备份和恢复

概览

etcd 备份与恢复

原生应用备份与恢复

备份仓库

操作步骤

相关操作

镜像仓库替换

操作步骤

创建应用备份计划

前提条件

操作步骤

相关操作

钩子

Overview

Backup Hook

Restore Hooks

运行原生应用

前提条件

操作步骤

下一步

概览

灵雀云容器平台 提供两种备份和恢复操作：

- 集群备份和恢复：备份并恢复控制平面数据，包括 etcd、Registry、日志和监控数据。
- 原生应用备份和恢复：基于 Velero 备份并恢复原生应用及其持久卷。

目录

集群备份和恢复

集群备份可保护控制平面状态和平台数据。

etcd 备份和恢复

etcd 是 ACP 的键值存储，用于持久化资源对象的状态。对于控制平面状态恢复场景，需要进行 etcd 备份。

如需详细说明，请参见 etcd Backup and Restore。

Registry 备份和恢复

有关 Registry 备份和恢复，请参见 Registry Data Backup and Recovery。

日志备份和恢复

当前支持 ClickHouse。请联系技术支持。

监控备份和恢复

有关监控备份和恢复，请参见 VictoriaMetrics Backup and Recovery。

原生应用备份和恢复

作为集群管理员，你可以使用 Velero 备份和恢复运行在 ACP 上的原生应用。

架构

原生应用备份和恢复由两个组件组成：

- 灵雀云容器平台 **Data Backup Essentials**：提供 UI，并安装在 global 集群上。
- 灵雀云容器平台 **Data Backup for Velero**：提供 Velero，并安装在工作负载集群上。

原生应用安装

要启用原生应用备份和恢复：

1. 从 灵雀云 Customer Portal 下载 灵雀云容器平台 **Data Backup Essentials** 和 灵雀云容器平台 **Data Backup for Velero**。
2. 上传软件包 到平台。
3. 在 global 集群上安装 Data Backup Essentials。
4. 在工作负载集群上安装 Data Backup for Velero。

安装完成后，请配置一个备份仓库以存储备份数据。

备份和恢复原生应用

你可以通过创建备份计划来备份原生应用，并通过执行恢复任务来恢复原生应用。

如需详细说明，请参见：

- 创建原生应用备份
- 恢复原生应用

备份注意事项

在配置备份策略之前，请考虑以下会影响备份和恢复策略的架构因素。

不可变 **OS** 集群

对于运行在 Immutable OS 上的集群，在灾难恢复场景中，不需要也不建议将 **VM Snapshot** 作为备份策略。

不建议使用 **VM Snapshot** 的原因：

- **Immutable OS** 设计：操作系统层是只读的，并由平台集中管理。当节点发生故障时，平台会自动创建一个配置正确的新节点。

- 分布式系统特性：Kubernetes 是一个分布式系统。VM Snapshot 无法捕获 etcd quorum 等分布式组件的一致状态。
- 灾难恢复限制：VM Snapshot 通常与原始数据存储在一起，无法应对站点级灾难。

推荐的备份方式：

- 集群状态：使用 etcd 备份来捕获控制平面状态
- 应用数据：对持久卷使用 PV 快照或 Restic
- 集群配置：使用 GitOps/IaC 进行配置管理

使用场景

在当前集群内备份和恢复

在误删除或故障后，将原生应用恢复到当前集群。恢复过程中，通常不需要修改原生应用资源。

跨集群原生应用迁移

常见场景包括：

- 跨数据中心进行开发和测试
- 在集群之间迁移资源
- 从生产集群复制到开发/测试集群

注意事项：

- 确保源集群和目标集群之间的 CPU 和内存规格相近
- 保持一致的网络模式，以避免资源恢复问题
- 如果子网不同，恢复后 Pod IP 将发生变化
- 在恢复数据之前，评估是否需要进行镜像迁移

etcd 备份与恢复

集群上的 etcd 服务是一个分布式键值存储，用于保存集群配置信息。etcd 部署在集群的所有控制平面节点上。

安装 灵雀云容器平台 Cluster Enhancer 插件后，会为集群配置自动创建一个 `EtcdBackupConfiguration` 资源。`EtcdBackupConfiguration` 包含备份数据源、控制平面节点路径、备份数据存储位置和备份方法。基于策略执行的每次备份都会生成一条新的备份记录，从而支持按需和定时的集群配置备份。

