OpenShift Container Platform 4.9

备份和恢复

备份和恢复 OpenShift Container Platform 集群

Last Updated: 2022-01-15

OpenShift Container Platform 4.9 备份和恢复

备份和恢复 OpenShift Container Platform 集群

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.

法律通告

摘要

本文档提供了备份集群数据以及从各种灾难场景中恢复的步骤。

目录

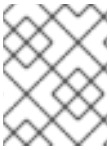
第 1 章 备份和恢复

1.1. OPENSHIFT CONTAINER PLATFORM 中备份和恢复操作概述

作为集群管理员，您可能需要在一段时间内停止 OpenShift Container Platform 集群，并在以后重启集群。重启集群的一些原因是您需要对集群执行维护或希望降低资源成本。在 OpenShift Container Platform 中，您可以 对集群执行安全关闭，以便在以后轻松重启集群。

您必须在关闭集群前 备份 etcd 数据 ；etcd 是 OpenShift Container Platform 的键值存储，它会保留所有资源对象的状态。etcd 备份在灾难恢复中扮演着关键角色。在 OpenShift Container Platform 中，您还可以 替换不健康的 etcd 成员。

当您希望集群再次运行时， 请安全地重启集群。



注意

集群的证书在安装日期后一年后过期。您可以关闭集群，并在证书仍有效时安全地重启集群。虽然集群自动检索过期的 control plane 证书，但您仍需要 批准证书签名请求(CSR)。

您可能会遇到 OpenShift Container Platform 无法按预期工作的一些情况，例如：

- 您有一个在重启后无法正常工作的集群，因为意外状况（如节点故障或网络连接问题）无法正常工作。
- 您已错误地删除了集群中的某些关键内容。
- 您丢失了大多数 control plane 主机，从而导致 etcd 仲裁丢失。

通过使用保存的 etcd 快照，始终可以通过将 集群恢复到之前的状态 来从灾难中恢复。

第 2 章 安全地关闭集群

本文档描述了安全关闭集群的过程。出于维护或者节约资源成本的原因，您可能需要临时关闭集群。

2.1. 先决条件

- 在关闭集群前进行 etcd 备份。

2.2. 关闭集群

您可以以安全的方式关闭集群，以便稍后重启集群。

注意

您可以在安装日期起的一年内关闭集群，并期望它可以正常重启。安装日期起一年后，集群证书会过期。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已进行 etcd 备份。

重要

执行此流程前务必要进行 etcd 备份，以便在重启集群遇到任何问题时可以恢复集群。

流程

1. 如果您要长时间关闭集群，请确定证书过期的日期。

```
$ oc -n openshift-kube-apiserver-operator get secret kube-apiserver-to-kubelet-signer -o jsonpath='{.metadata.annotations.auth\.openshift\.io/certificate-not-after}'
```

输出示例

```
2022-08-05T14:37:50Zuser@user:~ $ 1
```

1 为确保集群可以正常重启，请计划在指定的日期或之前重启集群。当集群重启时，可能需要您手动批准待处理的证书签名请求 (CSR) 来恢复 kubelet 证书。

2. 关闭集群中的所有节点。您可以从云供应商的 Web 控制台完成此操作，或者运行以下循环：

```
$ for node in $(oc get nodes -o jsonpath='{.items[*].metadata.name}'); do oc debug node/${node} -- chroot /host shutdown -h 1; done
```

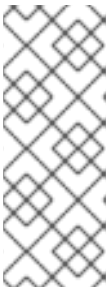
输出示例

```
Starting pod/ip-10-0-130-169us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
```

```
Shutdown scheduled for Mon 2021-09-13 09:36:17 UTC, use 'shutdown -c' to cancel.

Removing debug pod ...
Starting pod/ip-10-0-150-116us-east-2computeinternal-debug ...
To use host binaries, run `chroot /host`
Shutdown scheduled for Mon 2021-09-13 09:36:29 UTC, use 'shutdown -c' to cancel.
```

使用以下方法关闭节点可让 pod 安全终止，从而减少数据崩溃的可能性。



注意

在关闭前，不需要排空 OpenShift Container Platform 中附带的标准 pod 的 control plane 节点。

集群管理员负责确保在集群重启后，彻底重启自己的工作负载。如果因为自定义工作负载的原因已在关闭前排空 control plane 节点，您必须在重启后将 control plane 节点标记为可调度，然后集群才可以重新正常工作。

3. 关闭不再需要的集群依赖项，如外部存储或 LDAP 服务器。在进行操作前请务必查阅您的厂商文档。

其他资源

- 正常重启集群

第 3 章 正常重启集群

本文档论述了在安全关闭后重启集群的过程。

尽管在重启后集群应该可以正常工作，但可能会因为意外状况集群可能无法恢复，例如：

- 关机过程中的 etcd 数据崩溃
- 因硬件原因造成节点故障
- 网络连接问题。

如果集群无法恢复，请按照以下步骤恢复到以前的集群状态。

3.1. 先决条件

- 已安全关闭集群。

3.2. 重启集群

您可以在集群被安全关闭后重启它。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 此流程假设您安全关闭集群。

