-restore.sh** 可以向后兼容来接受这个单一的文件，它的格式需要是 **snapshot_db_kuberesources_<datetimestamp>.tar.gz**。

流程

1. 准备集群中要还原每个 master 主机。
   您应该在短时间内在所有 master 主机上运行还原脚本，以使集群成员可以大约在同时启动并形成仲裁。因此，建议为每一个 master 主机使用一个单独的终端，以便恢复脚本可以快速地在各个终端中同时运行。

   a. 将 etcd 备份目录复制到一个 master 主机上。
      此流程假设您将 **backup** 目录（其中包含 etcd 快照和静态 Kubernetes API 服务器资源）复制到 master 主机的 **/home/core/** 目录中。

   b. 访问 master 主机。

   c. 将**INITIAL_CLUSTER**变量设置为以**<name>=<url>**格式代表的成员列表。该变量将被传递到恢复脚本，并且对于每个成员必须完全相同。

      ```
      [core@ip-10-0-143-125 ~]$ export INITIAL_CLUSTER="etcd-member-ip-10-0-143-125.ec2.internal=https://etcd-0.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-35-108.ec2.internal=https://etcd-1.clustername.devcluster.openshift.com:2380,etcd-member-ip-10-0-10-16.ec2.internal=https://etcd-2.clustername.devcluster.openshift.com:2380"
      ```

   d. 如果启用了集群范围的代理，请确定已导出了 **NO_PROXY**、**HTTP_PROXY**和 **HTTPS_PROXY** 环境变量。

      提示

      您可以通过查看 **oc get proxy cluster -o yaml** 的输出来检查代理是否已启用。如果 **httpProxy**、**httpsProxy**和 **noProxy** 字段设置了值，则会启用代理。

   e. 在其他 master 主机上重复这些步骤（每个 master 都使用一个单独的终端）。确保在每个 master 主机上使用包含相同备份文件的备份目录。

2. 在所有 master 主机上运行恢复脚本。

   a. 在第一个 master 主机上启动**etcd-snapshot-restore.sh**脚本。传递两个参数：etcd 备份目录的路径，成员列表（由**INITIAL_CLUSTER**变量定义）。

      ```
      [core@ip-10-0-143-125 ~]$ sudo -E /usr/local/bin/etcd-snapshot-restore.sh /home/core/backup $INITIAL_CLUSTER
      Creating asset directory ./assets
      Downloading etcdctl binary..
      etcdctl version: 3.3.10
      API version: 3.3
      Backing up /etc/kubernetes/manifests/etcd-member.yaml to ./assets/backup/
      Stopping all static pods..
      ..stopping kube-scheduler-pod.yaml
      ..stopping kube-controller-manager-pod.yaml
      ```

```
..stopping kube-apiserver-pod.yaml
..stopping etcd-member.yaml
Stopping etcd..
Waiting for etcd-member to stop
Stopping kubelet..
Stopping all containers..
bd44e4bc942276eb1a6d4b48ecd9f5fe95570f54aa9c6b16939fa2d9b679e1ea
d88defb9da5ae623592b81619e3690faeb4fa645440e71c029812cb960ff586f
3920ced20723064a379739c4a586f909497a7b6705a5b3cf367d9b930f23a5f1
d470f7a2d962c90f3a21bcc021970bde96bc8908f317ec70f1c21720b322c25c
Backing up etcd data-dir..
Removing etcd data-dir /var/lib/etcd
Restoring etcd member etcd-member-ip-10-0-143-125.ec2.internal from snapshot..
2019-05-15 19:03:34.647589 I | pkg/netutil: resolving etcd-
0.clustername.devcluster.openshift.com:2380 to 10.0.143.125:2380
2019-05-15 19:03:34.883545 I | mvcc: restore compact to 361491
2019-05-15 19:03:34.915679 I | etcdserver/membership: added member
cbe982c74cbb42f [https://etcd-0.clustername.devcluster.openshift.com:2380] to cluster
807ae3bffc8d69ca
Starting static pods..
..starting kube-scheduler-pod.yaml
..starting kube-controller-manager-pod.yaml
..starting kube-apiserver-pod.yaml
..starting etcd-member.yaml
Starting kubelet..
```

b. 当恢复操作开始后，在其他 master 主机上运行脚本。

3. 验证 Machine Configs 已被使用。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令。

```
$ oc get machineconfigpool
NAME     CONFIG                                             UPDATED   UPDATING
master   rendered-master-50e7e00374e80b767fcc922bdfbc522b   True      False
```

