앱 보호

Astra Control Center

NetApp
November 21, 2023

목차

앱 보호 1
보호 개요 1
스냅샷 및 백업으로 애플리케이션 보호 1
앱 복원 5
SnapMirror 기술을 사용하여 원격 시스템에 애플리케이션을 복제합니다 7
애플리케이션 클론 복제 및 마이그레이션 12
앱 실행 후크 관리 14

앱 보호

보호 개요

Astra Control Center를 사용하여 앱에 대한 백업, 클론, 스냅샷 및 보호 정책을 생성할 수 있습니다. 앱을 백업하면 서비스 및 관련 데이터를 가능한 한 사용할 수 있습니다. 재해 시나리오 중에 백업에서 복원하면 애플리케이션 및 관련 데이터를 중단 없이 완벽하게 복구할 수 있습니다. 백업, 클론, 스냅샷을 사용하면 랜섬웨어, 우발적인 데이터 손실 및 환경 재해와 같은 일반적인 위협으로부터 보호할 수 있습니다. "Astra Control Center에서 사용 가능한 데이터 보호 유형과 사용 시기에 대해 알아보십시오".

또한 재해 복구에 대비하여 애플리케이션을 원격 클러스터로 복제할 수 있습니다.

애플리케이션 보호 워크플로우

다음 예제 워크플로를 사용하여 앱 보호를 시작할 수 있습니다.

[1개**]** 모든 앱을 보호합니다

앱을 즉시 보호하려면 "모든 앱의 수동 백업을 생성합니다".

[2개**]** 각 앱에 대한 보호 정책을 구성합니다

향후 백업 및 스냅샷 자동화 "각 앱에 대한 보호 정책을 구성합니다". 예를 들어 주별 백업과 일별 스냅샷으로 시작할 수 있으며 두 가지 모두에 대해 한 달 동안 보존할 수 있습니다. 수동 백업 및 스냅샷보다 보호 정책을 사용하여 백업 및 스냅샷을 자동화하는 것이 좋습니다.

[세 가지**]** 보호 정책을 조정합니다

앱과 사용 패턴이 변경되면 최적의 보호 기능을 제공하기 위해 필요에 따라 보호 정책을 조정합니다.

[네**]** 앱을 원격 클러스터로 복제합니다

"애플리케이션 복제" NetApp SnapMirror 기술을 사용하여 원격 클러스터로 Astra Control은 스냅샷을 원격 클러스터에 복제하여 비동기식 재해 복구 기능을 제공합니다.

[다섯**]** 재해가 발생할 경우 최신 백업 또는 복제를 사용하여 원격 시스템으로 앱을 복구합니다

데이터 손실이 발생하면 를 통해 복구할 수 있습니다 "최신 백업을 복원하는 중입니다" 각 앱에 대해 먼저 그런 다음 최신 스냅샷을 복구할 수 있습니다(사용 가능한 경우). 또는 원격 시스템에 복제를 사용할 수 있습니다.

스냅샷 및 백업으로 애플리케이션 보호

자동화된 보호 정책을 사용하거나 필요에 따라 스냅샷 및 백업을 수행하여 모든 애플리케이션을 보호합니다. Astra UI 또는 를 사용할 수 있습니다 "Astra Control API" 앱을 보호합니다.

Helm을 사용하여 앱을 배포하는 경우 Astra Control Center에 Helm 버전 3이 필요합니다. Helm 3으로 배포된 애플리케이션 관리 및 복제(또는 Helm 2에서 Helm 3으로 업그레이드)가 완벽하게 지원됩니다. Helm 2와 함께 배포된 앱은 지원되지 않습니다.

OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 만들면 프로젝트(또는 Kubernetes 네임스페이스)에 SecurityContext UID가 할당됩니다. Astra Control Center에서 앱을 보호하고 OpenShift의 다른 클러스터 또는 프로젝트로 앱을 이동하려면 해당 앱을 UID로 실행할 수 있는 정책을 추가해야 합니다. 예를 들어 다음 OpenShift CLI 명령은 WordPress 앱에 적절한 정책을 부여합니다.

```
oc new-project wordpress
oc adm policy add-scc-to-group anyuid system:serviceaccounts:wordpress
oc adm policy add-scc-to-user privileged -z default -n wordpress
```

앱 데이터 보호와 관련된 다음 작업을 수행할 수 있습니다.

- 보호 정책을 구성합니다
- 스냅샷을 생성합니다
- 백업을 생성합니다
- 스냅샷 및 백업을 봅니다
- 스냅샷을 삭제합니다
- 백업을 취소합니다
- 백업을 삭제합니다

보호 정책을 구성합니다

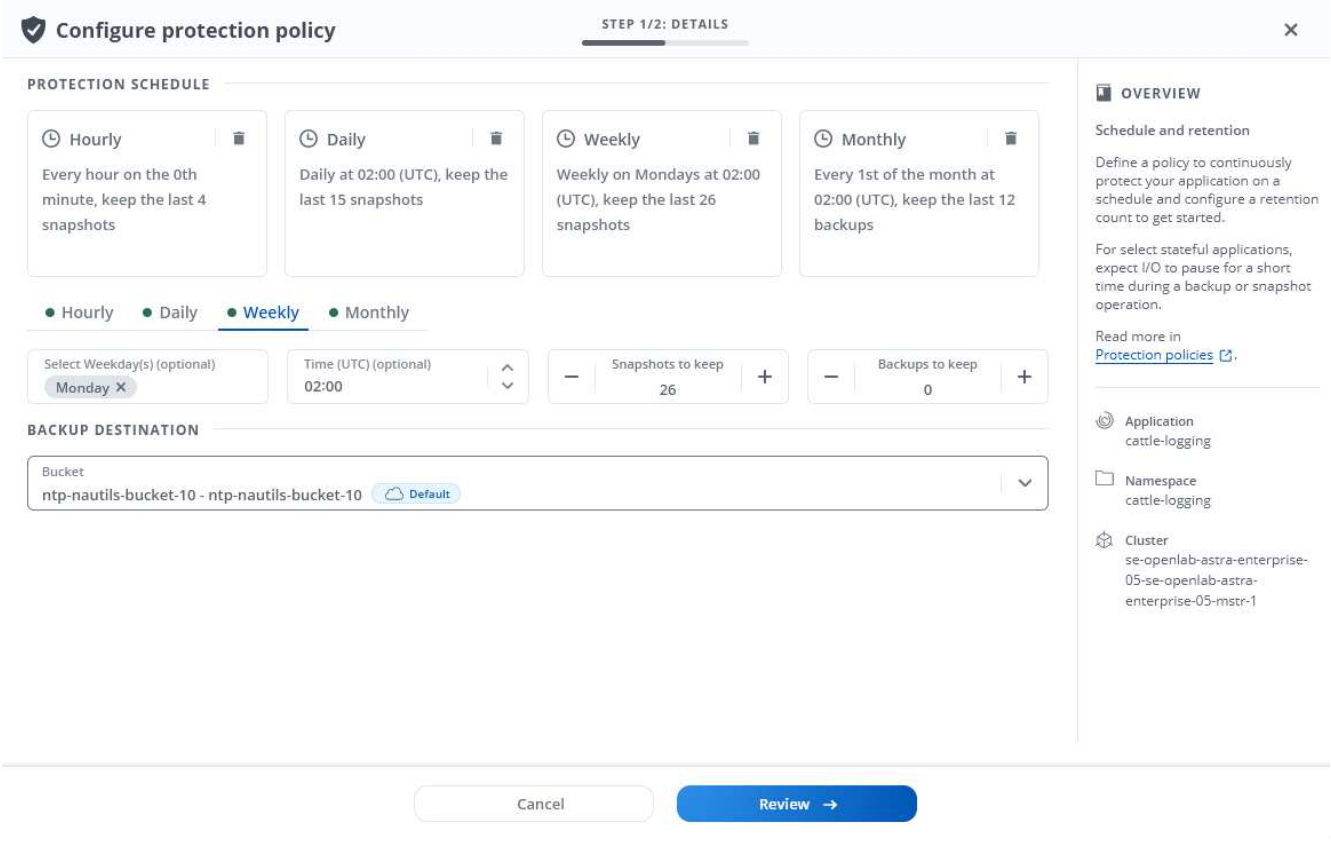
보호 정책은 정의된 일정에 따라 스냅샷, 백업 또는 둘 다를 생성하여 앱을 보호합니다. 시간별, 일별, 주별 및 월별 스냅샷과 백업을 생성하도록 선택할 수 있으며, 보존할 복제본 수를 지정할 수 있습니다. 예를 들어 보호 정책은 주별 백업과 일별 스냅샷을 생성하고 백업 및 스냅샷을 한 달 동안 보존할 수 있습니다. 스냅샷 및 백업을 생성하는 빈도와 보관 기간은 조직의 요구 사항에 따라 다릅니다.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. 보호 정책 구성 * 을 선택합니다.
4. 시간별, 일별, 주별 및 월별로 유지할 스냅샷 및 백업 수를 선택하여 보호 스케줄을 정의합니다.