流程

1. 启动所有依赖设备，如外部存储或 LDAP 服务器。
2. 启动所有集群机器。
   使用适合您的云环境的方法启动机器，例如从云供应商的 Web 控制台启动机器。

   等待大约 10 分钟，然后继续检查 control plane 节点的状态。
3. 验证所有 control plane 节点都已就绪。

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

   如果状态为 **Ready**，如以下输出中所示，则代表 control plane 节点已就绪：

   ```
   NAME                          STATUS   ROLES    AGE   VERSION
   ip-10-0-168-251.ec2.internal   Ready    master   75m   v1.22.1
   ip-10-0-170-223.ec2.internal   Ready    master   75m   v1.22.1
   ip-10-0-211-16.ec2.internal    Ready    master   75m   v1.22.1
   ```
4. 如果 control plane 节点*没有*就绪，请检查是否有待批准的证书签名请求 (CSR)。
   a. 获取当前 CSR 列表。

      ```
      $ oc get csr
      ```

b. 查看一个 CSR 的详细信息以验证其是否有效：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

c. 批准每个有效的 CSR：

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

5. 在 control plane 节点就绪后，验证所有 worker 节点是否已就绪。

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

如果状态为 **Ready**，如下所示，则代表 worker 节点已就绪：

```
NAME                         STATUS   ROLES    AGE   VERSION
ip-10-0-179-95.ec2.internal  Ready    worker   64m   v1.22.1
ip-10-0-182-134.ec2.internal Ready    worker   64m   v1.22.1
ip-10-0-250-100.ec2.internal Ready    worker   64m   v1.22.1
```

6. 如果 worker 节点 *未* 就绪，请检查是否有待批准的证书签名请求(CSR)。

a. 获取当前 CSR 列表。

```
$ oc get csr
```

b. 查看一个 CSR 的详细信息以验证其是否有效：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

c. 批准每个有效的 CSR：

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

7. 验证集群是否已正确启动。

a. 检查是否有降级的集群 Operator。

```
$ oc get clusteroperators
```

确定没有 **DEGRADED** 条件为 **True** 的集群 Operator。

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE
authentication                             4.9.0     True        False         False     59m
cloud-credential                           4.9.0     True        False         False     85m
cluster-autoscaler                         4.9.0     True        False         False     73m
config-operator                            4.9.0     True        False         False     73m
```

```
console                                    4.9.0     True        False         False      62m
csi-snapshot-controller                    4.9.0     True        False         False      66m
dns                                        4.9.0     True        False         False      76m
etcd                                       4.9.0     True        False         False      76m
...
```

b. 检查所有节点是否处于 **Ready** 状态：

```
$ oc get nodes
```

检查所有节点的状态是否为 **Ready**。

```
NAME                           STATUS   ROLES    AGE   VERSION
ip-10-0-168-251.ec2.internal   Ready    master   82m   v1.22.1
ip-10-0-170-223.ec2.internal   Ready    master   82m   v1.22.1
ip-10-0-179-95.ec2.internal    Ready    worker   70m   v1.22.1
ip-10-0-182-134.ec2.internal   Ready    worker   70m   v1.22.1
ip-10-0-211-16.ec2.internal    Ready    master   82m   v1.22.1
ip-10-0-250-100.ec2.internal   Ready    worker   69m   v1.22.1
```

如果集群无法正确启动，您可能需要使用 etcd 备份来恢复集群。

其他资源

- 如果集群重启后无法恢复，请参阅恢复到以前的集群状态。

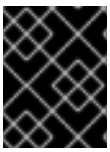
第 4 章 CONTROL PLANE 备份和恢复

4.1. 备份 ETCD

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

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

确保升级集群后执行 etcd 备份。这很重要，因为当恢复集群时，必须使用从同一 z-stream 发行版本中获取的 etcd 备份。例如，OpenShift Container Platform 4.y.z 集群必须使用从 4.y.z 中获得的 etcd 备份。



重要

通过在 control plane 主机上执行一次备份脚本来备份集群的 etcd 数据。不要为每个 control plane 主机进行备份。

在进行了 etcd 备份后，就可以恢复到一个以前的集群状态。

4.1.1. 备份 etcd 数据

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

重要

只保存单一 control plane 主机的备份。不要从集群中的每个 control plane 主机进行备份。

先决条件

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

提示

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

流程

1. 为 control plane 节点启动一个 debug 会话：

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

2. 将您的根目录改为主机：

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

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

4. 运行 **cluster-backup.sh** 脚本，输入保存备份的位置。

提示

cluster-backup.sh 脚本作为 etcd Cluster Operator 的一个组件被维护，它是 **etcdctl snapshot save** 命令的包装程序（wrapper）。

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

脚本输出示例

