OpenShift Container Platform 4.6

백업 및 복원

OpenShift Container Platform 클러스터 백업 및 복원

Last Updated: 2021-06-04

OpenShift Container Platform 4.6 백업 및 복원

OpenShift Container Platform 클러스터 백업 및 복원

법적 공지

초록

이 문서는 클러스터 데이터를 백업하고 다양한 재해 시나리오에서 복구하는 방법에 대해 설명합니다.

차례

1장. ETCD 백업

etcd는 모든 리소스 개체의 상태를 저장하는 OpenShift Container Platform의 키-값 형식의 저장소입니다.

클러스터의 etcd 데이터를 정기적으로 백업하고 OpenShift Container Platform 환경 외부의 안전한 위치에 백업 데이터를 저장하십시오. 설치 후 24 시간 내에 발생하는 첫 번째 인증서 교체가 완료되기 전까지 etcd 백업을 수행하지 마십시오. 인증서 교체가 완료되기 전에 실행하면 백업에 만료된 인증서가 포함됩니다. 또한 백업이 시스템 성능에 영향을 미칠 수 있으므로 사용량이 많지 않은 시간에 etcd 백업을 수행하는 것이 좋습니다.

클러스터를 업그레이드한 후 etcd 백업을 수행해야 합니다. 이는 클러스터를 복원할 때 동일한 z-stream 릴리스에서 가져온 etcd 백업을 사용해야 하므로 중요합니다. 예를 들어 OpenShift Container Platform 4.6.2 클러스터는 4.6.2에서 가져온 etcd 백업을 사용해야 합니다.

중요

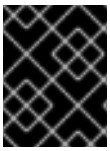
마스터 호스트에서 백업 스크립트를 실행하여 클러스터의 etcd 데이터를 백업합니다. 클러스터의 각 마스터 호스트마다 백업을 수행하지 마십시오.

etcd 백업 후 이전 클러스터 상태로 복원 할 수 있습니다.

etcd 인스턴스가 실행중인 모든 마스터 호스트에서 etcd 데이터 백업 프로세스를 수행할 수 있습니다.

1.1. ETCD 데이터 백업

다음 단계에 따라 etcd 스냅샷을 작성하고 정적 pod의 리소스를 백업하여 etcd 데이터를 백업합니다. 이 백업을 저장하여 etcd를 복원해야하는 경우 나중에 사용할 수 있습니다.



중요

단일 마스터 호스트의 백업만을 저장합니다. 클러스터의 각 마스터 호스트에서 백업을 수행하지 마십시오.

전제 조건

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 클러스터 전체의 프록시가 활성화되어 있는지 확인해야 합니다.

  작은 정보

  oc get proxy cluster -o yaml의 출력을 확인하여 프록시가 사용 가능한지 여부를 확인할 수 있습니다. **httpProxy**, **httpsProxy** 및 **noProxy** 필드에 값이 설정되어 있으면 프록시가 사용됩니다.

프로세스

1. 마스터 노드의 디버그 세션을 시작합니다.

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

2. 루트 디렉토리를 호스트로 변경하십시오.

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

3. 클러스터 전체의 프록시가 활성화되어 있는 경우 **NO_PROXY, HTTP_PROXY** 및 **https_proxy** 환경 변수를 내보내고 있는지 확인합니다.

4. **cluster-backup.sh** 스크립트를 실행하고 백업을 저장할 위치를 입력합니다.

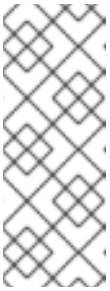
```
sh-4.4# /usr/local/bin/cluster-backup.sh /home/core/assets/backup
```

스크립트 출력 예

```
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
```

이 예제에서는 마스터 호스트의 **/home/core/assets/backup/** 디렉토리에 두 개의 파일이 생성됩니다.

- **snapshot_<datetimestamp>.db**:이 파일은 etcd 스냅샷입니다.
- **static_kuberesources_<datetimestamp>.tar.gz**: 이 파일에는 정적 pod 리소스가 포함되어 있습니다. etcd 암호화가 활성화되어 있는 경우 etcd 스냅 샷의 암호화 키도 포함됩니다.



참고

etcd 암호화가 활성화되어 있는 경우 보안상의 이유로 이 두 번째 파일을 etcd 스냅 샷과 별도로 저장하는 것이 좋습니다. 그러나 이 파일은 etcd 스냅 샷에서 복원하는데 필요합니다.

etcd 암호화는 키가 아닌 값만 암호화합니다. 이는 리소스 유형, 네임 스페이스 및 개체 이름은 암호화되지 않습니다.

2장. 비정상적인 ETCD 멤버 교체

이 문서는 비정상적인 단일 etcd 멤버를 교체하는 프로세스를 설명합니다.

이 프로세스는 시스템이 실행 중이 아니거나 노드가 준비되지 않았거나 etcd pod가 크래시 루프 상태에있는 등 etcd 멤버의 비정상적인 상태에 따라 다릅니다.

참고

대부분의 마스터 호스트가 손실되고 etcd 쿼럼 (정족수)의 손실이 발생한 경우 재해 복구 프로세스에 따라 이전 클러스터 상태로 복원해야 합니다.