   시간별, 일별, 주별 및 월별 스케줄을 동시에 정의할 수 있습니다. 보존 레벨을 설정하기 전에는 스케줄이 활성화되지 않습니다.

   다음 예에서는 스냅샷 및 백업의 경우 매시간, 일별, 주별 및 월별로 4개의 보호 스케줄을 설정합니다.



5. Review * 를 선택합니다.
6. 보호 정책 설정 * 을 선택합니다

결과

Astra Control Center는 사용자가 정의한 스케줄 및 보존 정책을 사용하여 스냅샷 및 백업을 생성하고 유지함으로써 데이터 보호 정책을 구현합니다.

스냅샷을 생성합니다

언제든지 주문형 스냅샷을 생성할 수 있습니다.

단계

1. 응용 프로그램 * 을 선택합니다.
2. 원하는 앱의 * Actions * 열에 있는 옵션 메뉴에서 * Snapshot * 을 선택합니다.
3. 스냅샷의 이름을 사용자 지정한 다음 * Review * 를 선택합니다.
4. 스냅샷 요약을 검토하고 * Snapshot * 을 선택합니다.

결과

스냅샷 프로세스가 시작됩니다. 데이터 보호 * > * 스냅샷 * 페이지의 * 작업 * 열에서 * 사용 가능 * 상태가 되면 스냅샷이 성공적으로 생성됩니다.

백업을 생성합니다

언제든지 앱을 백업할 수도 있습니다.

> Astra Control Center의 S3 버킷은 가용 용량을 보고하지 않습니다. Astra Control Center에서 관리하는 앱을 백업 또는 클론 생성하기 전에 ONTAP 또는 StorageGRID 관리 시스템에서 버킷 정보를 확인하십시오.

단계

1. 응용 프로그램 * 을 선택합니다.
2. 원하는 앱의 * Actions * 열에 있는 옵션 메뉴에서 * Backup * 을 선택합니다.
3. 백업 이름을 사용자 지정합니다.
4. 기존 스냅샷에서 앱을 백업할지 여부를 선택합니다. 이 옵션을 선택하면 기존 스냅샷 목록에서 선택할 수 있습니다.
5. 스토리지 버킷 목록에서 선택하여 백업 대상을 선택합니다.
6. Review * 를 선택합니다.
7. 백업 요약을 검토하고 * Backup * 을 선택합니다.

결과

Astra Control Center는 앱 백업을 생성합니다.

> 네트워크에 정전이 발생했거나 비정상적으로 느린 경우 백업 작업이 시간 초과될 수 있습니다. 이로 인해 백업이 실패합니다.

> 실행 중인 백업을 중지할 방법은 없습니다. 백업을 삭제해야 하는 경우 백업이 완료될 때까지 기다린 다음 의 지침을 따르십시오 백업을 삭제합니다. 실패한 백업을 삭제하려면 "Astra Control API를 사용합니다".

> 데이터 보호 작업(클론, 백업, 복원)과 후속 영구 볼륨 크기 조정 후 UI에 새 볼륨 크기가 표시되기까지 최대 20분이 지연됩니다. 데이터 보호 작업이 몇 분 내에 성공적으로 완료되며 스토리지 백엔드에 관리 소프트웨어를 사용하여 볼륨 크기 변경을 확인할 수 있습니다.

스냅샷 및 백업을 봅니다

Data Protection 탭에서 앱의 스냅샷 및 백업을 볼 수 있습니다.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.

   스냅샷은 기본적으로 표시됩니다.

3. 백업 목록을 보려면 * backups * 를 선택합니다.

스냅샷을 삭제합니다

더 이상 필요하지 않은 예약된 스냅샷 또는 주문형 스냅샷을 삭제합니다.

> 현재 복제 중인 스냅샷 복사본은 삭제할 수 없습니다.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. 원하는 스냅샷에 대한 * Actions * 열의 Options 메뉴에서 * Delete snapshot * 을 선택합니다.
4. 삭제를 확인하려면 "delete"라는 단어를 입력하고 * Yes, Delete snapshot * 을 선택합니다.

결과

Astra Control Center가 스냅샷을 삭제합니다.

백업을 취소합니다

진행 중인 백업을 취소할 수 있습니다.

> 백업을 취소하려면 백업이 실행 중 상태여야 합니다. 보류 중인 백업은 취소할 수 없습니다.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. Backups * 를 선택합니다.
4. 원하는 백업에 대한 * Actions * 열의 Options 메뉴에서 * Cancel * 을 선택합니다.
5. 삭제를 확인하려면 "취소"라는 단어를 입력하고 * 예, 백업 취소 * 를 선택합니다.

백업을 삭제합니다

더 이상 필요하지 않은 예약된 백업 또는 필요 시 백업을 삭제합니다.

> 실행 중인 백업을 중지할 방법은 없습니다. 백업을 삭제해야 하는 경우 백업이 완료될 때까지 기다린 후 다음 지침을 따르십시오. 실패한 백업을 삭제하려면 "Astra Control API를 사용합니다".

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. Backups * 를 선택합니다.
4. 원하는 백업에 대한 * Actions * 열의 Options 메뉴에서 * Delete backup * 을 선택합니다.
5. 삭제를 확인하려면 "delete"라는 단어를 입력하고 * Yes, Delete backup * 을 선택합니다.

결과

Astra Control Center가 백업을 삭제합니다.

앱 복원

Astra Control은 스냅샷 또는 백업에서 애플리케이션을 복원할 수 있습니다. 애플리케이션을

동일한 클러스터로 복구할 경우 기존 스냅샷에서 복구하는 속도가 빨라집니다. Astra Control UI 또는 를 사용할 수 있습니다 "Astra Control API" 앱을 복원합니다.

이 작업에 대해

- 응용 프로그램을 복원하기 전에 응용 프로그램의 스냅샷을 생성하거나 백업하는 것이 좋습니다. 이렇게 하면 복구에 실패한 경우 스냅샷 또는 백업에서 클론을 생성할 수 있습니다.
- Helm을 사용하여 앱을 배포하는 경우 Astra Control Center에 Helm 버전 3이 필요합니다. Helm 3으로 배포된 애플리케이션 관리 및 복제(또는 Helm 2에서 Helm 3으로 업그레이드)가 완벽하게 지원됩니다. Helm 2와 함께 배포된 앱은 지원되지 않습니다.
- 다른 클러스터로 복원하는 경우 클러스터에서 동일한 영구 볼륨 액세스 모드(예: ReadWriteMany)를 사용하고 있는지 확인합니다. 대상 영구 볼륨 액세스 모드가 다르면 복원 작업이 실패합니다.
- 네임스페이스 이름/ID 또는 네임스페이스 레이블에 의해 네임스페이스 제한이 있는 구성원 사용자는 동일한 클러스터 또는 조직 계정의 다른 클러스터에 있는 새 네임스페이스에 앱을 클론 복제하거나 복원할 수 있습니다. 그러나 동일한 사용자가 새 네임스페이스에서 복제되거나 복원된 앱에 액세스할 수 없습니다. 클론 또는 복원 작업을 통해 새 네임스페이스를 생성한 후 계정 관리자/소유자는 구성원 사용자 계정을 편집하고 영향을 받는 사용자의 역할 제약 조건을 업데이트하여 새 네임스페이스에 대한 액세스 권한을 부여할 수 있습니다.
- OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 만들면 프로젝트(또는 Kubernetes 네임스페이스)에 SecurityContext UID가 할당됩니다. Astra Control Center에서 앱을 보호하고 OpenShift의 다른 클러스터 또는 프로젝트로 앱을 이동하려면 해당 앱을 UID로 실행할 수 있는 정책을 추가해야 합니다. 예를 들어 다음 OpenShift CLI 명령은 WordPress 앱에 적절한 정책을 부여합니다.

```
oc new-project wordpress
oc adm policy add-scc-to-group anyuid system:serviceaccounts:wordpress
oc adm policy add-scc-to-user privileged -z default -n wordpress
```

