



OpenShift Container Platform 4.5

バックアップおよび復元

OpenShift Container Platform クラスターのバックアップおよび復元

OpenShift Container Platform 4.5 バックアップおよび復元

OpenShift Container Platform クラスターのバックアップおよび復元

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

法律上の通知

Copyright © 2021 | You need to change the HOLDER entity in the en-US/Backup_and_restore.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書では、クラスターのデータのバックアップと、さまざまな障害関連のシナリオでの復旧方法について説明します。

目次

第1章 ETCDのバックアップ	3
1.1. ETCD データのバックアップ	3
第2章 正常でない ETCD メンバーの置き換え	5
2.1. 前提条件	5
2.2. 正常でない ETCD メンバーの特定	5
2.3. 正常でない ETCD メンバーの状態の判別	5
2.4. 正常でない ETCD メンバーの置き換え	7
2.4.1. マシンが実行されていないか、またはノードが準備状態にない場合の正常でない etcd メンバーの置き換え	8
2.4.2. etcd Pod がクラッシュループしている場合の正常でない etcd メンバーの置き換え	15
第3章 クラスターの正常なシャットダウン	19
3.1. 前提条件	19
3.2. クラスターのシャットダウン	19
第4章 クラスターの正常な再起動	21
4.1. 前提条件	21
4.2. クラスターの再起動	21
第5章 障害復旧	24
5.1. 障害復旧について	24
5.2. マスターホストの失われた状態からのリカバリー	24
5.3. クラスターの直前の状態への復元	24
5.3.1. クラスターの直前の状態への復元	24
5.4. 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	30
5.4.1. 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	30

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となります。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。

クラスターのアップグレード後に必ず etcd バックアップを作成してください。これは、クラスターを復元する際に、同じ z-stream リリースから取得した etcd バックアップを使用する必要があるために重要になります。たとえば、OpenShift Container Platform 4.5.2 クラスターは、4.5.2 から取得した etcd バックアップを使用する必要があります。



重要

マスターホストでバックアップスクリプトの単一の呼び出しを実行して、クラスターの etcd データをバックアップします。各マスターホストのバックアップを取得しないでください。

etcd のバックアップを作成した後に、[クラスターの直前の状態への復元](#)を実行できます。

etcd インスタンスを持つマスターホストで [etcd データのバックアッププロセス](#)を実行できます。

1.1. ETCD データのバックアップ

以下の手順に従って、etcd スナップショットを作成し、静的 Pod のリソースをバックアップして etcd データをバックアップします。このバックアップは保存でき、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。



重要

単一マスターホストからのバックアップのみを保存します。クラスター内の各マスターホストからのバックアップは取りません。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできること。
- クラスター全体のプロキシが有効になっているかどうかを確認している。

ヒント

oc get proxy cluster -o yaml の出力を確認して、プロキシが有効にされているかどうかを確認できます。プロキシは、**httpProxy**、**httpsProxy**、および **noProxy** フィールドに値が設定されている場合に有効にされます。

手順

1. マスターノードのデバッグセッションを開始します。

■

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

2. ルートディレクトリーをホストに切り替えます。

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

3. クラスター全体のプロキシが有効になっている場合は、**NO_PROXY**、**HTTP_PROXY**、および **HTTPS_PROXY** 環境変数をエクスポートしていることを確認します。
4. **etcd-snapshot-backup.sh** スクリプトを実行し、バックアップの保存先となる場所を渡します。

ヒント

cluster-backup.sh スクリプトは etcd Cluster Operator のコンポーネントとして維持され、**etcdctl snapshot save** コマンドに関連するラッパーです。

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

スクリプトの出力例

```
1bf371f1b5a483927cd01bb593b0e12cff406eb8d7d0acf4ab079c36a0abd3f7
etcdctl version: 3.3.18
API version: 3.3
found latest kube-apiserver-pod: /etc/kubernetes/static-pod-resources/kube-apiserver-pod-7
found latest kube-controller-manager-pod: /etc/kubernetes/static-pod-resources/kube-
controller-manager-pod-8
found latest kube-scheduler-pod: /etc/kubernetes/static-pod-resources/kube-scheduler-pod-6
found latest etcd-pod: /etc/kubernetes/static-pod-resources/etcd-pod-2
Snapshot saved at /home/core/assets/backup/snapshot_2020-03-18_220218.db
snapshot db and kube resources are successfully saved to /home/core/assets/backup
```

この例では、マスターホストの **/home/core/assets/backup/** ディレクトリーにファイルが2つ作成されます。

- **snapshot_<timestamp>.db**: このファイルは etcd スナップショットです。
- **static_kuberresources_<timestamp>.tar.gz**: このファイルには、静的 Pod のリソースが含まれます。etcd 暗号化が有効にされている場合、etcd スナップショットの暗号化キーも含まれます。



注記

etcd 暗号化が有効にされている場合、セキュリティ上の理由から、この2つ目のファイルを etcd スナップショットとは別に保存することが推奨されます。ただし、このファイルは etcd スナップショットから復元するために必要になります。

etcd 暗号化はキーではなく値のみを暗号化することに注意してください。つまり、リソースタイプ、namespace、およびオブジェクト名は暗号化されません。

第2章 正常でない ETCD メンバーの置き換え

本書では、単一の正常でない etcd メンバーを置き換えるプロセスについて説明します。

このプロセスは、マシンが実行されていないか、またはノードが準備状態にないことによって etcd メンバーが正常な状態にないか、または etcd Pod がクラッシュループしているためにこれが正常な状態にないかによって異なります。



注記

大多数のマスターホストが失われ、etcd のクォーラム（定足数）の損失が発生した場合は、この手順ではなく、[直前のクラスター状態への復元](#)に向けた障害復旧手順を実行する必要があります。

コントロールプレーンの証明書が置き換えているメンバーで有効でない場合は、この手順ではなく、[期限切れのコントロールプレーン証明書からの回復手順](#)を実行する必要があります。

マスターノードが失われ、新規ノードが作成される場合、etcd クラスター Operator は新規 TLS 証明書の生成と、ノードの etcd メンバーとしての追加を処理します。