교체된 멤버에서 컨트롤 플레인 인증서가 유효하지 않은 경우 이 프로세스 대신 만료된 컨트롤 플레인 인증서 복구절차를 따라야합니다.

마스터 노드가 유실되고 새 노드가 생성되는 경우 etcd 클러스터 Operator는 새 TLS 인증서 생성 및 노드를 etcd 멤버로 추가하는 프로세스를 진행합니다.

2.1. 사전 요구 사항

- 비정상적인 etcd 멤버를 교체하기 전에 etcd 백업을 수행하십시오.

2.2. 비정상 ETCD 멤버 식별

클러스터에 비정상적인 etcd 멤버가 있는지 여부를 확인할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.

프로세스

1. 다음 명령을 사용하여 **EtcdMembersAvailable** 상태의 상태 조건을 확인하십시오.

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

2. 출력을 확인합니다.

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

이 출력 예는 **ip-10-0-131-183.ec2.internal** etcd 멤버가 비정상임을 보여줍니다.

2.3. 비정상적인 ETCD 멤버의 상태 확인

비정상적인 etcd 멤버를 교체하는 프로세스는 etcd가 다음의 어떤 상태에 있는지에 따라 달라집니다.

- 컴퓨터가 실행 중이 아니거나 노드가 준비되지 않았습니다.
- etcd pod가 크래시 루프 상태에 있습니다.

다음 프로세스에서는 etcd 멤버가 어떤 상태에 있는지를 확인합니다. 이를 통해 비정상 etcd 멤버를 대체하기 위해 수행해야하는 단계를 확인할 수 있습니다.

참고

시스템이 실행되고 있지 않거나 노드가 준비되지 않았지만 곧 정상 상태로 돌아올 것으로 예상되는 경우 etcd 멤버를 교체하기 위한 절차를 수행할 필요가 없습니다. etcd 클러스터 Operator는 머신 또는 노드가 정상 상태로 돌아 오면 자동으로 동기화됩니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 비정상적인 etcd 멤버를 식별하고 있습니다.

프로세스

1. **시스템이 실행되고 있지 않은지**를 확인합니다.

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

출력 예

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

(1) 이 출력은 노드와 노드 시스템의 상태를 나열합니다. 상태가 **running**이 아닌 경우 **시스템은 실행되지 않습니다**.

시스템이 실행되고 있지 않은 경우, *시스템이 실행되고 있지 않거나 노드가 준비되지 않은 비정상적인 etcd 멤버 교체* 프로세스를 수행하십시오.

2. **노드가 준비되지 않았는지** 확인합니다.
다음 조건 중 하나에 해당하면 **노드가 준비되지 않은 것**입니다.

- 시스템이 실행중인 경우 노드에 액세스할 수 있는지 확인하십시오.

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

출력 예

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

(1) **unreachable** 상태의 노드가 나열되면 **노드가 준비되지 않은 것**입니다.

- 노드에 여전히 액세스할 수 있는 경우 노드가 **NotReady**로 나열되어 있는지 확인하십시오.

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

출력 예

```
ip-10-0-131-183.ec2.internal   NotReady   master   122m   v1.19.0 (1)
```

1 노드가 **NotReady**로 표시되면 **노드가 준비되지 않은 것**입니다.

노드가 준비되지 않은 경우 *시스템이 실행되고 있지 않거나 노드가 준비되지 않은 비정상적인 etcd 멤버 교체* 프로세스를 수행하십시오.

3. **etcd pod가 크래시 루프 상태**인지 확인합니다.
시스템이 실행되고 있고 노드가 준비된 경우 etcd pod가 크래시 루프 상태인지 확인하십시오.

a. 모든 마스터 노드가 **Ready**로 표시되어 있는지 확인합니다.

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

출력 예

```
NAME                           STATUS   ROLES    AGE     VERSION
ip-10-0-131-183.ec2.internal   Ready    master   6h13m   v1.19.0
ip-10-0-164-97.ec2.internal    Ready    master   6h13m   v1.19.0
ip-10-0-154-204.ec2.internal   Ready    master   6h13m   v1.19.0
```

b. etcd pod의 상태가 **Error** 또는 **CrashloopBackoff**인지 확인하십시오.

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

출력 예

```
etcd-ip-10-0-131-183.ec2.internal           2/3     Error       7          6h9m  1
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
```

1 이 pod의 상태는 **Error**이므로 **etcd pod는 크래시 루프 상태**입니다.

etcd pod가 크래시 루프 상태인 경우 *etcd pod가 크래시 루프 상태인 비정상적인 etcd 멤버 교체* 프로세스를 수행하십시오.

2.4. 비정상적인 ETCD 멤버 교체

비정상적인 etcd 멤버의 상태에 따라 다음 절차 중 하나를 사용합니다.

- 시스템이 실행되고 있지 않거나 노드가 준비되지 않은 비정상적인 etcd 멤버 교체
- etcd pod가 크래시 루프 상태인 비정상적인 etcd 멤버 교체

2.4.1. 시스템이 실행되고 있지 않거나 노드가 준비되지 않은 비정상적인 etcd 멤버 교체

다음에서는 시스템이 실행되고 있지 않거나 노드가 준비되지 않은 경우의 비정상적인 etcd 멤버를 교체하는 프로세스에 대해 자세히 설명합니다.

