앱 보호

Astra Control Center

NetApp
November 21, 2023

This PDF was generated from https://docs.netapp.com/ko-kr/astra-control-center-2112/use/protection-overview.html on November 21, 2023. Always check docs.netapp.com for the latest.

목차

앱 보호 1
보호 개요 1
스냅샷 및 백업으로 애플리케이션 보호 1
앱 복원 5
애플리케이션 클론 복제 및 마이그레이션 6
앱 실행 후크 관리 8

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".

애플리케이션 보호 워크플로우

다음 예제 워크플로를 사용하여 앱 보호를 시작할 수 있습니다.

[1개] 모든 앱을 백업합니다

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] 선택 사항**:** 보호 정책을 조정합니다

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

[네] 재해가 발생할 경우 앱을 복원합니다

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).

스냅샷 및 백업으로 애플리케이션 보호

자동화된 보호 정책을 사용하거나 필요에 따라 스냅샷과 백업을 생성하여 앱을 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:WordPress의 OC adm policy add-SCC-to-user privileged-z default-n WordPress

보호 정책을 구성합니다

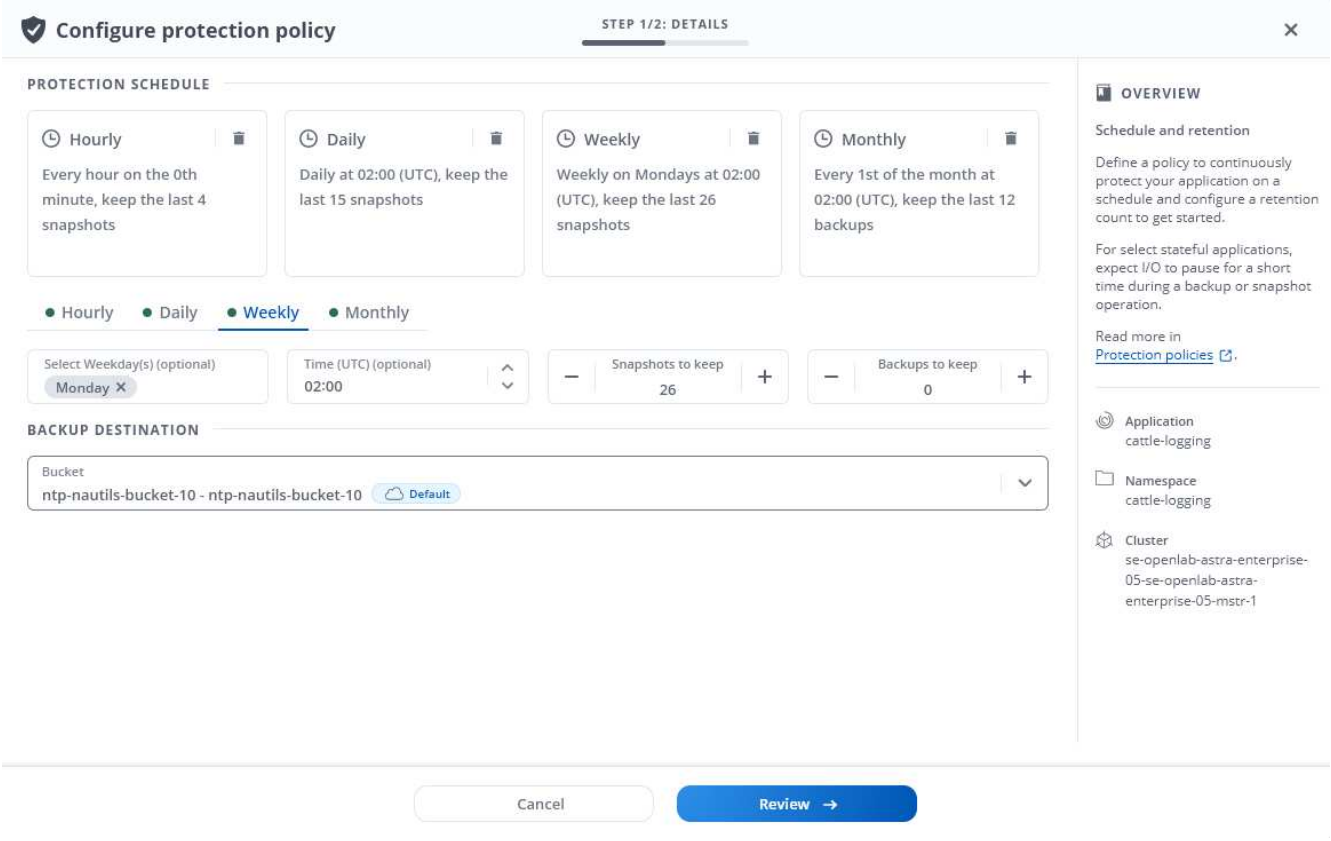
보호 정책은 정의된 일정에 따라 스냅샷, 백업 또는 둘 다를 생성하여 앱을 보호합니다. 시간별, 일별, 주별 및 월별 스냅샷과 백업을 생성하도록 선택할 수 있으며, 보존할 복제본 수를 지정할 수 있습니다. 예를 들어 보호 정책은 주별 백업과 일별 스냅샷을 생성하고 백업 및 스냅샷을 한 달 동안 보존할 수 있습니다. 스냅샷 및 백업을 생성하는 빈도와 보관 기간은 조직의 요구 사항에 따라 다릅니다.