目录

前提条件

要启用 etcd 备份：

1. 从 灵雀云 Customer Portal 下载 灵雀云容器平台 **Cluster Enhancer**。
2. 上传包 到平台。
3. 在集群上安装插件。

安装完成后，会自动创建一个 EtcdBackupConfiguration 资源。

工作原理

- etcd 备份由 灵雀云容器平台 **Cluster Enhancer** 提供。
- 支持本地存储和 S3 兼容的对象存储。本地备份会写入每个控制平面节点上的一个目录，默认是 `/var/cpaas/backup`。请为该目录规划足够的容量，以容纳 etcd 数据大小和保留周期。配置 S3 存储会在 S3 bucket 中创建一份额外副本；本地备份仍会继续生成。
- 对于 Immutable OS 集群，本地备份默认使用 `/var/cpaas/backup`。仍可通过 `remoteStorage.s3` 使用 S3 存储。

配置参考

你可以配置 `EtcdBackupConfiguration` 资源，以自定义备份计划、保留策略和存储选项。

计划与保留

- **`schedule`**：使用标准 cron 语法定义备份频率。
  - 示例：`0 0 * * *`（每天午夜运行备份）。
- **`localStorage`**：配置本地备份存储。
  - **`path`**：每个控制平面节点上存储备份的目录。默认值为 `/var/cpaas/backup`。请将其设置为一个能够容纳 etcd 数据大小和已配置保留周期的目录；请参见下方的存储说明。
  - **`ttl`**：备份文件的保留期，单位为秒。超过此时长的备份将被自动删除。
    - 示例：`7776000`（约 90 天）。
- **`paused`**：设置为 `true` 可临时暂停自动备份，而不删除配置。

> **WARNING**
>
> 在启用 etcd 备份之前，请先规划本地备份的存储容量。本地备份会累积在控制平面节点上，并会一直保留到其 `ttl` 过期，因此 `/var/cpaas/backup` 必须为预期的备份大小和保留周期预留足够的可用空间。

配置示例：

```yaml
apiVersion: enhancement.cluster.alauda.io/v1
kind: EtcdBackupConfiguration
metadata:
  name: etcd-backup-default
spec:
  schedule: "0 0 * * *"       # Run daily at 00:00
  paused: false               # Enable backups
  localStorage:
    path: /var/cpaas/backup   # Local backup directory on each control pl
ane node
    ttl: "7776000"            # Retain for 90 days
  # ... other fields
```

查看备份记录

要查看 etcd 备份记录，可以使用平台 UI 或命令行。

使用平台 UI

1. 在左侧导航栏中，单击 **Operation Center** > **Monitor** > **Dashboards**。
2. 单击页面右上角的 **Switch**。
3. 单击 **Cluster** → **etcd backup** 查看 etcd 备份记录。

使用 CLI

你可以通过检查 `EtcdBackupConfiguration` 资源的 `status` 字段来验证备份状态和历史记录：

```
kubectl get etcdbackupconfiguration etcd-backup-default -o yaml
```

输出包含一个 `status.records` 列表，其中包含每次备份的详细信息，包括：

- `backupTimestamp`：创建备份的时间。
- `fileName`：备份文件名称（例如，`snapshot-etcd-<date>-<time>-<ip>.tar`）。
- `result`：备份操作的结果（例如，`Success`）。

S3 备份配置

要为 etcd 备份启用 S3 存储，请按以下步骤操作：

前提条件

- 已在集群上安装 灵雀云容器平台 Cluster Enhancer。

步骤 1：创建 S3 Secret

准备你的 S3 访问凭据，并在 `cpaas-system` 命名空间中创建一个 Kubernetes secret：

```
export ACCESS_KEY="your-access-key"
export SECRET_KEY="your-secret-key"

kubectl create secret generic etcd-backup-s3-secret \
  --from-literal=ACCESS_KEY="$ACCESS_KEY" \
  --from-literal=SECRET_KEY="$SECRET_KEY" \
  --dry-run=client -n cpaas-system -o yaml | kubectl apply -f -
```