2.1. 前提条件

- 正常でない etcd メンバーを置き換える前に、[etcd バックアップ](#)を作成します。

2.2. 正常でない ETCD メンバーの特定

クラスターに正常でない etcd メンバーがあるかどうかを特定することができます。

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。

手順

1. 以下のコマンドを使用して **EtcMembersAvailable** ステータス条件のステータスを確認します。

```
$ oc get etcd -o=jsonpath='{range .items[0].status.conditions[?(@.type=="EtcMembersAvailable")]}{.message}{"\n"}
```

2. 出力を確認します。

```
2 of 3 members are available, ip-10-0-131-183.ec2.internal is unhealthy
```

この出力例は、**ip-10-0-131-183.ec2.internal** etcd メンバーが正常ではないことを示しています。

2.3. 正常でない ETCD メンバーの状態の判別

正常でない etcd メンバーを置き換える手順は、etcd メンバーが以下のどの状態にあるかによって異なります。

- マシンが実行されていないか、ノードが準備状態にない
- etcd Pod がクラッシュループしている。

以下の手順では、etcd メンバーがどの状態にあるかを判別します。これにより、正常でない etcd メンバーを置き換えるために実行する必要がある手順を確認できます。



注記

マシンが実行されていないか、またはノードが準備状態にないものの、すぐに正常な状態に戻ることが予想される場合は、etcd メンバーを置き換える手順を実行する必要はありません。etcd クラスター Operator はマシンまたはノードが正常な状態に戻ると自動的に同期します。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- 正常でない etcd メンバーを特定している。

手順

1. マシンが実行されていないかどうかを判別します。

```
$ oc get machines -A -ojsonpath='{range .items[*]}{@.status.nodeRef.name}{"\t"}{@.status.providerStatus.instanceState}{"\n"}' | grep -v running
```

出力例

```
ip-10-0-131-183.ec2.internal stopped 1
```

- 1** この出力には、ノードおよびノードのマシンのステータスを一覧表示されます。ステータスが **running** 以外の場合は、マシンは実行されていません。

マシンが実行されていない場合は、「マシンが実行されていないか、またはノードが準備状態にない場合の正常でない etcd メンバーの置き換え」の手順を実行します。

2. ノードが準備状態にないかどうかを判別します。
以下のシナリオのいずれかが true の場合、ノードは準備状態にありません。

- マシンが実行されている場合は、ノードに到達できないかどうかを確認します。

```
$ oc get nodes -o jsonpath='{range .items[*]}{"\n"}{@.metadata.name}{"\t"}{@range .spec.taints[*]}{.key}{ " " } | grep unreachable
```

出力例

```
ip-10-0-131-183.ec2.internal node-role.kubernetes.io/master  
node.kubernetes.io/unreachable node.kubernetes.io/unreachable 1
```

- 1** ノードが **unreachable** テイントと共に一覧表示される場合、ノードの準備はできていません。

- ノードが以前として到達可能である場合は、そのノードが **NotReady** として一覧表示されているかどうかを確認します。

```
$ oc get nodes -l node-role.kubernetes.io/master | grep "NotReady"
```

出力例

```
ip-10-0-131-183.ec2.internal NotReady master 122m v1.18.3 ❶
```

- ❶ ノードが **NotReady** として一覧表示されている場合、ノードの準備はできていません。

ノードの準備ができていない場合は、「マシンが実行されていないか、またはノードが準備状態にない場合の正常でない etcd メンバーの置き換え」の手順を実行します。

3. etcd Pod がクラッシュループしているかどうかを判別します。
マシンが実行され、ノードが準備できている場合は、etcd Pod がクラッシュループしているかどうかを確認します。
 - a. すべてのマスターノードが **Ready** として一覧表示されていることを確認します。

```
$ oc get nodes -l node-role.kubernetes.io/master
```

出力例

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-131-183.ec2.internal	Ready	master	6h13m	v1.18.3
ip-10-0-164-97.ec2.internal	Ready	master	6h13m	v1.18.3
ip-10-0-154-204.ec2.internal	Ready	master	6h13m	v1.18.3

- b. etcd Pod のステータスが **Error** または **CrashloopBackoff** のいずれかであるかどうかを確認します。

```
$ oc get pods -n openshift-etcd | grep etcd
```

出力例

etcd-ip-10-0-131-183.ec2.internal	2/3	Error	7	6h9m ❶
etcd-ip-10-0-164-97.ec2.internal	3/3	Running	0	6h6m
etcd-ip-10-0-154-204.ec2.internal	3/3	Running	0	6h6m

- ❶ この Pod のこのステータスは **Error** であるため、etcd Pod はクラッシュループしています。

etcd Pod がクラッシュループしている場合、etcd Pod がクラッシュループしている場合の正常でない etcd メンバーの置き換えについての手順を実行します。

2.4. 正常でない ETCD メンバーの置き換え

正常でない etcd メンバーの状態に応じて、以下のいずれかの手順を使用します。

- マシンが実行されていないか、またはノードが準備状態にない場合の正常でない etcd メンバーの置き換え
- etcd Pod がクラッシュループしている場合の正常でない etcd メンバーの置き換え

2.4.1. マシンが実行されていないか、またはノードが準備状態にない場合の正常でない etcd メンバーの置き換え

以下の手順では、マシンが実行されていないか、またはノードが準備状態にない場合の正常でない etcd メンバーを置き換える手順を説明します。

前提条件

- 正常でない etcd メンバーを特定している。
- マシンが実行されていないか、またはノードが準備状態にないことを確認している。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- etcd のバックアップを取得している。



重要

問題が発生した場合にクラスターを復元できるように、この手順を実行する前に etcd バックアップを作成しておくことは重要です。

手順

1. 正常でないメンバーを削除します。
 - a. 影響を受けるノード上に **ない** Pod を選択します。
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 pods -n openshift-etcd | grep etcd
```

出力例

```
etcd-ip-10-0-131-183.ec2.internal      3/3   Running   0      123m
etcd-ip-10-0-164-97.ec2.internal     3/3   Running   0      123m
etcd-ip-10-0-154-204.ec2.internal    3/3   Running   0      124m
```

- b. 実行中の etcd コンテナに接続し、影響を受けるノードにない Pod の名前を渡します。
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 rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

- c. メンバーの一覧を確認します。