사전 요구 사항

- 비정상적인 etcd 멤버를 식별했습니다.

- 시스템이 실행되고 있지 않거나 노드가 준비되지 않았음을 확인했습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- etcd 백업이 수행되었습니다.

중요

문제가 발생할 경우 클러스터를 복원할 수 있도록 이 프로세스를 수행하기 전에 etcd 백업을 수행해야합니다.

프로세스

1. 비정상적인 멤버를 제거합니다.
   a. 영향을 받는 노드에 *없는*pod를 선택합니다.
   클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

출력 예

```
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. 실행중인 etcd 컨테이너에 연결하고 영향을 받는 노드에 없는 pod 이름을 전달합니다.
   클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

   c. 멤버 목록을 확인합니다.

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

출력 예

```
+------------------+---------+-----------------------------+---------------------------+---------------
-----------+
|        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 |
+------------------+---------+-----------------------------+---------------------------+---------------
-----------+
```

이러한 값은 프로세스의 뒷부분에서 필요하므로 비정상 etcd 멤버의 ID와 이름을 기록해 두십시오.

d. **etcdctl member remove** 명령에 ID를 지정하여 비정상적인 etcd 멤버를 제거합니다.

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

출력 예

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

e. 멤버 목록을 다시 표시하고 멤버가 제거되었는지 확인합니다.

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

출력 예

```
+------------------+---------+--------------------------+--------------------------+----------------
-----------+
|        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 |
+------------------+---------+--------------------------+--------------------------+----------------
-----------+
```

이제 노드 쉘을 종료할 수 있습니다.

2. 삭제된 비정상 etcd 멤버의 이전 암호를 제거합니다.

a. 삭제된 비정상 etcd 멤버의 시크릿(secrets)을 나열합니다.

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

1 이 프로세스의 앞부분에서 기록한 비정상 etcd 멤버의 이름을 전달합니다.

다음 출력에 표시된대로 피어, 서빙 및 메트릭 시크릿이 있습니다.

출력 예

```
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. 제거된 비정상 etcd 멤버의 시크릿을 삭제합니다.

i. 피어 시크릿을 삭제합니다.

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

ii. 서빙 시크릿을 삭제합니다.

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

iii. 메트릭 시크릿을 삭제합니다.

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

3. 마스터 시스템을 삭제하고 다시 만듭니다. 이 시스템을 다시 만든 후에는 새 버전이 강제 실행되고 etcd는 자동으로 확장됩니다.
설치 프로그램에서 제공한 인프라를 실행 중이거나 Machine API를 사용하여 컴퓨터를 만든 경우 다음 단계를 수행합니다. 그렇지 않으면 원래 마스터를 만들 때 사용한 방법과 동일한 방법을 사용하여 새 마스터를 작성해야합니다.

a. 비정상 멤버의 컴퓨터를 가져옵니다.
클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

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

출력 예

```
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   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 이는 비정상 노드의 마스터 시스템 **ip-10-0-131-183.ec2.internal**입니다.

b. 시스템 설정을 파일 시스템의 파일에 저장합니다.

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

1 비정상 노드의 마스터 시스템의 이름을 지정합니다.

c. 이전 단계에서 만든 **new-master-machine.yaml** 파일을 편집하여 새 이름을 할당하고 불필요한 필드를 제거합니다.

i. 전체 **status** 섹션을 삭제합니다.

```
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** 필드를 새 이름으로 변경합니다.
이전 시스템과 동일한 기본 이름을 유지하고 마지막 번호를 사용 가능한 다음 번호로 변경하는 것이 좋습니다. 이 예에서 **clustername-8qw5l-master-0**을 **clustername-8qw5l-master-3**으로 변경되어 있습니다.

예를 들면 다음과 같습니다.

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

iii. 이전 단계의 새 시스템 이름을 사용하도록 **metadata.selfLink** 필드를 업데이트합니다.

```
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** 필드를 삭제합니다.

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

v. **metadata.annotations** 및 **metadata.generation** 필드를 제거합니다.

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

vi. **metadata.resourceVersion** 및 **metadata.uid** 필드를 제거합니다.

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

d. 비정상 멤버의 시스템을 삭제합니다.

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

1 비정상 노드의 마스터 시스템의 이름을 지정합니다.

e. 시스템이 삭제되었는지 확인합니다.

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

출력 예

```
NAME                                 PHASE     TYPE        REGION      ZONE         AGE
NODE                           PROVIDERID                          STATE
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   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
```

f. **new-master-machine.yaml** 파일을 사용하여 새 시스템을 만듭니다.

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

g. 새 시스템이 생성되었는지 확인합니다.

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

출력 예

```
NAME                                PHASE         TYPE        REGION      ZONE         AGE
NODE                           PROVIDERID                          STATE
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-master-3          Provisioning  m4.xlarge   us-east-1   us-east-1a
85s     ip-10-0-133-53.ec2.internal    aws:///us-east-1a/i-015b0888fe17bc2c8   running
(1)
clustername-8qw5l-worker-us-east-1a-wbtgd  Running     m4.large    us-east-1   us-
east-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) 새 시스템 **clustername-8qw5l-master-3**이 생성되고 단계가 **Provisioning**에서 **Running**으로 변경되면 시스템이 준비 상태가 됩니다.

