etcd

概览 **etcd** 实践 **etcd** 加密

概览

etcd 存储集群的 Kubernetes 控制平面状态。平台管理员使用 etcd 指南来了解加密、运行状况、容量考量，以及如何交接到备份和恢复操作步骤。

需求	继续阅读
启用并了解 etcd 加密密钥轮换。	etcd encryption
查看运行实践，并转到运行状况、容量和备份主题。	etcd practices
备份或恢复 etcd 数据。	etcd backup and restore

使用这些 etcd 页面来规划控制平面状态的加密、运行状况、容量和维护。当您需要保护或恢复 etcd 数据时，请继续阅读备份和恢复操作步骤。

etcd 实践

使用 etcd 实践，在执行会影响集群恢复或可用性的更改之前，规划控制平面状态操作。

实践领域	指导
高风险维护前备份	在进行控制平面维护、节点操作系统更改或其他可能影响集群状态的操作之前，先备份 etcd。请遵循 etcd 备份和恢复。
加密	当敏感的 Kubernetes 资源需要通过文档中的加密管理器在静态存储时进行加密时，请使用 etcd 加密。
容量和健康状况	通过你环境中可用的可观测性和集群操作工具，监控 etcd 健康状况、磁盘延迟和可用容量。
恢复范围	将 etcd 备份和恢复用于控制平面状态恢复。Registry 数据、监控数据、日志数据和应用数据需要各自独立的备份和恢复路径。

不要将此页面用作生产加固检查清单。请遵循特定主题的支持指南，以获取经过验证的恢复流程、规模限制和维护窗口。

etcd 加密

在 ACP 中安装并运行控制平面数据库加密，以自动轮换集群中的 etcd 数据加密密钥。

它确保存储在 etcd 中的敏感数据（例如 Secret 和 configmap）使用安全算法进行加密，从而增强集群的安全性。

目录

- 密钥策略
- 安装
  - 工作负载 / DCS 集群（随机策略）
  - Global 集群（确定性模式）
    - 先决条件
    - 插件参数
    - 准备根 Secret
    - 高级设置（chart 值，不在插件表单中公开）
- 工作原理
  - 默认配置
- 操作指南
  - 配置文件
  - 检查状态
  - 批准确定性密钥轮换
    - 手动批准（推荐）
    - 定时批准（不推荐）

密钥策略

该插件支持两种密钥策略。请选择与集群角色匹配的策略。

策略	使用时机	密钥生成方式
`random`	独立运行的工作负载和 DCS 集群的默认策略（无 etcd 同步）。	每次轮换都会在本地生成一个全新的随机密钥。
`deterministic`	`global` 集群必须使用此策略，尤其是在通过 Global Cluster Disaster Recovery 插件组成 DR 设置时。	每个密钥都根据共享根 Secret 和共享 SeedBundle 以确定性方式派生，因此对于相同的修订版本，Active 和 Standby 集群始终会生成相同的密钥。

在 `deterministic` 模式下，运行时的 Active / Standby 角色会自动检测；它不是手动安装参数。

安装

该插件根据目标集群支持两种安装路径：

- 工作负载 **/ DCS** 集群 — 使用默认的 `random` 密钥策略。
- **Global** 集群 — 必须使用 `deterministic` 密钥策略（请参阅下方的 Global 集群（确定性模式））。

有关通用的插件安装流程，请参阅 集群插件。

工作负载 **/ DCS** 集群（随机策略）

- 支持的集群类型：On-Premises、DCS。
- 无需其他安装参数 — 通过标准的集群插件流程进行安装。

Global 集群（确定性模式）

DR 对要求

对于 Global DR 对，请在 `deterministic` 模式下将插件同时安装在 **Active** 和 **Standby** 集群上，并在两端配置相同的 `replication_group_id` 和 `root_secret_name`（根密钥材料必须匹配）。不对称或不匹配的配置会导致 Standby 为每个修订版本派生出与 Active 不同的密钥，从而导致故障转移失败。