```
found latest kube-apiserver: /etc/kubernetes/static-pod-resources/kube-apiserver-pod-6
found latest kube-controller-manager: /etc/kubernetes/static-pod-resources/kube-controller-manager-pod-7
found latest kube-scheduler: /etc/kubernetes/static-pod-resources/kube-scheduler-pod-6
found latest etcd: /etc/kubernetes/static-pod-resources/etcd-pod-3
ede95fe6b88b87ba86a03c15e669fb4aa5bf0991c180d3c6895ce72eaade54a1
etcdctl version: 3.4.14
API version: 3.4
{"level":"info","ts":1624647639.0188997,"caller":"snapshot/v3_snapshot.go:119","msg":"created temporary db file","path":"/home/core/assets/backup/snapshot_2021-06-25_190035.db.part"}
{"level":"info","ts":"2021-06-25T19:00:39.030Z","caller":"clientv3/maintenance.go:200","msg":"opened snapshot stream; downloading"}
{"level":"info","ts":1624647639.0301006,"caller":"snapshot/v3_snapshot.go:127","msg":"fetching snapshot","endpoint":"https://10.0.0.5:2379"}
{"level":"info","ts":"2021-06-25T19:00:40.215Z","caller":"clientv3/maintenance.go:208","msg":"completed snapshot read; closing"}
{"level":"info","ts":1624647640.6032252,"caller":"snapshot/v3_snapshot.go:142","msg":"fetched snapshot","endpoint":"https://10.0.0.5:2379","size":"114 MB","took":1.584090459}
{"level":"info","ts":1624647640.6047094,"caller":"snapshot/v3_snapshot.go:152","msg":"saved","path":"/home/core/assets/backup/snapshot_2021-06-25_190035.db"}
Snapshot saved at /home/core/assets/backup/snapshot_2021-06-25_190035.db
{"hash":3866667823,"revision":31407,"totalKey":12828,"totalSize":114446336}
snapshot db and kube resources are successfully saved to /home/core/assets/backup
```

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

- **snapshot_<datetimestamp>.db**：这个文件是 etcd 快照。**cluster-backup.sh** 脚本确认其有效。
- **static_kuberesources_<datetimestamp>.tar.gz**：此文件包含静态 pod 的资源。如果启用了 etcd 加密，它也包含 etcd 快照的加密密钥。

注意

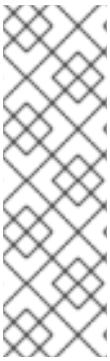
如果启用了 etcd 加密，建议出于安全考虑，将第二个文件与 etcd 快照分开保存。但是，需要这个文件才能从 etcd 快照中进行恢复。

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

4.2. 替换不健康的 ETCD 成员

本文档描述了替换一个不健康 etcd 成员的过程。

此过程取决于 etcd 成员不健康的原因，如机器没有运行，或节点未就绪，或 etcd pod 处于 crashlooping 状态。



注意

如果您丢失了大多数 control plane 主机，并导致 etcd 仲裁丢失，则必须遵循灾难恢复流程恢复到以前的集群状态，而不是这个过程。

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

如果 control plane 节点丢失并且创建了一个新节点，etcd 集群 Operator 将处理生成新 TLS 证书并将节点添加为 etcd 成员。

4.2.1. 先决条件

- 在替换不健康的 etcd 成员，需要进行 etcd 备份。

4.2.2. 找出一个不健康的 etcd 成员

您可以识别集群是否有不健康的 etcd 成员。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。

流程

1. 使用以下命令检查 **EtcdMembersAvailable** 状态条件的状态：

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

2. 查看输出：

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

这个示例输出显示 **ip-10-0-131-183.ec2.internal** etcd 成员不健康。

4.2.3. 确定不健康的 etcd 成员的状态

替换不健康 etcd 成员的步骤取决于 etcd 的以下状态：

- 机器没有运行或者该节点未就绪
- etcd pod 处于 crashlooping 状态

此流程决定了 etcd 成员处于哪个状态。这可让您了解替换不健康的 etcd 成员要遵循的步骤。

注意

如果您知道机器没有运行或节点未就绪，但它们应该很快返回健康状态，那么您就不需要执行替换 etcd 成员的流程。当机器或节点返回一个健康状态时，etcd cluster Operator 将自动同步。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已找到不健康的 etcd 成员。

流程

1. 检查 **机器是否没有运行**:

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

输**出示例**

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

(1) 此输出列出了节点以及节点机器的状态。如果状态不是 **running**，则**代表机器没有运行**。

如果**机器没有运行**，按照 *替换机器没有运行或节点没有就绪的非健康 etcd 成员*过程进行操作。

2. 确定 **节点是否未就绪**。
如果以下任何一种情况是正确的，则代表**节点没有就绪**。

- 如果机器正在运行，检查节点是否不可访问：

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

输**出示例**

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

(1) 如果节点带有 **unreachable** 污点，则**节点没有就绪**。

- 如果该节点仍然可访问，则检查该节点是否列为 **NotReady**:

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

输**出示例**

```
ip-10-0-131-183.ec2.internal   NotReady   master   122m   v1.22.1 (1)
```

(1) 如果节点列表为 **NotReady**，则 **该节点没有就绪**。

如果节点没有就绪，按照*替换机器没有运行或节点没有就绪的 etcd 成员*的步骤进行操作。

3. 确定 etcd Pod 是否处于 crashlooping 状态。
如果机器正在运行并且节点已就绪，请检查 etcd pod 是否处于 crashlooping 状态。

a. 验证所有 control plane 节点都列为 **Ready** ：

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

输出示例