새 시스템을 만드는 데 몇 분이 소요될 수 있습니다. etcd 클러스터 Operator는 머신 또는 노드가 정상 상태로 돌아 오면 자동으로 동기화됩니다.

검증

1. 모든 etcd pod가 올바르게 실행되고 있는지 확인합니다.
   클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

출력 예

```
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
```

이전 명령의 출력에 두 개의 pod만 나열되는 경우 수동으로 etcd 재배포를 강제 수행할 수 있습니다. 클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

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

(1) **forceRedeploymentReason** 값은 고유해야하므로 타임 스탬프가 추가됩니다.

2. 정확히 세 개의 etcd 멤버가 있는지 확인합니다.

   a. 실행중인 etcd 컨테이너에 연결하고 영향을 받는 노드에 없는 pod 이름을 전달합니다.
   클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

b. 멤버 목록을 확인합니다.

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

출력 예

```
+------------------+---------+--------------------------+--------------------------+--------------
----------+
|        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 |
+------------------+---------+--------------------------+--------------------------+--------------
----------+
```

이전 명령의 출력에 세 개 이상의 etcd 멤버가 나열된 경우 원하지 않는 멤버를 신중하게 제거해야 합니다.

주의

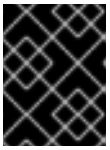
올바른 etcd 멤버를 제거하십시오. 좋은 etcd 멤버를 제거하면 쿼럼 손실이 발생할 수 있습니다.

2.4.2. etcd pod가 크래시 루프 상태인 비정상적인 etcd 멤버 교체

이 단계에서는 etcd pod가 크래시 루프 상태에 있는 경우 비정상 etcd 멤버를 교체하는 방법을 설명합니다.

전제 조건

- 비정상적인 etcd 멤버를 식별했습니다.
- etcd pod가 크래시 루프 상태에 있는것으로 확인되었습니다.
- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- etcd 백업이 수행되었습니다.



중요

문제가 발생할 경우 클러스터를 복원할 수 있도록 이 프로세스를 수행하기 전에 etcd 백업을 수행해야합니다.

프로세스

1. 크래시 루프 상태에 있는 etcd pod를 중지합니다.

   a. 크래시 루프 상태의 노드를 디버깅합니다.
   클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

   ```
   $ oc debug node/ip-10-0-131-183.ec2.internal 1
   ```

   1 이를 비정상 노드의 이름으로 변경합니다.

   b. 루트 디렉토리를 호스트로 변경하십시오.

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

   c. kubelet 매니페스트 디렉토리에서 기존 etcd pod 파일을 이동합니다.

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

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

   d. etcd 데이터 디렉토리를 다른 위치로 이동합니다.

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

   이제 노드 쉘을 종료할 수 있습니다.

2. 비정상적인 멤버를 제거합니다.

   a. 영향을 받는 노드에 *없는* pod를 선택합니다.
   클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

   ```
   $ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
   ```

   출력 예

   ```
   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. 실행중인 etcd 컨테이너에 연결하고 영향을 받는 노드에 없는 pod 이름을 전달합니다.
   클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

   ```
   $ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
   ```

   c. 멤버 목록을 확인합니다.

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

   출력 예

```
+-----------------+---------+----------------------------+-------------------------+---------------
----------+
|        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 |
+-----------------+---------+----------------------------+-------------------------+---------------
----------+
```

이러한 값은 프로세스의 뒷부분에서 필요하므로 비정상 etcd 멤버의 ID와 이름을 기록해 두십시오.

d. **etcdctl member remove** 명령에 ID를 지정하여 비정상적인 etcd 멤버를 제거합니다.

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

출력 예

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

e. 멤버 목록을 다시 표시하고 멤버가 제거되었는지 확인합니다.

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

출력 예

```
+-----------------+---------+----------------------------+-------------------------+---------------
----------+
|        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 |
+-----------------+---------+----------------------------+-------------------------+---------------
----------+
```

이제 노드 쉘을 종료할 수 있습니다.

3. 삭제된 비정상 etcd 멤버의 이전 암호를 제거합니다.

a. 삭제된 비정상 etcd 멤버의 시크릿(secrets)을 나열합니다.

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

1 이 프로세스의 앞부분에서 기록한 비정상 etcd 멤버의 이름을 전달합니다.

다음 출력에 표시된대로 피어, 서빙 및 메트릭 시크릿이 있습니다.

출력 예

```
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. 제거된 비정상 etcd 멤버의 시크릿을 삭제합니다.

i. 피어 시크릿을 삭제합니다.

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

ii. 서빙 시크릿을 삭제합니다.

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

iii. 메트릭 시크릿을 삭제합니다.