先决条件

- 集群必须是 `global` 集群。当 `global.cluster.name` 为 `global` 时，插件会强制使用 `etcdEncryption.keyStrategy=deterministic`；对于标记为 `is-global=true` 的集群，插件配置 UI 也会将该字段默认设置为 `deterministic`。

Global Cluster Disaster Recovery 版本

在以 `deterministic` 模式安装 Control Plane Database Encryption 之前，必须先安装 **v4.3.7** 或更高版本的 **Global Cluster Disaster Recovery** 插件。早期版本不会复制共享的 SeedBundle，因此 Standby 集群无法派生出匹配的密钥。

插件参数

插件表单公开了以下用于确定性安装的参数。

插件参数	必填	默认值	用途
`key_strategy`	是	`random`；对于 `global` 集群，动态默认为 `deterministic`。	在每次轮换生成随机密钥和确定性派生之间进行选择。
`replication_group_id`	是	插件 UI 会预填当前集群名称（不建议 — 请参阅下文）。	标识 Active / Standby DR 组，并参与 HKDF 域分离。
`root_secret_name`	是	`etcd-derivation-root`	`kube-system` 中存储确定性根密钥

插件参数	必填	默认值	用途
			材料的 Secret 名称。

`rootSecretRef.namespace` 固定为 `kube-system`，不会在表单中公开。

`key_strategy`

值	含义
`random`	单集群随机轮换。
`deterministic`	基于共享根 Secret 和 SeedBundle 派生密钥。

对于 `global` 集群，此值必须为 `deterministic`。

`replication_group_id`

属性	值
用途	标识一个 Active / Standby DR 复制组，并参与 HKDF 域分离。
一致性要求	同一 DR 组中的 Active 和 Standby 集群必须使用完全相同的值。
UI 中的默认值	当前集群名称。
是否建议保留默认值	否。

建议：

- 对于使用 Global Cluster Disaster Recovery 插件构建的 DR 集群，同一 DR 组的两端必须共享相同的 `replication_group_id`。
- 为避免泄露真实的业务、环境或集群身份，即使 UI 预填了集群名称，也不要重复使用该名称。请在安装前规划一个单独的不透明标识符。
- UUID 风格的标识符可能比较合适，例如 `6f1b9e2c-7c3a-4a9c-8b72-2fd0f8f3c1ab`。

要从命令行生成 UUID 风格的值：

```
openssl rand -hex 16 | sed -E 's/^(.{8})(.{4})(.{4})(.{4})(.{12})$/\1-\2-\3-\4-\5/'
```

> **在两个集群上使用相同的值**
>
> 只生成一次该值，然后在 DR 组的 Active 和 Standby 集群上配置相同的 `replication_group_id`。在两端生成不同的值会导致加密密钥不一致。

`root_secret_name`

属性	值
用途	指向确定性根密钥 Secret。
默认值	`etcd-derivation-root`
固定命名空间	`kube-system`
跨集群要求	同一复制组中的所有集群必须使用相同的根密钥材料。

插件表单仅公开 Secret 名称；渲染时命名空间始终为 `kube-system`。

准备根 Secret

确定性根 Secret 必须在安装插件之前存在于 `kube-system` 中。请分两步准备：

1. 将 32 字节的密码学强随机材料生成到 `./root-key.bin` 中。这些字节必须来自高熵源，通常是操作系统 CSPRNG。`openssl rand` 是安全的默认选项；使用操作系统 CSPRNG 的等效命令（例如 `head -c 32 /dev/urandom > ./root-key.bin`）也可以正常工作。不要使用 shell `$RANDOM`、语言级 `rand()` 或任何基于时间设定种子的生成器。

   ```
   openssl rand -out ./root-key.bin 32
   ```

2. 使用该文件在 `kube-system` 中创建 Secret：

```
kubectl -n kube-system create secret generic etcd-derivation-root \
  --from-file=root-key=./root-key.bin
```