```
NAME                           STATUS   ROLES    AGE     VERSION
ip-10-0-131-183.ec2.internal   Ready    master   6h13m   v1.22.1
ip-10-0-164-97.ec2.internal    Ready    master   6h13m   v1.22.1
ip-10-0-154-204.ec2.internal   Ready    master   6h13m   v1.22.1
```

b. 检查 etcd pod 的状态是否为 **Error** 或 **CrashLoopBackOff**:

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

输出示例

```
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 由于此 pod 的状态是 **Error**，因此 **etcd pod 为 crashlooping 状态。**

如果 **etcd pod 为 crashlooping 状态，**请按照*替换 etcd pod 处于 crashlooping 状态的不健康的 etcd 成员*的步骤进行操作。

4.2.4. 替换不健康的 etcd 成员

根据不健康的 etcd 成员的状态，使用以下一个流程：

- 替换机器没有运行或节点未就绪的不健康 etcd 成员
- 替换其 etcd Pod 处于 crashlooping 状态的不健康 etcd 成员

4.2.4.1. 替换机器没有运行或节点未就绪的不健康 etcd 成员

此流程详细介绍了替换因机器没有运行或节点未就绪造成不健康的 etcd 成员的步骤。

先决条件

- 您已找出不健康的 etcd 成员。
- 您已确认机器没有运行，或者该节点未就绪。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已进行 etcd 备份。

重要

执行此流程前务必要进行 etcd 备份，以便在遇到任何问题时可以恢复集群。

流程

1. 删除不健康的成员。

   a. 选择一个*不在*受影响节点上的 pod:
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

   输出示例

   ```
   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. 连接到正在运行的 etcd 容器，传递没有在受影响节点上的 pod 的名称：
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

   c. 查看成员列表：

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

   输出示例

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

   记录不健康的 etcd 成员的 ID 和名称，因为稍后需要这些值。

   d. 通过向 **etcdctl member remove** 命令提供 ID 来删除不健康的 etcd 成员 :

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

   输出示例

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

e. 再次查看成员列表，并确认成员已被删除：

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

输出示例

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

现在您可以退出节点 shell。

2. 删除已删除的不健康 etcd 成员的旧 secret。

a. 列出已删除的不健康 etcd 成员的 secret。

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

1 传递您之前在这个过程中记录的不健康 etcd 成员的名称。

有一个对等的、服务和指标的 secret，如以下输出所示：

输出示例

```
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. 删除已删除的不健康 etcd 成员的 secret。

i. 删除 peer（对等）secret:

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

ii. 删除 serving secret:

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

iii. 删除 metrics secret:

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

3. 删除并重新创建 control plane 机器。重新创建此机器后，会强制一个新修订版本并自动扩展 etcd。
   如果您正在运行安装程序置备的基础架构，或者您使用 Machine API 创建机器，请按照以下步骤执行。否则，您必须使用最初创建 master 时使用的相同方法创建新的 master。

   a. 获取不健康成员的机器。
      在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

      输出示例

```
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   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) 这是不健康节点的 control plane 机器 **ip-10-0-131-183.ec2.internal**。

   b. 将机器配置保存到文件系统中的一个文件中：

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

      (1) 为不健康的节点指定 control plane 机器的名称。

   c. 编辑上一步中创建的 **new-master-machine.yaml** 文件，以分配新名称并删除不必要的字段。

      i. 删除整个 **status** 部分：

```
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. 将 **metadata.name** 字段更改为新名称。
建议您保留与旧机器相同的基础名称，并将结束号码改为下一个可用数字。在本例中，**clustername-8qw5l-master-0** 改为 **clustername-8qw5l-master-3**。

例如：

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

iii. 更新 **metadata.selfLink** 字段，使用上一步中的新机器名称。

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

iv. 删除 **spec.providerID** 字段：

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

v. 删除 **metadata.annotations** 和 **metadata.generation** 字段：

```
    annotations:
```

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

vi. 删除 **metadata.resourceVersion** 和 **metadata.uid** 字段：

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

d. 删除不健康成员的机器：

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

1 为不健康的节点指定 control plane 机器的名称。

e. 验证机器是否已删除：

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

输出示例

```
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. 使用 **new-master-machine.yaml** 文件创建新机器：

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

g. 验证新机器是否已创建：

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

输出示例

```
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) 新机器 **clustername-8qw5l-master-3** 将被创建，并当阶段从 **Provisioning** 变为 **Running** 后就可以使用。

创建新机器可能需要几分钟时间。当机器或节点返回一个健康状态时，etcd cluster Operator 将自动同步。

验证

1. 验证所有 etcd pod 是否都正常运行。
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

   输**出示例**

```
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
```

   如果上一命令的输出只列出两个 pod，您可以手动强制重新部署 etcd。在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