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

4. etcd를 강제로 재배포합니다.
클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

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

1 **forceRedeploymentReason** 값은 고유해야하므로 타임 스탬프가 추가됩니다.

etcd 클러스터 Operator가 재배포를 수행하면 모든 마스터 노드에서 etcd pod가 작동하는지 확인합니다.

검증

- 새 멤버가 사용 가능하고 정상적인 상태에 있는지 확인합니다.

a. 실행중인 etcd 컨테이너에 다시 연결합니다.
cluster-admin 사용자로 클러스터에 액세스할 수 있는 터미널에서 다음 명령을 실행합니다.

```
$ oc rsh -n openshift-etcd etcd-ip-10-0-154-204.ec2.internal
```

b. 모든 멤버가 정상인지 확인합니다.

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

출력 예

```
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
```

3장. 클러스터를 안전하게 종료

이 문서에서는 클러스터를 안전하게 종료하는 프로세스를 설명합니다. 유지 관리를 위해 또는 리소스 비용을 절약하기 위해 일시적으로 클러스터를 종료해야 할 수 있습니다.

3.1. 전제 조건

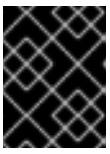
- 클러스터를 종료하기 전에 etcd 백업을 만듭니다.

3.2. 클러스터 종료

나중에 클러스터를 다시 시작하기 위해 안전한 방법으로 클러스터를 종료할 수 있습니다.

전제 조건

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- etcd 백업이 수행되었습니다.



중요

클러스터를 다시 시작할 때 문제가 발생할 경우 클러스터를 복원 할 수 있도록 이 단계를 수행하기 전에 etcd 백업을 해 두는 것이 중요합니다.

프로세스

1. 클러스터의 모든 노드를 종료합니다. 클라우드 제공 업체의 웹 콘솔에서 이 작업을 수행하거나 다음 명령을 사용할 수 있습니다.

   a. 노드 목록을 가져옵니다.

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

   b. 모든 노드를 종료합니다.

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

   이러한 방법 중 하나를 사용하여 노드를 종료하면 pod가 정상적으로 종료되어 데이터 손상 가능성을 줄일 수 있습니다.

참고

종료하기 전에 OpenShift Container Platform과 함께 제공되는 표준 pod의 마스터 노드를 드레인할 필요가 없습니다.

클러스터 관리자는 클러스터를 다시 시작한 후 워크로드를 완전히 다시 시작해야 합니다. 사용자 지정 워크로드로 인해 종료하기 전에 마스터 노드를 드레이한 경우 다시 시작한 후 클러스터가 다시 작동하기 전에 마스터 노드를 스케줄 대상으로 표시해야합니다.

2. 외부 스토리지 또는 LDAP 서버와 같이 더 이상 필요하지 않은 클러스터 종속성을 중지합니다. 이 작업을 수행하기 전에 공급 업체의 설명서를 확인하십시오.

추가 리소스

- 클러스터를 정상적으로 다시 시작

4장. 클러스터를 정상적으로 다시 시작

이 문서에서는 정상 종료 후 클러스터를 다시 시작하는 프로세스에 대해 설명합니다.

다시 시작한 후 클러스터가 정상적으로 작동할 것으로 예상되지만 예상치 못한 상황으로 인해 클러스터가 복구되지 않을 수 있습니다. 예를 들면 다음과 같습니다.

- 종료 중 etcd 데이터 손상
- 하드웨어로 인한 노드 오류
- 네트워크 연결 문제

클러스터를 복구할 수 없는 경우 다음 단계에 따라이전 클러스터 상태로 복원 합니다.

4.1. 전제 조건

- 클러스터가 정상적으로 종료되었습니다.

4.2. 클러스터를 다시 시작

클러스터가 정상적으로 종료된 후 클러스터를 다시 시작할 수 있습니다.

전제 조건

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 이 프로세스에서는 클러스터를 정상적으로 종료하고 있는 것을 전제로 하고 있습니다.

프로세스

1. 외부 스토리지 또는 LDAP 서버와 같은 클러스터의 종속 장치를 시작합니다.
2. 모든 클러스터 시스템을 시작합니다.
   클라우드 제공 업체의 웹 콘솔에서 시스템을 시작하는 것과 같이 클라우드 환경에 적합한 방법을 사용하여 시스템을 시작합니다.

   약 10 분 정도 기다린 후 마스터 노드의 상태를 확인합니다.
3. 모든 마스터 노드가 준비 상태인지 확인합니다.

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

   다음 출력에 표시된 대로 마스터 노드의 상태가 **Ready**인 경우 마스터 노드가 준비되었음을 의미합니다.

   ```
   NAME                          STATUS   ROLES    AGE   VERSION
   ip-10-0-168-251.ec2.internal   Ready    master   75m   v1.19.0
   ip-10-0-170-223.ec2.internal   Ready    master   75m   v1.19.0
   ip-10-0-211-16.ec2.internal    Ready    master   75m   v1.19.0
   ```
4. 마스터 노드가 준비 *되지 않은* 경우 승인해야하는 보류중인 인증서 서명 요청(CSR)이 있는지 확인합니다.

a. 현재 CSR의 목록을 가져옵니다.

```
$ oc get csr
```

b. CSR의 세부 사항을 검토하여 CSR이 유효한지 확인합니다.