如果使用其他 Secret 名称，请将上方的 `etcd-derivation-root` 替换为要作为 `root_secret_name` 输入的值。

> **在两个集群上使用相同的密钥材料**
>
> 只生成一次 `./root-key.bin`，然后将同一个文件传输到复制组中的每个集群，并在每个集群上执行 `kubectl create secret` 步骤。不要为每个集群重新生成密钥材料 — 不同的根密钥字节会导致加密密钥不一致。

高级设置（**chart** 值，不在插件表单中公开）

▶ 显示高级 chart 值

> **将 Global 集群的 activationPolicy 设置为 manual**
>
> 对于 Global Active/Standby 对，请明确将 `activationPolicy` 设置为 `manual`。在 operator 确认 Standby 集群上的匹配修订版本已准备就绪之前，新的密钥修订版本不得在 Active 集群上变为活动状态。如果省略 `activationPolicy`，插件会默认使用带有 `30m` 延迟的 `timed-approval` — 有关其风险，请参阅定时批准。

工作原理

安装后，`etcd-encryption-manager` 控制器会部署在 `kube-system` 命名空间中，该控制器会：

- 定期轮换 etcd 数据加密密钥。
- 保留最近的 8 个密钥，以支持回滚。
- 更新所有控制平面节点上的加密配置。

- 触发 `kube-apiserver` 热加载新密钥。
- 自动迁移资源，以使用新密钥重新加密数据。

在整个过程中，集群稳定性都会得到保持。

当插件以 `deterministic` 模式安装在 `global` DR 对上时，控制器还会：

- 在运行时检测自身是 **Active** 还是 **Standby** — 这不是手动配置。
- 在 Active 侧生成 SeedBundle，并由 Global Cluster Disaster Recovery 插件将其复制到 Standby。
- 在 Standby 侧根据共享的 SeedBundle 和根 Secret 派生相同的每个修订版本密钥，因此发生故障转移时无需手动复制密钥即可生成完全相同的加密密钥。

默认配置

参数	值
加密资源	secrets, configmaps
加密算法	256-bit AES-GCM
轮换间隔	168 小时（7 天）

操作指南

配置文件

路径	内容
`/etc/kubernetes/encryption-provider.conf`	当前加密配置
`/etc/kubernetes/encryption-provider-history.bak`	历史密钥记录（用于恢复）

路径	内容
`/etc/kubernetes/encryption-provider-bak/`	已过期的加密配置版本

检查状态

运行以下命令以检查当前轮换状态：

```
kubectl get EtcdEncryptionConfig default -o yaml
```

示例输出：

```yaml
apiVersion: cluster.alauda.io/v1alpha1
kind: EtcdEncryptionConfig
metadata:
  name: default
spec:
  resources:
    - secrets
    - configmaps
  rotationInterval: 168h0m0s
  type: aesgcm
status:
  deployStatus:
    192.168.100.1:
      revision: 3
      state: Success
    192.168.100.2:
      revision: 3
      state: Success
    192.168.100.3:
      revision: 3
      state: Success
  migration:
    completeTimestamp: "2025-05-27T05:47:01Z"
    resources:
      - secrets
      - configmaps
    revision: 3
    state: Success
  revision: 3
```

批准确定性密钥轮换

当加密管理器在 `deterministic` 模式下轮换密钥时，新修订版本会在 Active 集群上进入 `WaitingApproval` 阶段。在 operator 批准之前，该密钥不会生效。设置此门控的原因是，Active 集群不得使用 Standby 集群尚未加载的密钥加密数据 — 否则发生故障转移时，Standby 将无法解密使用新密钥加密的密文。

批准始终仅在 **Active** 集群上执行。请勿修补 Standby 的 `EtcdKeySeedBundleState`，也不要直接编辑任何资源的 `status` 字段。

手动批准（推荐）

步骤 **1 —** 确定待处理的修订版本。

登录 **Active** 集群的控制平面节点，并列出所有 bundle 状态：

```
kubectl get eksbs \
  -o custom-columns='NAME:.metadata.name,REVISION:.spec.revision,PHASE:.status.phase,APPROVED:.spec.approved'
```