```
sh-4.2# etcdctl member list -w table
```

出力例

```

+-----+-----+-----+-----+-----+
-----+
|   ID   | STATUS |   NAME   |   PEER ADDRS   |   CLIENT
ADDRS   |
+-----+-----+-----+-----+-----+
-----+
| 6fc1e7c9db35841d | started | ip-10-0-131-183.ec2.internal | https://10.0.131.183:2380 |
https://10.0.131.183:2379 |
| 757b6793e2408b6c | started | ip-10-0-164-97.ec2.internal | https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+-----+-----+-----+-----+-----+
-----+

```

これらの値はこの手順で後ほど必要となるため、ID および正常でない etcd メンバーの名前を書き留めておきます。

- d. ID を **etcdctl member remove** コマンドに指定して、正常でない etcd メンバーを削除します。

```
sh-4.2# etcdctl member remove 6fc1e7c9db35841d
```

出力例

```
Member 6fc1e7c9db35841d removed from cluster baa565c8919b060e
```

- e. メンバーの一覧を再度表示し、メンバーが削除されたことを確認します。

```
sh-4.2# etcdctl member list -w table
```

出力例

```

+-----+-----+-----+-----+-----+
-----+
|   ID   | STATUS |   NAME   |   PEER ADDRS   |   CLIENT
ADDRS   |
+-----+-----+-----+-----+-----+
-----+
| 757b6793e2408b6c | started | ip-10-0-164-97.ec2.internal | https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+-----+-----+-----+-----+-----+
-----+

```

これでノードシェルを終了できます。

2. 削除された正常でない etcd メンバーの古いシークレットを削除します。

- a. 削除された正常でない etcd メンバーのシークレットを一覧表示します。

```
$ oc get secrets -n openshift-etcd | grep ip-10-0-131-183.ec2.internal 1
```

- 1 この手順で先ほど書き留めた正常でない etcd メンバーの名前を渡します。

以下の出力に示されるように、ピア、サービング、およびメトリクスシークレットがあります。

出力例

```
etcd-peer-ip-10-0-131-183.ec2.internal      kubernetes.io/tls      2    47m
etcd-serving-ip-10-0-131-183.ec2.internal  kubernetes.io/tls      2    47m
etcd-serving-metrics-ip-10-0-131-183.ec2.internal kubernetes.io/tls      2
47m
```

- b. 削除された正常でない etcd メンバーのシークレットを削除します。

- i. ピアシークレットを削除します。

```
$ oc delete secret -n openshift-etcd etcd-peer-ip-10-0-131-183.ec2.internal
```

- ii. 提供シークレットを削除します。

```
$ oc delete secret -n openshift-etcd etcd-serving-ip-10-0-131-183.ec2.internal
```

- iii. メトリクスシークレットを削除します。

```
$ oc delete secret -n openshift-etcd etcd-serving-metrics-ip-10-0-131-183.ec2.internal
```

3. マスターマシンを削除し、再作成します。このマシンが再作成されると、新規リビジョンが強制的に実行され、etcd は自動的にスケールアップします。

インストーラーでプロビジョニングされるインフラストラクチャーを実行している場合、またはマシン API を使用してマシンを作成している場合は、以下の手順を実行します。それ以外の場合は、最初に作成する際に使用した方法と同じ方法を使用して新規マスターを作成する必要があります。

- a. 正常でないメンバーのマシンを取得します。

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 machines -n openshift-machine-api -o wide
```

出力例

```
NAME                                PHASE  TYPE      REGION  ZONE  AGE
NODE                                PROVIDERID  STATE
clustername-8qw5l-master-0          Running m4.xlarge us-east-1 us-east-1a
3h37m ip-10-0-131-183.ec2.internal  aws:///us-east-1a/i-0ec2782f8287dfb7e  stopped
1
clustername-8qw5l-master-1          Running m4.xlarge us-east-1 us-east-1b
3h37m ip-10-0-154-204.ec2.internal  aws:///us-east-1b/i-096c349b700a19631  running
clustername-8qw5l-master-2          Running m4.xlarge us-east-1 us-east-1c
3h37m ip-10-0-164-97.ec2.internal  aws:///us-east-1c/i-02626f1dba9ed5bba  running
clustername-8qw5l-worker-us-east-1a-wbtgd Running m4.large us-east-1 us-east-1a
1a 3h28m ip-10-0-129-226.ec2.internal  aws:///us-east-1a/i-010ef6279b4662ced
```

```

running
clustername-8qw5l-worker-us-east-1b-lrdxb Running m4.large us-east-1 us-east-1b
3h28m ip-10-0-144-248.ec2.internal aws:///us-east-1b/i-0cb45ac45a166173b running
clustername-8qw5l-worker-us-east-1c-pkg26 Running m4.large us-east-1 us-east-
1c 3h28m ip-10-0-170-181.ec2.internal aws:///us-east-1c/i-06861c00007751b0a
running

```

- 1 これは正常でないノードのマスターマシンです (**ip-10-0-131-183.ec2.internal**)。

- b. マシン設定をファイルシステムのファイルに保存します。

```

$ oc get machine clustername-8qw5l-master-0 \ 1
-n openshift-machine-api \
-o yaml \
> new-master-machine.yaml

```

- 1 正常でないノードのマスターマシンの名前を指定します。

- c. 直前の手順で作成された **new-master-machine.yaml** ファイルを編集し、新しい名前を割り当て、不要なフィールドを削除します。

- i. **status** セクション全体を削除します。

```

status:
  addresses:
    - address: 10.0.131.183
      type: InternalIP
    - address: ip-10-0-131-183.ec2.internal
      type: InternalDNS
    - address: ip-10-0-131-183.ec2.internal
      type: Hostname
  lastUpdated: "2020-04-20T17:44:29Z"
  nodeRef:
    kind: Node
    name: ip-10-0-131-183.ec2.internal
    uid: acca4411-af0d-4387-b73e-52b2484295ad
  phase: Running
  providerStatus:
    apiVersion: awsproviderconfig.openshift.io/v1beta1
    conditions:
      - lastProbeTime: "2020-04-20T16:53:50Z"
        lastTransitionTime: "2020-04-20T16:53:50Z"
        message: machine successfully created
        reason: MachineCreationSucceeded
        status: "True"
        type: MachineCreation
    instanceId: i-0fdb85790d76d0c3f
    instanceState: stopped
    kind: AWSMachineProviderStatus