应用快照后，master 服务器的**currentConfig**将与 etcd 快照被创建时的 ID相匹配。master 的 currentConfig名称格式为 **rendered-master-<currentConfig>** 。

4. 验证所有 master 主机已启动并加入集群。

a. 访问一个 master 主机，连接到正在运行的 etcd 容器。

```
[core@ip-10-0-143-125 ~] id=$(sudo crictl ps --name etcd-member | awk 'FNR==2{ print
$1}') && sudo crictl exec -it $id /bin/sh
```

b. 在 etcd 容器中，导出连接到 etcd 所需的变量。

```
sh-4.3# export ETCDCTL_API=3 ETCDCTL_CACERT=/etc/ssl/etcd/ca.crt
ETCDCTL_CERT=$(find /etc/ssl/ -name *peer*crt) ETCDCTL_KEY=$(find /etc/ssl/ -
name *peer*key)
```

c. 在 etcd 容器中，执行**etcdctl member list**，检查三个成员的状态都为“started”。

```
sh-4.3#  etcdctl member list -w table
```

```
+-----------------+---------+-----------------------------------------+-----------------------------
-------------------------------+-------------------------+
|        ID       | STATUS  |                  NAME                   |                        PEER ADDRS
|       CLIENT ADDRS        |
+-----------------+---------+-----------------------------------------+-----------------------------
-------------------------------+-------------------------+
| 29e461db6be4eaaa | started | etcd-member-ip-10-0-164-170.ec2.internal | https://etcd-
2.clustername.devcluster.openshift.com:2380 | https://10.0.164.170:2379 |
|  cbe982c74cbb42f | started | etcd-member-ip-10-0-143-125.ec2.internal | https://etcd-
0.clustername.devcluster.openshift.com:2380 | https://10.0.143.125:2379 |
| a752f80bcb0da3e8 | started |    etcd-member-ip-10-0-156-2.ec2.internal | https://etcd-
1.clustername.devcluster.openshift.com:2380 |    https://10.0.156.2:2379 |
+-----------------+---------+-----------------------------------------+-----------------------------
-------------------------------+-------------------------+
```

请注意，每个新成员最多可能需要 20 分钟才能启动。

3.4. 从 CONTROL PLANE 证书已过期的情况下恢复

3.4.1. 从 control plane 证书已过期的情况下恢复

在 control plane 证书已过期时，按照以下过程进行恢复。

先决条件

- 具有对 master 主机的 SSH 访问权限。

流程

1. 以 root 用户身份访问具有过期证书的 master 主机。
2. 获取相应的**cluster-kube-apiserver-operator**镜像。

```
# RELEASE_IMAGE=<release_image> 1
```

1 例如，**<release_image>** 的值可以是 **quay.io/openshift-release-dev/ocp-release:4.3.0-x86_64**。Repository Tags 页面包括了可用的标签列表 。

```
# KAO_IMAGE=$( oc adm release info --registry-config='/var/lib/kubelet/config.json'
"${RELEASE_IMAGE}" --image-for=cluster-kube-apiserver-operator )
```

3. 抓取（pull）**cluster-kube-apiserver-operator** 镜像。

```
# podman pull --authfile=/var/lib/kubelet/config.json "${KAO_IMAGE}"
```

4. 创建一个恢复 API 服务器。

```
# podman run -it --network=host -v /etc/kubernetes/:/etc/kubernetes/:Z --
entrypoint=/usr/bin/cluster-kube-apiserver-operator "${KAO_IMAGE}" recovery-apiserver
create
```

5. 在以上命令的输出中运行**export KUBECONFIG**命令，此过程后面的**oc**命令需要它。

```
# export KUBECONFIG=/<path_to_recovery_kubeconfig>/admin.kubeconfig
```

