OpenShift Container Platform 4.1

バックアップおよび復元

OpenShift Container Platform 4.1 クラスターのリカバリー

Last Updated: 2020-04-16

OpenShift Container Platform 4.1 バックアップおよび復元

OpenShift Container Platform 4.1 クラスターのリカバリー

法律上の通知

概要

本書では、さまざまな障害シナリオから OpenShift Container Platform 4.1 クラスターのリカバリーを実行する方法について説明します。

目次

第1章 ETCD のバックアップ

etcd は OpenShift Container Platform のキーと値のストアであり、すべてのリソースオブジェクトの状態を保存します。

クラスターの etcd データを定期的にバックアップし、OpenShift Container Platform 環境外の安全な場所に保存するのが理想的です。インストールの 24 時間後に行われる最初の証明書のローテーションが完了するまで etcd のバックアップを実行することはできません。ローテーションの完了前に実行すると、バックアップに期限切れの証明書が含まれることになります。さらに、ピーク時には障害が発生させる要素となる可能性があるため、ピーク時以外に etcd バックアップを取得することも推奨されています。

etcd のバックアップを取得した後に、マスターホストが失われた状態からの復旧や、クラスターの直前の状態への復元を実行できます。

etcd データのバックアッププロセスは、適切な証明書が提供されている etcd クラスターに接続できるマスターホストで実行できます。

1.1. ETCD データのバックアップ

以下の手順に従い、スナップショットを作成して etcd データをバックアップします。このスナップショットは保存でき、etcd を復元する必要がある場合に後で使用することができます。

単一マスターホストからのスナップショットのみを保存する必要があります。クラスター内の各マスターホストからのスナップショットは必要ありません。

前提条件

- マスターホストへの SSH アクセス。

手順

1. root ユーザーとしてマスターホストにアクセスします。
2. **etcd-snapshot-backup.sh** スクリプトを実行し、etcd スナップショットの保存先となる場所を渡します。

```
$ sudo /usr/local/bin/etcd-snapshot-backup.sh ./assets/backup/snapshot.db
```

この例では、スナップショットはマスターホストの **./assets/backup/snapshot.db** に保存されます。

第2章 障害復旧

2.1. 障害復旧について

この障害復旧ドキュメントでは、OpenShift Container Platform クラスターで発生する可能性のある複数の障害のある状態からの復旧方法についての管理者向けの情報を提供しています。管理者は、クラスターの状態を機能する状態に戻すために、以下の 1 つまたは複数の手順を実行する必要がある場合があります。

マスターホストの失われた状態からのリカバリー

このソリューションは、大多数のマスターホストが失われたために etcd クォーラム(定足数) が失われ、クラスターがオフラインになる状態に対応します。etcd のバックアップを取得し、かつ少なくとも 1 つ以上の正常なマスターホストが残っている限り、以下の手順に従ってクラスターを復旧できます。
該当する場合は、コントロールプレーン証明書の期限切れの状態からのリカバリーが必要になる場合もあります。

クラスターの直前の状態への復元

このソリューションは、管理者が重要なものを削除した場合など、クラスターを直前の状態に復元する必要がある状態に対応します。etcd バックアップを取得している限り、以下の手順に従ってクラスターを直前の状態に復元できます。
該当する場合は、コントロールプレーン証明書の期限切れの状態からのリカバリーが必要になる場合もあります。

コントロールプレーン証明書の期限切れの状態からのリカバリー

このソリューションは、コントロールプレーン証明書の期限が切れた状態に対応します。たとえば、インストールの 24 時間後に行われる最初の証明書のローテーション前にクラスターをシャットダウンする場合、証明書はローテーションされず、期限切れになります。以下の手順に従って、コントロールプレーン証明書の期限切れの状態からのリカバリーを実行できます。

2.2. マスターホストの失われた状態からのリカバリー

以下では、マスターホストの完全に失われた状態からのリカバリープロセスについて説明します。大多数のマスターホストが失われたために etcd クォーラム(定足数) が失われ、クラスターがオフラインになる状態などがこれに該当します。この手順では、少なくとも 1 つ以上の正常なマスターホストがあることを前提とします。

概観すると、この手順では以下を実行します。

1. 残りのマスターホストでクォーラム(定足数) を復元します。
2. 新規マスターホストを作成します。
3. DNS およびロードバランサーエントリーを修正します。
4. etcd をフルメンバーシップに拡張します。

大多数のマスターホストが失われた場合、etcd スナップショットのバックアップ が必要となり、残りのマスターホストで etcd クォーラム (定足数) を復元する必要があります。

2.2.1. マスターホストの失われた状態からのリカバリー

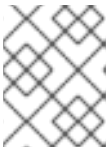
以下の手順に従って、1 つ以上のマスターホストの失われた状態からのリカバリーを実行します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてのクラスターへのアクセスがあること。
- 残りのマスターホストへの SSH アクセス。
- 大多数のマスターの失われた状態からのリカバリーの場合は etcd スナップショットのバックアップ。