```

- ii. **metadata.name** フィールドを新規の名前に変更します。

古いマシンと同じベース名を維持し、最後の番号を次に利用可能な番号に変更することが推奨されます。この例では、**clustername-8qw5l-master-0** は **clustername-8qw5l-master-3** に変更されています。

以下は例になります。

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
  name: clustername-8qw5l-master-3
  ...
```

- iii. **metadata.selfLink** フィールドを、直前の手順からの新規のマシン名を使用するように更新します。

```
apiVersion: machine.openshift.io/v1beta1
kind: Machine
metadata:
  ...
  selfLink: /apis/machine.openshift.io/v1beta1/namespaces/openshift-machine-api/machines/clustername-8qw5l-master-3
  ...
```

- iv. **spec.providerID** フィールドを削除します。

```
providerID: aws:///us-east-1a/i-0fdb85790d76d0c3f
```

- v. **metadata.annotations** および **metadata.generation** フィールドを削除します。

```
annotations:
  machine.openshift.io/instance-state: running
  ...
generation: 2
```

- vi. **metadata.resourceVersion** および **metadata.uid** フィールドを削除します。

```
resourceVersion: "13291"
uid: a282eb70-40a2-4e89-8009-d05dd420d31a
```

- d. 正常でないメンバーのマシンを削除します。

```
$ oc delete machine -n openshift-machine-api clustername-8qw5l-master-0 1
```

- 1** 正常でないノードのマスターマシンの名前を指定します。

- e. マシンが削除されたことを確認します。

```
$ oc get machines -n openshift-machine-api -o wide
```

出力例

NAME NODE	PHASE PROVIDERID	TYPE	REGION STATE	ZONE	AGE
clustername-8qw5l-master-1 3h37m ip-10-0-154-204.ec2.internal	Running aws:///us-east-1b/i-096c349b700a19631	m4.xlarge	us-east-1	us-east-1b	running
clustername-8qw5l-master-2 3h37m ip-10-0-164-97.ec2.internal	Running aws:///us-east-1c/i-02626f1dba9ed5bba	m4.xlarge	us-east-1	us-east-1c	running
clustername-8qw5l-worker-us-east-1a-wbtgd 1a 3h28m ip-10-0-129-226.ec2.internal	Running aws:///us-east-1a/i-010ef6279b4662ced	m4.large	us-east-1	us-east-1a	running
clustername-8qw5l-worker-us-east-1b-lrdxb 3h28m ip-10-0-144-248.ec2.internal	Running aws:///us-east-1b/i-0cb45ac45a166173b	m4.large	us-east-1	us-east-1b	running
clustername-8qw5l-worker-us-east-1c-pkg26 1c 3h28m ip-10-0-170-181.ec2.internal	Running aws:///us-east-1c/i-06861c00007751b0a	m4.large	us-east-1	us-east-1c	running

- f. **new-master-machine.yaml** ファイルを使用して新規マシンを作成します。

```
$ oc apply -f new-master-machine.yaml
```

- g. 新規マシンが作成されたことを確認します。

```
$ oc get machines -n openshift-machine-api -o wide
```

出力例

NAME NODE	PHASE PROVIDERID	TYPE	REGION STATE	ZONE	AGE
clustername-8qw5l-master-1 3h37m ip-10-0-154-204.ec2.internal	Running aws:///us-east-1b/i-096c349b700a19631	m4.xlarge	us-east-1	us-east-1b	running
clustername-8qw5l-master-2 3h37m ip-10-0-164-97.ec2.internal	Running aws:///us-east-1c/i-02626f1dba9ed5bba	m4.xlarge	us-east-1	us-east-1c	running
clustername-8qw5l-master-3 85s ip-10-0-133-53.ec2.internal	Provisioning aws:///us-east-1a/i-015b0888fe17bc2c8	m4.xlarge	us-east-1	us-east-1a	running
clustername-8qw5l-worker-us-east-1a-wbtgd 1a 3h28m ip-10-0-129-226.ec2.internal	Running aws:///us-east-1a/i-010ef6279b4662ced	m4.large	us-east-1	us-east-1a	running
clustername-8qw5l-worker-us-east-1b-lrdxb 1b 3h28m ip-10-0-144-248.ec2.internal	Running aws:///us-east-1b/i-0cb45ac45a166173b	m4.large	us-east-1	us-east-1b	running
clustername-8qw5l-worker-us-east-1c-pkg26 east-1c 3h28m ip-10-0-170-181.ec2.internal	Running aws:///us-east-1c/i-06861c00007751b0a	m4.large	us-east-1	us-east-1c	running

- ① 新規マシン **clustername-8qw5l-master-3** が作成され、**Provisioning** から **Running** にフェーズが変更されると準備状態になります。

新規マシンが作成されるまでに数分の時間がかかる場合があります。etcd クラスター Operator はマシンまたはノードが正常な状態に戻ると自動的に同期します。

検証

1. すべての etcd Pod が適切に実行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 pods -n openshift-etcd | grep etcd
```

出力例

```
etcd-ip-10-0-133-53.ec2.internal      3/3   Running   0       7m49s
etcd-ip-10-0-164-97.ec2.internal     3/3   Running   0       123m
etcd-ip-10-0-154-204.ec2.internal    3/3   Running   0       124m
```

直前のコマンドの出力に 2 つの Pod のみが一覧表示される場合、etcd の再デプロイメントを手動で強制できます。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 patch etcd cluster -p="{\"spec\": {\"forceRedeploymentReason\": \"recovery-\"$( date --rfc-3339=ns )\"}}\"} --type=merge ①
```

- ① **forceRedeploymentReason** 値は一意である必要があります。そのため、タイムスタンプが付加されます。

2. 3 つの etcd メンバーがあることを確認します。

- a. 実行中の etcd コンテナに接続し、影響を受けるノードになかった Pod の名前を渡します。
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 rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

- b. メンバーの一覧を確認します。