6. 等待恢复 API 服务器启动。

```
# until oc get namespace kube-system 2>/dev/null 1>&2; do echo 'Waiting for recovery
apiserver to come up.'; sleep 1; done
```

7. 运行**regenerate-certificates**命令。它会修复 API 中的证书，覆盖本地驱动器上的旧证书，然后重新启动静态 Pod 来使用它们。

```
# podman run -it --network=host -v /etc/kubernetes/:/etc/kubernetes/:Z --
entrypoint=/usr/bin/cluster-kube-apiserver-operator "${KAO_IMAGE}" regenerate-certificates
```

8. 在 API 中修复证书之后，运行以下命令为 control plane 强制执行新的 rollout。由于 kubelet 使用内部负载平衡器连接到 API 服务器，因此它将在其他节点上重新安装。

```
# oc patch kubeapiserver cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$(
date --rfc-3339=ns )"'"}}' --type=merge
```

```
# oc patch kubecontrollermanager cluster -p='{"spec": {"forceRedeploymentReason":
"recovery-'"$( date --rfc-3339=ns )"'"}}' --type=merge
```

```
# oc patch kubescheduler cluster -p='{"spec": {"forceRedeploymentReason": "recovery-'"$(
date --rfc-3339=ns )"'"}}' --type=merge
```

9. 使用一个有效用户创建引导 kubeconfig。

   a. 运行**recover-kubeconfig.sh**脚本，然后将输出保存到名为**kubeconfig**的文件中。

   ```
   # recover-kubeconfig.sh > kubeconfig
   ```

   b. 将**kubeconfig**文件复制到所有 master 主机并把它移到/**etc**/**kubernetes**/ **kubeconfig** 。

   c. 获取用于验证连接 API 服务器的 CA 证书。

   ```
   # oc get configmap kube-apiserver-to-kubelet-client-ca -n openshift-kube-apiserver-
   operator --template='{{ index .data "ca-bundle.crt" }}' > /etc/kubernetes/kubelet-ca.crt
   ```

   d. 将/**etc**/**kubernetes**/**kubelet-ca.crt**文件复制到所有其他 master 主机和节点上。

   e. 将**machine-config-daemon-force**文件添加到所有 master 主机和节点上，以强制 Machine Config 守护程序接受此证书更新。

   ```
   # touch /run/machine-config-daemon-force
   ```

10. 在所有 master 上恢复 kubelet。

    a. 在 master 主机上停止 kubelet。

    ```
    # systemctl stop kubelet
    ```

    b. 删除旧的 kubelet 数据。

```
# rm -rf /var/lib/kubelet/pki /var/lib/kubelet/kubeconfig
```

c. 重新启动 kubelet。

```
# systemctl start kubelet
```

d. 在所有其他 master 主机上重复这些步骤。

11. 如有必要，在 worker 节点上恢复 kubelet。
在还原 master 节点后，worker 节点可能会自行恢复。您可以通过运行**oc get nodes**命令进行验证。如果 worker 节点没有被列出，请在每个 worker 节点上执行以下步骤。

a. 停止 kubelet。

```
# systemctl stop kubelet
```

b. 删除旧的 kubelet 数据。

```
# rm -rf /var/lib/kubelet/pki /var/lib/kubelet/kubeconfig
```

c. 重新启动 kubelet。

```
# systemctl start kubelet
```

12. 批准待批准的 **node-bootstrapper** 证书签名请求 (CSR)。

a. 获取当前CSR列表。

```
# oc get csr
```

b. 查看CSR的详细信息以验证其是否有效。

```
# oc describe csr <csr_name> 1
```

1 **<csr_name>**是当前CSR列表中CSR的名称。

c. 批准每个有效的CSR。

```
# oc adm certificate approve <csr_name>
```

确保批准所有待批准的 **node-bootstrapper** CSR。

13. 因为已不再需要，请销毁恢复 API 服务器。

```
# podman run -it --network=host -v /etc/kubernetes/:/etc/kubernetes/:Z --
entrypoint=/usr/bin/cluster-kube-apiserver-operator "${KAO_IMAGE}" recovery-apiserver
destroy
```

等待 control plane 重新启动并获取新证书。这可能最多需要 10 分钟。