找到包含 `PHASE=WaitingApproval` 和 `APPROVED=false` 的行。记录其修订版本号：

```
REVISION=<revision-number>
STATE="seed-bundle-state-${REVISION}"
BUNDLE="seed-bundle-${REVISION}"
```

步骤 **2 —** 验证 **Standby** 是否就绪。

登录 **Standby** 集群的控制平面节点，并检查相同的修订版本：

```
kubectl get eksbs "$STATE" \
  -o custom-columns='REVISION:.spec.revision,PHASE:.status.phase,PREPARED_AT:.status.preparedAt'
```

对于预期的修订版本，输出必须显示 `PHASE=Prepared`。如果对象缺失、仍处于 `Preparing` 或 `Failed` 状态，请停止操作；继续之前，先调查 Global Cluster Disaster Recovery 复制路径。

步骤 **3 —** 交叉检查派生输入。

在 **Active** 和 **Standby** 集群上分别运行以下命令，并比较输出。`replicationGroupID`、`dimension` 和 `derivationAlgorithm` 必须完全匹配：

```
kubectl get eksb "$BUNDLE" \
  -o custom-columns='REVISION:.spec.revision,GROUP:.spec.replicationGroupID,DIMENSION:.spec.dimension,ALGORITHM:.spec.derivationAlgorithm'
```

步骤 **4** — 批准。

在 **Active** 集群上批准待处理的修订版本：

```
kubectl patch eksbs "$STATE" \
  --type=merge -p '{"spec":{"approved":true}}'
```

此修补操作会记录批准结果。此后，激活、部署到控制平面节点以及数据迁移将异步进行。

步骤 **5** — 确认激活完成。

在 **Active** 集群上检查 bundle 状态和迁移进度：

```
kubectl get eksbs "$STATE" \
  -o custom-columns='REVISION:.spec.revision,PHASE:.status.phase,MESSAG
E:.status.message'
kubectl get eec default \
  -o custom-columns='REVISION:.status.revision,MIGRATION_STATE:.status.mi
gration.state,MIGRATION_REVISION:.status.migration.revision'
```

满足以下条件时，轮换完成：

- Bundle 状态达到 `PHASE=Active`。
- `eec/default` 显示新的 `REVISION`，其状态为 `MIGRATION_STATE=Success`。

要检查 **Active** 集群上的每个节点的部署进度：

```
kubectl get eec default \
  -o go-template='{{range $node, $status := .status.deployStatus}}{{print
f "%s\t%s\t%d\t%s\n" $node $status.state $status.revision $status.messag
e}}{{end}}'
```

每个控制平面节点都必须针对新修订版本报告 `state=Success`。如果 bundle 一直处于 `Activating`，则表示部署或迁移仍在进行中（或失败后正在重试）。

定时批准（不推荐）

> **定时批准会在故障转移期间带来数据丢失风险**
>
> 对于 Global Active/Standby 对，不建议使用 `timed-approval`。`approvalDelay` 计时器（默认值为 `30m`）是 Active 集群上的本地倒计时 — 它不会检查 Standby 是否确实已加载新密钥。如果 etcd 同步缓慢、Standby 网络发生分区，或 Standby 控制平面正在维护中，截止时间可能在 Standby 仍未准备就绪时到期。在此时间窗口内发生故障转移，会导致 Standby 无法解密使用新密钥加密的数据。

当 `activationPolicy` 设置为 `timed-approval` 时，控制器会在创建 `EtcdKeySeedBundleState` 后经过 `approvalDelay` 时间自动批准待处理的修订版本。延迟时间必须足够长，以便完整的 Standby 准备流程完成：

阶段	操作	典型持续时间
同步	Global Cluster Disaster Recovery 插件将 SeedBundle 复制到 Standby	数秒（取决于跨地域延迟）
协调	Standby 控制器派生密钥并更新本地 EncryptionConfig	数秒
部署	Deployer 将新的 provider 配置写入所有 Standby 控制平面节点	数秒到数分钟（随节点数量扩展）
重新加载	Standby kube-apiserver 加载新的 provider	数秒到数分钟（取决于是否为加密 provider 配置启用了自动重新加载）