```
sh-4.2# etcdctl member list -w table
```

出力例

```
+-----+-----+-----+-----+-----+
+-----+
| ID      | STATUS | NAME          | PEER ADDRS      | CLIENT
ADDRS    |
+-----+-----+-----+-----+-----+
+-----+
| 5eb0d6b8ca24730c | started | ip-10-0-133-53.ec2.internal | https://10.0.133.53:2380 |
https://10.0.133.53:2379 |
| 757b6793e2408b6c | started | ip-10-0-164-97.ec2.internal | https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| ca8c2990a0aa29d1 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+-----+-----+-----+-----+-----+
+-----+
```

直前のコマンドの出力に 4 つ以上の etcd メンバーが表示される場合、不要なメンバーを慎重に削除する必要があります。



警告

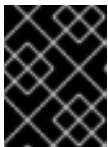
必ず適切な etcd メンバーを削除します。適切な etcd メンバーを削除すると、クォーラム (定足数) が失われる可能性があります。

2.4.2. etcd Pod がクラッシュループしている場合の正常でない etcd メンバーの置き換え

この手順では、etcd Pod がクラッシュループしている場合の正常でない etcd メンバーを置き換える手順を説明します。

前提条件

- 正常でない etcd メンバーを特定している。
- etcd Pod がクラッシュループしていることを確認している。
- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- etcd のバックアップを取得している。



重要

問題が発生した場合にクラスターを復元できるように、この手順を実行する前に etcd バックアップを作成しておくことは重要です。

手順

1. クラッシュループしている etcd Pod を停止します。
 - a. クラッシュループしているノードをデバッグします。
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 debug node/ip-10-0-131-183.ec2.internal ①
```

- ① これを正常でないノードの名前に置き換えます。

- b. ルートディレクトリーをホストに切り替えます。

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

- c. 既存の etcd Pod ファイルを kubelet マニフェストディレクトリーから移動します。

```
sh-4.2# mkdir /var/lib/etcd-backup
```

-

```
sh-4.2# mv /etc/kubernetes/manifests/etcd-pod.yaml /var/lib/etcd-backup/
```

- d. etcd データディレクトリーを別の場所に移動します。

```
sh-4.2# mv /var/lib/etcd/ /tmp
```

これでノードシェルの終了できます。

2. 正常でないメンバーを削除します。

- a. 影響を受けるノード上に **ない** Pod を選択します。

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 pods -n openshift-etcd | grep etcd
```

出力例

```
etcd-ip-10-0-131-183.ec2.internal      2/3   Error    7      6h9m
etcd-ip-10-0-164-97.ec2.internal     3/3   Running  0      6h6m
etcd-ip-10-0-154-204.ec2.internal    3/3   Running  0      6h6m
```

- b. 実行中の etcd コンテナに接続し、影響を受けるノードにない Pod の名前を渡します。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 rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

- c. メンバーの一覧を確認します。

```
sh-4.2# etcdctl member list -w table
```

出力例

```
+-----+-----+-----+-----+-----+
+-----+
| ID      | STATUS | NAME           | PEER ADDRS      | CLIENT
ADDRS    |        |                |                 |
+-----+-----+-----+-----+-----+
+-----+
| 62bcf33650a7170a | started | ip-10-0-131-183.ec2.internal | https://10.0.131.183:2380 |
https://10.0.131.183:2379 |
| b78e2856655bc2eb | started | ip-10-0-164-97.ec2.internal | https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| d022e10b498760d5 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380 |
https://10.0.154.204:2379 |
+-----+-----+-----+-----+-----+
+-----+
```

これらの値はこの手順で後ほど必要となるため、ID および正常でない etcd メンバーの名前を書き留めておきます。

- d. ID を `etcdctl member remove` コマンドに指定して、正常でない etcd メンバーを削除します。

```
sh-4.2# etcdctl member remove 62bcf33650a7170a
```

出力例

```
Member 62bcf33650a7170a removed from cluster ead669ce1fbfb346
```

- e. メンバーの一覧を再度表示し、メンバーが削除されたことを確認します。

```
sh-4.2# etcdctl member list -w table
```

出力例

```
+-----+-----+-----+-----+-----+
+-----+
| ID      | STATUS | NAME           | PEER ADDRS      | CLIENT
ADDRS    |
+-----+-----+-----+-----+-----+
+-----+
| b78e2856655bc2eb | started | ip-10-0-164-97.ec2.internal | https://10.0.164.97:2380 |
https://10.0.164.97:2379 |
| d022e10b498760d5 | started | ip-10-0-154-204.ec2.internal | https://10.0.154.204:2380
| https://10.0.154.204:2379 |
+-----+-----+-----+-----+-----+
+-----+
```

これでノードシェルを終了できます。

3. 削除された正常でない etcd メンバーの古いシークレットを削除します。

- a. 削除された正常でない etcd メンバーのシークレットを一覧表示します。

```
$ oc get secrets -n openshift-etcd | grep ip-10-0-131-183.ec2.internal 1
```

- 1** この手順で先ほど書き留めた正常でない etcd メンバーの名前を渡します。

以下の出力に示されるように、ピア、サービング、およびメトリクスシークレットがあります。

出力例

```
etcd-peer-ip-10-0-131-183.ec2.internal      kubernetes.io/tls      2    47m
etcd-serving-ip-10-0-131-183.ec2.internal  kubernetes.io/tls      2    47m
etcd-serving-metrics-ip-10-0-131-183.ec2.internal kubernetes.io/tls      2
47m
```

- b. 削除された正常でない etcd メンバーのシークレットを削除します。

- i. ピアシークレットを削除します。

```
$ oc delete secret -n openshift-etcd etcd-peer-ip-10-0-131-183.ec2.internal
```

- ii. 提供シークレットを削除します。

```
$ oc delete secret -n openshift-etcd etcd-serving-ip-10-0-131-183.ec2.internal
```

- iii. メトリクスシークレットを削除します。

```
$ oc delete secret -n openshift-etcd etcd-serving-metrics-ip-10-0-131-183.ec2.internal
```

4. etcd の再デプロイメントを強制的に実行します。

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 patch etcd cluster -p='{ "spec": { "forceRedeploymentReason": "single-master-recovery-$( date --rfc-3339=ns )" } }' --type=merge 1
```

- 1** **forceRedeploymentReason** 値は一意である必要があります。そのため、タイムスタンプが付加されます。

etcd クラスター Operator が再デプロイを実行する場合、すべてのマスターノードで etcd Pod が機能していることを確認します。

検証

- 新しいメンバーが利用可能で、正常な状態にあることを確認します。
 - a. 再度実行中の etcd コンテナに接続します。
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 rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