手順

1. 残りのマスターホストでクォーラム(定足数) を復元します。



注記

この手順は、大多数のマスターが失敗している場合にのみ必要になります。大多数のマスターが依然として利用可能な場合にはこの手順を省略できます。

   a. etcd スナップショットファイルを残りのマスターホストにコピーします。
   この手順では、**snapshot.db** というスナップショットを、マスターホストの /**home**/**core**/ ディレクトリーにコピーしていることを前提とします。

   b. 残りのマスターホストにアクセスします。

   c. **INITIAL_CLUSTER** 変数を、 **<name>=<url>** 形式でメンバー一覧に追加します。この変数は復元スクリプトに渡されます。 この手順では、この時点で単一メンバーのみが存在することを前提とします。

```
[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. **etcd-snapshot-restore.sh** スクリプトを実行します。
   2 つのパラメーターを **etcd-snapshot-restore.sh** スクリプトに渡します。 それらは、バックアップされた etcd スナップショットファイルへのパスと、**INITIAL_CLUSTER** 変数で定義されるメンバーの一覧です。

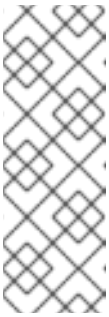
```
[core@ip-10-0-143-125 ~]$ sudo /usr/local/bin/etcd-snapshot-restore.sh /home/core/snapshot.db $INITIAL_CLUSTER
Creating asset directory ./assets
Downloading etcdctl binary..
etcdctl version: 3.3.10
API version: 3.3
Backing up /etc/kubernetes/manifests/etcd-member.yaml to ./assets/backup/
Stopping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

etcd-snapshot-restore.sh スクリプトが完了すると、クラスターでは単一メンバーの etcd クラスターを確認でき、API サービスは再起動を開始します。これには最長 15 分の時間がかかる可能性があります。

クラスターにアクセスできるターミナルで、以下のコマンドを実行し、クラスターが準備状態にあることを確認します。

```
$ 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
v1.13.4+db7b699c3
```



注記

置き換えられる古い etcd メンバーがすべてシャットダウンしていることを確認します。シャットダウンしていない場合には、それらが新規クラスターへの接続を試行し、以下のようなエラーをログに報告します。

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

2. 新規マスターホストを作成します。
クラスターでマシン API が有効にされており、かつ機能する場合、OpenShift **machine-api** Operator が復元すると、新規マスターが作成されます。しかし、**machine-api** Operator が有効にされていない場合は、最初にマスターを作成するために使用した方法と同じ方法を使って新規マスターを作成する必要があります。

   また、これらの新規マスターホストの証明書署名要求 (CSR) を承認する必要もあります。クラスターに追加された各マシンについて 2 つの保留状態の CSR が生成されます。

   a. クラスターにアクセスできるターミナルで、以下のコマンドを実行し、CSR を承認します。

      i. 現在の CSR の一覧を取得します。

      ```
      $ oc get csr
      ```

ii. CSR の詳細をレビューし、これが有効であることを確認します。

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

iii. それぞれの有効な CSR を承認します。

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

クラスターに追加されたそれぞれのマスターについての保留状態のクライアントおよびサーバー CSR の両方を承認します。

b. クラスターにアクセスできるターミナルで、以下のコマンドを実行し、マスターが準備状態にあることを確認します。

```
$ 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
v1.13.4+db7b699c3
ip-10-0-156-255.us-east-2.compute.internal   Ready    master   92s   v1.13.4+db7b699c3
ip-10-0-162-178.us-east-2.compute.internal   Ready    master   70s   v1.13.4+db7b699c3
```

3. DNS エントリーを修正します。

a. AWS コンソールで、プライベート DNS ゾーンにある etcd-0、etcd-1、および etcd-2 Route 53 レコードを確認し、必要な場合には、値を適切な新規のプライベート IP アドレスに更新します。具体的な方法については、AWS ドキュメントの「 Editing Records 」を参照してください。
クラスターにアクセスできるターミナルで以下のコマンドを実行して、インスタンスのプライベート IP アドレスを取得できます。

```
$ 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. ロードバランサーのエントリーを更新します。
クラスターで管理されるロードバランサーを使用している場合、エントリーは自動的に更新されます。このロードバランサーを使用していない場合には、お使いのロードバランサーをマスターホストの現在のアドレスで更新してください。

負荷分散が AWS によって管理されている場合には、ロードバランサーのエントリーを更新する方法について、AWS ドキュメントの「Register or Deregister Targets by IP Address 」を参照してください。

5. etcd をフルメンバーシップに拡張します。

a. etcd を復元したマスターで、一時的な etcd の証明書に署名するサービスをセットアップします。

i. 元のマスターにアクセスし、以下のコマンドを使用して **cluster-admin** ユーザーとしてクラスターにログインします。