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 * 열에서 드롭다운 목록을 선택합니다.
3. 스냅샷 * 을 선택합니다.
4. 스냅샷의 이름을 사용자 지정한 다음 * Review * 를 선택합니다.
5. 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 * 열에서 드롭다운 목록을 선택합니다.
3. 백업 * 을 선택합니다.
4. 백업 이름을 사용자 지정합니다.
5. 기존 스냅샷에서 앱을 백업할지 여부를 선택합니다. 이 옵션을 선택하면 기존 스냅샷 목록에서 선택할 수 있습니다.
6. 스토리지 버킷 목록에서 선택하여 백업 대상을 선택합니다.
7. Review * 를 선택합니다.
8. 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.

단계

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. 원하는 스냅샷의 * Actions * 열에서 드롭다운 목록을 선택합니다.
4. 스냅샷 삭제 * 를 선택합니다.
5. 삭제를 확인하려면 "delete"라는 단어를 입력하고 * Yes, Delete snapshot * 을 선택합니다.

결과

Astra Control Center가 스냅샷을 삭제합니다.

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".

1. 응용 프로그램 * 을 선택한 다음 앱 이름을 선택합니다.
2. 데이터 보호 * 를 선택합니다.
3. Backups * 를 선택합니다.
4. 원하는 백업의 * Actions * 열에서 드롭다운 목록을 선택합니다.
5. 백업 삭제 * 를 선택합니다.
6. 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.

> 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.

> 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:WordPress의 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. 복구할 스냅샷 또는 백업의 * Actions * 열에서 드롭다운 목록을 선택합니다.
5. Restore application * 을 선택합니다.
6. * 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.
7. * 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 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.

애플리케이션 클론 복제 및 마이그레이션

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

> 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.

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.

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

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.

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:WordPress의 OC adm policy add-SCC-to-user privileged-z default-n WordPress

필요한 것

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

단계

1. 응용 프로그램 * 을 선택합니다.
2. 다음 중 하나를 수행합니다.
   - 원하는 앱의 * Actions * 열에서 드롭다운 목록을 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 앱 클론이 * 응용 프로그램 * 페이지의 "사용 가능" 상태가 되면 클론 작업이 성공적으로 수행됩니다.

> 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 스냅샷 전후에 실행할 수 있는 사용자 지정 스크립트입니다. 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.

기본 실행 후크 및 정규식

일부 애플리케이션의 경우 Astra Control은 NetApp에서 제공하는 기본 실행 후크와 함께 제공되며, 스냅샷 전후에 고정 및 고정 작업을 처리합니다. Astra Control은 정규식을 사용하여 앱의 컨테이너 이미지를 다음과 같은 앱에 일치시킵니다.