- b. すべてのメンバーが正常であることを確認します。

```
sh-4.2# etcdctl endpoint health --cluster
```

出力例

```
https://10.0.131.183:2379 is healthy: successfully committed proposal: took = 16.671434ms
https://10.0.154.204:2379 is healthy: successfully committed proposal: took = 16.698331ms
https://10.0.164.97:2379 is healthy: successfully committed proposal: took = 16.621645ms
```

第3章 クラスターの正常なシャットダウン

本書では、クラスターを正常にシャットダウンするプロセスについて説明します。メンテナンスの目的で、またはリソースコストの節約のためにクラスターを一時的にシャットダウンする必要がある場合があります。

3.1. 前提条件

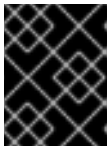
- クラスターをシャットダウンする前に [etcd バックアップ](#) を作成します。

3.2. クラスターのシャットダウン

クラスターを正常な状態でシャットダウンし、後で再起動できるようにします。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- etcd のバックアップを取得している。



重要

クラスターの再起動時に問題が発生した場合にクラスターを復元できるように、この手順を実行する前に etcd バックアップを作成しておくことは重要です。

手順

1. クラスターのすべてのノードをシャットダウンします。これは、クラウドプロバイダーの Web コンソールから行うか、以下のコマンドを使用できます。
 - a. ノードの一覧を取得します。

```
$ nodes=$(oc get nodes -o jsonpath='{.items[*].metadata.name}')
```

- b. すべてのノードをシャットダウンします。

```
$ for node in ${nodes[@]}
do
  echo "==== Shut down $node ====="
  ssh core@$node sudo shutdown -h 1
done
```

これらの方法のいずれかを使用してノードをシャットダウンすると、Pod は正常に終了するため、データが破損する可能性が低減します。



注記

シャットダウン前に OpenShift Container Platform に同梱される標準 Pod のマスターノードをドレイン (解放) する必要はありません。

クラスター管理者は、クラスターの再起動後に独自のワークロードのクリーンな再起動を実行する必要があります。カスタムワークロードが原因でシャットダウン前にマスターノードをドレイン (解放) した場合は、再起動後にクラスターが再び機能する前にマスターノードをスケジュール対象としてマークする必要があります。

2. 外部ストレージや LDAP サーバーなど、不要になったクラスター依存関係をすべて停止します。この作業を行う前に、ベンダーのドキュメントを確認してください。

追加リソース

- [クラスターの正常な再起動](#)

第4章 クラスターの正常な再起動

本書では、正常なシャットダウン後にクラスターを再起動するプロセスについて説明します。

クラスターは再起動後に機能することが予想されますが、クラスターは以下の例を含む予期しない状態によって回復しない可能性があります。

- シャットダウン時の etcd データの破損
- ハードウェアが原因のノード障害
- ネットワーク接続の問題

クラスターが回復しない場合は、[クラスターの以前の状態に復元する手順](#)を実行します。

4.1. 前提条件

- [クラスターを正常にシャットダウン](#)している。

4.2. クラスターの再起動

クラスターの正常なシャットダウン後にクラスターを再起動できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- この手順では、クラスターを正常にシャットダウンしていることを前提としています。

手順

1. 外部ストレージや LDAP サーバーなどのクラスターの依存関係すべてをオンにします。
2. すべてのクラスターマシンを起動します。
クラウドプロバイダーの Web コンソールなどでマシンを起動するには、ご使用のクラウド環境に適した方法を使用します。

約 10 分程度待機してから、マスターノードのステータス確認に進みます。

3. すべてのマスターノード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
```

以下の出力に示されているように、マスターノードのステータスが **Ready** の場合、マスターノードは準備状態にあります。

```
NAME                                STATUS ROLES  AGE  VERSION
ip-10-0-168-251.ec2.internal        Ready  master    75m  v1.18.3
ip-10-0-170-223.ec2.internal        Ready  master    75m  v1.18.3
ip-10-0-211-16.ec2.internal         Ready  master    75m  v1.18.3
```

4. マスターノードが準備状態に **ない** 場合、承認する必要がある保留中の証明書署名要求 (CSR) があるかどうかを確認します。

- 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> ❶
```

❶ <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>
```

5. マスターノードが準備状態になった後に、すべてのワーカーノードが準備状態にあることを確認します。

```
$ oc get nodes -l node-role.kubernetes.io/worker
```

以下の出力に示されているように、ワーカーノードのステータスが **Ready** の場合、ワーカーノードは準備状態にあります。

```
NAME                                STATUS ROLES  AGE  VERSION
ip-10-0-179-95.ec2.internal        Ready  worker    64m  v1.18.3
ip-10-0-182-134.ec2.internal       Ready  worker    64m  v1.18.3
ip-10-0-250-100.ec2.internal       Ready  worker    64m  v1.18.3
```

6. ワーカーノードが準備状態に **ない** 場合、承認する必要がある保留中の証明書署名要求 (CSR) があるかどうかを確認します。

- 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> ❶
```

❶ <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>
```

7. クラスターが適切に起動していることを確認します。

- a. パフォーマンスが低下したクラスター Operator がないことを確認します。

```
$ oc get clusteroperators
```

DEGRADED 条件が **True** に設定されているクラスター Operator がないことを確認します。

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED
authentication	4.5.0	True	False	59m
cloud-credential	4.5.0	True	False	85m
cluster-autoscaler	4.5.0	True	False	73m
config-operator	4.5.0	True	False	73m
console	4.5.0	True	False	62m
csi-snapshot-controller	4.5.0	True	False	66m
dns	4.5.0	True	False	76m
etcd	4.5.0	True	False	76m
...				

- b. すべてのノードが **Ready** 状態にあることを確認します。