(1) **forceRedeploymentReason** 值必须是唯一的，这就是为什么附加时间戳的原因。

2. 验证只有三个 etcd 成员。
   a. 连接到正在运行的 etcd 容器，传递没有在受影响节点上的 pod 的名称：
      在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

   b. 查看成员列表：

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

输出示例

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

如果上一命令的输出列出了超过三个 etcd 成员，您必须删除不需要的成员。

警告

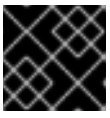
确保删除正确的 etcd 成员；如果删除了正常的 etcd 成员则有可能会导致仲裁丢失。

4.2.4.2. 替换其 etcd Pod 处于 crashlooping 状态的不健康 etcd 成员

此流程详细介绍了替换因 etcd pod 处于 crashlooping 状态造成不健康的 etcd 成员的步骤。

先决条件

- 您已找出不健康的 etcd 成员。
- 已确认 etcd pod 处于 crashlooping 状态。
- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已进行 etcd 备份。



重要

执行此流程前务必要进行 etcd 备份，以便在遇到任何问题时可以恢复集群。

流程

1. 停止处于 crashlooping 状态的 etcd pod。

   a. 对处于 crashlooping 状态的节点进行调试。
   在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

1 使用不健康节点的名称来替换它。

b. 将您的根目录改为主机：

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

c. 将现有 etcd pod 文件从 Kubelet 清单目录中移出：

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

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

d. 将 etcd 数据目录移到不同的位置：

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

现在您可以退出节点 shell。

2. 删除不健康的成员。

a. 选择一个*不在*受影响节点上的 pod。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

输**出示例**

```
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. 连接到正在运行的 etcd 容器，传递没有在受影响节点上的 pod 的名称。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

c. 查看成员列表：

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

输**出示例**

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

记录不健康的 etcd 成员的 ID 和名称，因为稍后需要这些值。

d. 通过向 **etcdctl member remove** 命令提供 ID 来删除不健康的 etcd 成员：

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

输出示例

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

e. 再次查看成员列表，并确认成员已被删除：

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

输出示例

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

现在您可以退出节点 shell。

3. 删除已删除的不健康 etcd 成员的旧 secret。

a. 列出已删除的不健康 etcd 成员的 secret。

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

1 传递您之前在这个过程中记录的不健康 etcd 成员的名称。

有一个对等的、服务和指标的 secret，如以下输出所示：

输出示例

```
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. 删除已删除的不健康 etcd 成员的 secret。

i. 删除 peer（对等）secret:

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

ii. 删除 serving secret:

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

iii. 删除 metrics secret:

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

4. 强制 etcd 重新部署。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

(1) **forceRedeploymentReason** 值必须是唯一的，这就是为什么附加时间戳的原因。

当 etcd 集群 Operator 执行重新部署时，它会确保所有 control plane 节点都有可正常工作的 etcd pod。

验证

- 确认新成员可用且健康。

a. 连接到正在运行的 etcd 容器：
在一个终端中使用 cluster-admin 用户连接到集群，运行以下命令：

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

b. 验证所有成员是否健康：

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

输出示例

```
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
```

4.3. 灾难恢复

4.3.1. 关于灾难恢复

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

重要

灾难恢复要求您至少有一个健康的 control plane 主机。

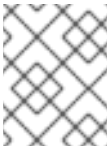
恢复到一个以前的集群状态

如果您希望将集群恢复到一个以前的状态时（例如，管理员错误地删除了一些关键信息），则可以使用这个解决方案。这包括您丢失了大多数 control plane 主机并导致 etcd 仲裁丢失，且集群离线的情况。只要您执行了 etcd 备份，就可以按照这个步骤将集群恢复到之前的状态。
如果适用，您可能还需要从过期的 control plane 证书中恢复。

警告

在一个正在运行的集群中恢复到以前的集群状态是破坏性的，而不稳定的操作。这仅应作为最后的手段使用。

在执行恢复前，请参阅关于恢复集群状态以了解有关对集群的影响的更多信息。



注意

如果大多数 master 仍可用，且仍有 etcd 仲裁，请按照以下步骤替换一个不健康的 etcd 成员。

从 control plane 证书已过期的情况下恢复

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

4.3.2. 恢复到一个以前的集群状态

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

4.3.2.1. 关于恢复集群状态

您可以使用 etcd 备份将集群恢复到以前的状态。在以下情况中可以使用这个方法进行恢复：

- 集群丢失了大多数 control plane 主机（仲裁丢失）。
- 管理员删除了一些关键内容，必须恢复才能恢复集群。

警告

在一个正在运行的集群中恢复到以前的集群状态是破坏性的，而不稳定的操作。这仅应作为最后的手段使用。

如果您可以使用 Kubernetes API 服务器检索数据，则代表 etcd 可用，且您不应该使用 etcd 备份来恢复。

恢复 etcd 实际相当于把集群返回到以前的一个状态，所有客户端都会遇到一个有冲突的、并行历史记录。这会影响 kubelet、Kubernetes 控制器、SDN 控制器和持久性卷控制器等监视组件的行为。