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. 스냅샷에서 복구하려면 * 스냅샷 * 아이콘을 선택한 상태로 유지합니다. 그렇지 않으면 * Backups * 아이콘을 선택하여 백업에서 복원합니다.
4. 복원하려는 스냅샷 또는 백업의 * 작업 * 열에 있는 옵션 메뉴에서 * 응용 프로그램 복원 * 을 선택합니다.
5. * Restore details *: 복원된 앱에 대한 세부 정보를 지정합니다. 기본적으로 현재 클러스터와 네임스페이스가 표시됩니다. 앱을 원래 상태로 복원하려면 이 값을 그대로 두십시오. 이렇게 하면 앱이 이전 버전으로 되돌아갑니다. 다른 클러스터 또는 네임스페이스로 복원하려는 경우 이 값을 변경합니다.
   - 앱의 이름과 네임스페이스를 입력합니다.
   - 앱의 대상 클러스터를 선택합니다.
   - Review * 를 선택합니다.

   > 이전에 삭제된 네임스페이스에 복원하는 경우 복원 프로세스의 일부로 동일한 이름의 새 네임스페이스가 만들어집니다. 이전에 삭제된 네임스페이스에서 앱을 관리할 권한이 있는 사용자는 새로 다시 생성된 네임스페이스에 대한 권한을 수동으로 복원해야 합니다.

6. * 복원 요약 *: 복원 작업에 대한 세부 정보를 검토하고 "복원"을 입력한 다음 * 복원 * 을 선택합니다.

결과

Astra Control Center는 사용자가 제공한 정보를 기반으로 앱을 복원합니다. 앱을 제자리에 복원한 경우 기존 영구 볼륨의 콘텐츠가 복원된 앱의 영구 볼륨 내용으로 바뀝니다.

> 데이터 보호 작업(클론, 백업, 복원)과 후속 영구 볼륨 크기 조정 후 웹 UI에 새 볼륨 크기가 표시되기까지 최대 20분이 지연됩니다. 데이터 보호 작업이 몇 분 내에 성공적으로 완료되며 스토리지 백엔드에 관리 소프트웨어를 사용하여 볼륨 크기 변경을 확인할 수 있습니다.

SnapMirror 기술을 사용하여 원격 시스템에 애플리케이션을 복제합니다

Astra Control을 사용하면 NetApp SnapMirror 기술의 비동기식 복제 기능을 사용하여 낮은 RPO(복구 시점 목표) 및 낮은 RTO(복구 시간 목표)로 애플리케이션에 대한 비즈니스 연속성을 구축할 수 있습니다. 이 기능을 구성하면 애플리케이션에서 클러스터 간에 데이터 및 애플리케이션 변경사항을 복제할 수 있습니다.

백업/복구와 복제를 비교하려면 을 참조하십시오 "데이터 보호 개념".

다음과 같은 사내 전용, 하이브리드 및 멀티 클라우드 시나리오와 같은 다양한 시나리오에서 앱을 복제할 수 있습니다.

- 사내 사이트 A에서 사내 사이트 B로
- Cloud Volumes ONTAP를 사용하여 사내에서 클라우드로 전환
- Cloud Volumes ONTAP를 사용하는 클라우드를 사내에서 운영
- Cloud Volumes ONTAP를 사용하는 클라우드(동일한 클라우드 공급자 내의 서로 다른 지역 또는 다른 클라우드 공급자 간)

Astra Control은 사내 클러스터, 사내 클러스터, 클라우드(Cloud Volumes ONTAP 사용) 또는 클라우드 간(Cloud Volumes ONTAP에서 Cloud Volumes ONTAP로) 애플리케이션을 복제할 수 있습니다.

> 다른 클러스터 또는 사이트에서 실행 중인 다른 앱을 반대 방향으로 동시에 복제할 수 있습니다. 예를 들어, 애플리케이션 A, B, C를 데이터 센터 1에서 데이터 센터 2로 복제하고 애플리케이션 X, Y, Z를 데이터 센터 2에서 데이터 센터 1로 복제할 수 있습니다.

Astra Control을 사용하면 애플리케이션 복제와 관련된 다음 작업을 수행할 수 있습니다.

- 복제 관계를 설정합니다
- 대상 클러스터에서 복제된 앱을 온라인 상태로 전환(페일오버)
- 페일오버된 복제 다시 동기화
- 애플리케이션 복제를 역으로 수행합니다
- 애플리케이션을 원래 소스 클러스터로 페일백합니다
- 애플리케이션 복제 관계를 삭제합니다

복제 사전 요구 사항

를 참조하십시오 "복제 사전 요구 사항" 시작하기 전에.

복제 관계를 설정합니다

복제 관계를 설정하려면 복제 정책을 구성하는 다음 작업이 필요합니다.

- Astra Control에서 애플리케이션 스냅샷을 얼마나 자주 생성할지 선택(앱의 Kubernetes 리소스 및 각 앱의 볼륨에 대한 볼륨 스냅샷 포함)
- 복제 일정 선택(Kubernetes 리소스 및 영구 볼륨 데이터 포함)
- 스냅샷을 촬영할 시간 설정

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 애플리케이션 페이지에서 * 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 데이터 보호 > 복제 탭에서 * 복제 정책 구성 * 을 선택합니다. 또는 애플리케이션 보호 상자에서 작업 옵션을 선택하고 * 복제 정책 구성 * 을 선택합니다.
4. 다음 정보를 입력하거나 선택합니다.
   - 타겟 클러스터
   - * 대상 스토리지 클래스 *: 대상 ONTAP 클러스터에서 쌍을 이루는 SVM을 사용하는 스토리지 클래스를 선택하거나 입력합니다.
   - * 복제 유형 *: "비동기"는 현재 사용 가능한 유일한 복제 유형입니다.
   - * 대상 네임스페이스 *: 대상 클러스터에 대한 새 또는 기존 대상 네임스페이스를 입력합니다.

     ⓘ 선택한 네임스페이스의 충돌하는 모든 리소스를 덮어씁니다.

   - * 복제 빈도 *: Astra Control이 스냅샷을 생성하여 대상에 복제할 빈도를 설정합니다.
   - * 오프셋 *: Astra Control에서 스냅샷을 생성할 시간(분)을 설정합니다. 다른 예약된 작업과 일치하지 않도록 오프셋을 사용할 수 있습니다. 예를 들어 10:02부터 5분마다 스냅샷을 만들려는 경우 오프셋 분으로 "02"를 입력합니다. 결과는 10:02, 10:07, 10:12 등이 될 것입니다
5. 다음 * 을 선택하고 요약을 검토하고 * 저장 * 을 선택합니다.

   ⓘ 첫 번째 일정이 발생하기 전에 상태가 "APP-MIRROR"로 표시됩니다.

   Astra Control은 복제에 사용되는 애플리케이션 스냅샷을 생성합니다.

6. 응용 프로그램 스냅샷 상태를 보려면 * 응용 프로그램 * > * 스냅샷 * 탭을 선택합니다.

   스냅샷 이름은 "replication-schedule-<string>" 형식을 사용합니다. Astra Control은 복제에 사용된 마지막 스냅샷을 보존합니다. 복제를 성공적으로 완료한 후에는 이전의 모든 복제 스냅샷이 삭제됩니다.

결과

그러면 복제 관계가 생성됩니다.

Astra Control은 관계를 수립함으로써 다음과 같은 조치를 수행합니다.