```
$ oc get nodes
```

すべてのノードのステータスが **Ready** であることを確認します。

NAME	STATUS	ROLES	AGE	VERSION
ip-10-0-168-251.ec2.internal	Ready	master	82m	v1.18.3
ip-10-0-170-223.ec2.internal	Ready	master	82m	v1.18.3
ip-10-0-179-95.ec2.internal	Ready	worker	70m	v1.18.3
ip-10-0-182-134.ec2.internal	Ready	worker	70m	v1.18.3
ip-10-0-211-16.ec2.internal	Ready	master	82m	v1.18.3
ip-10-0-250-100.ec2.internal	Ready	worker	69m	v1.18.3

クラスターが適切に起動しなかった場合、etcd バックアップを使用してクラスターを復元する必要がある場合があります。

追加リソース

- クラスターが再起動後に回復しない場合に etcd バックアップを使用して復元する方法については、「[クラスターの直前の状態への復元](#)」を参照してください。

第5章 障害復旧

5.1. 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。



重要

障害復旧には、1つ以上の正常なマスターホストが必要です。

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。これには、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も含まれます。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。



注記

大多数のマスターが依然として利用可能であり、etcd のクォーラムがある場合は、手順に従って[単一の正常でない etcd メンバーの置き換え](#)を実行します。

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。

5.2. マスターホストの失われた状態からのリカバリー

OpenShift Container Platform 4.4 の時点では、マスターホストの失われた状態から回復するために、[直前のクラスターの状態に復元する](#)手順を実行します。



注記

大多数のマスターが依然として利用可能であり、etcd のクォーラムがある場合は、手順に従って[単一の正常でない etcd メンバーの置き換え](#)を実行します。

5.3. 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。

5.3.1. クラスターの直前の状態への復元

保存された etcd バックアップを使用して、クラスターの以前の状態に戻すことができます。etcd バックアップを使用して単一のコントロールパネルホストを復元します。次に、etcd クラスター Operator は残りのマスターホストへのスケーリングを処理します。



重要

クラスターを復元する際に、同じ z-stream リリースから取得した etcd バックアップを使用する必要があります。たとえば、OpenShift Container Platform 4.5.2 クラスターは、4.5.2 から取得した etcd バックアップを使用する必要があります。

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。
- リカバリーホストとして使用する正常なマスターホスト。
- マスターホストへの SSH アクセス。
- etcd スナップショットと静的 Pod のリソースの両方を含むバックアップディレクトリー（同じバックアップから取られるもの）。ディレクトリー内のファイル名は、**snapshot_<timestamp>.db** および **static_kuberesources_<timestamp>.tar.gz** の形式にする必要があります。

手順

1. リカバリーホストとして使用するコントロールプレーンホストを選択します。これは、復元操作を実行するホストです。
2. リカバリーホストを含む、各コントロールプレーンノードへの SSH 接続を確立します。Kubernetes API サーバーは復元プロセスの開始後にアクセスできなくなるため、コントロールプレーンノードにはアクセスできません。このため、別のターミナルで各コントロールプレーンホストに SSH 接続を確立することが推奨されます。



重要

この手順を完了しないと、復元手順を完了するためにマスターホストにアクセスすることができなくなり、この状態からクラスターを回復できなくなります。

3. etcd バックアップディレクトリーをリカバリーコントロールプレーンホストにコピーします。この手順では、etcd スナップショットおよび静的 Pod のリソースを含む **backup** ディレクトリーを、リカバリーコントロールプレーンホストの **/home/core/** ディレクトリーにコピーしていることを前提としています。
4. 他のすべてのコントロールプレーンノードで静的 Pod を停止します。



注記

リカバリーホストで Pod を手動で停止する必要はありません。リカバリースクリプトは、リカバリーホストの Pod を停止します。

- a. リカバリーホストではないコントロールプレーンホストにアクセスします。
- b. 既存の etcd Pod ファイルを kubelet マニフェストディレクトリーから移動します。

```
[core@ip-10-0-154-194 ~]$ sudo mv /etc/kubernetes/manifests/etcd-pod.yaml /tmp
```

- c. etcd Pod が停止していることを確認します。

```
[core@ip-10-0-154-194 ~]$ sudo crictl ps | grep etcd | grep -v operator
```

コマンドの出力は空であるはずですが、空でない場合は、数分待機してから再度確認します。

- d. 既存の Kubernetes API サーバー Pod ファイルを kubelet マニフェストディレクトリーから移動します。

```
[core@ip-10-0-154-194 ~]$ sudo mv /etc/kubernetes/manifests/kube-apiserver-pod.yaml /tmp
```

- e. Kubernetes API サーバー Pod が停止していることを確認します。

```
[core@ip-10-0-154-194 ~]$ sudo crictl ps | grep kube-apiserver | grep -v operator
```

コマンドの出力は空であるはずですが、空でない場合は、数分待機してから再度確認します。

- f. etcd データディレクトリーを別の場所に移動します。

```
[core@ip-10-0-154-194 ~]$ sudo mv /var/lib/etcd/ /tmp
```

- g. リカバリーホストではない他のマスターホストでこの手順を繰り返します。

- リカバリーコントロールプレーンホストにアクセスします。
- クラスター全体のプロキシが有効になっている場合は、**NO_PROXY**、**HTTP_PROXY**、および **HTTPS_PROXY** 環境変数をエクスポートしていることを確認します。

ヒント

oc get proxy cluster -o yaml の出力を確認して、プロキシが有効にされているかどうかを確認できます。プロキシは、**httpProxy**、**httpsProxy**、および **noProxy** フィールドに値が設定されている場合に有効にされます。

- リカバリーコントロールプレーンホストで復元スクリプトを実行し、パスを etcd バックアップディレクトリーに渡します。

```
[core@ip-10-0-143-125 ~]$ sudo -E /usr/local/bin/cluster-restore.sh /home/core/backup
```

スクリプトの出力例

```
...stopping kube-scheduler-pod.yaml
...stopping kube-controller-manager-pod.yaml
...stopping etcd-pod.yaml
...stopping kube-apiserver-pod.yaml
Waiting for container etcd to stop
.complete
Waiting for container etcdctl to stop
```

```

.....complete
Waiting for container etcd-metrics to stop
complete
Waiting for container kube-controller-manager to stop
complete
Waiting for container kube-apiserver to stop
.....complete
Waiting for container kube-scheduler to stop
complete
Moving etcd data-dir /var/lib/etcd/member to /var/lib/etcd-backup
starting restore-etcd static pod
starting kube-apiserver-pod.yaml
static-pod-resources/kube-apiserver-pod-7/kube-apiserver-pod.yaml
starting kube-controller-manager-pod.yaml
static-pod-resources/kube-controller-manager-pod-7/kube-controller-manager-pod.yaml
starting kube-scheduler-pod.yaml
static-pod-resources/kube-scheduler-pod-8/kube-scheduler-pod.yaml

```

8. すべてのマスターホストで kubelet サービスを再起動します。

a. リカバリーホストから以下のコマンドを実行します。