```
[core@ip-10-0-143-125 ~]$ oc login https://localhost:6443
```

```
Authentication required for https://localhost:6443 (openshift)
Username: kubeadmin
Password:
Login successful.
```

ii. **kube-etcd-signer-server** イメージのプル仕様を取得します。

```
[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. **tokenize-signer.sh** スクリプトを実行します。
-E フラグを **sudo** に渡し、環境変数がスクリプトに適切に渡されるようにします。

```
[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) 復元したばかりの元のマスターのホスト名です。 ここに署名側のデプロイが行われる必要があります。

iv. 生成されたファイルを使用して、署名側の Pod を作成します。

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

v. 署名側がこのマスターノードでリッスンしていることを確認します。

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

b. 新規マスターホストを etcd クラスターに追加します。

i. 新規マスターホストの 1 つにアクセスし、以下のコマンドを使用して **cluster-admin** ユーザーとしてクラスターにログインします。

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

ii. **etcd-member-recover.sh** スクリプトで必要な 2 つの環境変数をエクスポートします。

```
[core@ip-10-0-156-255 ~]$ export SETUP_ETCD_ENVIRONMENT=$(sudo oc adm
release info --image-for setup-etcd-environment --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. **etcd-member-recover.sh** スクリプトを実行します。
-E フラグを **sudo** に渡し、環境変数がスクリプトに適切に渡されるようにします。

```
[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) 署名側のサーバーが実行されている元のマスターの IP アドレスと、新規メンバーの etcd 名の両方を指定します。

iv. 新規マスターホストが etcd メンバー一覧に追加されていることを確認します。

A. 元のマスターにアクセスし、実行中の etcd コンテナーに接続します。

```
[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. etcd コンテナーで、etcd に接続するために必要な変数をエクスポートします。

```
sh-4.2# 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. etcd コンテナーで、**etcdctl member list** を実行し、新規メンバーが一覧表示されていることを確認します。

```
sh-4.2#  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 |
+------------------+---------+-----------------------------------------+----------------------
----------------------------------------+--------------------------+
```

新規メンバーが起動するまで最長 10 分の時間がかかる可能性があることに注意してください。

v. これらの手順を繰り返し実行し、etcd のフルメンバーシップに達するまで他の新規マスターホストを追加します。

c. すべてのメンバーが復元された後に、署名側 Pod は不要になるため、これを削除します。クラスターにアクセスできるターミナルで、以下のコマンドを実行します。

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

この手順の完了後、すべてのサービスを復元するまでに数分かかる場合があります。たとえば、**oc login** を使用した認証は、OAuth サーバー Pod が再起動するまですぐに機能しない可能性があります。

2.3. クラスターの直前の状態への復元

クラスターを直前の状態に復元するには、スナップショットを作成して、事前に etcd データのバックアップを行っている必要があります。このスナップショットを使用して、クラスターの状態を復元します。

2.3.1. クラスターの直前の状態への復元

保存された etcd スナップショットを使用して、クラスターの以前の状態に戻すことができます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてのクラスターへのアクセスがあること。

- マスターホストへの SSH アクセス。
- etcd スナップショットのバックアップ。

注記

クラスター内のすべてのマスターホストに同じ etcd スナップショットファイルを使用する必要があります。

手順

1. 復元するクラスター内のそれぞれのマスターホストを準備します。
   短期間でマスターホストのすべてに復元スクリプトを実行し、クラスターのメンバーがほぼ同時に起動し、クォーラム(定足数) を構成できるようにする必要がありなす。そのため、それぞれのマスターホストを別々のターミナルに分け、復元スクリプトをそれぞれに対して迅速に開始できるようにすることが推奨されます。

   a. etcd スナップショットをマスターホストにコピーします。
      この手順では、**snapshot.db** というスナップショットを、マスターホストの **/home/core/** ディレクトリーにコピーしていることを前提とします。

   b. マスターホストにアクセスします。

   c. **INITIAL_CLUSTER** 変数を、 **<name>=<url>** 形式でメンバー一覧に追加します。この変数は復元スクリプトに渡され、各メンバーに同一の変数が使用される必要があります。

      ```
      [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. これらの手順を他のマスターホストで繰り返し実行します。 その際、各マスターホストをそれぞれのターミナルで処理します。各マスターホストで同じ etcd スナップショットファイルを使用するようにしてください。

2. すべてのマスターホストで復元スクリプトを実行します。

   a. 最初のマスターホストで **etcd-snapshot-restore.sh** スクリプトを開始します。スナップショットファイルのパスと、**INITIAL_CLUSTER** 変数で定義されるメンバーの一覧の 2 つのパラメーターを渡します。

      ```
      [core@ip-10-0-143-125 ~]$ sudo /usr/local/bin/etcd-snapshot-restore.sh /home/core/snapshot.db $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. 復元が開始されたら、他のマスターホストでスクリプトを実行します。

3. マシン設定 (Machine Config) が適用されていることを確認します。
クラスターにアクセスできるターミナルで、**cluster-admin** ユーザーとして以下のコマンドを実行します。

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

スナップショットが適用されると、マスターの **currentConfig** は etcd スナップショットの取られた時点の ID に一致します。マスターの **currentConfig** 名は **rendered-master-<currentConfig>** 形式の名前になります。

4. すべてのマスターホストが起動しており、クラスターに参加していることを確認します。

a. マスターホストにアクセスし、実行中の etcd コンテナーに接続します。