- MariaDB
  - 일치하는 정규식:\bmariadb\b
- MySQL
  - 일치 정규식:\bmysql\b
- PostgreSQL
  - 일치하는 정규식:\bpostgresql\b

일치하는 항목이 있으면 해당 앱에 대한 NetApp 제공 기본 실행 후크가 앱의 활성 실행 후크 목록에 나타나고, 해당 앱의 스냅샷을 생성하면 해당 후크가 자동으로 실행됩니다. 사용자 지정 앱 중 하나에 정규식과 일치하는 유사한 이미지 이름이 있는 경우(기본 실행 후크를 사용하지 않으려는 경우) 이미지 이름을 변경할 수 있습니다. 또는 해당 앱에 대한 기본 실행 후크를 비활성화하고 대신 사용자 지정 후크를 사용합니다.

기본 실행 후크는 삭제하거나 수정할 수 없습니다.

사용자 정의 실행 후크에 대한 중요 참고 사항

앱에 대한 실행 후크를 계획할 때 다음 사항을 고려하십시오.

- Astra Control을 사용하려면 실행 가능한 셸 스크립트 형식으로 실행 후크를 작성해야 합니다.
- 스크립트 크기는 128KB로 제한됩니다.
- Astra Control은 실행 후크 설정 및 모든 일치 기준을 사용하여 스냅샷에 적용할 후크를 결정합니다.
- 모든 실행 후크 오류는 소프트 장애이며, 후크에 장애가 발생해도 다른 후크와 스냅샷이 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.

> 실행 후크는 실행 중인 응용 프로그램의 기능을 줄이거나 완전히 비활성화하기 때문에 사용자 지정 실행 후크가 실행되는 시간을 최소화해야 합니다.

스냅샷이 실행되면 실행 후크 이벤트가 다음 순서로 발생합니다.

1. NetApp에서 제공하는 기본 사전 스냅샷 실행 후크는 해당 컨테이너에서 실행됩니다.
2. 해당되는 모든 사용자 지정 사전 스냅샷 실행 후크는 해당 컨테이너에서 실행됩니다. 필요에 따라 사용자 지정 사전 스냅샷 후크를 생성하고 실행할 수 있지만 스냅샷이 보장되거나 구성 가능해지기 전에 이러한 후크의 실행 순서가 보장되지 않습니다.
3. 스냅샷이 수행됩니다.
4. 해당되는 모든 사용자 지정 사후 스냅샷 실행 후크는 해당 컨테이너에서 실행됩니다. 필요에 따라 사용자 지정 사후 스냅샷 후크를 생성하고 실행할 수 있지만 스냅샷 후에 이러한 후크를 실행하는 순서는 보장되거나 구성할 수 없습니다.
5. NetApp에서 제공하는 모든 기본 사후 스냅샷 실행 후크는 해당 컨테이너에서 실행됩니다.

> 운영 환경에서 실행 후크 스크립트를 사용하려면 항상 해당 스크립트를 테스트해야 합니다. 'kubbeck exec' 명령을 사용하여 스크립트를 편리하게 테스트할 수 있습니다. 운영 환경에서 실행 후크를 활성화한 후 결과 스냅샷을 테스트하여 정합성이 보장되는지 확인합니다. 앱을 임시 네임스페이스에 클론 복제하고 스냅샷을 복구한 다음 앱을 테스트하여 이 작업을 수행할 수 있습니다.

기존 실행 후크를 봅니다

앱에 대한 기존 맞춤형 또는 NetApp에서 제공한 기본 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 스냅샷)을 확인할 수 있습니다. 실행 후크를 둘러싼 이벤트 로그를 보려면 왼쪽 탐색 영역의 * Activity * 페이지로 이동합니다.

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 Linux 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. 새 후크 추가 * 를 선택합니다.
4. 후크 세부 정보 * 영역에서 후크를 실행해야 하는 시기에 따라 * 사전 스냅샷 * 또는 * 사후 스냅샷 * 을 선택합니다.
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.
   - 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.
   - 클립보드에서 사용자 정의 스크립트를 붙여 넣습니다.
     i. 클립보드에서 붙여넣기 * 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.
9. 후크 추가 * 를 선택합니다.