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

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

c. 각각의 유효한 CSR을 승인합니다.

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

5. 마스터 노드가 준비되면 모든 작업자 노드가 준비되었는지 확인합니다.

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

다음 출력에 표시된 대로 작업자 노드의 상태가 **Ready**인 경우 작업자 노드는 준비된 것입니다.

```
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-179-95.ec2.internal    Ready    worker   64m   v1.19.0
ip-10-0-182-134.ec2.internal   Ready    worker   64m   v1.19.0
ip-10-0-250-100.ec2.internal   Ready    worker   64m   v1.19.0
```

6. 작업자 노드가 준비*되지 않은* 경우 승인해야하는 보류중인 인증서 서명 요청(CSR)이 있는지 확인합니다.

a. 현재 CSR의 목록을 가져옵니다.

```
$ oc get csr
```

b. CSR의 세부 사항을 검토하여 CSR이 유효한지 확인합니다.

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

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

c. 각각의 유효한 CSR을 승인합니다.

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

7. 클러스터가 제대로 시작되었는지 확인합니다.

a. 성능이 저하된 클러스터 Operator가 없는지 확인합니다.

```
$ oc get clusteroperators
```

DEGRADED 조건이 **True**로 설정된 클러스터 Operator가 없는지 확인합니다.

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
```

```
SINCE
authentication                             4.6.0     True        False         False      59m
cloud-credential                           4.6.0     True        False         False      85m
cluster-autoscaler                         4.6.0     True        False         False      73m
config-operator                            4.6.0     True        False         False      73m
console                                    4.6.0     True        False         False      62m
csi-snapshot-controller                    4.6.0     True        False         False      66m
dns                                        4.6.0     True        False         False      76m
etcd                                       4.6.0     True        False         False      76m
...
```

b. 모든 노드가 **Ready** 상태에 있는지 확인합니다.

```
$ oc get nodes
```

모든 노드의 상태가 **Ready** 상태인지 확인합니다.