步骤 2：配置 EtcdBackupConfiguration

修改 `EtcdBackupConfiguration` 资源，添加带有 S3 配置的 `remoteStorage` 字段：

```
spec:
  remoteStorage:
    s3:
      endpoint: "your-s3-endpoint"  # e.g.: https://s3.bucket.com
      region: "your-s3-region"
      bucket: "your-s3-bucket"
      dir: "your-s3-bucket-dir"
      skipTLSVerify: false  # Set to true only for self-signed certificat
es
      secretRef: etcd-backup-s3-secret
```

步骤 3：验证备份

触发一次手动 etcd 备份以验证配置：

```
# Set environment variables
token="your-platform-token"
platform_url="your-platform-url"
cluster_name="your-cluster-name"

# Trigger backup
curl $platform_url/kubernetes/$cluster_name/apis/enhancement.cluster.alauda.io/v1/etcdbackupconfigurations/etcd-backup-default/exec \
  -k -H "Authorization: Bearer $token"
```

备份完成后，请确认你的 S3 bucket 中存在备份文件。

在进行恢复准备时，请使用备份记录中显示的 `fileName` 和已配置的 `remoteStorage.s3.dir` 值，从 S3 下载 tar 归档：

```
aws --endpoint-url <s3-endpoint> s3 cp \
  s3://<bucket>/<dir>/<backup-file-name> \
  /tmp/snapshot-etcd.tar

mkdir -p /tmp/etcd-restore
tar -xf /tmp/snapshot-etcd.tar -C /tmp/etcd-restore
find /tmp/etcd-restore -name 'snapshot-etcd-*.db' -type f
```

etcd 恢复

> 警告：
>
> - 此操作会对 etcd 集群执行破坏性恢复，并覆盖现有数据。继续之前，请确保你有有效的备份快照。
> - 该操作步骤风险较高。如果你不确定如何操作，请联系技术支持。
> - 在恢复过程中，Kubernetes API Server 将不可用。

前提条件

- Kubernetes 集群使用主机名部署（`kubectl get node` 显示主机名作为节点名称）。

- 已有一个 etcd 备份快照可用。
- 集群因 etcd 节点故障而出现异常（例如，超过一半的控制平面节点宕机）。
- 此恢复操作步骤专为 **3** 节点控制平面 集群设计。如果你的集群有 5 个或更多控制平面节点，请联系技术支持以获取帮助。
- 在 Immutable OS 上，请在开始恢复前确认每个控制平面节点上都存在 `/var/lib/etcd`、`/etc/kubernetes/manifests/etcd.yaml` 和 `/etc/kubernetes/pki/etcd/`。
- 在 Immutable OS 上，`etcdctl` 可能不在宿主机的 `PATH` 中。请在 `/var/lib/containerd` 下查找，或使用包含兼容 `etcdctl` 二进制文件的容器镜像。

步骤 **1**：备份原始数据并修改 **etcd** 配置

在所有控制平面节点上执行以下命令：

```
# Create backup directory
mkdir -p /root/backup_$(date +%Y%m%d%H)/old-etcd/

# Stop kubelet
systemctl stop kubelet

# Prepare etcdctl binary
ETCDCTL_PATH="$(find /var/lib/containerd/ -name etcdctl -type f | tail -
1)"
if [ -z "$ETCDCTL_PATH" ]; then
  echo "etcdctl was not found under /var/lib/containerd. Use a container
image that includes a compatible etcdctl binary."
  exit 1
fi
cp "$ETCDCTL_PATH" /root/etcdctl
chmod +x /root/etcdctl

# Remove etcd containers
crictl ps -a | grep etcd | awk '{print $1}' | xargs -r crictl rm -f

# Backup etcd data and kubernetes configuration
cp -a /var/lib/etcd/* /root/backup_$(date +%Y%m%d%H)/old-etcd/
rm -rf /var/lib/etcd/*
cp -r /etc/kubernetes/ /root/backup_$(date +%Y%m%d%H)/old-etcd/

# Modify etcd.yaml to use existing cluster state
sed -i /initial-cluster-state=/d /etc/kubernetes/manifests/etcd.yaml
sed -i '/initial-cluster=/a\    - --initial-cluster-state=existing' /etc/
kubernetes/manifests/etcd.yaml
```