- 대상에서 네임스페이스 생성(없는 경우)

- 소스 앱의 PVC에 해당하는 대상 네임스페이스에 PVC를 생성합니다.
- 애플리케이션 정합성이 보장되는 초기 Snapshot을 만듭니다.
- 초기 스냅샷을 사용하여 영구 볼륨의 SnapMirror 관계를 설정합니다.

데이터 보호 페이지에는 복제 관계 상태 및 상태가 표시됩니다. <상태>|<관계 수명 주기 상태>

예: Normal | 설정합니다

아래에서 복제 상태 및 상태에 대해 자세히 알아보십시오.

대상 클러스터에서 복제된 앱을 온라인 상태로 전환(페일오버)

Astra Control을 사용하면 복제된 애플리케이션을 대상 클러스터로 "페일오버"할 수 있습니다. 이 절차는 복제 관계를 중지하고 대상 클러스터에서 앱을 온라인으로 전환합니다. 이 절차를 수행해도 소스 클러스터에서 앱이 중지되지 않습니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 애플리케이션 페이지에서 * 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 데이터 보호 > 복제 탭의 작업 메뉴에서 * 페일오버 * 를 선택합니다.
4. 페일오버 페이지에서 정보를 검토하고 * 페일오버 * 를 선택합니다.

결과

페일오버 절차로 인해 다음 작업이 수행됩니다.

- 대상 클러스터에서 최신 복제 스냅샷을 기반으로 앱이 시작됩니다.
- 소스 클러스터와 앱(작동 중인 경우)이 중지되지 않고 계속 실행됩니다.
- 복제 상태가 "페일오버 중"으로 변경되고, 완료되면 "페일오버 실패"로 변경됩니다.
- 소스 앱의 보호 정책은 장애 조치 시 소스 앱에 있는 일정에 따라 대상 앱에 복사됩니다.
- Astra Control은 소스 및 대상 클러스터와 해당 상태 모두에서 앱을 표시합니다.

페일오버된 복제 다시 동기화

재동기화 작업은 복제 관계를 다시 설정합니다. 관계의 소스를 선택하여 소스 또는 타겟 클러스터에 데이터를 유지할 수 있습니다. 이 작업은 SnapMirror 관계를 다시 설정하여 원하는 방향으로 볼륨 복제를 시작합니다.

이 프로세스는 복제를 다시 설정하기 전에 새 대상 클러스터에서 앱을 중지합니다.

> 재동기화 프로세스 중에 수명 주기 상태가 "설정 중"으로 표시됩니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 애플리케이션 페이지에서 * 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 데이터 보호 > 복제 탭의 작업 메뉴에서 * 재동기화 * 를 선택합니다.

4. 재동기화 페이지에서 보존할 데이터가 포함된 소스 또는 대상 앱 인스턴스를 선택합니다.

> ! 대상의 데이터를 덮어쓰므로 재동기화 소스를 신중하게 선택합니다.

5. 계속하려면 * 재동기화 * 를 선택하십시오.
6. "resync"를 입력하여 확인합니다.
7. 예, 재동기화 * 를 선택하여 완료합니다.

결과

- 복제 페이지에는 복제 상태로 "설정 중"이 표시됩니다.
- Astra Control은 새 대상 클러스터에서 애플리케이션을 중지합니다.
- Astra Control은 SnapMirror 재동기화를 사용하여 선택한 방향으로 영구 볼륨 복제를 다시 설정합니다.
- 복제 페이지에는 업데이트된 관계가 표시됩니다.

애플리케이션 복제를 역으로 수행합니다

원래 소스 클러스터로 계속 복제하면서 애플리케이션을 대상 클러스터로 이동하기 위한 계획된 작업입니다. Astra Control은 소스 클러스터에서 애플리케이션을 중지하고 대상 클러스터에 앱을 페일오버하기 전에 데이터를 대상에 복제합니다.

이 경우 소스와 대상을 스와핑합니다. 원래 소스 클러스터가 새 대상 클러스터가 되고 원래 타겟 클러스터가 새 소스 클러스터가 됩니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 애플리케이션 페이지에서 * 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 데이터 보호 > 복제 탭의 동작 메뉴에서 * 역방향 복제 * 를 선택합니다.
4. 역방향 복제 페이지에서 정보를 검토하고 계속하려면 * 역방향 복제 * 를 선택합니다.

결과

역방향 복제의 결과로 다음 작업이 수행됩니다.

- 원본 소스 앱의 Kubernetes 리소스 에 대한 스냅샷이 촬영됩니다.
- 앱의 Kubernetes 리소스를 삭제하여 원본 소스 앱의 Pod를 정상적으로 중지할 수 있습니다(PVC 및 PVS를 그대로 둡니다).
- 포드가 종료된 후 앱 볼륨의 스냅샷이 촬영되고 복제됩니다.
- SnapMirror 관계가 끊어져 타겟 볼륨이 읽기/쓰기 준비가 되었습니다.
- 앱의 Kubernetes 리소스는 원래 소스 애플리케이션이 종료된 후 복제된 볼륨 데이터를 사용하여 사전 종료 Snapshot에서 복원됩니다.
- 복제는 반대 방향으로 다시 설정됩니다.

애플리케이션을 원래 소스 클러스터로 페일백합니다

Astra Control을 사용하면 다음과 같은 일련의 작업을 통해 "장애 조치" 작업 후에 "장애 복구"를 달성할 수 있습니다. 이 워크플로우에서 원래 복제 방향을 복구하기 위해 Astra Control은 복제 방향을 바꾸기 전에 애플리케이션 변경 사항을 원래 소스 클러스터로 복제(재동기화)합니다.

이 프로세스는 대상에 대한 장애 조치를 완료한 관계로부터 시작되며 다음 단계를 포함합니다.

- 페일오버된 상태로 시작합니다.
- 관계를 다시 동기화합니다.
- 복제를 역으로 수행합니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 애플리케이션 페이지에서 * 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 데이터 보호 > 복제 탭의 작업 메뉴에서 * 재동기화 * 를 선택합니다.
4. 장애 복구 작업의 경우 페일오버된 앱을 재동기화 작업의 소스로 선택합니다(기록된 모든 데이터 장애 조치 유지).
5. "resync"를 입력하여 확인합니다.
6. 예, 재동기화 * 를 선택하여 완료합니다.
7. 재동기화가 완료되면 데이터 보호 > 복제 탭의 동작 메뉴에서 * 역방향 복제 * 를 선택합니다.
8. 역방향 복제 페이지에서 정보를 검토하고 * 역방향 복제 * 를 선택합니다.

결과

이렇게 하면 "재동기화" 및 "역관계" 작업의 결과가 결합되어 원래 소스 클러스터에서 애플리케이션이 온라인 상태가 되고 복제가 원래 대상 클러스터로 다시 시작됩니다.

애플리케이션 복제 관계를 삭제합니다

관계를 삭제하면 두 개의 별도 앱이 서로 관계가 없습니다.

단계

1. Astra Control 왼쪽 탐색 모음에서 * 응용 프로그램 * 을 선택합니다.
2. 애플리케이션 페이지에서 * 데이터 보호 * > * 복제 * 탭을 선택합니다.
3. 데이터 보호 > 복제 탭의 애플리케이션 보호 상자 또는 관계 다이어그램에서 * 복제 관계 삭제 * 를 선택합니다.

결과

복제 관계를 삭제하면 다음과 같은 작업이 수행됩니다.

- 관계가 설정되었지만 대상 클러스터에서 앱이 아직 온라인 상태가 되지 않은 경우(장애 발생) Astra Control은 초기화 중에 생성된 PVC를 유지하고 "비어 있는" 관리 앱을 대상 클러스터에 남겨두고 생성된 백업을 유지할 수 있도록 대상 앱을 유지합니다.
- 대상 클러스터에서 앱이 온라인 상태가 된 경우(장애 발생), Astra Control은 PVC 및 대상 앱을 유지합니다. 이제 소스 및 대상 앱이 독립 앱으로 취급됩니다. 백업 스케줄은 두 애플리케이션 모두에 유지되지만 서로 연결되지 않습니다.