当 etcd 中的内容与磁盘上的实际内容不匹配时，可能会导致 Operator churn，从而导致 Kubernetes API 服务器、Kubernetes 控制器管理器、Kubernetes 调度程序和 etcd 的 Operator 在磁盘上的文件与 etcd 中的内容冲突时卡住。这可能需要手动操作来解决问题。

在极端情况下，集群可能会丢失持久性卷跟踪，删除已不存在的关键工作负载，重新镜像机器，以及重写带有过期证书的 CA 捆绑包。

4.3.2.2. 恢复到一个以前的集群状态

您可以使用保存的 etcd 备份来恢复以前的集群状态，或恢复丢失了大多数 control plane 主机的集群。

重要

恢复集群时，必须使用同一 z-stream 发行版本中获取的 etcd 备份。例如，OpenShift Container Platform 4.7.2 集群必须使用从 4.7.2 开始的 etcd 备份。

先决条件

- 使用具有 **cluster-admin** 角色的用户访问集群。
- 用作恢复主机的健康 control plane 主机。
- SSH 对 control plane 主机的访问。
- 包含从同一备份中获取的 etcd 快照和静态 pod 资源的备份目录。该目录中的文件名必须采用以下格式: **snapshot_<datetimestamp>.db 和 static_kuberesources_<datetimestamp>.tar.gz** 。

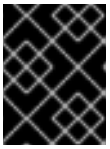
重要

对于非恢复 control plane 节点，不需要建立 SSH 连接或停止静态 pod。您可以逐个删除并重新创建其他非恢复 control plane 机器。

流程

1. 选择一个要用作恢复主机的 control plane 主机。这是您要在其中运行恢复操作的主机。
2. 建立到每个 control plane 节点（包括恢复主机）的 SSH 连接。

恢复过程启动后，Kubernetes API 服务器将无法访问，因此您无法访问 control plane 节点。因此，建议在一个单独的终端中建立到每个control plane 主机的 SSH 连接。



重要

如果没有完成这个步骤，将无法访问 control plane 主机来完成恢复过程，您将无法从这个状态恢复集群。

3. 将 etcd 备份目录复制复制到恢复 control plane 主机上。
此流程假设您将 **backup** 目录（其中包含 etcd 快照和静态 pod 资源）复制到恢复 control plane 主机的 **/home/core/** 目录中。

4. 在任何其他 control plane 节点上停止静态 pod。

注意

不需要手动停止恢复主机上的 pod。恢复脚本将停止恢复主机上的 pod。

   a. 访问不是恢复主机的 control plane 主机。

   b. 将现有 etcd pod 文件从 Kubelet 清单目录中移出：

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

   c. 验证 etcd pod 是否已停止。

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

   命令输出应该为空。如果它不是空的，请等待几分钟后再重新检查。

   d. 将现有 Kubernetes API 服务器 pod 文件移出 kubelet 清单目录中：

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

   e. 验证 Kubernetes API 服务器 pod 是否已停止。

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

   命令输出应该为空。如果它不是空的，请等待几分钟后再重新检查。

   f. 将 etcd 数据目录移到不同的位置：

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

   g. 在其他不是恢复主机的 control plane 主机上重复此步骤。

5. 访问恢复 control plane 主机。

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

提示

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

7. 在恢复 control plane 主机上运行恢复脚本，提供到 etcd 备份目录的路径：

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

脚本输出示例

```
...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. 在所有 control plane 主机上重启 kubelet 服务。

   a. 在恢复主机中运行以下命令：

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

   b. 在所有其他 control plane 主机上重复此步骤。

9. 批准待处理的 CSR：

   a. 获取当前 CSR 列表。

   ```
   $ oc get csr
   ```

   输出示例

   ```
   NAME        AGE    SIGNERNAME                                    REQUESTOR
   CONDITION
   ```

```
csr-2s94x   8m3s   kubernetes.io/kubelet-serving                 system:node:<node_name>
Pending (1)
csr-4bd6t   8m3s   kubernetes.io/kubelet-serving                 system:node:<node_name>
Pending (2)
csr-4hl85   13m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
(3)
csr-zhhhp   3m8s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending
(4)
...
```

(1) (2) 一个待处理的 kubelet 服务 CSR（用于用户置备的安装）。

(3) (4) 一个待处理的 **node-bootstrapper** CSR。

b. 查看一个 CSR 的详细信息以验证其是否有效：

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

(1) **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

c. 批准每个有效的 **node-bootstrapper** CSR：

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

d. 对于用户置备的安装，请批准每个有效的 kubelet 服务 CSR：

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

10. 确认单个成员 control plane 已被成功启动。

a. 从恢复主机上，验证 etcd 容器是否正在运行。

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

输出示例

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

b. 从恢复主机上，验证 etcd pod 是否正在运行。

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

注意

如果您试图在运行这个命令前运行 **oc login** 并接收以下错误，请等待一些时间以便身份验证控制器启动并再次尝试。

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

输**出示例**

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

如果状态是 **Pending**，或者输出中列出了多个正在运行的 etcd pod，请等待几分钟，然后再次检查。

c. 对不是恢复主机的 control plane 主机重复此步骤。

11. 删除并重新创建其他非恢复(control plane)机器，逐个删除和重新创建。重新创建这些机器后，会强制一个新的修订版本，etcd 会自动扩展。
如果您正在运行安装程序置备的基础架构，或者您使用 Machine API 创建机器，请按照以下步骤执行。否则，您必须使用最初创建 master 节点时所用的相同方法创建新的 master 节点。

警告

不要为恢复主机删除和重新创建计算机。

a. 为丢失的 control plane 主机之一获取机器。
在一个终端中使用 cluster-admin 用户连接到集群，运行以下命令：

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

输出示例：