注意：请验证 `/etc/kubernetes/manifests/etcd.yaml` 中 `--initial-cluster-state=existing` 的缩进。

步骤 2：复制备份快照

将选定的 etcd 备份快照复制到第一个控制平面节点上的 `/tmp` 目录，并将其命名为 `snapshot.db`。

如果备份 tar 归档文件已保存在本地，则从 `/var/cpaas/backup` 中提取它：

```
mkdir -p /tmp/etcd-restore
tar -xf "$(ls -1t /var/cpaas/backup/snapshot-etcd-*.tar | head -1)" -C /t
mp/etcd-restore
cp "$(find /tmp/etcd-restore -name 'snapshot-etcd-*.db' -type f | head -
1)" /tmp/snapshot.db
```

如果备份 tar 归档文件存储在 S3 中，请先下载它，然后再提取快照文件：

```
aws --endpoint-url <s3-endpoint> s3 cp \
  s3://<bucket>/<dir>/<backup-file-name> \
  /tmp/snapshot-etcd.tar

mkdir -p /tmp/etcd-restore
tar -xf /tmp/snapshot-etcd.tar -C /tmp/etcd-restore
cp "$(find /tmp/etcd-restore -name 'snapshot-etcd-*.db' -type f | head -
1)" /tmp/snapshot.db
```

步骤 3：恢复 etcd

在第一个控制平面节点上执行以下脚本以恢复快照。

> 注意： 以下脚本假定集群为 3 节点控制平面。如果你的集群有 5 个或更多节点，请联系技术支持。

```
#!/usr/bin/env bash

# Set etcd node IPs (Replace with actual IPs)
export ETCD_1=1.1.1.1
export ETCD_2=2.2.2.2
export ETCD_3=3.3.3.3

# Set corresponding node hostnames (Replace with actual hostnames)
export ETCD_1_HOSTNAME=etcd-1
export ETCD_2_HOSTNAME=etcd-2
export ETCD_3_HOSTNAME=etcd-3

export ETCDCTL_API=3

# Loop for 3 nodes. Adjust '1 2 3' if you have a different number of node
s.
for n in 1 2 3; do
  ip_var=ETCD_${n}
  host_var=ETCD_${n}_HOSTNAME

  ip=${!ip_var}
  host=${!host_var}

  echo "Restoring for node: ${host} (${ip})..."

  rm -rf /tmp/etcd
  /root/etcdctl snapshot restore /tmp/snapshot.db \
    --cert=/etc/kubernetes/pki/etcd/server.crt \
    --key=/etc/kubernetes/pki/etcd/server.key \
    --cacert=/etc/kubernetes/pki/etcd/ca.crt \
    --skip-hash-check=true \
    --data-dir=/tmp/etcd \
    --name "${host}" \
    --initial-cluster \
      ${ETCD_1_HOSTNAME}=https://${ETCD_1}:2380,\
${ETCD_2_HOSTNAME}=https://${ETCD_2}:2380,\
${ETCD_3_HOSTNAME}=https://${ETCD_3}:2380 \
    --initial-advertise-peer-urls https://"${ip}":2380 && \
    mv /tmp/etcd /root/etcd_"${host}"

  echo "Restoration for ${host} completed. Data directory: /root/etcd_${h
ost}"
done
```

脚本完成后，会在 `/root` 目录中生成三个目录（`etcd_$host`）。

步骤 4：分发恢复后的数据

1. 将恢复后的数据目录传输到相应的控制平面节点。使用 `scp` 或类似工具，将步骤 3 中生成的目录（`/root/etcd_<hostname>`）从第一个节点复制到其他节点。
   例如，传输到 **etcd-2** 和 **etcd-3**：

   ```
   # Replace <etcd-2-ip> and <etcd-3-ip> with actual IPs
   scp -r /root/etcd_etcd-2 root@<etcd-2-ip>:/root/
   scp -r /root/etcd_etcd-3 root@<etcd-3-ip>:/root/
   ```