복제 관계 상태 및 관계 수명 주기 상태입니다

Astra Control은 복제 관계의 관계 상태와 수명 주기의 상태를 표시합니다.

복제 관계 상태

다음 상태는 복제 관계의 상태를 나타냅니다.

- * 정상 *: 관계가 설정되었거나 설정되었으며 최근 스냅샷이 성공적으로 전송되었습니다.
- * 경고 *: 관계가 페일오버되었거나 페일오버되었습니다(따라서 소스 앱을 더 이상 보호하지 않음).
- * 심각 *
  - 관계가 설정 또는 페일오버되고 마지막 조정 시도가 실패했습니다.
  - 관계가 성립되고 새로운 PVC의 추가를 조정하기 위한 마지막 시도가 실패합니다.
  - 관계가 설정되지만(성공한 스냅샷은 복제되고 페일오버는 가능) 가장 최근의 스냅샷이 실패했거나 복제하지 못했습니다.

복제 수명 주기 상태입니다

다음 상태는 복제 주기의 여러 단계를 반영합니다.

- * 설정 *: 새 복제 관계가 생성됩니다. Astra Control은 필요한 경우 네임스페이스를 생성하고, 대상 클러스터의 새 볼륨에 지속적인 PVC(Volume Claim)를 생성하여 SnapMirror 관계를 생성합니다. 이 상태는 복제가 재동기화 중이거나 복제 재동기화 중임을 나타낼 수도 있습니다.
- * 설정됨 *: 복제 관계가 있습니다. Astra Control은 주기적으로 PVC가 사용 가능한지 확인하고, 복제 관계를 확인하고, 정기적으로 앱의 스냅샷을 생성하고, 앱에서 새로운 소스 PVC를 식별합니다. 이 경우 Astra Control은 복제에 포함할 리소스를 생성합니다.
- * 페일오버 *: Astra Control은 SnapMirror 관계를 중단시키고 마지막으로 성공한 복제 애플리케이션 Snapshot에서 앱의 Kubernetes 리소스를 복원합니다.
- * 페일오버됨 *: Astra Control은 소스 클러스터에서 복제를 중지하고, 대상에서 최근(성공한) 복제 앱 Snapshot을 사용하고, Kubernetes 리소스를 복원합니다.
- * 재동기화 *: Astra Control SnapMirror 재동기화를 사용하여 재동기화 소스의 새 데이터를 재동기화 대상으로 재동기화합니다. 이 작업은 동기화 방향에 따라 대상의 일부 데이터를 덮어쓸 수 있습니다. Astra Control은 대상 네임스페이스에서 실행 중인 앱을 중지하고 Kubernetes 앱을 제거합니다. 재동기화 프로세스 중에 상태가 "설정 중"으로 표시됩니다.
- * 후진 *: 은 원래 소스 클러스터로 계속 복제하면서 애플리케이션을 대상 클러스터로 이동하기 위한 계획된 작업입니다. Astra Control은 소스 클러스터에서 애플리케이션을 중지하고, 대상 클러스터에 앱을 페일오버하기 전에 데이터를 대상에 복제합니다. 역방향 복제 중에 상태가 "설정 중"으로 표시됩니다.
- * 삭제 *:
  - 복제 관계가 설정되었지만 아직 페일오버되지 않은 경우 Astra Control은 복제 중에 생성된 PVC를 제거하고 대상 관리 앱을 삭제합니다.
  - 복제가 이미 실패한 경우 Astra Control은 PVC 및 대상 앱을 유지합니다.

애플리케이션 클론 복제 및 마이그레이션

기존 앱을 클론 복제하여 동일한 Kubernetes 클러스터 또는 다른 클러스터에 중복 앱을

생성합니다. Astra Control Center에서 앱을 클론하면 애플리케이션 구성 및 영구 스토리지의 클론이 생성됩니다.

Kubernetes 클러스터 간에 애플리케이션 및 스토리지를 이동해야 하는 경우 클로닝에 도움이 될 수 있습니다. 예를 들어, CI/CD 파이프라인과 Kubernetes 네임스페이스 전체에서 워크로드를 이동할 수 있습니다. Astra UI 또는 를 사용할 수 있습니다 "Astra Control API" 앱을 클론 복제 및 마이그레이션합니다.

필요한 것

앱을 다른 클러스터로 클론 복제하려면 기본 버킷이 필요합니다. 첫 번째 버킷을 추가하면 기본 버킷을 사용할 수 있습니다.

이 작업에 대해

- StorageClass가 명시적으로 설정된 앱을 배포하고 앱을 복제해야 하는 경우 타겟 클러스터에 원래 지정된 StorageClass가 있어야 합니다. 명시적으로 StorageClass를 동일한 StorageClass가 없는 클러스터로 설정한 애플리케이션을 클론 복제하면 실패합니다.
- Jenkins CI의 운영자 배포 인스턴스를 복제하는 경우 영구 데이터를 수동으로 복원해야 합니다. 이는 앱 배포 모델의 제한 사항입니다.
- Astra Control Center의 S3 버킷은 가용 용량을 보고하지 않습니다. Astra Control Center에서 관리하는 앱을 백업 또는 클론 생성하기 전에 ONTAP 또는 StorageGRID 관리 시스템에서 버킷 정보를 확인하십시오.
- 애플리케이션 백업 또는 애플리케이션 복구 중에 버킷 ID를 선택적으로 지정할 수 있습니다. 그러나 애플리케이션 클론 작업에서는 항상 정의된 기본 버킷을 사용합니다. 클론의 버킷을 변경할 수 있는 옵션은 없습니다. 어떤 버킷이 사용되는지 제어하려는 경우 이 두 가지 방법을 사용할 수 있습니다 "버킷 기본값을 변경합니다" 또는 을 수행합니다 "백업" 뒤에 가 있습니다 "복원" 별도.
- 네임스페이스 이름/ID 또는 네임스페이스 레이블에 의해 네임스페이스 제한이 있는 구성원 사용자는 동일한 클러스터 또는 조직 계정의 다른 클러스터에 있는 새 네임스페이스에 앱을 클론 복제하거나 복원할 수 있습니다. 그러나 동일한 사용자가 새 네임스페이스에서 복제되거나 복원된 앱에 액세스할 수 없습니다. 클론 또는 복원 작업을 통해 새 네임스페이스를 생성한 후 계정 관리자/소유자는 구성원 사용자 계정을 편집하고 영향을 받는 사용자의 역할 제약 조건을 업데이트하여 새 네임스페이스에 대한 액세스 권한을 부여할 수 있습니다.

OpenShift 고려 사항

- 클러스터 간에 앱을 복제하는 경우 소스 클러스터와 대상 클러스터는 OpenShift의 배포 환경과 동일해야 합니다. 예를 들어 OpenShift 4.7 클러스터에서 앱을 클론하는 경우 OpenShift 4.7인 대상 클러스터를 사용합니다.
- OpenShift 클러스터에서 앱을 호스팅하기 위한 프로젝트를 만들면 프로젝트(또는 Kubernetes 네임스페이스)에 SecurityContext UID가 할당됩니다. Astra Control Center에서 앱을 보호하고 OpenShift의 다른 클러스터 또는 프로젝트로 앱을 이동하려면 해당 앱을 UID로 실행할 수 있는 정책을 추가해야 합니다. 예를 들어 다음 OpenShift CLI 명령은 WordPress 앱에 적절한 정책을 부여합니다.

```
oc new-project wordpress
oc adm policy add-scc-to-group anyuid system:serviceaccounts:wordpress
oc adm policy add-scc-to-user privileged -z default -n wordpress
```

단계

1. 응용 프로그램 * 을 선택합니다.
2. 다음 중 하나를 수행합니다.
    - 원하는 앱의 * Actions * 열에서 Options 메뉴를 선택합니다.
    - 원하는 앱의 이름을 선택하고 페이지 오른쪽 상단의 상태 드롭다운 목록을 선택합니다.