```
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-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2           Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.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 这是用于丢失的 control plane 主机 **ip-10-0-131-183.ec2.internal** 的 control plane 机器。

b. 将机器配置保存到文件系统中的一个文件中：

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

1 为丢失的 control plane 主机指定 control plane 机器的名称。

c. 编辑上一步中创建的 **new-master-machine.yaml** 文件，以分配新名称并删除不必要的字段。

i. 删除整个 **status** 部分：

```
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. 将 **metadata.name** 字段更改为新名称。
建议您保留与旧机器相同的基础名称，并将结束号码改为下一个可用数字。在本例中，cluster **name-8qw5l-master-0** 被改为 **clustername-8qw5l-master-3** ：

```
apiVersion: machine.openshift.io/v1beta1
```

```
kind: Machine
metadata:
  ...
  name: clustername-8qw5l-master-3
  ...
```

iii. 更新 **metadata.selfLink** 字段以使用上一步中的新机器名称：

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

iv. 删除 **spec.providerID** 字段：

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

v. 删除 **metadata.annotations** 和 **metadata.generation** 字段：

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

vi. 删除 **metadata.resourceVersion** 和 **metadata.uid** 字段：

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

d. 删除丢失的 control plane 主机的机器：

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

1 为丢失的 control plane 主机指定 control plane 机器的名称。

e. 验证机器是否已删除：

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

输出示例：

```
NAME                                PHASE     TYPE        REGION      ZONE         AGE
NODE                           PROVIDERID                          STATE
clustername-8qw5l-master-1          Running   m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2          Running   m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.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. 使用 **new-master-machine.yaml** 文件创建新机器：

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

g. 验证新机器是否已创建：

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

输出示例：

```
NAME                                PHASE          TYPE        REGION      ZONE         AGE
NODE                          PROVIDERID                          STATE
clustername-8qw5l-master-1          Running        m4.xlarge   us-east-1   us-east-1b
3h37m   ip-10-0-143-125.ec2.internal   aws:///us-east-1b/i-096c349b700a19631   running
clustername-8qw5l-master-2          Running        m4.xlarge   us-east-1   us-east-1c
3h37m   ip-10-0-154-194.ec2.internal   aws:///us-east-1c/i-02626f1dba9ed5bba   running
clustername-8qw5l-master-3          Provisioning   m4.xlarge   us-east-1   us-east-1a
85s     ip-10-0-173-171.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) 新机器 **clustername-8qw5l-master-3** 会被创建，并在阶段从 **Provisioning** 变为 **Running** 后就绪。

创建新机器可能需要几分钟时间。当机器或节点返回一个健康状态时，etcd cluster Operator 将自动同步。

h. 对不是恢复主机的 control plane 主机重复这些步骤。

12. 在一个单独的终端窗口中，使用以下命令以具有 **cluster-admin** 角色的用户身份登录到集群：

```
$ oc login -u <cluster_admin> (1)
```

(1) 对于 **<cluster_admin>**，使用 **cluster-admin** 角色指定一个用户名。

13. 强制 etcd 重新部署。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

(1) **forceRedeploymentReason** 值必须是唯一的，这就是为什么附加时间戳的原因。

当 etcd cluster Operator 执行重新部署时，现有节点开始使用与初始 bootstrap 扩展类似的新 pod。

14. 验证所有节点是否已更新至最新的修订版本。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

查看 etcd 的**NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新的修订。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

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

(1) 在本例中，最新的修订版本号是 **7**。

如果输出包含多个修订号，如 **2 个节点为修订版本 6；1 个节点为修订版本 7**，这意味着更新仍在进行中。等待几分钟后重试。

15. 在重新部署 etcd 后，为 control plane 强制进行新的 rollout。由于 kubelet 使用内部负载平衡器连接到 API 服务器，因此 Kubernetes API 将在其他节点上重新安装自己。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令。

    a. 为 Kubernetes API 服务器强制进行新的推出部署：

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

    验证所有节点是否已更新至最新的修订版本。

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

    查看 **NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新版本。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

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

    (1) 在本例中，最新的修订版本号是 **7**。

    如果输出包含多个修订号，如 **2 个节点为修订版本 6；1 个节点为修订版本 7**，这意味着更新仍在进行中。等待几分钟后重试。

    b. 为 Kubernetes 控制器管理器强制进行新的推出部署：

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

验证所有节点是否已更新至最新的修订版本。

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

查看 **NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新版本。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

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

(1) 在本例中，最新的修订版本号是 **7**。

如果输出包含多个修订号，如 **2 个节点为修订版本 6；1 个节点为修订版本 7**，这意味着更新仍在进行中。等待几分钟后重试。

c. 为 Kubernetes 调度程序强制进行新的推出部署：

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

验证所有节点是否已更新至最新的修订版本。

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

查看 **NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新版本。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

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

(1) 在本例中，最新的修订版本号是 **7**。

如果输出包含多个修订号，如 **2 个节点为修订版本 6；1 个节点为修订版本 7**，这意味着更新仍在进行中。等待几分钟后重试。

16. 验证所有 control plane 主机是否已启动并加入集群。
在一个终端中使用 **cluster-admin** 用户连接到集群，运行以下命令：

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

输出示例