2. 将数据恢复到每个控制平面节点上的 etcd 数据目录（`/var/lib/etcd`）。

   ```
   # On etcd-1:
   cp -r /root/etcd_etcd-1/member/* /var/lib/etcd/

   # On etcd-2:
   cp -r /root/etcd_etcd-2/member/* /var/lib/etcd/

   # On etcd-3:
   cp -r /root/etcd_etcd-3/member/* /var/lib/etcd/
   ```

步骤 5：重启集群组件

在所有控制平面节点上执行以下命令：

```
# Remove Kubernetes control plane containers
crictl ps -a | grep -E "kube-api|kube-sche|kube-contro" | awk '{print
$1}' | xargs -r crictl rm -f

# Restart kubelet
systemctl restart kubelet
```

步骤 6：验证恢复

1. 检查 etcd 集群是否健康。你可以在任意一个 `etcd` Pod 内执行此命令，或使用宿主机上的 `etcdctl` 二进制文件：

```
# Using etcdctl on the host
export ETCDCTL_API=3
/root/etcdctl --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/server.crt \
  --key=/etc/kubernetes/pki/etcd/server.key \
  --endpoints=https://127.0.0.1:2379 \
  endpoint health
```

2. 检查 Kubernetes Pod 是否正常运行：

```
kubectl get po -n kube-system
```

3. 在所有节点（包括控制平面节点和 worker 节点）上重启 kubelet，以确保所有组件重新连接到已恢复的 etcd：

```
systemctl restart kubelet
```

配置管理

如需修改默认的 etcd 备份配置，请联系技术支持以获取详细的配置选项和高级设置。

原生应用备份与恢复

Backup repository

备份仓库存储应用和配置数据（例如命名空间和应用），并支持从备份文件快速恢复。

目录

操作步骤

1. 在左侧导航中，点击 **Clusters** > **Backup and Recovery**。
2. 切换到 **Backup Repository Management** 标签页。
3. 点击 **Create Backup Repository** 并按以下参数配置。

Parameter	Description
Region	对象存储数据中心的地域。提示：MinIO 填写 "minio"。
Addressing Style	地址访问方式。默认：对象存储为 virtual-hosted–style，文件存储和 MinIO 集成为 path-based。

Parameter	Description
Endpoint	对象存储的外部访问地址（HTTP REST API），使用服务端点，例如 `https://cos.<region>.myqcloud.com`。注意：不要使用包含桶名的公共桶域名，例如 `https://<bucket name>.cos.<region>.myqcloud.com`。
Bucket	存储桶名称。
Folder	用于存储备份数据的桶内文件夹前缀，例如 `backups`。
Secret Id/Secret Key	用于认证对象存储的访问密钥。
Skip TLS Verify	如果外部存储未使用有效证书，启用此选项以跳过证书验证。
CA Certificate	如需 CA 验证，上传 Base64 编码的 CA 证书。

4. 点击 **Create**。

相关操作

删除备份仓库

如果备份计划引用了该仓库，删除后该计划运行时将导致备份失败。请先更新备份计划，引用其他仓库后再删除。

1. 在左侧导航中，点击 **Clusters** > **Backup and Recovery**。
2. 切换到 **Backup Repository Management** 标签页。
3. 点击备份仓库右侧的 **Delete** 按钮，然后点击 **Delete** 确认删除。

创建应用备份

创建应用备份计划，用于定义备份数据的范围（按命名空间）、备份存储位置、方式及相关参数。每次按计划运行都会生成新的备份记录，实现对所选命名空间内应用资源的按需或定期自动备份。

目录

前提条件

- 已安装备份组件。安装说明请参见应用安装。
- 已配置备份仓库。

操作步骤

WARNING

- 备份应用数据时需包含 PersistentVolumeClaims（PVC）。不支持绑定到 `hostPath` PersistentVolumes 的 PVC。
- 为确保可靠性和数据完整性，不建议备份数据库数据（例如 MySQL-PXC、Redis），数据库备份请使用 Data Services。
- 避免在备份的命名空间内进行读写、更新和删除操作，以防止迁移后出现漂移和不一致。