3. 클론 * 을 선택합니다.
4. * 클론 세부 정보 *: 클론에 대한 세부 정보 지정:
    - 이름을 입력합니다.
    - 클론의 네임스페이스를 입력합니다.
    - 클론의 대상 클러스터를 선택합니다.
    - 기존 스냅샷이나 백업에서 클론을 생성할지 여부를 선택합니다. 이 옵션을 선택하지 않으면 Astra Control Center는 앱의 현재 상태에서 클론을 생성합니다.
5. * 소스 *: 기존 스냅샷 또는 백업에서 복제하도록 선택한 경우 사용할 스냅샷 또는 백업을 선택합니다.
6. Review * 를 선택합니다.
7. * 클론 요약 *: 클론에 대한 세부 정보를 검토하고 * 클론 * 을 선택합니다.

결과

Astra Control Center는 사용자가 제공한 정보를 기반으로 해당 앱을 복제합니다. 새 애플리케이션 클론이 에 있을 때 클론 작업이 성공적으로 수행됩니다 `Available` 상태를 표시합니다.

> 데이터 보호 작업(클론, 백업, 복원)과 후속 영구 볼륨 크기 조정 후 UI에 새 볼륨 크기가 표시되기까지 최대 20분이 지연됩니다. 데이터 보호 작업이 몇 분 내에 성공적으로 완료되며 스토리지 백엔드에 관리 소프트웨어를 사용하여 볼륨 크기 변경을 확인할 수 있습니다.

앱 실행 후크 관리

실행 후크는 관리되는 앱의 데이터 보호 작업과 함께 실행되도록 구성할 수 있는 사용자 지정 작업입니다. 예를 들어 데이터베이스 앱이 있는 경우 실행 후크를 사용하여 스냅샷 전에 모든 데이터베이스 트랜잭션을 일시 중지하고 스냅샷이 완료된 후 트랜잭션을 다시 시작할 수 있습니다. 따라서 애플리케이션 정합성이 보장되는 스냅샷이 보장됩니다.

실행 후크 유형

Astra Control은 실행 가능 시점을 기준으로 다음과 같은 유형의 실행 후크를 지원합니다.

- 사전 스냅샷
- 사후 스냅샷
- 사전 백업
- 백업 후
- 사후 복원

사용자 정의 실행 후크에 대한 중요 참고 사항

앱에 대한 실행 후크를 계획할 때 다음 사항을 고려하십시오.

- 실행 후크는 스크립트를 사용하여 작업을 수행해야 합니다. 많은 실행 후크가 동일한 스크립트를 참조할 수 있습니다.

- Astra Control에는 실행 후크가 실행 가능한 셸 스크립트 형식으로 기록하는 데 사용하는 스크립트가 필요합니다.
- 스크립트 크기는 96KB로 제한됩니다.
- Astra Control은 실행 후크 설정과 모든 일치 기준을 사용하여 스냅샷, 백업 또는 복구 작업에 적용할 수 있는 후크를 결정합니다.
- 모든 실행 후크 장애는 소프트 장애이며, 후크가 실패하더라도 다른 후크와 데이터 보호 작업은 계속 시도됩니다. 그러나 후크가 실패하면 * Activity * 페이지 이벤트 로그에 경고 이벤트가 기록됩니다.
- 실행 후크를 생성, 편집 또는 삭제하려면 소유자, 관리자 또는 구성원 권한이 있는 사용자여야 합니다.
- 실행 후크를 실행하는 데 25분 이상 걸리는 경우 후크에 장애가 발생하고 반환 코드가 "N/A"인 이벤트 로그 항목이 생성됩니다. 영향을 받는 모든 스냅샷은 시간 초과되어 실패로 표시되며, 그 결과 이벤트 로그 항목이 시간 초과를 나타냅니다.
- 임시 데이터 보호 작업의 경우 모든 후크 이벤트가 생성되고 * Activity * 페이지 이벤트 로그에 저장됩니다. 그러나 예약된 데이터 보호 작업의 경우 후크 장애 이벤트만 이벤트 로그에 기록됩니다(예약된 데이터 보호 작업 자체에서 생성되는 이벤트는 계속 기록됨).

> 실행 후크는 실행 중인 응용 프로그램의 기능을 줄이거나 완전히 비활성화하기 때문에 사용자 지정 실행 후크가 실행되는 시간을 최소화해야 합니다. 연결된 실행 후크와 함께 백업 또는 스냅샷 작업을 시작한 다음 취소하면 백업 또는 스냅샷 작업이 이미 시작된 경우에도 후크를 실행할 수 있습니다. 즉, 백업 후 실행 후크는 백업이 완료된 것으로 가정할 수 없습니다.

실행 순서

데이터 보호 작업이 실행되면 실행 후크 이벤트가 다음 순서로 발생합니다.

1. 해당되는 모든 사용자 정의 사전 작업 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사전 작업 후크를 만들고 실행할 수 있지만, 이 후크의 실행 순서는 보장되거나 구성할 수 없습니다.
2. 데이터 보호 작업이 수행됩니다.
3. 해당되는 모든 사용자 지정 작업 후 실행 후크는 해당 컨테이너에서 실행됩니다. 필요한 만큼 사용자 지정 사후 작업 후크를 만들고 실행할 수 있지만 작업 후 후크의 실행 순서는 보장되거나 구성할 수 없습니다.

같은 유형의 실행 후크를 여러 개 생성하는 경우(예: 사전 스냅샷) 해당 후크의 실행 순서는 보장되지 않습니다. 그러나 다른 유형의 후크를 실행하는 순서는 보장됩니다. 예를 들어, 5가지 유형의 후크가 모두 있는 구성의 실행 순서는 다음과 같습니다.

1. 예비 후크가 실행되었습니다
2. 사전 스냅샷 후크가 실행되었습니다
3. 사후 스냅샷 후크가 실행되었습니다
4. 백업 후 후크가 실행되었습니다
5. 복원 후 후크가 실행되었습니다

시나리오 번호 2에서 이 구성의 예를 볼 수 있습니다 후크가 실행될지 여부를 결정합니다.

> 운영 환경에서 실행 후크 스크립트를 사용하려면 항상 해당 스크립트를 테스트해야 합니다. 'kubbeck exec' 명령을 사용하여 스크립트를 편리하게 테스트할 수 있습니다. 운영 환경에서 실행 후크를 사용하도록 설정한 후 결과 스냅샷과 백업을 테스트하여 정합성이 보장되는지 확인합니다. 앱을 임시 네임스페이스에 클론 복제하고, 스냅샷 또는 백업을 복원한 다음 앱을 테스트하여 이 작업을 수행할 수 있습니다.

후크가 실행될지 여부를 결정합니다

다음 표를 사용하여 사용자 지정 실행 후크가 앱에 대해 실행되는지 여부를 확인할 수 있습니다.

모든 상위 수준 앱 작업은 스냅샷, 백업 또는 복원의 기본 작업 중 하나를 실행하는 것으로 구성됩니다. 시나리오에 따라 클론 작업은 이러한 작업의 다양한 조합으로 구성되므로 클론 작업이 실행되는 실행 후크는 달라집니다.

데이터 이동 없이 복원 작업을 수행하려면 기존 스냅샷 또는 백업이 필요하므로 이러한 작업은 스냅샷 또는 백업 후크를 실행하지 않습니다.

> 를 시작한 다음 스냅샷이 포함된 백업을 취소하고 연결된 실행 후크가 있는 경우 일부 후크가 실행될 수 있고 그렇지 않은 백업이 있을 수 있습니다. 즉, 백업 후 실행 후크는 백업이 완료된 것으로 가정할 수 없습니다. 연결된 실행 후크와 함께 취소된 백업의 경우 다음 사항에 유의하십시오.
>
> - 예비 백업 및 예비 후크는 항상 실행됩니다.
> - 백업에 새 스냅샷이 포함되어 있고 스냅샷이 시작된 경우 사전 스냅샷 및 사후 스냅샷 후크가 실행됩니다.
> - 스냅샷을 시작하기 전에 백업을 취소하면 사전 스냅샷 및 사후 스냅샷 후크가 실행되지 않습니다.