```
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
```

为确保所有工作负载在恢复过程后返回到正常操作，请重启存储 Kubernetes API 信息的每个 pod。这包括 OpenShift Container Platform 组件，如路由器、Operator 和第三方组件。

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

4.3.2.3. 恢复持久性存储状态的问题和解决方法

如果您的 OpenShift Container Platform 集群使用任何形式的持久性存储，集群的状态通常存储在 etcd 外部。它可能是在 pod 中运行的 Elasticsearch 集群，或者在 **StatefulSet** 对象中运行的数据库。从 etcd 备份中恢复时，还会恢复 OpenShift Container Platform 中工作负载的状态。但是，如果 etcd 快照是旧的，其状态可能无效或过期。

重要

持久性卷（PV）的内容绝不会属于 etcd 快照的一部分。从 etcd 快照恢复 OpenShift Container Platform 集群时，非关键工作负载可能会访问关键数据，反之亦然。

以下是生成过时状态的一些示例情况：

- MySQL 数据库在由 PV 对象支持的 pod 中运行。从 etcd 快照恢复 OpenShift Container Platform 不会使卷恢复到存储供应商上，且不会生成正在运行的 MySQL pod，尽管 pod 会重复尝试启动。您必须通过在存储供应商中恢复卷，然后编辑 PV 以指向新卷来手动恢复这个 pod。

- Pod P1 使用卷 A，它附加到节点 X。如果另一个 pod 在节点 Y 上使用相同的卷，则执行 etcd 恢复时，pod P1 可能无法正确启动，因为卷仍然被附加到节点 Y。OpenShift Container Platform 并不知道附加，且不会自动分离它。发生这种情况时，卷必须从节点 Y 手动分离，以便卷可以在节点 X 上附加，然后 pod P1 才可以启动。

- 在执行 etcd 快照后，云供应商或存储供应商凭证会被更新。这会导致任何依赖于这些凭证的 CSI 驱动程序或 Operator 无法正常工作。您可能需要手动更新这些驱动程序或 Operator 所需的凭证。

- 在生成 etcd 快照后，会从 OpenShift Container Platform 节点中删除或重命名设备。Local Storage Operator 会为从 **/dev/disk/by-id** 或 **/dev** 目录中管理的每个 PV 创建符号链接。这种情况可能会导致本地 PV 引用不再存在的设备。
  要解决这个问题，管理员必须：

  1. 手动删除带有无效设备的 PV。
  2. 从对应节点中删除符号链接。
  3. 删除 **LocalVolume** 或 **LocalVolumeSet** 对象（请参阅 *Storage → Configuring persistent storage → Persistent storage → Persistent storage → Deleting the Local Storage Operator Resources*）。

其他资源

- 如需有关如何创建堡垒主机来访问 OpenShift Container Platform 实例和使用 SSH 的 control plane 节点，请参阅 访问主机。

4.3.3. 从 control plane 证书已过期的情况下恢复

4.3.3.1. 从 control plane 证书已过期的情况下恢复

集群可以从过期的 control plane 证书中自动恢复。

但是，您需要手动批准待处理的 **node-bootstrapper** 证书签名请求（CSR）来恢复 kubelet 证书。对于用户置备的安装，您可能需要批准待处理的 kubelet 服务 CSR。

使用以下步骤批准待处理的 CSR：

流程

1. 获取当前 CSR 列表。

```
$ oc get csr
```

输**出示例**

```
NAME        AGE    SIGNERNAME                                    REQUESTOR
CONDITION
csr-2s94x   8m3s   kubernetes.io/kubelet-serving                 system:node:<node_name>
Pending 1
csr-4bd6t   8m3s   kubernetes.io/kubelet-serving                 system:node:<node_name>
Pending 2
csr-4hl85   13m    kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending 3
csr-zhhhp   3m8s   kubernetes.io/kube-apiserver-client-kubelet
system:serviceaccount:openshift-machine-config-operator:node-bootstrapper   Pending 4
...
```

1 2 一个待处理的 kubelet 服务 CSR（用于用户置备的安装）。

3 4 一个待处理的 **node-bootstrapper** CSR。

2. 查看一个 CSR 的详细信息以验证其是否有效：

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

1 **<csr_name>** 是当前 CSR 列表中 CSR 的名称。

3. 批准每个有效的 **node-bootstrapper** CSR：

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

4. 对于用户置备的安装，请批准每个有效的 kubelet 服务 CSR：

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