1 基本信息

1. 在左侧导航栏点击 **Clusters** > **Backup and Recovery**。
2. 切换到 **Backup Management** 标签页。
3. 点击 **Create Backup Policy** > **Create Application Backup**，并按如下配置参数。
   - **Backup Resource Type**：**Kubernetes Resources** 包含所选命名空间内的所有 Kubernetes 资源文件。**PVCs** 是用于备份绑定到持久卷的应用数据的持久卷声明。不支持绑定到 `hostPath` 卷的 PVC。
     - 如果 PVC 使用的存储资源的 **Recycle Strategy** 为 **Retain**，则只需备份 Kubernetes 资源。
     - 如果 PVC 使用的存储资源的 **Recycle Strategy** 为 **Delete**，则需同时备份 Kubernetes 资源和 PVC。
   - **Backup Repository**：选择已通过连通性验证的仓库，或点击创建备份仓库。创建仓库后点击 **OK and Create Application Backup** 返回继续。
4. 配置完基本信息后，点击 **Next**。

2 备份资源

备份所选命名空间下的应用资源。

集群中未导入的命名空间不会显示，需先将其导入项目后才能备份。

1. 选择一个或多个要备份的 **Namespaces**。
2. （可选） 配置资源过滤选项以细化备份范围：

- **Included Namespace Resources**：选择要包含在备份中的特定命名空间范围资源类型。仅备份所选命名空间内的这些资源类型。支持模糊搜索和多选。
- **Included Cluster Resources**：选择要包含在备份中的特定集群范围资源类型。仅备份明确选择的集群范围资源。默认不备份任何集群范围资源，除非包含。支持模糊搜索和多选。
- **Excluded Namespace Resources**：选择要排除在备份之外的命名空间范围资源类型。Velero 不会备份这些资源类型。支持模糊搜索和多选。
- **Label Selectors**：添加标签选择器过滤资源。仅备份匹配配置表达式的资源。配置多个选择器时，采用 OR 逻辑组合（由 Velero 的 `orLabelSelectors` 字段支持）。注意：此项与单一 `labelSelector` 互斥，备份时只能使用一种模式。每个选择器支持两种匹配方式：
  matchLabels - 简单的键值匹配：

参数	类型	说明
Key	string (必填)	标签键
Value	string (必填)	标签值

  matchExpressions - 复杂条件匹配：

参数	类型	说明
Key	string (必填)	标签键
Operator	string (必填)	取值为 `In`、`NotIn`、`Exists`、`DoesNotExist` 中之一
Values	array (条件性必填)	值数组。`In` / `NotIn` 时必填，`Exists` / `DoesNotExist` 时必须为空

  仅按标签键匹配（不指定值）时，使用带 `Exists` 操作符的 **matchExpressions**。

> **NOTE**
>
> 如果匹配的资源挂载了 PVC，且备份模式包含 PVC 数据（基于 `defaultVolumesToFsBackup`），则挂载的 PVC 数据也会被备份，除非通过 Pod 注解（例如 `backup.velero.io/backup-volumes-excludes`）排除。该注解仅排除文件系统备份的卷，若配置了卷快照，排除的卷可能仍通过快照备份。

3. 点击 **Next**。

3 方式

配置备份计划。

- **Backup once only**：创建后立即执行。设置 **Backup retention period** 后，超过保留期的备份会自动清理。
- **Scheduled**：设置 **Backup rule**，定期执行策略。支持 crontab 表达式。可使用平台预设的 **Backup rule templates**，并按需调整。推荐最小频率：备份 **Kubernetes** 资源和 **Persistent Volume Claims** 每日一次；备份 **Kubernetes** 资源 每小时一次。

4 高级配置

如有需要，配置自定义钩子以在备份过程中执行。

相关操作

手动执行备份计划

手动执行已创建的计划（包括周期规则的计划）。每次执行都会生成新的备份记录。

1. 在左侧导航栏点击 **Clusters** > **Backup and Recovery**。
2. 切换到 **Backup Management** 标签页。
3. 在计划右侧点击 **Execute Backup**，然后确认。

导出备份任务日志

手动导出指定计划的备份任务日志。备份任务进行中时不支持导出日志。

操作步骤

1. 在左侧导航栏点击 **Clusters** > **Backup and Recovery**。
2. 切换到 **Backup Management** 标签页。
3. 点击 **Backup Schedule Name** 查看备份记录，然后在 **Backup Records** 区域点击对应记录右侧的 **Export Log**。