```
[core@ip-10-0-143-125 ~]$ sudo systemctl restart kubelet.service
```

b. 他のすべてのマスターホストでこの手順を繰り返します。

9. 単一メンバーのコントロールプレーンが正常に起動していることを確認します。

a. リカバリーホストから etcd コンテナが実行中であることを確認します。

```
[core@ip-10-0-143-125 ~]$ sudo crictl ps | grep etcd | grep -v operator
```

出力例

```

3ad41b7908e32
36f86e2eeaaaffe662df0d21041eb22b8198e0e58abeeae8c743c3e6e977e8009
About a minute ago   Running           etcd              0
7c05f8af362f0

```

b. リカバリーホストから、etcd Pod が実行されていることを確認します。

```
[core@ip-10-0-143-125 ~]$ oc get pods -n openshift-etcd | grep etcd
```



注記

このコマンドを実行する前に **oc login** の実行を試行し、以下のエラーを受信すると、認証コントローラーが起動し、再試行するまでしばらく待機します。

```
Unable to connect to the server: EOF
```

出力例

NAME	READY	STATUS	RESTARTS	AGE
etcd-ip-10-0-143-125.ec2.internal	1/1	Running	1	2m47s

ステータスが **Pending** の場合や出力に複数の実行中の etcd Pod が一覧表示される場合、数分待機してから再度チェックを行います。

- etcd の再デプロイメントを強制的に実行します。

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 patch etcd cluster -p='{ "spec": { "forceRedeploymentReason": "recovery-"$( date --rfc-3339=ns )"' }' --type=merge 1
```

- forceRedeploymentReason** 値は一意である必要があります。そのため、タイムスタンプが付加されます。

etcd クラスター Operator が再デプロイメントを実行すると、初期ブートストラップのスケールアップと同様に、既存のノードが新規 Pod と共に起動します。

- すべてのノードが最新のリリースに更新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 etcd -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

etcd の **NodeInstallerProgressing** 状況条件を確認し、すべてのノードが最新のリリースであることを確認します。更新が正常に実行されると、この出力には **AllNodesAtLatestRevision** が表示されます。

```
AllNodesAtLatestRevision
3 nodes are at revision 7 1
```

- この例では、最新のリリース番号は **7** です。

出力に **2 nodes are at revision 6; 1 nodes are at revision 7** などの複数のリリース番号が含まれる場合、これは更新が依然として進行中であることを意味します。数分待機した後に再試行します。

- etcd の再デプロイ後に、コントロールプレーンの新規ロールアウトを強制的に実行します。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、Kubernetes API サーバーは他のノードに再インストールされます。

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。

- kubeapiserver** を更新します。

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

すべてのノードが最新のリリースに更新されていることを確認します。


```
$ oc get kubeapiserver -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}\n}{.message}\n}'
```

NodeInstallerProgressing 状況条件を確認し、すべてのノードが最新のリリース番号であることを確認します。更新が正常に実行されると、この出力には **AllNodesAtLatestRevision** が表示されます。

```
AllNodesAtLatestRevision
3 nodes are at revision 7 ①
```

① この例では、最新のリリース番号は 7 です。

出力に **2 nodes are at revision 6; 1 nodes are at revision 7** などの複数のリリース番号が含まれる場合、これは更新が依然として進行中であることを意味します。数分待機した後に再試行します。

b. **kubecontrollermanager** を更新します。

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

すべてのノードが最新のリリース番号に更新されていることを確認します。

```
$ oc get kubecontrollermanager -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}\n}{.message}\n}'
```

NodeInstallerProgressing 状況条件を確認し、すべてのノードが最新のリリース番号であることを確認します。更新が正常に実行されると、この出力には **AllNodesAtLatestRevision** が表示されます。

```
AllNodesAtLatestRevision
3 nodes are at revision 7 ①
```

① この例では、最新のリリース番号は 7 です。

出力に **2 nodes are at revision 6; 1 nodes are at revision 7** などの複数のリリース番号が含まれる場合、これは更新が依然として進行中であることを意味します。数分待機した後に再試行します。

c. **kubescheduler** を更新します。

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

すべてのノードが最新のリリース番号に更新されていることを確認します。

```
$ oc get kubescheduler -o=jsonpath='{range .items[0].status.conditions[?(@.type=="NodeInstallerProgressing")]}{.reason}\n}{.message}\n}'
```

NodeInstallerProgressing 状況条件を確認し、すべてのノードが最新のリリース番号であることを確認します。更新が正常に実行されると、この出力には **AllNodesAtLatestRevision** が表示されます。

```
AllNodesAtLatestRevision
3 nodes are at revision 7 ❶
```

- ❶ この例では、最新のリビジョン番号は 7 です。

出力に **2 nodes are at revision 6; 1 nodes are at revision 7** などの複数のリビジョン番号が含まれる場合、これは更新が依然として進行中であることを意味します。数分待機した後、再試行します。

- 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。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 pods -n openshift-etcd | grep etcd
```

出力例

```
etcd-ip-10-0-143-125.ec2.internal      2/2   Running   0    9h
etcd-ip-10-0-154-194.ec2.internal      2/2   Running   0    9h
etcd-ip-10-0-173-171.ec2.internal      2/2   Running   0    9h
```

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。

5.4. 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

5.4.1. 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

OpenShift Container Platform 4.4.8 の時点で、クラスターはコントロールプレーン証明書の期限切れの状態から自動的にリカバリーできます。以前のバージョンで必要であった手動の手順を実行する必要がなくなりました。

例外として、kubelet 証明書を回復するために保留状態の **node-bootstrapper** 証明書署名要求 (CSR) を手動で承認する必要があります。

保留中の **node-bootstrapper** CSR を承認するには、以下の手順に従います。

手順

- 現在の CSR の一覧を取得します。

```
$ oc get csr
```

- 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> ❶
```

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

- それぞれの有効な **node-bootstrapper** CSR を承認します。

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