```
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.19.0
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.19.0
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.19.0
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.19.0
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.19.0
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.19.0
```

클러스터가 제대로 시작되지 않은 경우 etcd 백업을 사용하여 클러스터를 복원해야 할 수 있습니다.

추가 리소스

- 클러스터를 다시 시작한 후 복구하지 못한 경우 etcd 백업을 사용하여 복원하는 방법은이전 클러스터 상태로 복구에서 참조하십시오.

5장. 재해 복구

5.1. 재해 복구 정보

재해 복구 문서에서는 관리자에게 OpenShift Container Platform 클러스터에서 발생할 수있는 여러 재해 상황을 복구하는 방법에 대한 정보를 제공합니다. 관리자는 클러스터를 작동 상태로 복원하려면 다음 절차 중 하나 이상을 수행해야합니다.

이전 클러스터 상태로 복구

클러스터를 이전 상태로 복원하려는 경우 (예: 관리자가 일부 주요 정보를 삭제한 경우) 이 솔루션을 사용할 수 있습니다. 이에는 대부분의 마스터 호스트가 손실되고 etcd 쿼럼이 손실되고 클러스터가 오프라인인 상태에서도 사용할 수 있습니다. etcd 백업을 수행한 경우 이 절차에 따라 클러스터를 이전 상태로 복원할 수 있습니다.
해당되는 경우 만료 된 컨트롤 플레인 인증서 복구 를 수행해야 할 수도 있습니다.

참고

대다수의 마스터를 여전히 사용할 수 있고 etcd 쿼럼이 있는 경우, 절차에 따라 비정상적인 단일 etcd 멤버 교체를 실행합니다.

만료된 컨트롤 플레인 인증서 복구

컨트롤 플레인 인증서가 만료된 경우 이 솔루션을 사용할 수 있습니다. 예를 들어, 설치 후 24 시간 내에 발생하는 첫 번째 인증서 교체 전에 클러스터를 종료하면 인증서가 교체되지 않고 만료됩니다. 다음 단계에 따라 만료된 컨트롤 플레인 인증서를 복구할 수 있습니다.

5.2. 손실된 마스터 호스트에서 복구

OpenShift Container Platform 4.4부터 손실된 마스터 호스트에서 복구하려면 다음의 이전 클러스터 상태로 복구 단계를 수행합니다.

참고

대다수의 마스터를 여전히 사용할 수 있고 etcd 쿼럼이 있는 경우, 절차에 따라 비정상적인 단일 etcd 멤버 교체를 실행합니다.

5.3. 이전 클러스터 상태로 복구

클러스터를 이전 상태로 복구하려면 스냅샷을 작성하여 etcd 데이터 백업을 수행해야합니다. 이 스냅샷을 사용하여 클러스터 상태를 복구합니다.

5.3.1. 이전 클러스터 상태로 복구

저장된 etcd 백업을 사용하여 이전 클러스터 상태로 다시 복원할 수 있습니다. etcd 백업을 사용하여 단일 컨트롤 패널 호스트를 복원합니다. 다음으로 etcd 클러스터 Operator는 나머지 마스터 호스트의 스케일링을 처리합니다.

중요

클러스터를 복원할 때 동일한 z-stream 릴리스에서 가져온 etcd 백업을 사용해야 합니다. 예를 들어 OpenShift Container Platform 4.6.2 클러스터는 4.6.2에서 가져온 etcd 백업을 사용해야 합니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- 마스터 호스트에 대한 SSH 액세스 권한이 있어야 합니다.
- 동일한 백업에서 가져온 etcd 스냅샷과 정적 pod 리소스가 모두 포함된 백업 디렉토리입니다. 디렉토리의 파일 이름은 **snapshot_<datetimestamp>.db** 및 **static_kuberesources_<datetimestamp>.tar.gz** 형식이어야합니다.

프로세스

1. 복구 호스트로 사용할 컨트롤 플레인 호스트를 선택합니다. 이는 복구 작업을 실행할 호스트입니다.
2. 복구 호스트를 포함하여 각 컨트롤 플레인 노드에 SSH 연결을 설정합니다.
   복구 프로세스가 시작된 후에는 Kubernetes API 서버에 액세스할 수 없으므로 컨트롤 플레인 노드에 액세스할 수 없습니다. 따라서 다른 터미널에서 각 컨트롤 플레인 호스트에 대한 SSH 연결을 설정하는 것이 좋습니다.

   중요

   이 단계를 완료하지 않으면 마스터 호스트에 액세스하여 복구 프로세스를 완료할 수 없으며 이 상태에서 클러스터를 복구할 수 없습니다.

3. etcd 백업 디렉토리를 복구 컨트롤 플레인 호스트에 복사합니다.
   이 단계에서는 etcd 스냅샷 및 정적 pod의 리소스가 포함된 **backup** 디렉터리를 복구 컨트롤 플레인 호스트의 **/home/core/** 디렉터리에 복사하는 것을 전제로하고 있습니다.
4. 다른 모든 컨트롤 플레인 노드에서 고정 pod를 중지합니다.

   참고

   복구 호스트에서 pod를 수동으로 중지할 필요는 없습니다. 복구 스크립트는 복구 호스트에서 pod를 중지합니다.

   a. 복구 호스트가 아닌 컨트롤 플레인 호스트에 액세스합니다.

   b. kubelet 매니페스트 디렉토리에서 기존 etcd pod 파일을 이동합니다.

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

   c. etcd pod가 중지되었는지 확인합니다.

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

   이 명령의 출력은 비어 있어야합니다. 비어 있지 않은 경우 몇 분 기다렸다가 다시 확인하십시오.

   d. kubelet 매니페스트 디렉토리에서 기존 Kubernetes API 서버 pod 파일을 이동합니다.

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

e. Kubernetes API 서버 pod가 중지되었는지 확인합니다.

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

이 명령의 출력은 비어 있어야합니다. 비어 있지 않은 경우 몇 분 기다렸다가 다시 확인하십시오.

f. etcd 데이터 디렉토리를 다른 위치로 이동합니다.

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

g. 복구 호스트가 아닌 다른 마스터 호스트에서 이 단계를 반복합니다.

5. 복구 컨트롤 플레인 호스트에 액세스합니다.

6. 클러스터 전체의 프록시가 활성화되어 있는 경우 **NO_PROXY, HTTP_PROXY** 및 **https_proxy** 환경 변수를 내보내고 있는지 확인합니다.

작은 정보

oc get proxy cluster -o yaml의 출력을 확인하여 프록시가 사용 가능한지 여부를 확인할 수 있습니다. **httpProxy**, **httpsProxy** 및 **noProxy** 필드에 값이 설정되어 있으면 프록시가 사용됩니다.

7. 복구 컨트롤 플레인 호스트에서 복원 스크립트를 실행하고 etcd 백업 디렉터리에 경로를 전달합니다.

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

스크립트 출력 예

```
...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. 모든 마스터 호스트에서 kubelet 서비스를 다시 시작합니다.

   a. 복구 호스트에서 다음 명령을 실행합니다.

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

   b. 다른 모든 마스터 호스트에서 이 단계를 반복합니다.

9. 단일 멤버 컨트롤 플레인이 제대로 시작되었는지 확인합니다.

   a. 복구 호스트에서 etcd 컨테이너가 실행 중인지 확인합니다.

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

   출력 예

   ```
   3ad41b7908e32
   36f86e2eeaaffe662df0d21041eb22b8198e0e58abeeae8c743c3e6e977e8009
   About a minute ago   Running             etcd                                0
   7c05f8af362f0
   ```

   b. 복구 호스트에서 etcd pod가 실행 중인지 확인합니다.

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

   참고

   이 명령을 실행하기 전에 **oc login**을 실행하여 다음 오류가 발생하면 인증 컨트롤러가 시작될 때까지 잠시 기다렸다가 다시 시도하십시오.

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

   출력 예

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

   Pending 상태에 있거나 출력에 여러 실행중인 etcd pod가 나열되어 있는 경우 몇 분 기다렸다가 다시 확인합니다.