运行原生应用恢复任务

您可以在以下场景中，基于现有的原生应用备份记录执行原生应用恢复任务，将原生应用快速恢复到目标命名空间：

- Kubernetes 资源被误删，需要恢复。
- 需要将原生应用数据迁移到同一集群中的其他命名空间。
- 需要将原生应用资源迁移到平台中其他集群的命名空间。
- 生产集群中备份的 Kubernetes 资源需要恢复到灾难恢复集群。

目录

前提条件

- 集群存储备份文件的对象存储中存在一次成功的原生应用备份。
- 如果 PersistentVolume 的 **reclaimPolicy** 为 **Retain**，请在将数据恢复到已绑定的 PVC 之前，删除对应 PersistentVolume（PV）上的 `spec.claimRef.uid` 和 `spec.claimRef.resourceVersion`。
- 对于跨集群恢复（数据迁移），请确保目标集群和备份文件所在的集群能够读取同一个备份文件。可通过以下两种方式实现：
  - 目标集群和备份文件所在的集群连接到同一个对象存储。
  - 预先将所需的备份文件复制到与目标集群连接的对象存储中。
- 如果在原生应用备份中选择的资源类型为 **Backup Kubernetes resources and persistent volume claims**，请确保目标集群的 **StorageClass** 名称与源集群一致。否则，请在 **Advanced Recovery Target Settings** 中配置源存储类与目标存储类之间的映射关系。

操作步骤

1. 在左侧导航栏中，单击 **Clusters** > **Backup and Recovery**。
2. 切换到 **Recovery Management** 选项卡。
3. 单击 **Execute Application Recovery Task**。
4. 按如下方式配置参数。

参数	说明
Backup Repository	选择一个已通过连通性校验的存储库，或单击 Create Backup Repository。提示：创建存储库后，单击 **OK and Create Application Backup** 返回并继续操作，或单击 **Create** 返回存储库列表。
Recovery File Configuration	选择用于存储备份数据的备份文件。提示：文件名前缀为备份策略名称；仅可选择备份成功的文件。
Recovery Target Configuration	**Namespaces**：执行数据恢复的命名空间，也是备份数据的源命名空间。可选范围为备份策略中设置的命名空间。系统会根据您的选择，将备份数据恢复到相同的命名空间。提示：如需恢复到集群中的其他命名空间，请配置 **Advanced Recovery Target Settings**。

参数	说明
Advanced Recovery Target Settings	将原本计划恢复到源命名空间的备份数据恢复到集群中的任意命名空间（已有命名空间或新建命名空间）。**Source Namespace**：所选命名空间。**Target Namespace**：执行数据恢复的命名空间；可以是已有命名空间，也可以通过输入不存在的名称来创建新命名空间。提示：如果原生应用备份资源类型选择了 **Backup Kubernetes Resources and Persistent Volume Claims**，请确保目标集群 **StorageClass** 名称与源集群一致。否则，请在高级选项中配置源和目标存储类名称，平台将使用新的存储类存储数据。

5. 单击右上角的 **YAML** 切换到 YAML 编辑模式。要配置恢复期间运行的命令，请参见 Configuring Hooks。
   注意：默认情况下，备份文件会与目标命名空间中的资源进行比较。仅恢复备份文件中存在但命名空间中缺失的数据。不会覆盖同名资源或增量资源（命名空间中存在但备份文件中缺失的资源）。
   如需覆盖命名空间中已存在的同名资源：

   1. 单击右上角的 **YAML** 切换到 YAML 编辑模式。
   2. 在 `.spec` 下添加 `existingResourcePolicy: update`，并按如下所示将 `pods` 添加到 `excludedResources`：`excludedResources: ["pods"]`。

   提示：此方法无法覆盖绑定到 PVC 的 PersistentVolume 中的原生应用数据。

6. 单击 **Start**。

下一步

1. **Namespace Import**：在跨集群或跨平台迁移后，需将命名空间手动导入 **Project Management** 中对应的项目。否则，恢复后的原生应用可能不会显示在平台 UI 中。
2. **Reconfigure Fixed IP**：在跨集群或跨平台迁移后，计算组件中容器的固定 IP 地址会发生变化。如有需要，请手动更新部署、DaemonSet 和 StatefulSet 的容器组 **Static IP Address** 参数。