시나리오	작동	기존 스냅샷	더 많은 워크로드 추가/제거	네임스페이스	클러스터	스냅샷 후크가 실행됩니다	백업 후크가 실행됩니다	후크 실행을 복원합니다
1	복제	해당 없음	해당 없음	신규	동일합니다	예	해당 없음	예
2	복제	해당 없음	해당 없음	신규	다릅니다	예	예	예
3	복제 또는 복원	예	해당 없음	신규	동일합니다	해당 없음	해당 없음	예
4	복제 또는 복원	해당 없음	예	신규	동일합니다	해당 없음	해당 없음	예
5	복제 또는 복원	예	해당 없음	신규	다릅니다	해당 없음	예	예
6	복제 또는 복원	해당 없음	예	신규	다릅니다	해당 없음	해당 없음	예
7	복원	예	해당 없음	기존	동일합니다	해당 없음	해당 없음	예
8	복원	해당 없음	예	기존	동일합니다	해당 없음	해당 없음	예
9	스냅샷	해당 없음	해당 없음	해당 없음	해당 없음	예	해당 없음	해당 없음
10	백업	해당 없음	해당 없음	해당 없음	해당 없음	예	예	해당 없음
11	백업	예	해당 없음	해당 없음	해당 없음	해당 없음	예	해당 없음

기존 실행 후크를 봅니다

앱의 기존 사용자 지정 실행 후크를 볼 수 있습니다.

단계

1. 응용 프로그램 * 으로 이동한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.

   결과 목록에서 사용 가능하거나 비활성화된 실행 후크를 모두 볼 수 있습니다. 후크의 상태, 소스 및 실행 시간(사전 또는 사후 작업)을 확인할 수 있습니다. 실행 후크를 둘러싼 이벤트 로그를 보려면 왼쪽 탐색 영역의 * Activity * 페이지로 이동합니다.

기존 스크립트 보기

업로드된 기존 스크립트를 볼 수 있습니다. 또한 이 페이지에서 사용 중인 스크립트와 해당 스크립트를 사용하는 후크를 확인할 수 있습니다.

단계

1. 계정 * 으로 이동합니다.
2. 스크립트 * 탭을 선택합니다.

   이 페이지에서는 업로드된 기존 스크립트 목록을 볼 수 있습니다. Used By* 열에는 각 스크립트를 사용하는 실행 후크가 표시됩니다.

스크립트를 추가합니다

실행 후크가 참조할 수 있는 스크립트를 하나 이상 추가할 수 있습니다. 많은 실행 후크가 동일한 스크립트를 참조할 수 있으므로 하나의 스크립트만 변경하여 여러 실행 후크를 업데이트할 수 있습니다.

단계

1. 계정 * 으로 이동합니다.
2. 스크립트 * 탭을 선택합니다.
3. 추가 * 를 선택합니다.
4. 다음 중 하나를 수행합니다.
   - 사용자 지정 스크립트를 업로드합니다.
     i. 파일 업로드 * 옵션을 선택합니다.
     ii. 파일을 찾아 업로드합니다.
     iii. 스크립트에 고유한 이름을 지정합니다.
     iv. (선택 사항) 다른 관리자가 스크립트에 대해 알아야 하는 참고 사항을 입력합니다.
     v. Save script * 를 선택합니다.
   - 클립보드에서 사용자 정의 스크립트를 붙여 넣습니다.
     i. 붙여넣기 또는 형식 * 옵션을 선택합니다.

ii. 텍스트 필드를 선택하고 필드에 스크립트 텍스트를 붙여 넣습니다.

iii. 스크립트에 고유한 이름을 지정합니다.

iv. (선택 사항) 다른 관리자가 스크립트에 대해 알아야 하는 참고 사항을 입력합니다.

5. Save script * 를 선택합니다.

결과

새 스크립트가 * 스크립트 * 탭의 목록에 나타납니다.

스크립트를 삭제합니다

스크립트가 더 이상 필요하지 않고 실행 후크에서 사용되지 않는 경우 시스템에서 스크립트를 제거할 수 있습니다.

단계

1. 계정 * 으로 이동합니다.
2. 스크립트 * 탭을 선택합니다.
3. 제거할 스크립트를 선택하고 * Actions * 열에서 메뉴를 선택합니다.
4. 삭제 * 를 선택합니다.

> 스크립트가 하나 이상의 실행 후크에 연결되어 있으면 * 삭제 * 작업을 사용할 수 없습니다. 스크립트를 삭제하려면 먼저 연결된 실행 후크를 편집하여 다른 스크립트에 연결합니다.

사용자 지정 실행 후크를 만듭니다

앱의 사용자 정의 실행 후크를 만들 수 있습니다. 을 참조하십시오 "실행 후크 예" 후크 예 실행 후크를 만들려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

> 실행 후크로 사용할 사용자 정의 쉘 스크립트를 작성할 때는 특정 명령을 실행하거나 실행 파일에 대한 전체 경로를 제공하지 않는 한 파일 시작 부분에 적절한 셸을 지정해야 합니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.
3. 추가 * 를 선택합니다.
4. Hook Details * (후크 세부 정보 *) 영역에서 * Operation * (작업 *) 드롭다운 메뉴에서 작업 유형을 선택하여 후크를 실행할 시기를 결정합니다.
5. 후크의 고유한 이름을 입력합니다.
6. (선택 사항) 실행 중에 후크에 전달할 인수를 입력하고 각 인수 뒤에 Enter 키를 눌러 각 인수를 기록합니다.
7. Container Images * (컨테이너 이미지 *) 영역에서 응용 프로그램에 포함된 모든 컨테이너 이미지에 대해 후크를 실행해야 하는 경우 * Apply to all container images * (모든 컨테이너 이미지에 적용) 확인란을 활성화합니다. 대신 후크가 하나 이상의 지정된 컨테이너 이미지에만 동작해야 하는 경우 일치시킬 * 컨테이너 이미지 이름 필드에 컨테이너 이미지 이름을 입력합니다.
8. Script * 영역에서 다음 중 하나를 수행합니다.

- 새 스크립트를 추가합니다.
  i. 추가 * 를 선택합니다.
  ii. 다음 중 하나를 수행합니다.
    - 사용자 지정 스크립트를 업로드합니다.
      I. 파일 업로드 * 옵션을 선택합니다.
      II. 파일을 찾아 업로드합니다.
      III. 스크립트에 고유한 이름을 지정합니다.
      IV. (선택 사항) 다른 관리자가 스크립트에 대해 알아야 하는 참고 사항을 입력합니다.
      V. Save script * 를 선택합니다.
    - 클립보드에서 사용자 정의 스크립트를 붙여 넣습니다.
      I. 붙여넣기 또는 형식 * 옵션을 선택합니다.
      II. 텍스트 필드를 선택하고 필드에 스크립트 텍스트를 붙여 넣습니다.
      III. 스크립트에 고유한 이름을 지정합니다.
      IV. (선택 사항) 다른 관리자가 스크립트에 대해 알아야 하는 참고 사항을 입력합니다.
- 목록에서 기존 스크립트를 선택합니다.

  이렇게 하면 실행 후크에 이 스크립트를 사용하도록 지시합니다.

9. 후크 추가 * 를 선택합니다.

실행 후크의 상태를 확인합니다

스냅샷, 백업 또는 복원 작업이 실행된 후에 작업의 일부로 실행된 실행 후크의 상태를 확인할 수 있습니다. 이 상태 정보를 사용하여 실행 후크를 유지할지, 수정하거나 삭제할 것인지 결정할 수 있습니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. 데이터 보호 * 탭을 선택합니다.
3. 스냅샷 * 을 선택하여 실행 중인 스냅샷을 보거나 * 백업 * 을 선택하여 실행 중인 백업을 확인합니다.

   후크 상태 * 는 작업이 완료된 후 실행 후크의 상태를 표시합니다. 상태 위로 마우스를 가져가면 자세한 정보를 볼 수 있습니다. 예를 들어, 스냅샷 중에 실행 후크 오류가 발생한 경우 해당 스냅샷의 후크 상태 위로 마우스를 이동하면 실패한 실행 후크 목록이 표시됩니다. 각 오류의 원인을 확인하려면 왼쪽 탐색 영역의 * Activity * 페이지를 확인하십시오.