```
[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. etcd コンテナーで、etcd に接続するために必要な変数をエクスポートします。

```
sh-4.2# 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. etcd コンテナーで、**etcdctl member list** を実行し、3 つのメンバーが起動済みであることを確認します。

```
sh-4.2#  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 |
+-----------------+---------+-----------------------------------------+-----------------------------
-------------------------------+-------------------------+
```

それぞれの新規メンバーが起動するまで最長 10 分の時間がかかる可能性があることに注意してください。

2.4. コントロールプレーン証明書の期限切れの状態からのリカバリー

2.4.1. コントロールプレーン証明書の期限切れの状態からのリカバリー

以下の手順に従い、コントロールプレーンの証明書の期限が切れの状態からのリカバリーを実行します。

前提条件

- マスターホストへの SSH アクセス。

手順

1. root ユーザーとして、証明書が期限切れになっているマスターホストにアクセスします。
2. リリースの **cluster-kube-apiserver-operator** イメージ参照を取得します。

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

1 **<release_image>** 値の例は **quay.io/openshift-release-dev/ocp-release:4.1.0** になります。

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

3. **cluster-kube-apiserver-operator** イメージをプルします。

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

4. リカバリー API サーバーを作成します。

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

5. 上記コマンドの出力から **export KUBECONFIG** コマンドを実行します。 これはこの手順の後の部分の **oc** コマンドで必要になります。

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

6. リカバリー API サーバーの起動を待機します。

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

7. **regenerate-certificates** コマンドを実行します。これは API の証明書を修正し、ローカルドライブの古い証明書を上書きし、静的 Pod を再起動して上書き内容を取得できるようにします。

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

8. 証明書が API で修正されたら、以下のコマンドを使用してコントロールプレーンの新規ロールアウトを強制実行します。これは、kubelet が内部ロードバランサーを使用して API サーバーに接続されているため、他のノードにも再インストールされます。

```
# 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. 有効なユーザーとしてブートストラップ kubeconfig を作成します。

   a. **recover-kubeconfig.sh** スクリプトを実行し、出力を **kubeconfig**というファイルに保存します。

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

   b. **kubeconfig** ファイルをすべてのマスターホストにコピーし、これを **/etc/kubernetes/kubeconfig** に移動します。

   c. API サーバーからの接続を検証するために使用する CA 証明書を取得します。

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

   d. **/etc/kubernetes/ca.crt** ファイルをその他すべてのマスターホストおよびノードにコピーします。

   e. すべてのマスターホストおよびノードに **machine-config-daemon-force** ファイルを追加して、Machine Config Daemon がこの証明書の更新を受け入れるようにします。

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

10. すべてのマスターで kubelet を回復します。

a. マスターホストで、kubelet を停止します。

```
# systemctl stop kubelet
```

b. 古い kubelet データを削除します。

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

c. kubelet を再起動します。

```
# systemctl start kubelet
```

d. 他のすべてのマスターホストでこれらの手順を繰り返し実行します。

11. 必要な場合には、ワーカーノードで kubelet を回復します。
マスターノードの復元後、ワーカーノードはそれら自体を復元する可能性があります。これは、**oc get nodes** コマンドを実行して確認できます。ワーカーノードが一覧表示されない場合には、各ワーカーノードで以下の手順を実行します。

a. kubelet を停止します。

```
# systemctl stop kubelet
```

b. 古い kubelet データを削除します。

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

c. kubelet を再起動します。

```
# systemctl start kubelet
```

12. 保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を承認します。

a. 現在の CSR の一覧を取得します。

```
# oc get csr
```

b. CSR の詳細をレビューし、これが有効であることを確認します。

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

1 **<csr_name>** は、現行の CSR の一覧からの CSR の名前です。

c. それぞれの有効な CSR を承認します。

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

保留状態のすべての **node-bootstrapper** CSR を承認するようにしてください。

13. リカバリー API サーバーは不要になるため、これを削除します。

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

コントロールプレーンが再起動し、新規証明書を取得するまで待機します。これには、最長 10 分の時間がかかる可能性があります。