默认的 `30m` 为典型的小型控制平面单地域部署提供了余量。如果设置涉及跨地域复制、大型控制平面，或未启用自动 provider 重新加载的集群，请增大该值。

必需的监控

如果选择 `timed-approval`，则在启用此策略之前，必须确认 **Active** 和 **Standby** 集群上的监控和告警均正常工作。

加密管理器会公开以下 Prometheus 指标：

指标

指标	类型	描述
`etcd_encryption_manager_bundle_waiting_approval_seconds`	Gauge	当前 bundl `WaitingA` 状态的秒数 `activati` （`manual` `timed-ap` 标记。
`etcd_encryption_manager_timed_approval_remaining_seconds`	Gauge	定时批准截 期前的剩余 值表示截止 过。仅当 ` approval` 于 `WaitingA` 状态时发出
`etcd_encryption_manager_bundle_activating`	Gauge	当任意 bun 于 `Activa` 时为 `1`。
`etcd_encryption_manager_bundle_activation_failed`	Gauge	当 `Activa` bundle 存在 署或迁移失 `1`。

内置告警规则

插件包含以下告警：

告警	严重性	条件
`EtcdEncryptionManagerApprovalRequired`	warning	`manual` bun 于 `WaitingAppr` 状态已 ≥ 5 分钟
`EtcdEncryptionManagerApprovalOverdue`	critical	`manual` bun 于

告警	严重性	条件
		`WaitingAppr` 状态已 ≥ 30 分
`EtcdEncryptionManagerTimedApprovalDeadlineApproaching`	warning	`timed-appro` bundle 距离自 准剩余 ≤ 10 分
`EtcdEncryptionManagerTimedApprovalDeadlinePassed`	critical	`timed-appro` bundle 仍处于 `WaitingAppr` 状态，但其截 间已过。
`EtcdEncryptionManagerActivationFailed`	critical	`Activating` bundle 存在部 迁移失败。
`EtcdEncryptionManagerActivationStuck`	critical	Bundle 处于 `Activating` 已 ≥ 30 分钟。

这些告警会在每个集群上独立触发。对于 Global 集群，需要采取行动的关键告警是 **Active** 集群上的 `EtcdEncryptionManagerTimedApprovalDeadlineApproaching` — 它会提供 10 分钟窗口，让你检查 Standby 是否为 `Prepared`，如果不是，则切换到 `manual`。

> **跨集群关联**
>
> 每个集群都独立运行自己的指标和告警。内置告警会检测本地问题 — 例如 Active 上卡住的 `WaitingApproval`，或 Standby 上失败的激活。要关联两个集群（例如“Active 即将自动批准，但 Standby 尚未处于 `Prepared` 状态”），请将两个集群的指标聚合到一个中央 Prometheus / Thanos / Victoria Metrics 实例中。

紧急响应

如果触发 `EtcdEncryptionManagerTimedApprovalDeadlineApproaching` 告警，而 Standby 尚未就绪，请将 Active 切换到 `manual` 以暂停自动批准。在 **Active** 集群上：

```
kubectl patch eec default \
  --type=merge -p '{"spec":{"deterministic":{"activationPolicy":"manua
l"}}}'
```

确认当前待处理的 bundle 已切换：

```
kubectl get eksbs \
  -o custom-columns='NAME:.metadata.name,REVISION:.spec.revision,PHASE:.s
tatus.phase,APPROVED:.spec.approved'
```

Bundle 必须仍显示 `APPROVED=false`。之后切换回 `timed-approval` 只会影响未来的 bundle — 当前 bundle 仍由手动控制。如果 bundle 已经进入 `Activating`，则截止时间已经生效；请按照上述激活检查进行操作，不要假定更改策略会将其回滚。