相关操作

重试

如果恢复任务失败，可以重试该任务。

> **TIP**
>
> 重试将生成一条新的恢复记录，您可以通过新的恢复记录查看任务执行状态。

操作步骤

1. 在左侧导航栏中，单击 **Clusters** > **Backup and Recovery**。
2. 切换到 **Recovery Management** 选项卡。
3. 单击失败恢复记录右侧的 **Retry**，并确认。

下载原生应用恢复任务日志

每次执行恢复任务时，都会生成一条恢复记录。您可以通过恢复记录查看执行状态和详情（YAML），并手动下载原生应用恢复任务的操作日志。

操作步骤

1. 在左侧导航栏中，单击 **Clusters** > **Backup and Recovery**。
2. 切换到 **Recovery Management** 选项卡。
3. 单击恢复记录右侧的 **Export Log**。

镜像仓库替换

针对跨集群和跨平台的恢复场景，使用此操作步骤恢复应用镜像，使其在迁移后能够从正确的仓库拉取。

- 镜像仓库迁移：当镜像仓库发生变化时，更新应用镜像引用为新的仓库地址。
- 镜像同步：当将镜像同步到仓库内的其他项目（例如从测试环境推广到生产环境）时，更新应用镜像引用为新的版本。

目录

操作步骤

1. 在执行恢复任务前，创建一个 ConfigMap，用于映射旧镜像仓库到新镜像仓库。将以下示例中的占位符替换为实际内容，并保存为 `change-registry-config.yaml`。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: change-registry-config
  namespace: cpaas-system
  labels:
    velero.io/plugin-config: ""
    alauda.io/change-registry: RestoreItemAction
data:
  old: <Old Image Repository Address>
  new: <New Image Repository Address>
```

2. 在目标集群的控制平面节点上，创建该 ConfigMap：

```
kubectl apply -f change-registry-config.yaml
```

3. 创建完成后，继续执行应用恢复任务。

Hooks

目录

Overview

自定义钩子是在集群内容器中运行的扩展命令。通过配置钩子，您可以在备份和恢复过程中执行定制操作。对于特殊配置需求，请联系技术支持。

Backup Hook

在备份执行过程中，当 Pod 正在备份时，可以在 Pod 内的容器中执行一个或多个命令。这些命令可以在 Pod 备份操作完成之前（`pre`）或之后（`post`）运行。

钩子可以配置在 **schedule** 资源的 `.spec.template.spec.hooks` 字段或 **backup** 资源的 `.spec.hooks` 字段下。

配置示例及参数说明：

```
hooks:
    resources:
      -
        # 钩子名称，显示在日志中。
        name: my-hook
        # （可选）运行钩子的命名空间（数组）。未设置时，钩子在所有命名空间运行。
        includedNamespaces:
        - '*'
        # （可选）排除的命名空间（数组）。
        excludedNamespaces:
        - some-namespace
        # 钩子运行的资源。目前仅支持 pods。
        includedResources:
        - pods
        # （可选）目标资源的标签选择器。
        labelSelector:
          matchLabels:
            app: velero
            component: server
        # 备份前执行的 Pod 操作（数组）。仅支持 exec 钩子。
        pre:
          -
            exec:
              # （可选）执行命令的容器名称。未设置时，默认为 Pod 中的第一个容器。
              container: my-container
              # 要执行的命令（数组）。必填。
              command:
                - /bin/uname
                - -a
              # （可选）命令执行失败策略：Continue 或 Fail。默认：Fail。
              onError: Fail
              # （可选）命令执行超时时间。默认：30s。
              timeout: 10s
        # 备份后执行的 Pod 操作（数组）。仅支持 exec 钩子。
        post:
          # 同 pre
```

Restore Hooks

备份与恢复组件支持通过钩子在恢复任务执行期间或之后执行自定义操作。

支持两种类型的恢复钩子：

- initContainer 恢复钩子：向恢复的 Pod 添加 init 容器，在应用容器启动前执行必要的初始化操作。
- exec 恢复钩子：用于在恢复的 Pod 容器中执行命令或脚本。

恢复钩子可在 **restore** 资源的 `.spec.hooks` 字段中设置。配置示例及参数说明：