10. etcd를 강제로 재배포합니다.
    클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

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

    1 **forceRedeploymentReason** 값은 고유해야하므로 타임 스탬프가 추가됩니다.

    etcd 클러스터 Operator가 재배포를 실행하면 기존 노드가 초기 부트 스트랩 확장과 유사한 새 pod를 사용하기 시작합니다.

11. 모든 노드가 최신 버전으로 업데이트되었는지 확인합니다.
클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

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

etcd의 **NodeInstallerProgressing** 상태 조건을 확인하고 모든 노드가 최신 버전인지 확인합니다. 업데이트가 성공적으로 실행되면 출력에 **AllNodesAtLatestRevision**이 표시됩니다.

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

(1) 이 예에서 최신 버전 번호는 **7**입니다.

출력에 **2 nodes are at revision 6; 1 nodes are at revision 7**와 같은 여러 버전 번호가 표시되면 이는 업데이트가 아직 진행 중임을 의미합니다. 몇 분 기다린 후 다시 시도합니다.

12. etcd를 재배포한 후 컨트롤 플레인에 새 롤아웃을 강제 실행합니다. kubelet이 내부 로드 밸런서를 사용하여 API 서버에 연결되어 있으므로 Kubernetes API 서버는 다른 노드에 다시 설치됩니다.
cluster-admin 사용자로 클러스터에 액세스할 수있는 터미널에서 다음 명령을 실행합니다.

a. **kubeapiserver**를 업데이트합니다.

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

모든 노드가 최신 버전으로 업데이트되었는지 확인합니다.

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

NodeInstallerProgressing 상태 조건을 확인하고 모든 노드가 최신 버전인지 확인합니다. 업데이트가 성공적으로 실행되면 출력에 **AllNodesAtLatestRevision**이 표시됩니다.

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

(1) 이 예에서 최신 버전 번호는 **7**입니다.

출력에 **2 nodes are at revision 6; 1 nodes are at revision 7**와 같은 여러 버전 번호가 표시되면 이는 업데이트가 아직 진행 중임을 의미합니다. 몇 분 기다린 후 다시 시도합니다.

b. **kubecontrollermanager**를 업데이트합니다.

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

모든 노드가 최신 버전으로 업데이트되었는지 확인합니다.

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

NodeInstallerProgressing 상태 조건을 확인하고 모든 노드가 최신 버전인지 확인합니다. 업데이트가 성공적으로 실행되면 출력에 **AllNodesAtLatestRevision**이 표시됩니다.

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

(1) 이 예에서 최신 버전 번호는 **7**입니다.

출력에 **2 nodes are at revision 6; 1 nodes are at revision 7**와 같은 여러 버전 번호가 표시되면 이는 업데이트가 아직 진행 중임을 의미합니다. 몇 분 기다린 후 다시 시도합니다.

c. **kubescheduler**를 업데이트합니다.

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

모든 노드가 최신 버전으로 업데이트되었는지 확인합니다.

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

NodeInstallerProgressing 상태 조건을 확인하고 모든 노드가 최신 버전인지 확인합니다. 업데이트가 성공적으로 실행되면 출력에 **AllNodesAtLatestRevision**이 표시됩니다.

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

(1) 이 예에서 최신 버전 번호는 **7**입니다.

출력에 **2 nodes are at revision 6; 1 nodes are at revision 7**와 같은 여러 버전 번호가 표시되면 이는 업데이트가 아직 진행 중임을 의미합니다. 몇 분 기다린 후 다시 시도합니다.

13. 모든 마스터 호스트가 클러스터를 시작하여 참여하고 있는지 확인합니다.
클러스터에 액세스할 수 있는 터미널에서 **cluster-admin** 사용자로 다음 명령을 실행합니다.

```
$ oc get pods -n openshift-etcd | grep -v etcd-quorum-guard | grep etcd
```

출력 예

```
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
```

이 프로세스를 완료한 후 모든 서비스를 복구하는데 몇 분 정도 걸릴 수 있습니다. 예를 들어, OAuth 서버 pod가 다시 시작될 때까지 **oc login**을 사용한 인증이 즉시 작동하지 않을 수 있습니다.

5.4. 만료된 컨트롤 플레인 인증서에서 복구

5.4.1. 만료된 컨트롤 플레인 인증서에서 복구

OpenShift Container Platform 4.4.8에서 클러스터는 만료된 컨트롤 플레인 인증서에서 자동으로 복구될 수 있습니다. 더 이상 이전 버전에서 필요했던 수동 단계를 수행할 필요가 없습니다.

예외적으로 kubelet 인증서를 복구하려면 대기 중인 **node-bootstrapper** 인증서 서명 요청 (CSR)을 수동으로 승인해야합니다.

보류중인 **node-bootstrapper** CSR을 승인하려면 다음 단계를 수행합니다.

프로세스

1. 현재 CSR의 목록을 가져옵니다.

```
$ oc get csr
```

2. CSR의 세부 사항을 검토하여 CSR이 유효한지 확인합니다.

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

1 **<csr_name>**은 현재 CSR 목록에 있는 CSR의 이름입니다.

3. 각각의 유효한 **node-bootstrapper** CSR을 승인합니다.

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