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. 비활성화하려는 후크에 대해 * 작업 * 드롭다운을 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 후크에 대한 * 작업 * 드롭다운을 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
```

저작권 정보

Copyright © 2023 NetApp, Inc. All Rights Reserved. 미국에서 인쇄된 본 문서의 어떠한 부분도 저작권 소유자의 사전 서면 승인 없이는 어떠한 형식이나 수단(복사, 녹음, 녹화 또는 전자 검색 시스템에 저장하는 것을 비롯한 그래픽, 전자적 또는 기계적 방법)으로도 복제될 수 없습니다.

NetApp이 저작권을 가진 자료에 있는 소프트웨어에는 아래의 라이센스와 고지사항이 적용됩니다.

본 소프트웨어는 NetApp에 의해 '있는 그대로' 제공되며 상품성 및 특정 목적에의 적합성에 대한 명시적 또는 묵시적 보증을 포함하여(이에 제한되지 않음) 어떠한 보증도 하지 않습니다. NetApp은 대체품 또는 대체 서비스의 조달, 사용 불능, 데이터 손실, 이익 손실, 영업 중단을 포함하여(이에 국한되지 않음), 이 소프트웨어의 사용으로 인해 발생하는 모든 직접 및 간접 손해, 우발적 손해, 특별 손해, 징벌적 손해, 결과적 손해의 발생에 대하여 그 발생 이유, 책임론, 계약 여부, 엄격한 책임, 불법 행위(과실 또는 그렇지 않은 경우)와 관계없이 어떠한 책임도 지지 않으며, 이와 같은 손실의 발생 가능성이 통지되었다 하더라도 마찬가지입니다.

NetApp은 본 문서에 설명된 제품을 언제든지 예고 없이 변경할 권리를 보유합니다. NetApp은 NetApp의 명시적인 서면 동의를 받은 경우를 제외하고 본 문서에 설명된 제품을 사용하여 발생하는 어떠한 문제에도 책임을 지지 않습니다. 본 제품의 사용 또는 구매의 경우 NetApp에서는 어떠한 특허권, 상표권 또는 기타 지적 재산권이 적용되는 라이센스도 제공하지 않습니다.

본 설명서에 설명된 제품은 하나 이상의 미국 특허, 해외 특허 또는 출원 중인 특허로 보호됩니다.

제한적 권리 표시: 정부에 의한 사용, 복제 또는 공개에는 DFARS 252.227-7013(2014년 2월) 및 FAR 52.227-19(2007년 12월)의 기술 데이터-비상업적 품목에 대한 권리(Rights in Technical Data -Noncommercial Items) 조항의 하위 조항 (b)(3)에 설명된 제한사항이 적용됩니다.

여기에 포함된 데이터는 상업용 제품 및/또는 상업용 서비스(FAR 2.101에 정의)에 해당하며 NetApp, Inc.의 독점 자산입니다. 본 계약에 따라 제공되는 모든 NetApp 기술 데이터 및 컴퓨터 소프트웨어는 본질적으로 상업용이며 개인 비용만으로 개발되었습니다. 미국 정부는 데이터가 제공된 미국 계약과 관련하여 해당 계약을 지원하는 데에만 데이터에 대한 전 세계적으로 비독점적이고 양도할 수 없으며 재사용이 불가능하며 취소 불가능한 라이센스를 제한적으로 가집니다. 여기에 제공된 경우를 제외하고 NetApp, Inc.의 사전 서면 승인 없이는 이 데이터를 사용, 공개, 재생산, 수정, 수행 또는 표시할 수 없습니다. 미국 국방부에 대한 정부 라이센스는 DFARS 조항 252.227-7015(b)(2014년 2월)에 명시된 권한으로 제한됩니다.

상표 정보

NETAPP, NETAPP 로고 및 http://www.netapp.com/TM에 나열된 마크는 NetApp, Inc.의 상표입니다. 기타 회사 및 제품 이름은 해당 소유자의 상표일 수 있습니다.