스크립트 사용을 봅니다

Astra Control 웹 UI에서 특정 스크립트를 사용하는 실행 후크를 확인할 수 있습니다.

단계

1. 계정 * 을 선택합니다.
2. 스크립트 * 탭을 선택합니다.

스크립트 목록의 * Used By * 열에 목록의 각 스크립트를 사용하는 후크에 대한 세부 정보가 포함되어 있습니다.

3. 관심 있는 스크립트에 대해 * Used By *(사용 대상 *) 열에서 정보를 선택합니다.

   스크립트를 사용하는 후크의 이름 및 스크립트를 실행하도록 구성된 작업 유형과 함께 더 자세한 목록이 나타납니다.

실행 후크를 비활성화합니다

앱 스냅샷 전후에 실행 후크가 실행되지 않도록 임시로 설정하려면 실행 후크를 사용하지 않도록 설정할 수 있습니다. 실행 후크를 비활성화하려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.
3. 비활성화할 후크의 경우 * Actions * 열에서 옵션 메뉴를 선택합니다.
4. 비활성화 * 를 선택합니다.

실행 후크를 삭제합니다

더 이상 필요 없는 경우 실행 후크를 완전히 제거할 수 있습니다. 실행 후크를 삭제하려면 소유자, 관리자 또는 구성원 권한이 있어야 합니다.

단계

1. 응용 프로그램 * 을 선택한 다음 관리되는 응용 프로그램의 이름을 선택합니다.
2. Execution hook * 탭을 선택합니다.
3. 삭제할 후크의 경우 * Actions * 열에서 옵션 메뉴를 선택합니다.
4. 삭제 * 를 선택합니다.

실행 후크 예

다음 예제를 사용하여 실행 후크를 구조화하는 방법에 대해 알아보십시오. 이러한 후크를 템플릿 또는 테스트 스크립트로 사용할 수 있습니다.

간단한 성공 사례

다음은 표준 출력 및 표준 오류에 성공하여 메시지를 기록하는 간단한 후크의 예입니다.

```
#!/bin/sh

# success_sample.sh
#
# A simple noop success hook script for testing purposes.
#
# args: None
```

```
#


#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running success_sample.sh"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

단순한 성공 사례**(bash** 버전**)**

다음은 bash용으로 작성된 표준 출력 및 표준 오류에 성공하여 메시지를 쓰는 간단한 후크의 예입니다.

```
#!/bin/bash

# success_sample.bash
#
# A simple noop success hook script for testing purposes.
#
# args: None


#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running success_sample.bash"

# exit with 0 to indicate success
```

```
info "exit 0"
exit 0
```

간단한 성공 사례**(zsh** 버전**)**

다음은 Z 셸에 대해 작성된 표준 출력 및 표준 오류에 성공하여 메시지를 기록하는 간단한 후크의 예입니다.

```
#!/bin/zsh

# success_sample.zsh
#
# A simple noop success hook script for testing purposes.
#
# args: None
#



#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}



#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}

#
```

```
# main
#

# log something to stdout
info "running success_sample.zsh"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

인수 성공 예제

다음 예제에서는 후크에 args를 사용하는 방법을 보여 줍니다.

```
#!/bin/sh

# success_sample_args.sh
#
# A simple success hook script with args for testing purposes.
#
# args: Up to two optional args that are echoed to stdout

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
```

```
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running success_sample_args.sh"


# collect args
arg1=$1
arg2=$2

# output args and arg count to stdout
info "number of args: $#"
info "arg1 ${arg1}"
info "arg2 ${arg2}"

# exit with 0 to indicate success
info "exit 0"
exit 0
```

사전 스냅샷/사후 스냅샷 후크의 예

다음 예제에서는 사전 스냅샷 및 사후 스냅샷 후크에 대해 동일한 스크립트를 사용하는 방법을 보여 줍니다.

```
#!/bin/sh

# success_sample_pre_post.sh
#
# A simple success hook script example with an arg for testing purposes
# to demonstrate how the same script can be used for both a prehook and
posthook
#
# args: [pre|post]

# unique error codes for every error case
ebase=100
eusage=$((ebase+1))
ebadstage=$((ebase+2))
epre=$((ebase+3))
epost=$((ebase+4))
```

```
#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# Would run prehook steps here
#
prehook() {
    info "Running noop prehook"
    return 0
}

#
# Would run posthook steps here
#
posthook() {
    info "Running noop posthook"
    return 0
}
```

```
#
# main
#

# check arg
stage=$1
if [ -z "${stage}" ]; then
    echo "Usage: $0 <pre|post>"
    exit ${eusage}
fi

if [ "${stage}" != "pre" ] && [ "${stage}" != "post" ]; then
    echo "Invalid arg: ${stage}"
    exit ${ebadstage}
fi

# log something to stdout
info "running success_sample_pre_post.sh"

if [ "${stage}" = "pre" ]; then
    prehook
    rc=$?
    if [ ${rc} -ne 0 ]; then
        error "Error during prehook"
    fi
fi

if [ "${stage}" = "post" ]; then
    posthook
    rc=$?
    if [ ${rc} -ne 0 ]; then
        error "Error during posthook"
    fi
fi

exit ${rc}
```

실패 예

다음 예제에서는 후크의 장애를 처리하는 방법을 보여 줍니다.

```
#!/bin/sh

# failure_sample_arg_exit_code.sh
#
```

```
# A simple failure hook script for testing purposes.
#
# args: [the exit code to return]
#



#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}



#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}



#
# main
#

# log something to stdout
info "running failure_sample_arg_exit_code.sh"

argexitcode=$1

# log to stderr
error "script failed, returning exit code ${argexitcode}"
```

```
# exit with specified exit code
exit ${argexitcode}
```

자세한 정보 표시 실패 예

다음 예제에서는 더 자세한 정보 로깅을 사용하여 후크의 오류를 처리하는 방법을 보여 줍니다.

```
#!/bin/sh

# failure_sample_verbose.sh
#
# A simple failure hook script with args for testing purposes.
#
# args: [The number of lines to output to stdout]


#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
```

```
# main
#

# log something to stdout
info "running failure_sample_verbose.sh"


# output arg value to stdout
linecount=$1
info "line count ${linecount}"

# write out a line to stdout based on line count arg
i=1
while [ "$i" -le ${linecount} ]; do
    info "This is line ${i} from failure_sample_verbose.sh"
    i=$(( i + 1 ))
done

error "exiting with error code 8"
exit 8
```

종료 코드 예제에 오류가 발생했습니다

다음 예제에서는 종료 코드와 함께 후크 실패를 보여 줍니다.

```
#!/bin/sh

# failure_sample_arg_exit_code.sh
#
# A simple failure hook script for testing purposes.
#
# args: [the exit code to return]
#



#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}



#
```

```
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running failure_sample_arg_exit_code.sh"

argexitcode=$1

# log to stderr
error "script failed, returning exit code ${argexitcode}"

# exit with specified exit code
exit ${argexitcode}
```

실패 후 성공 예

다음 예제에서는 후크가 처음 실행될 때 후크가 실패하지만 두 번째 실행 후에 후크가 발생하는 방법을 보여 줍니다.

```
#!/bin/sh

# failure_then_success_sample.sh
#
# A hook script that fails on initial run but succeeds on second run for
testing purposes.
#
# Helpful for testing retry logic for post hooks.
#
```

```
# args: None
#

#
# Writes the given message to standard output
#
# $* - The message to write
#
msg() {
    echo "$*"
}


#
# Writes the given information message to standard output
#
# $* - The message to write
#
info() {
    msg "INFO: $*"
}

#
# Writes the given error message to standard error
#
# $* - The message to write
#
error() {
    msg "ERROR: $*" 1>&2
}


#
# main
#

# log something to stdout
info "running failure_success sample.sh"


if [ -e /tmp/hook-test.junk ] ; then
    info "File does exist.  Removing /tmp/hook-test.junk"
    rm /tmp/hook-test.junk
    info "Second run so returning exit code 0"
    exit 0
else
```

```
    info "File does not exist.  Creating /tmp/hook-test.junk"
    echo "test" > /tmp/hook-test.junk
    error "Failed first run, returning exit code 5"
    exit 5
fi
```